合肥大學
HEFEI UNIVERSITY

An Introduction to Optimization

Thomas Weise (汤卫思)
tweise@hfuu.edu.cn

Institute of Applied Optimization (IAO)
School of Artificial Intelligence and Big Data
Hefei University
Hefei, Anhui, China

应用优化研究所
人工智能与大数据学院
合肥大学
中国安徽省合肥市

Version: 2026-01-29

Outline

Introduction

What is Optimization?

- Are are three ways to approach this topic.

What is Optimization?

- Are are three ways to approach this topic

  1. Optimization is an extension of school mathematics into a field where equations are no longer enough and exact solutions cannot always be reached.

What is Optimization?

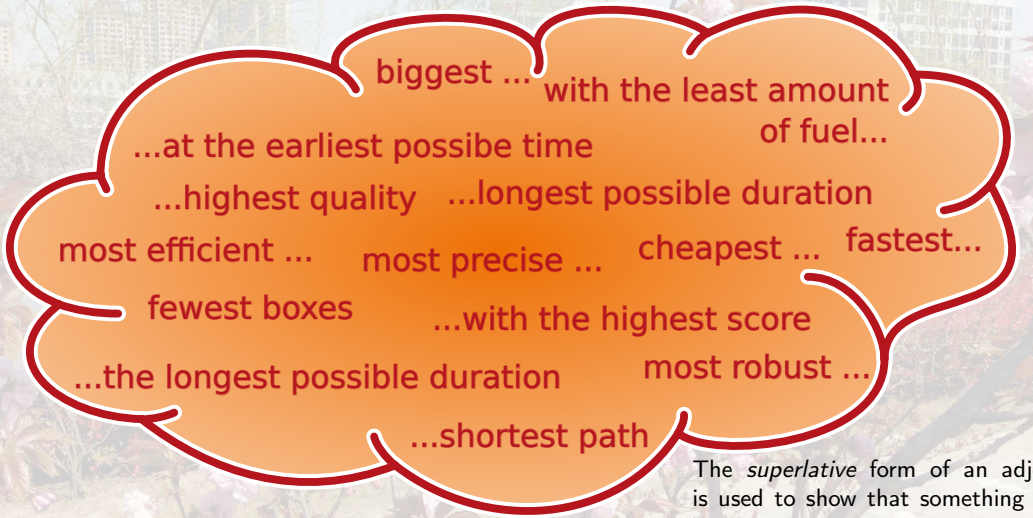
- Are are three ways to approach this topic
  1. Optimization is an extension of school mathematics into a field where equations are no longer enough and exact solutions cannot always be reached.
  2. Optimization is the art of solving hard problems.

What is Optimization?

- Are are three ways to approach this topic
  1. Optimization is an extension of school mathematics into a field where equations are no longer enough and exact solutions cannot always be reached.
  2. Optimization is the art of solving hard problems.
  3. Optimization means searching for superlatives.

Optimization = Search for Superlatives

- Optimization means finding superlatives.



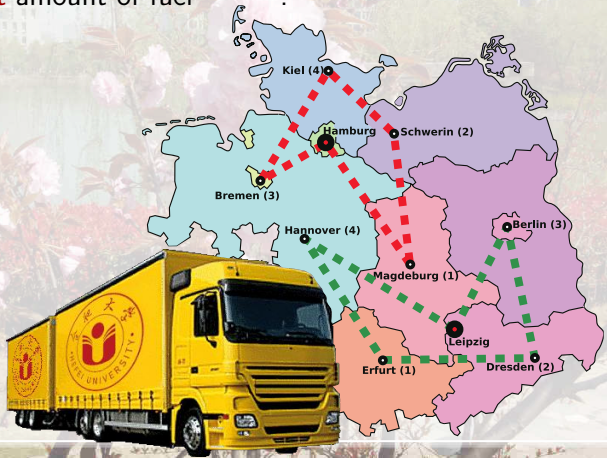
The *superlative* form of an adjective is used to show that something has a quality to the greatest or least degree.

Optimization = Search for Superlatives

- Optimization means finding superlatives.
- Find the shortest path from start to goal[23,39].

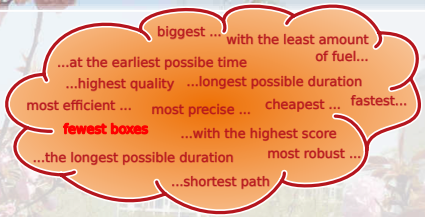
Optimization = Search for Superlatives

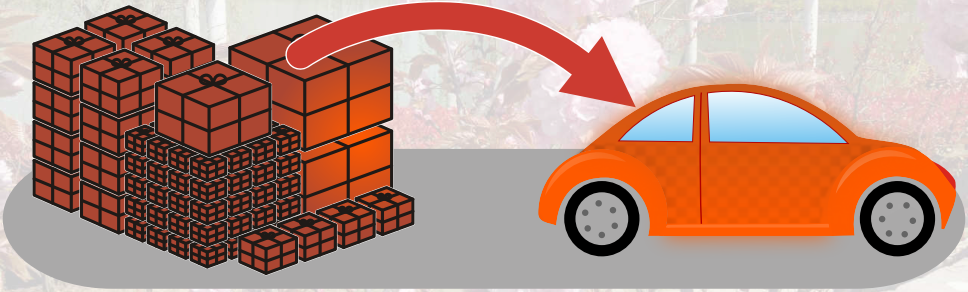
- Optimization means finding superlatives.
- Find the shortest path from start to goal[23,39].
- Pick up and deliver packages from different places to customers using the least amount of fuel[40,49–51].



Optimization = Search for Superlatives

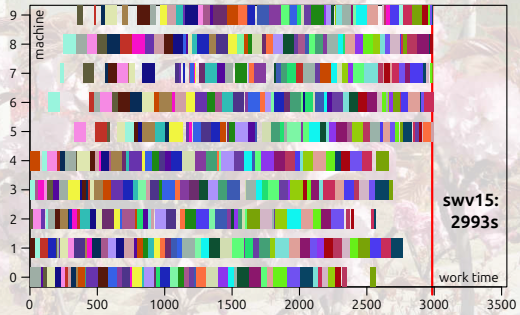
- Optimization means finding superlatives.
- Find the shortest path from start to goal[23,39].
- Pick up and deliver packages from different places to customers using the least amount of fuel[40,49–51].
- Pack a set of things into the fewest boxes[56,57].



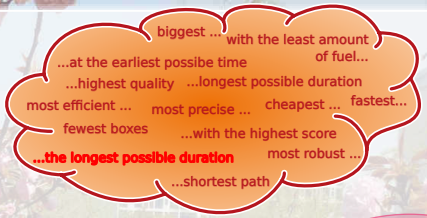


Optimization = Search for Superlatives

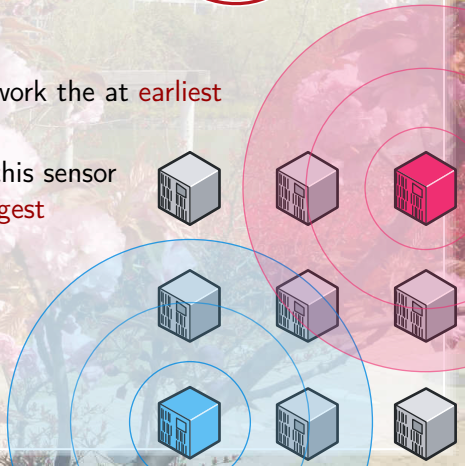
- Optimization means finding superlatives.
- Find the shortest path from start to goal[23,39].
- Pick up and deliver packages from different places to customers using the least amount of fuel[40,49–51].
- Pack a set of things into the fewest boxes[56,57].
- Assign tasks to machines such that we can finish our work the at earliest possible time.



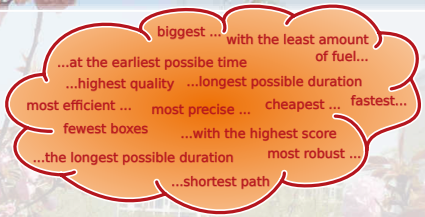
Optimization = Search for Superlatives



- Optimization means finding superlatives.
- Find the shortest path from start to goal[23,39].
- Pick up and deliver packages from different places to customers using the least amount of fuel[40,49–51].
- Pack a set of things into the fewest boxes[56,57].
- Assign tasks to machines such that we can finish our work the at earliest possible time.
- Find a strategy to manage the power of the nodes in this sensor network so that full coverage is guaranteed for the longest possible duration.



Optimization = Search for Superlatives

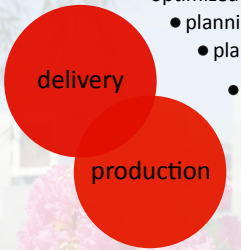


- Optimization means finding superlatives.
- Find the shortest path from start to goal[23,39].
- Pick up and deliver packages from different places to customers using the least amount of fuel[40,49–51].
- Pack a set of things into the fewest boxes[56,57].
- Assign tasks to machines such that we can finish our work the at earliest possible time.
- Find a strategy to manage the power of the nodes in this sensor network so that full coverage is guaranteed for the longest possible duration.
- And so on.

Business

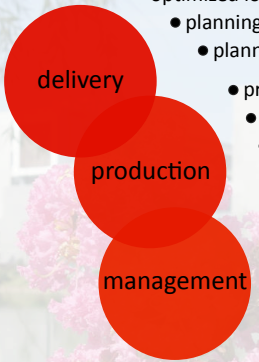
- optimized logistics (business-to-customer)
  - planning and scheduling of maintenance visits
    - planning and scheduling of supply visits

Business



- optimized logistics (business-to-customer)
- planning and scheduling of maintenance visits
- planning and scheduling of supply visits
- production planning and scheduling
- optimized assignment of jobs/orders to machines
- optimization of production processes
- optimization of stock-keeping
- optimization of intra-enterprise logistics
- optimization of supply chains
- optimization of factory layouts and intra-factory logistics

Business

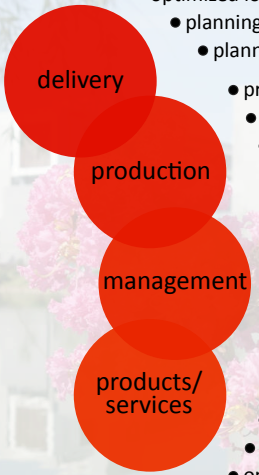


- optimized logistics (business-to-customer)
- planning and scheduling of maintenance visits
- planning and scheduling of supply visits

- production planning and scheduling
- optimized assignment of jobs/orders to machines
- optimization of production processes
- optimization of stock-keeping
- optimization of intra-enterprise logistics
- optimization of supply chains
- optimization of factory layouts and intra-factory logistics

- scheduling of employee work
- optimal assignment of employees to tasks or customers
- optimized locations for new branch offices
  (based current or predicted future customers)

Business



- optimized logistics (business-to-customer)
- planning and scheduling of maintenance visits
- planning and scheduling of supply visits

- production planning and scheduling
- optimized assignment of jobs/orders to machines
- optimization of production processes
- optimization of stock-keeping
- optimization of intra-enterprise logistics
- optimization of supply chains
- optimization of factory layouts and intra-factory logistics

- scheduling of employee work
- optimal assignment of employees to tasks or customers
- optimized locations for new branch offices
  (based current or predicted future customers)

- optimization of product design
- optimization of product feature configuration
- optimization of service offers
- improved tailoring of products/services to customers

Business

- optimized logistics (business-to-customer)
- planning and scheduling of maintenance visits
- planning and scheduling of supply visits

- production planning and scheduling
- optimized assignment of jobs/orders to machines
- optimization of production processes
- optimization of stock-keeping
- optimization of intra-enterprise logistics
- optimization of supply chains
- optimization of factory layouts and intra-factory logistics

- scheduling of employee work
- optimal assignment of employees to tasks or customers
- optimized locations for new branch offices
  (based current or predicted future customers)

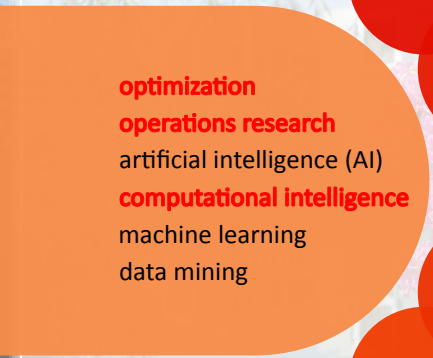
- optimization of product design
- optimization of product feature configuration
- optimization of service offers
- improved tailoring of products/services to customers

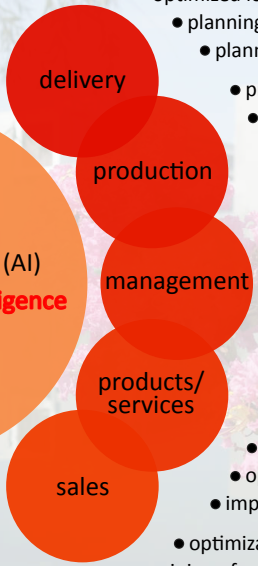
- optimization of pricing and offers
- mining of customer data for targeted offers

Business


optimization
operations research
artificial intelligence (AI)
computational intelligence
machine learning
data mining
delivery
production
management
products/
services
sales
optimized logistics (business-to-customer)
planning and scheduling of maintenance visits
planning and scheduling of supply visits
production planning and scheduling
optimized assignment of jobs/orders to machines
optimization of production processes
optimization of stock-keeping
optimization of intra-enterprise logistics
optimization of supply chains
optimization of factory layouts and intra-factory logistics
scheduling of employee work
optimal assignment of employees to tasks or customers
optimized locations for new branch offices
(based current or predicted future customers)
optimization of product design
optimization of product feature configuration
optimization of service offers
improved tailoring of products/services to customers
optimization of pricing and offers
mining of customer data for targeted offers

Business





- optimized logistics (business-to-customer)
- planning and scheduling of maintenance visits
- planning and scheduling of supply visits

- production planning and scheduling
- optimized assignment of jobs/orders to machines
- optimization of production processes
- optimization of stock-keeping
- optimization of intra-enterprise logistics
- optimization of supply chains
- optimization of factory layouts and intra-factory logistics

- scheduling of employee work
- optimal assignment of employees to tasks or customers
- optimized locations for new branch offices
  (based current or predicted future customers)

- optimization of product design
- optimization of product feature configuration
- optimization of service offers
- improved tailoring of products/services to customers

- optimization of pricing and offers
- mining of customer data for targeted offers

Optimization

- There are incredibly many problems from a very wide area where we can use optimization and Operations Research (OR).

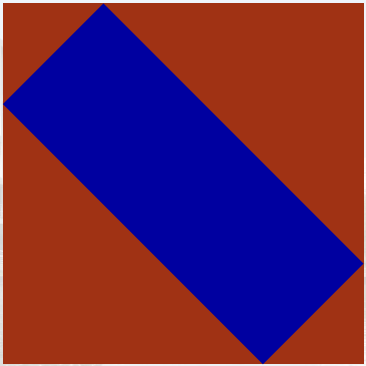
Optimization

- There are incredibly many problems from a very wide area where we can use optimization and Operations Research (OR).
- How is this related to what you already learned?

Problems that we can Solve with Equations

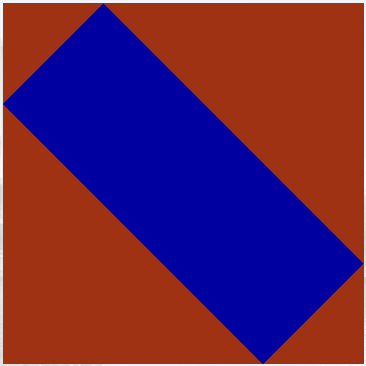
A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.



A Problem that we can Solve with (High School Maths) Equations

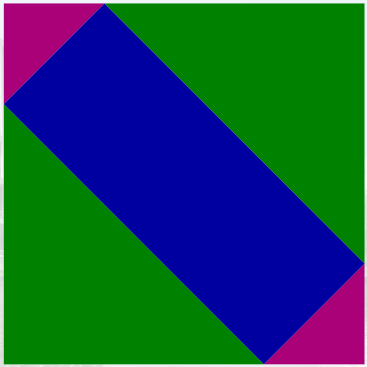
- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?



A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

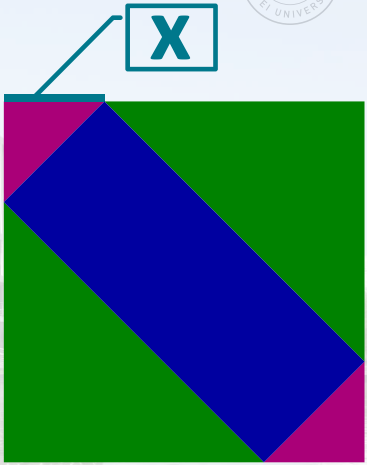


A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

$$A_{\blacksquare}(x) = \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right)$$



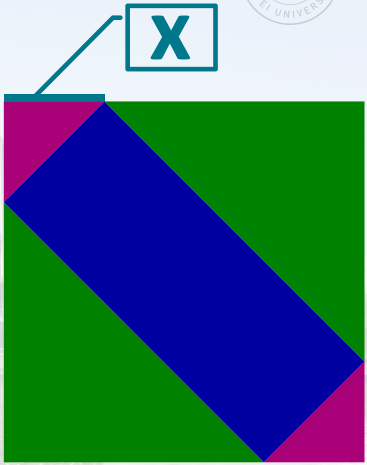
A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

$$A_{\blacksquare}(x) = \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right)$$

$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2$$



A Problem that we can Solve with (High School Maths) Equations

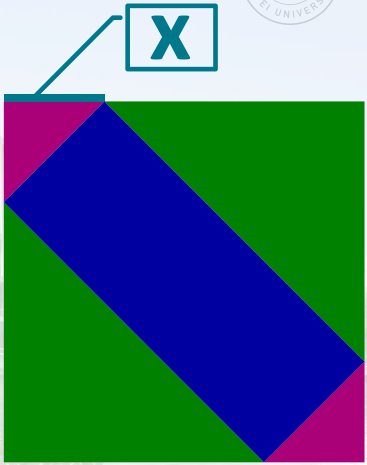
- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

$$A_{\blacksquare}(x) = (13\text{cm}^2) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right)$$

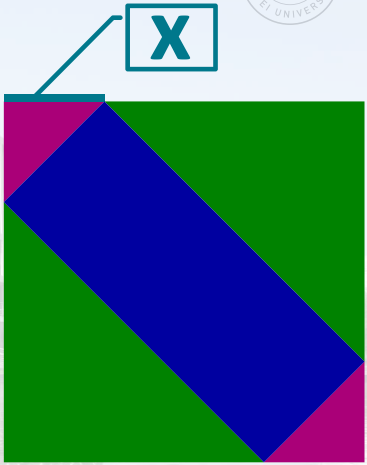
$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2$$

$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - 169\text{cm}^2 + 26\text{cm} * x - x^2$$



A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?



$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

$$A_{\blacksquare}(x) = \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right)$$

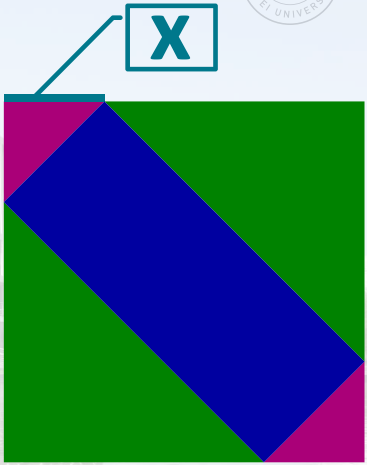
$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2$$

$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - 169\text{cm}^2 + 26\text{cm} * x - x^2$$

$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?



$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

$$A_{\blacksquare}(x) = \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right)$$

$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2$$

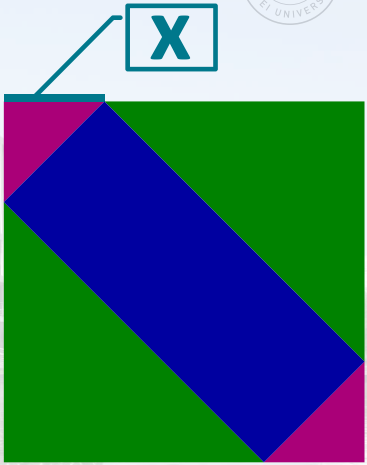
$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - 169\text{cm}^2 + 26\text{cm} * x - x^2$$

$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

$$A'_{\blacksquare}(x) = -4x + 26\text{cm}$$

A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?



$$A_{\blacksquare} = A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle}$$

$$A_{\blacksquare}(x) = \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right)$$

$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2$$

$$A_{\blacksquare}(x) = 169\text{cm}^2 - x^2 - 169\text{cm}^2 + 26\text{cm} * x - x^2$$

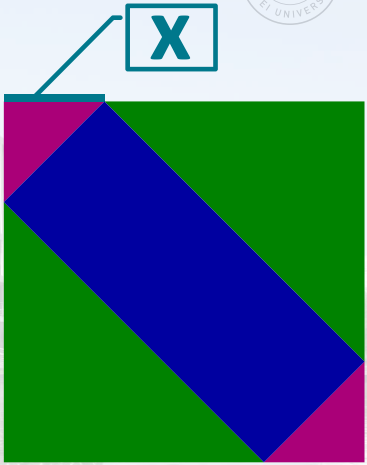
$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

$$A'_{\blacksquare}(x) = -4x + 26\text{cm}$$

$$0 = -4\hat{x} + 26\text{cm}$$

A Problem that we can Solve with (High School Maths) Equations

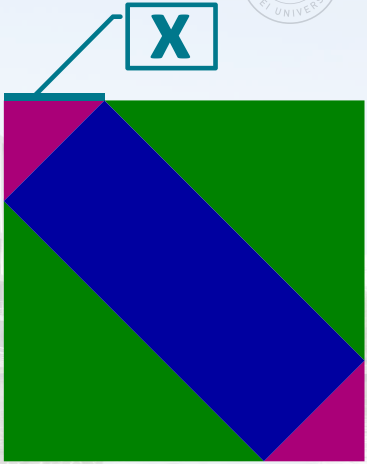
- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?



$$
\begin{aligned}
A_{\blacksquare} &= A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle} \\
A_{\blacksquare}(x) &= \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right) \\
A_{\blacksquare}(x) &= 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2 \\
A_{\blacksquare}(x) &= 169\text{cm}^2 - x^2 - 169\text{cm}^2 + 26\text{cm} * x - x^2 \\
A_{\blacksquare}(x) &= -2x^2 + 26\text{cm} * x \\
A'_{\blacksquare}(x) &= -4x + 26\text{cm} \\
0 &= -4\hat{x} + 26\text{cm} \\
4\hat{x} &= 26\text{cm}
\end{aligned}
$$

A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?



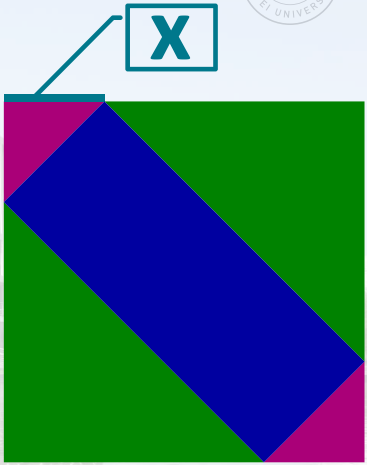
$$
\begin{aligned}
A_{\blacksquare} &= A_{\blacksquare} - 2A_{\blacktriangledown} - 2A_{\blacktriangle} \\
A_{\blacksquare}(x) &= \left(13\text{cm}^2\right) - 2\left(\tfrac{1}{2}x^2\right) - 2\left(\tfrac{1}{2}(13\text{cm} - x)^2\right) \\
A_{\blacksquare}(x) &= 169\text{cm}^2 - x^2 - (13\text{cm} - x)^2 \\
A_{\blacksquare}(x) &= 169\text{cm}^2 - x^2 - 169\text{cm}^2 + 26\text{cm} * x - x^2 \\
A_{\blacksquare}(x) &= -2x^2 + 26\text{cm} * x \\
A'_{\blacksquare}(x) &= -4x + 26\text{cm} \\
0 &= -4\hat{x} + 26\text{cm} \\
4\hat{x} &= 26\text{cm} \\
\hat{x} &= 6.5\text{cm}
\end{aligned}
$$

A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

$$\hat{x} = 6.5\text{cm}$$



A Problem that we can Solve with (High School Maths) Equations

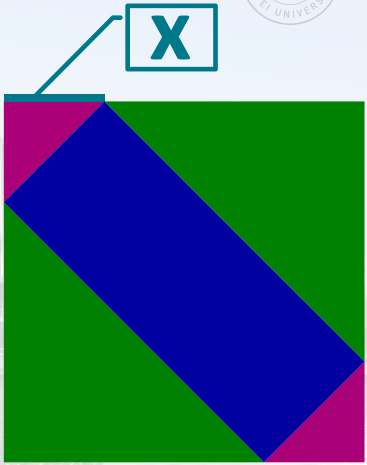
- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

$$\hat{x} = 6.5\text{cm}$$

$$\widehat{A_{\blacksquare}} = A_{\blacksquare}(\hat{x}) = A_{\blacksquare}(6.5\text{cm})$$

$$\widehat{A_{\blacksquare}} = -2(6.5\text{cm})^2 + 26\text{cm} * 6.5\text{cm}$$



A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

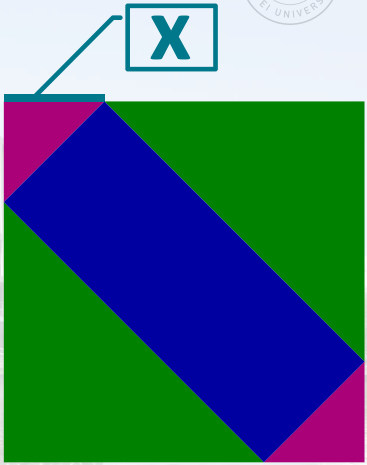
$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

$$\hat{x} = 6.5\text{cm}$$

$$\widehat{A_{\blacksquare}} = A_{\blacksquare}(\hat{x}) = A_{\blacksquare}(6.5\text{cm})$$

$$\widehat{A_{\blacksquare}} = -2(6.5\text{cm})^2 + 26\text{cm} * 6.5\text{cm}$$

$$\widehat{A_{\blacksquare}} = 84.5\text{cm}^2$$



A Problem that we can Solve with (High School Maths) Equations

- A rectangle should be placed inside a square with side length 13cm as sketched.
- What area has the largest such rectangle?

$$A_{\blacksquare}(x) = -2x^2 + 26\text{cm} * x$$

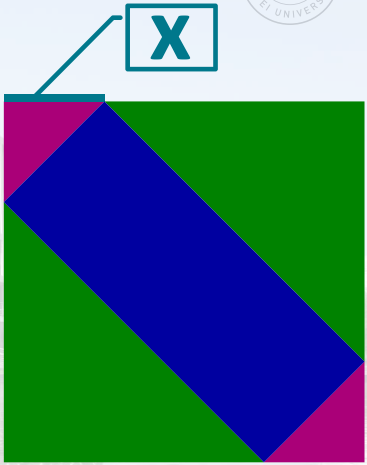
$$\hat{x} = 6.5\text{cm}$$

$$\widehat{A_{\blacksquare}} = A_{\blacksquare}(\hat{x}) = A_{\blacksquare}(6.5\text{cm})$$

$$\widehat{A_{\blacksquare}} = -2(6.5\text{cm})^2 + 26\text{cm} * 6.5\text{cm}$$

$$\widehat{A_{\blacksquare}} = 84.5\text{cm}^2$$

Solved.



Problems that we can solve with Equations

- We can actually solve a lot of problems.

Problems that we can solve with Equations

- We can actually solve a lot of problems.
- We just used an equation.

Problems that we can solve with Equations

- We can actually solve a lot of problems.
- We just used an equation.
- We did computations in multiple steps.

Problems that we can solve with Equations

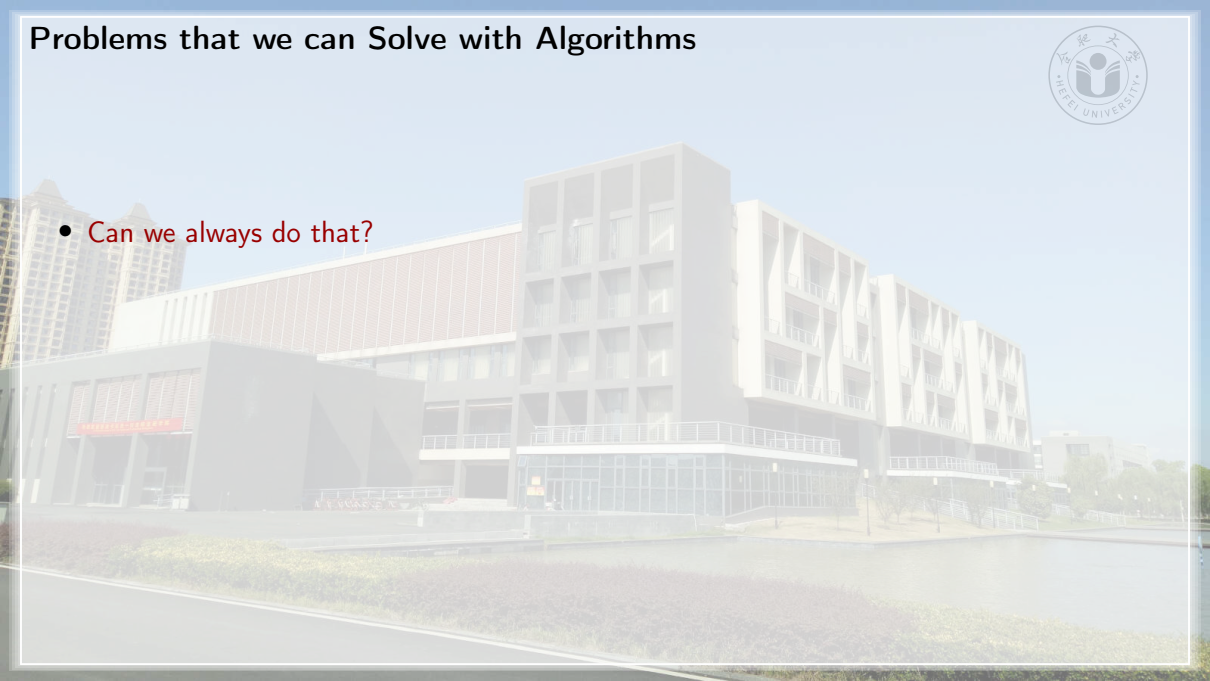
- We can actually solve a lot of problems.
- We just used an equation.
- We did computations in multiple steps.
- Regardless how the previous problem would be parameterized (say, 16cm instead of 13cm), we could perform the exactly same steps.

Problems that we can solve with Equations

- We can actually solve a lot of problems.
- We just used an equation.
- We did computations in multiple steps.
- Regardless how the previous problem would be parameterized (say, 16cm instead of 13cm), we could perform the exactly same steps.
- Can we always do that?

Problems that we can Solve with an Algorithm

Problems that we can Solve with Algorithms



- Can we always do that?

Problems that we can Solve with Algorithms

- Can we always do that?
- Can we always solve problems with a pre-defined number of steps?

Problems that we can Solve with Algorithms

- Can we always do that?
- Can we always solve problems with a pre-defined number of steps?
- No.

Problems that we can Solve with Algorithms

- Can we always do that?
- Can we always solve problems with a pre-defined number of steps?
- No.
- There are problems that we cannot solve with equations, but with algorithms.

Problems that we can Solve with Algorithms

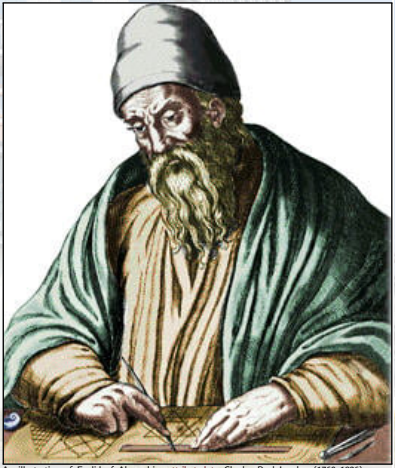
- Can we always do that?
- Can we always solve problems with a pre-defined number of steps?
- No.
- There are problems that we cannot solve with equations, but with algorithms.
- And some problems require us to use algorithms which perform different numbers of steps of different inputs.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?

A Problem that we can Solve with (a High School) Algorithm

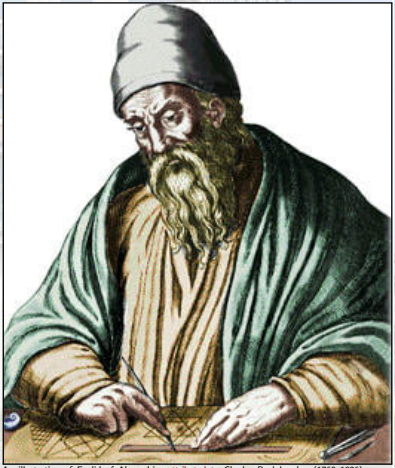
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($Εὐκλείδης$), who lived about 300 *before Common Era (BCE)*.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

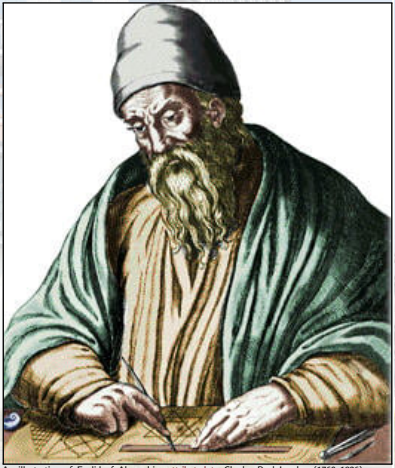
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($E\acute{\upsilon}\kappa\lambda\varepsilon\acute{\iota}\delta\eta\varsigma$), who lived about 300 *before Common Era (BCE)*.
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vikidia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

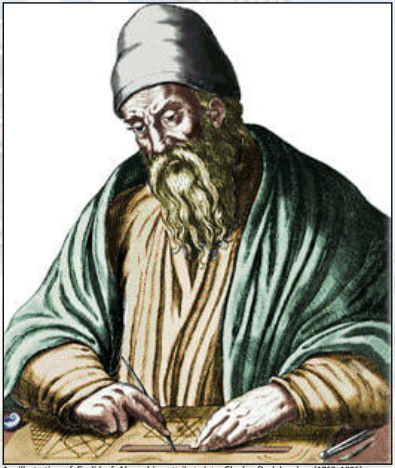
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($E\acute{\upsilon}\kappa\lambda\varepsilon\acute{\iota}\delta\eta\varsigma$), who lived about 300 *before Common Era (BCE)*.
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vikidia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

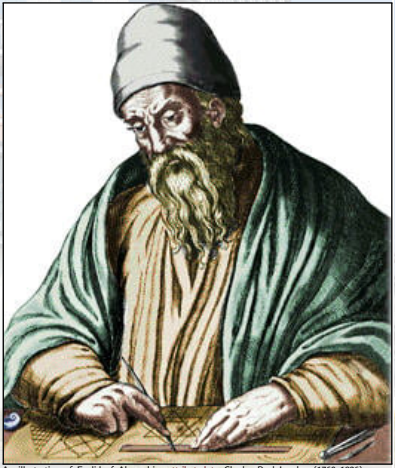
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($E\acute{\upsilon}\kappa\lambda\varepsilon\acute{\iota}\delta\eta\varsigma$), who lived about 300 *before Common Era (BCE)*.
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vikidia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

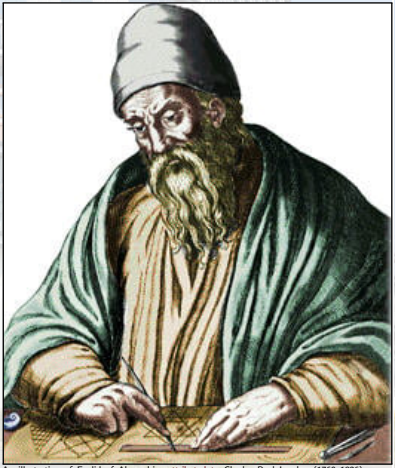
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($E\acute{\upsilon}\kappa\lambda\varepsilon\acute{\iota}\delta\eta\varsigma$), who lived about 300 *before Common Era (BCE)*.
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.
- Without loss of generality, let's assume that $a > b$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

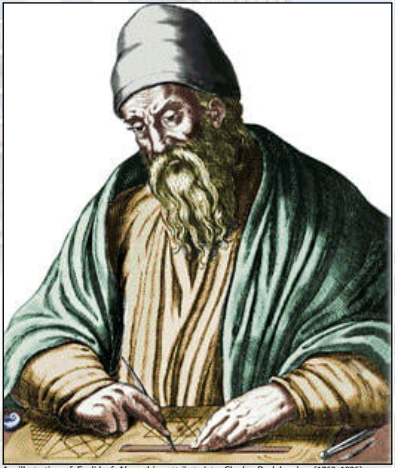
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($E\acute{\upsilon}\kappa\lambda\varepsilon\acute{\iota}\delta\eta\varsigma$), who lived about 300 *before Common Era (BCE)*.
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.
- Without loss of generality, let's assume that $a > b$.
- Then it holds that $c = a - b = (i - j)g$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

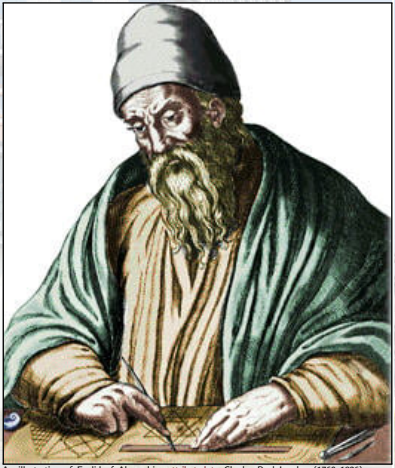
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ can be computed with the Euclidean algorithm[5,19,20] by Euclid of Alexandria ($E\acute{\upsilon}\kappa\lambda\varepsilon\acute{\iota}\delta\eta\varsigma$), who lived about 300 *before Common Era (BCE)*.
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.
- Without loss of generality, let's assume that $a > b$.
- Then it holds that $c = a - b = (i - j)g$.
- Thus, g also divides c without remainder, i.e., $\gcd(a, b) = \gcd(c, b)$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vikidia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

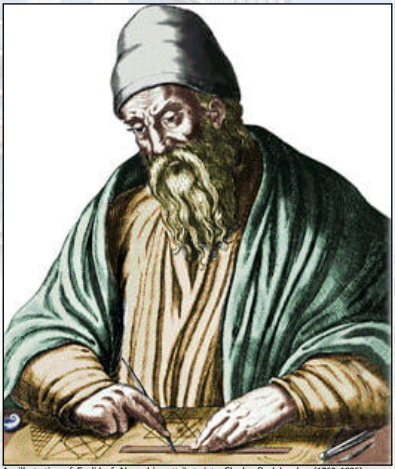
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.
- Without loss of generality, let's assume that $a > b$.
- Then it holds that $c = a - b = (i - j)g$.
- Thus, g also divides c without remainder, i.e., $\gcd(a, b) = \gcd(c, b)$.
- It must also be that $a - b < a$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

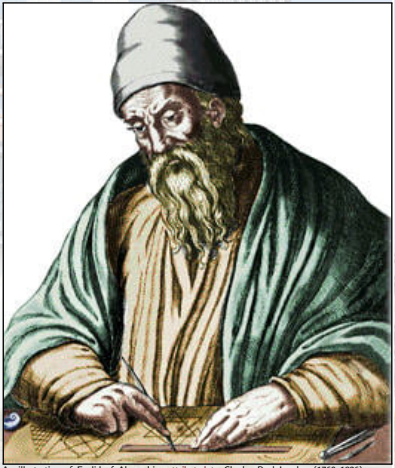
- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- The $\gcd$ of two positive natural numbers $a \in \mathbb{N}_1$ and $b \in \mathbb{N}_1$ is the largest number $g \in \mathbb{N}_1 = \gcd(a, b)$ which divides both a and b without remainder.
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.
- Without loss of generality, let's assume that $a > b$.
- Then it holds that $c = a - b = (i - j)g$.
- Thus, g also divides c without remainder, i.e., $\gcd(a, b) = \gcd(c, b)$.
- It must also be that $a - b < a$.
- We can replace a with $a - b$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vinkdia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- If $a = b$, then obviously $\gcd(a, b) = a = b$.
- Otherwise, we know that $a = ig$ for some $i \in \mathbb{N}_1$ and that $b = jg$ for some $j \in \mathbb{N}_1$.
- Without loss of generality, let's assume that $a > b$.
- Then it holds that $c = a - b = (i - j)g$.
- Thus, g also divides c without remainder, i.e., $\gcd(a, b) = \gcd(c, b)$.
- It must also be that $a - b < a$.
- We can replace a with $a - b$.
- We can repeatedly subtract the smaller from the larger number until we "converge."

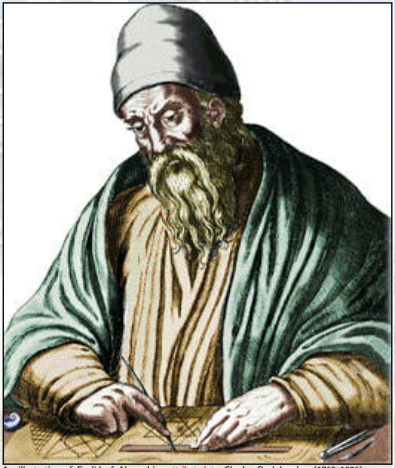


An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vinkdia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$

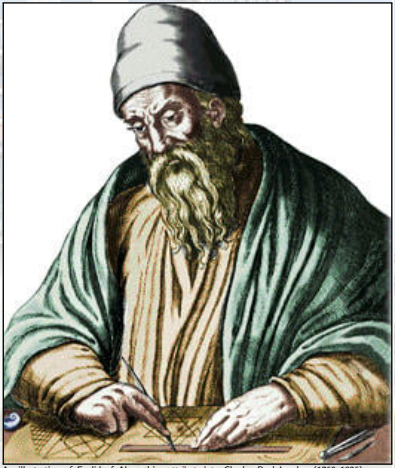


An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084

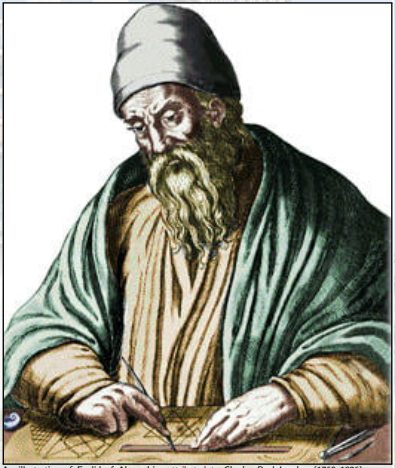


An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084
164 1647	93 8084	70 3563

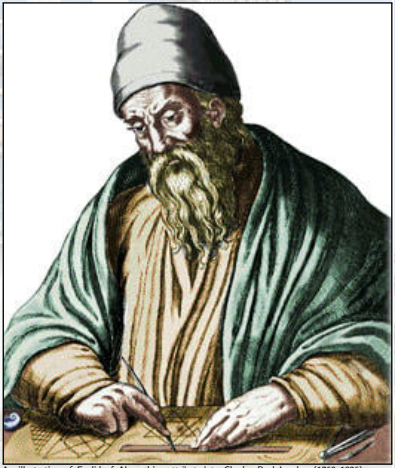


An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084
164 1647	93 8084	70 3563
93 8084	70 3563	23 4521

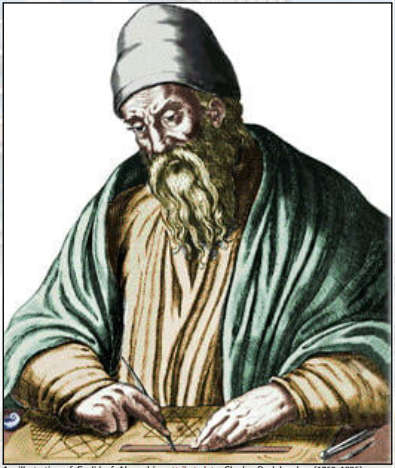


An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084
164 1647	93 8084	70 3563
93 8084	70 3563	23 4521
70 3563	23 4521	46 9042

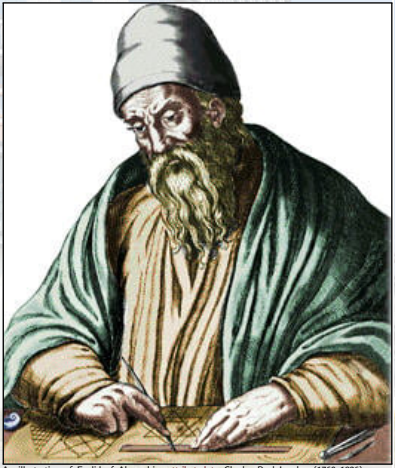


An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084
164 1647	93 8084	70 3563
93 8084	70 3563	23 4521
70 3563	23 4521	46 9042
46 9042	23 4521	23 4521



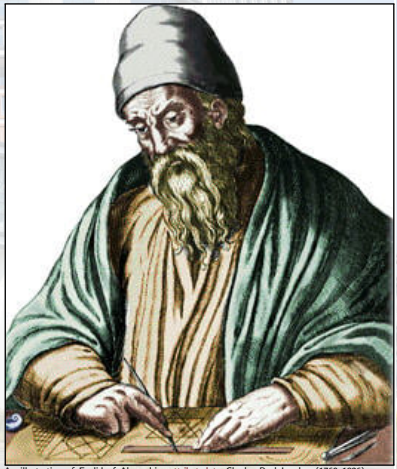
An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084
164 1647	93 8084	70 3563
93 8084	70 3563	23 4521
70 3563	23 4521	46 9042
46 9042	23 4521	23 4521

- $\gcd(257\,9731, 164\,1647) = 23\,4521$.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

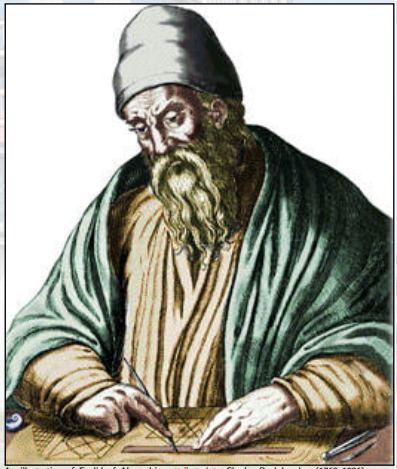
A Problem that we can Solve with (a High School) Algorithm

- What is the greatest common divisor ($\gcd$) of $a = 257\,9731$ and $b = 164\,1647$?
- We can repeatedly subtract the smaller from the larger number until we "converge."

bigger number: a	**smaller number:** b	$a - b$
257 9731	164 1647	93 8084
164 1647	93 8084	70 3563
93 8084	70 3563	23 4521
70 3563	23 4521	46 9042
46 9042	23 4521	23 4521

- $\gcd(257\,9731, 164\,1647) = 23\,4521$.

Solved.



An illustration of Euclid of Alexandria, attributed to Charles Paul Landon (1760–1826). Source: Vividia, where it is listed as *domaine public*.

Euclidean Algorithm

- The number of steps that the algorithm needs depends on the input.

Euclidean Algorithm

- The number of steps that the algorithm needs depends on the input.
- The Euclidean Algorithm can be implemented more efficiently using "division remainders" instead of subtractions.

Euclidean Algorithm

- The number of steps that the algorithm needs depends on the input.
- The Euclidean Algorithm can be implemented more efficiently using "division remainders" instead of subtractions.
- It can be made even more efficient using a binary variant ... that already existed in China in the first century Common Era (CE)[5], published in the famous *Jiu Zhang Suanshu* (九章算术)[14,15,27,38,42].

Euclidean Algorithm

- The number of steps that the algorithm needs depends on the input.
- The Euclidean Algorithm can be implemented more efficiently using "division remainders" instead of subtractions.
- It can be made even more efficient using a binary variant ... that already existed in China in the first century Common Era (CE)[5], published in the famous *Jiu Zhang Suanshu* (九章算术)[14,15,27,38,42].
- What does efficient even mean?

Euclidean Algorithm

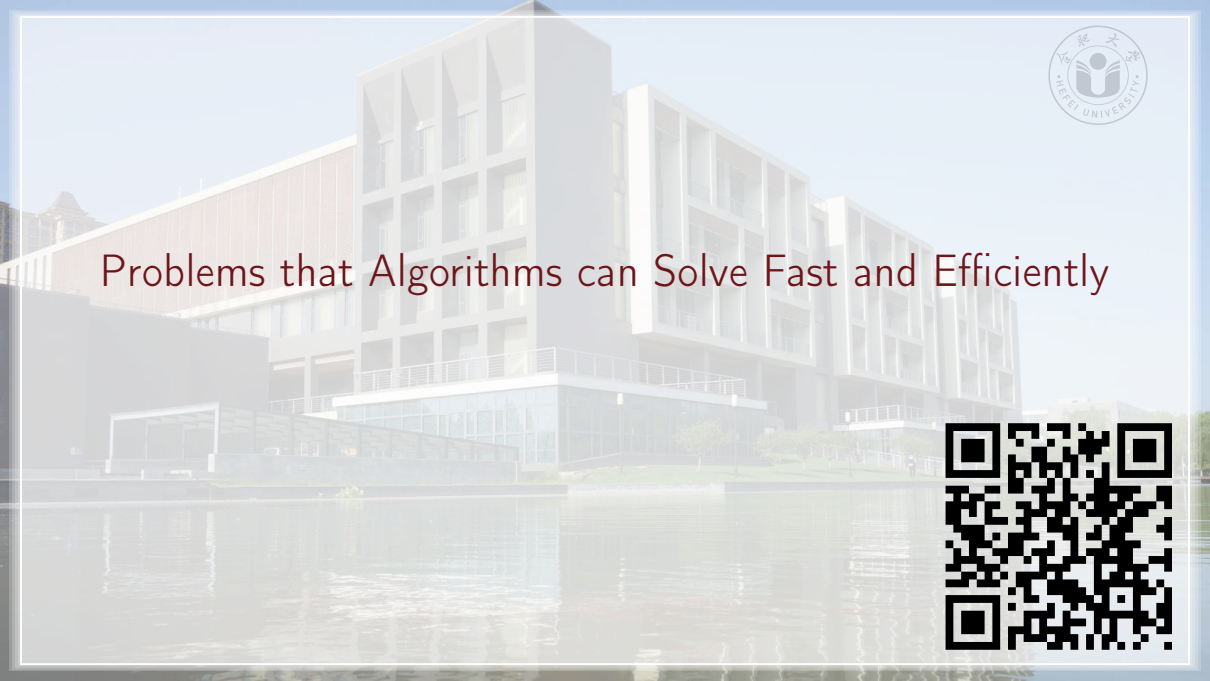
- The number of steps that the algorithm needs depends on the input.
- The Euclidean Algorithm can be implemented more efficiently using "division remainders" instead of subtractions.
- It can be made even more efficient using a binary variant ... that already existed in China in the first century Common Era (CE)[5], published in the famous *Jiu Zhang Suanshu* (九章算术)[14,15,27,38,42].
- What does efficient even mean?
- efficient = fast

Euclidean Algorithm

- The number of steps that the algorithm needs depends on the input.
- The Euclidean Algorithm can be implemented more efficiently using "division remainders" instead of subtractions.
- It can be made even more efficient using a binary variant ... that already existed in China in the first century Common Era (CE)[5], published in the famous *Jiu Zhang Suanshu* (九章算术)[14,15,27,38,42].
- What does efficient even mean?
- efficient = fast and does not need much memory

Euclidean Algorithm

- The number of steps that the algorithm needs depends on the input.
- The Euclidean Algorithm can be implemented more efficiently using "division remainders" instead of subtractions.
- It can be made even more efficient using a binary variant ... that already existed in China in the first century Common Era (CE)[5], published in the famous *Jiu Zhang Suanshu* (九章算术)[14,15,27,38,42].
- What does efficient even mean?
- efficient = fast and does not need much memory
- (Side note: The binary Euclidean algorithm can be computed in $c * (\log a + \log b)$ steps where $c > 0$ is some constance. It needs two memory cells[4,5].)



Problems that Algorithms can Solve Fast and Efficiently

Find the Shortest Path from Start to Goal

Find the Shortest Path from Start to Goal

- I am at the starting point 合肥大学南二区北大门.

Find the Shortest Path from Start to Goal

- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.

Find the Shortest Path from Start to Goal

- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- How do I get there the fastest?

Find the Shortest Path from Start to Goal

- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- How do I get there the fastest?
- We know the campus map.

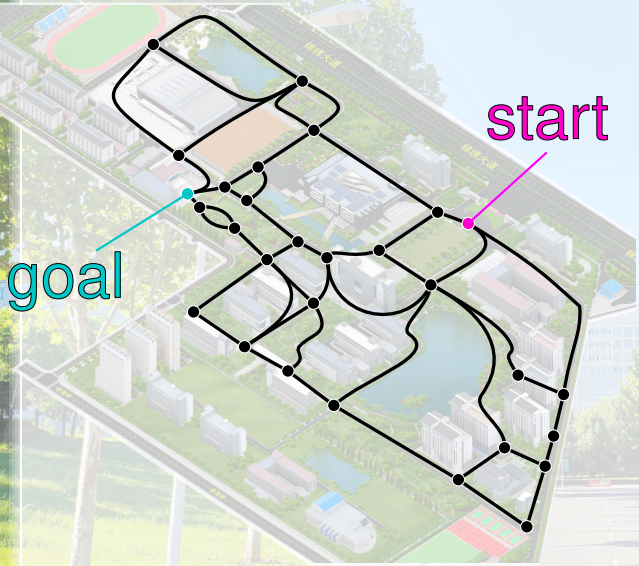
Find the Shortest Path from Start to Goal

- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- How do I get there the fastest?
- We know the campus map.
- We want to compute the shortest path (before actually walking it).

Find the Shortest Path from Start to Goal

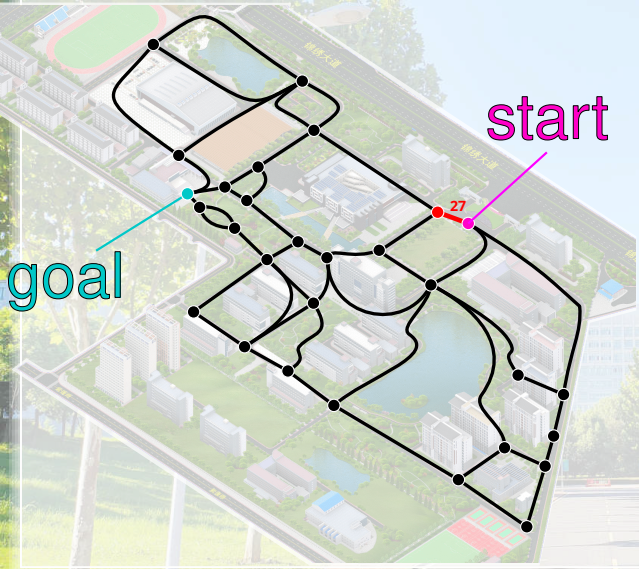
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- How do I get there the fastest?
- We know the campus map.
- We want to compute the shortest path (before actually walking it).
- We know all the intersections where I could make turns.

Find the Shortest Path from Start to Goal



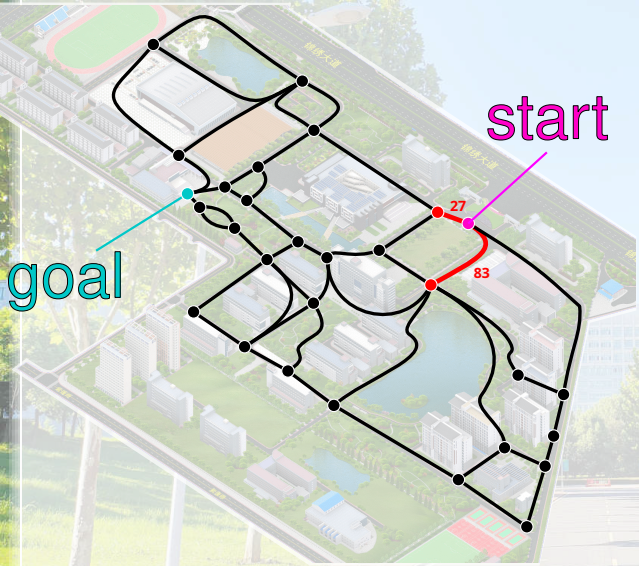
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- How do I get there the fastest?
- We know the campus map.
- We want to compute the shortest path (before actually walking it).
- We know all the intersections where I could make turns.

Find the Shortest Path from Start to Goal



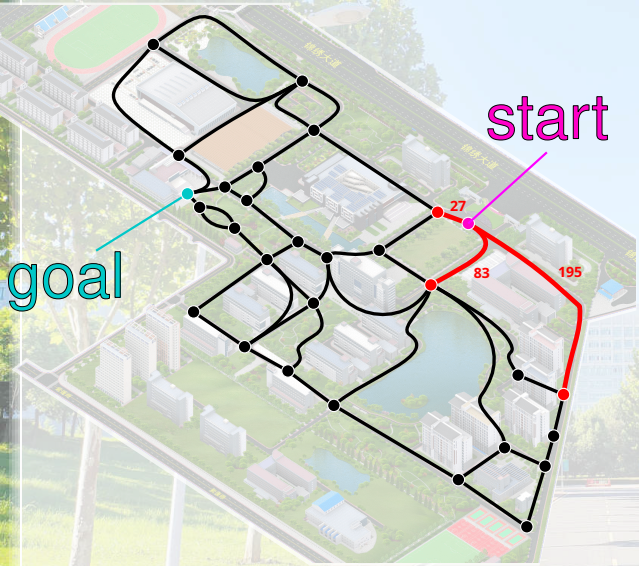
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- For example, I could walk for 27s to this intersection.

Find the Shortest Path from Start to Goal



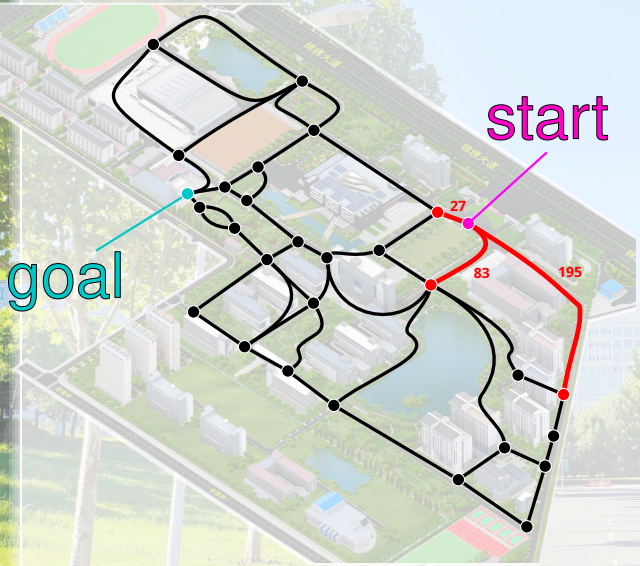
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- For example, I could walk for $27s$ to this intersection.
- Or for $83s$ to that one.

Find the Shortest Path from Start to Goal



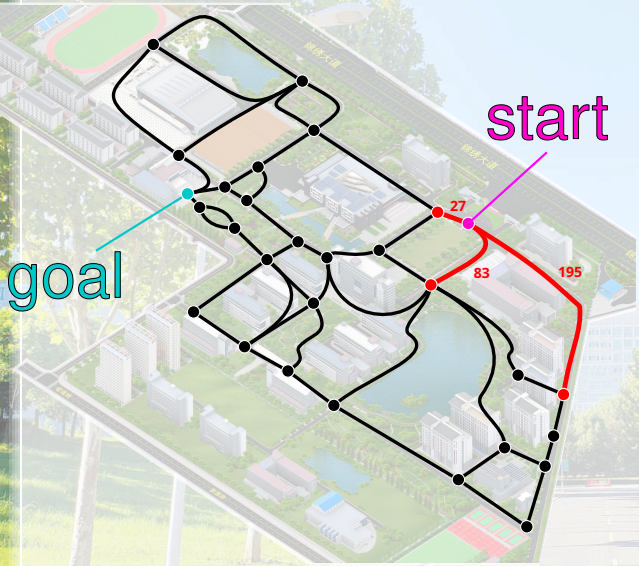
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- For example, I could walk for $27s$ to this intersection.
- Or for $83s$ to that one.
- Or for $195s$ to that one.

Find the Shortest Path from Start to Goal



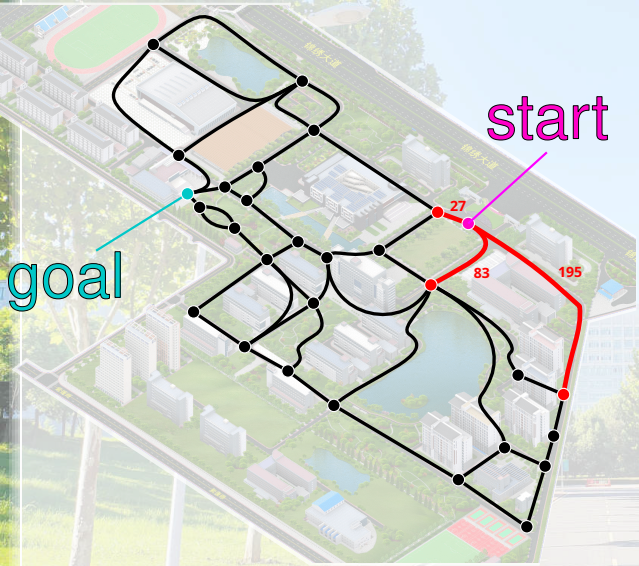
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- For example, I could walk for $27s$ to this intersection.
- Or for $83s$ to that one.
- Or for $195s$ to that one.
- Which one should we pick?

Find the Shortest Path from Start to Goal



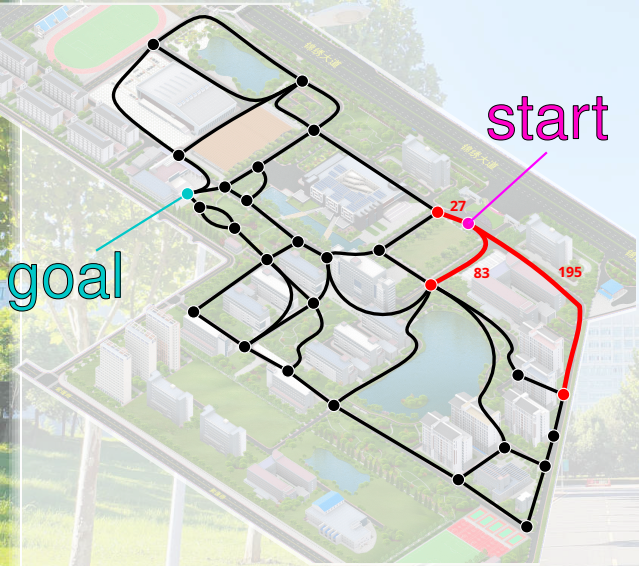
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s$, (b) $83s$, (c) $195s$.

Find the Shortest Path from Start to Goal



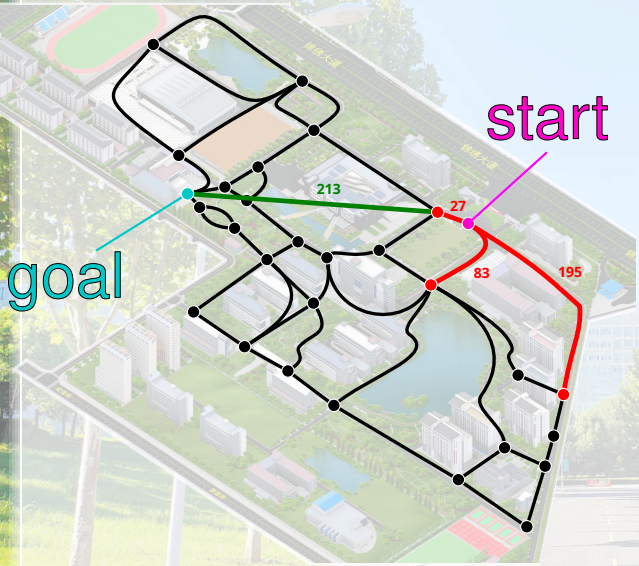
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s$, (b) $83s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.

Find the Shortest Path from Start to Goal



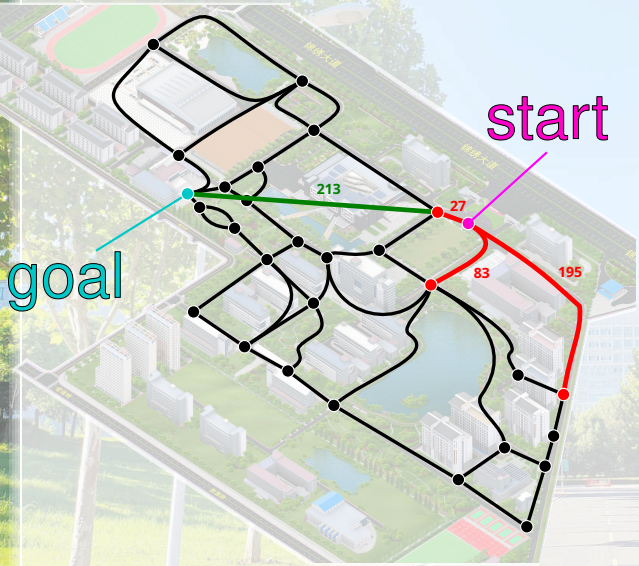
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s$, (b) $83s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



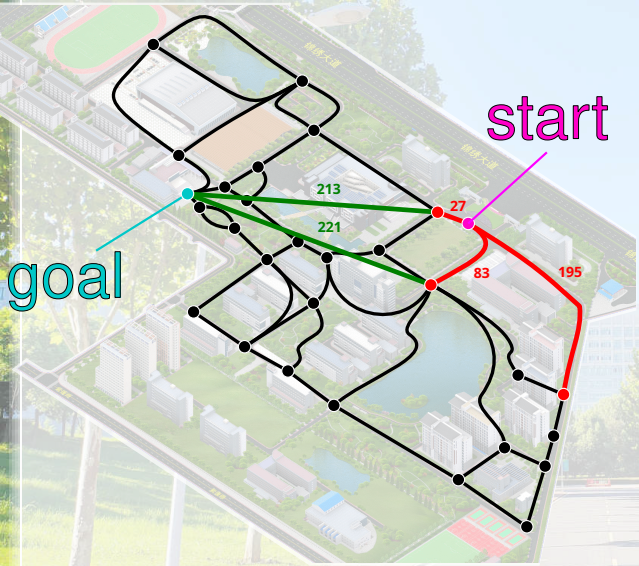
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s$, (b) $83s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



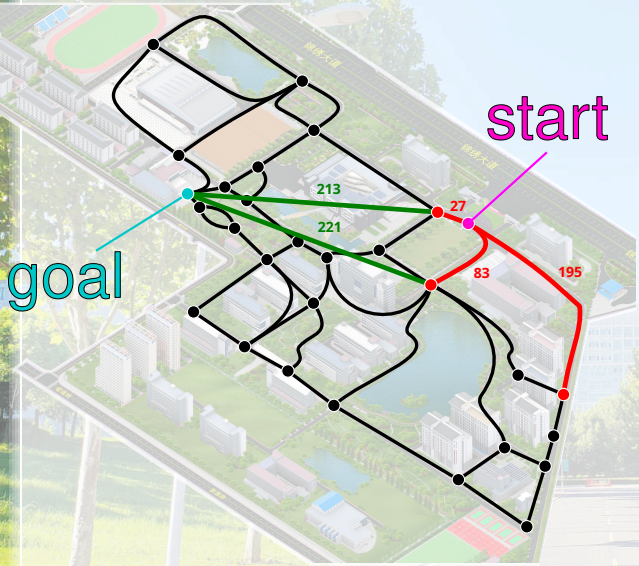
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s + 213s$, (b) $83s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



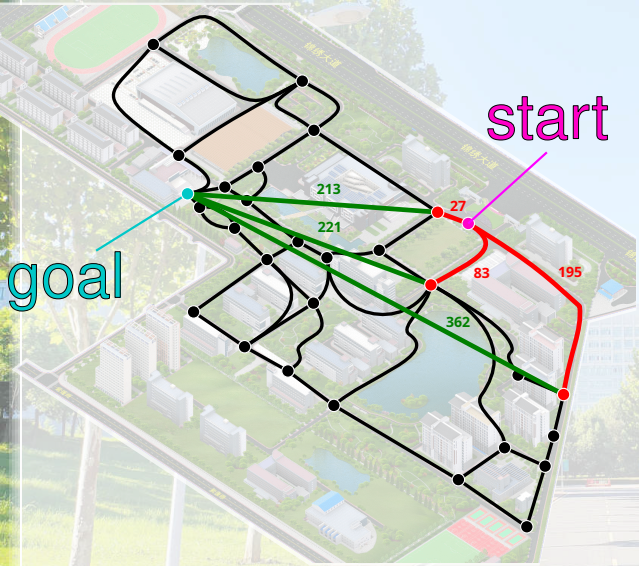
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s + 213s$, (b) $83s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



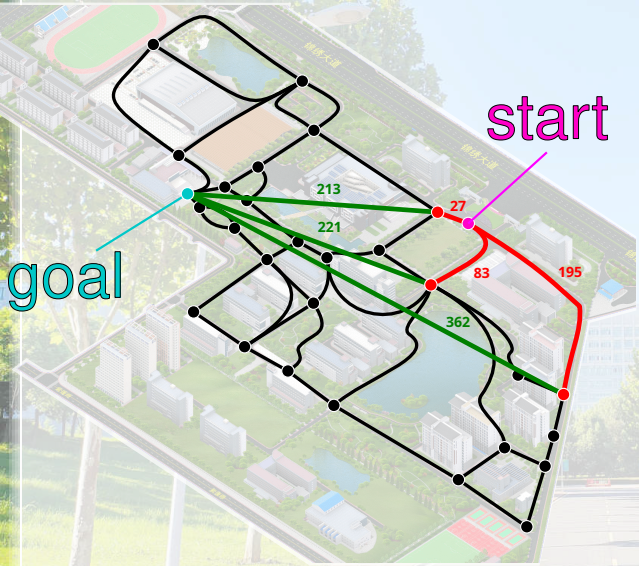
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s + 213s$, (b) $83s + 221s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



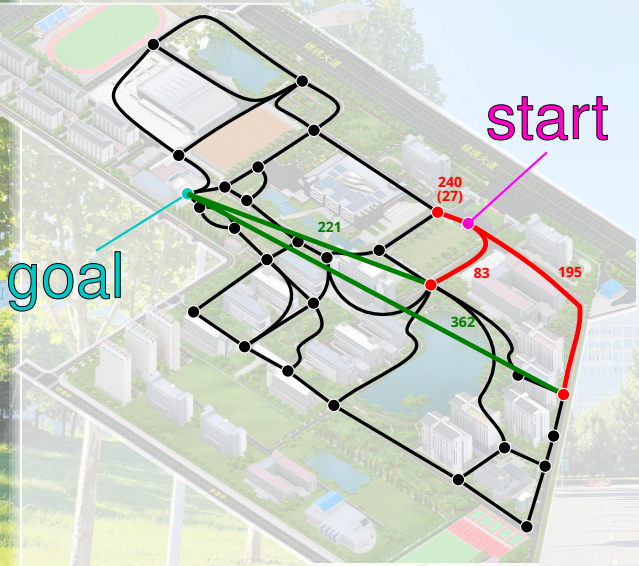
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s + 213s$, (b) $83s + 221s$, (c) $195s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



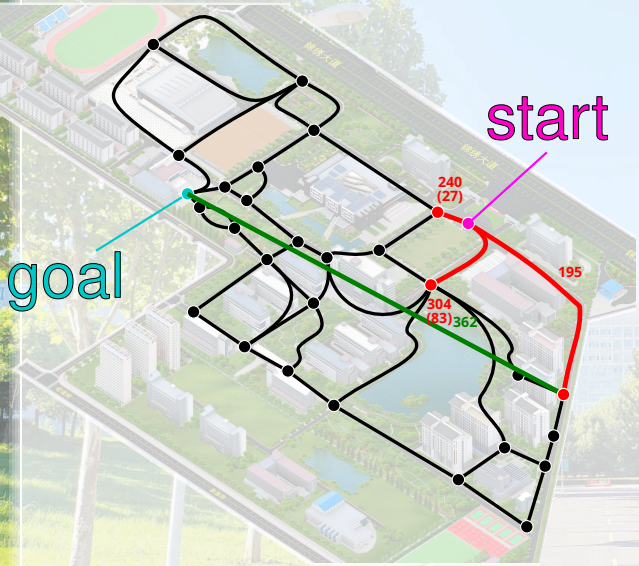
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices: (a) $27s + 213s$, (b) $83s + 221s$, (c) $195s + 362s$.
- For each intersection, we can compute the airline distance (as the crow flies) to the 食堂.
- The actual walking distance can never be shorter than that.

Find the Shortest Path from Start to Goal



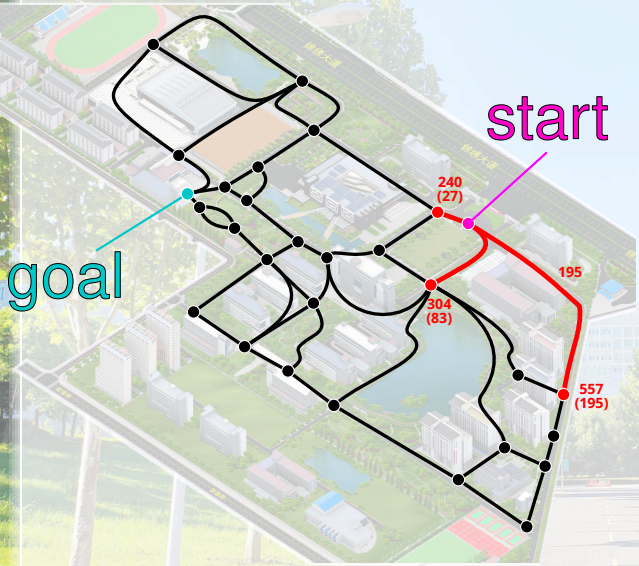
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices:
  (a) $240s = 27s + 213s$, (b) $83s + 221s$,
  (c) $195s + 362s$.

Find the Shortest Path from Start to Goal



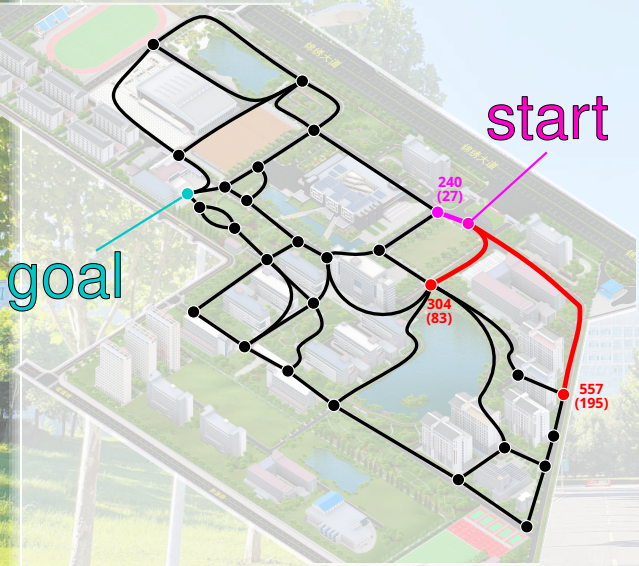
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices:
  (a) $240s = 27s + 213s$,
  (b) $304s = 83s + 221s$, (c) $195s + 362s$.

Find the Shortest Path from Start to Goal



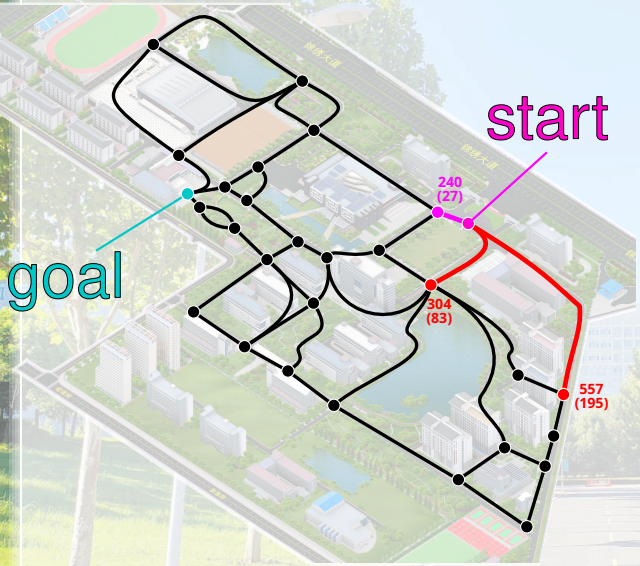
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices:
  (a) $240s = 27s + 213s$,
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.

Find the Shortest Path from Start to Goal



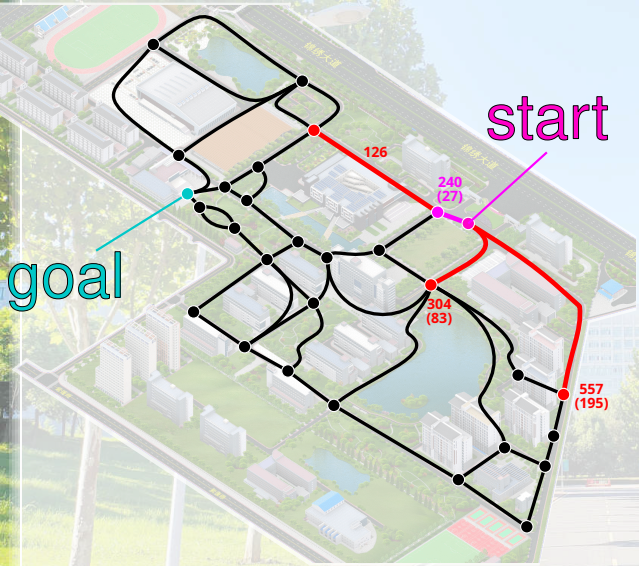
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 choices:
  (a) $240s = 27s + 213s$,
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.
- The most interesting candidate for the first step is clearly (a).

Find the Shortest Path from Start to Goal



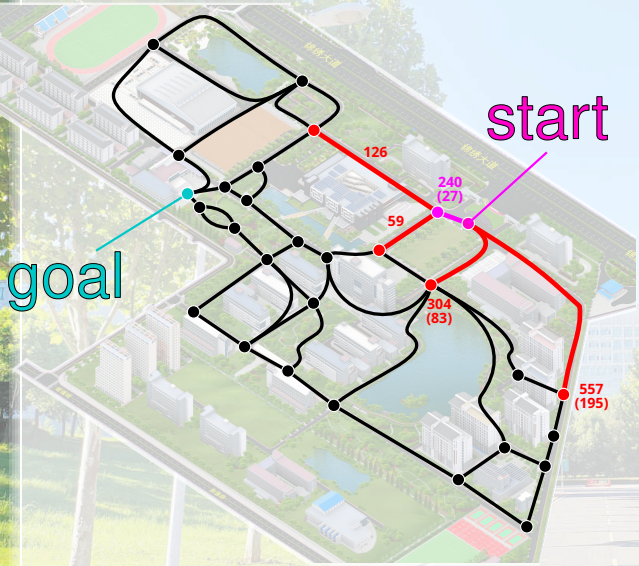
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- We have 2 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.

Find the Shortest Path from Start to Goal



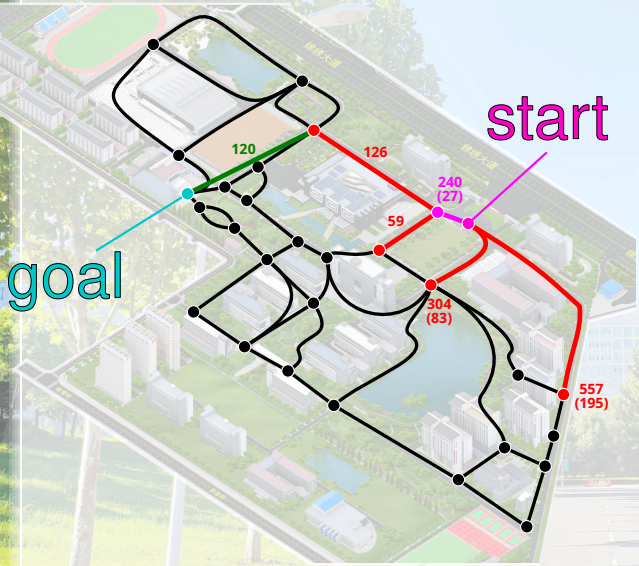
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- We have 2 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.
- From (a), we can go to (d) by walking for $126s$.

Find the Shortest Path from Start to Goal



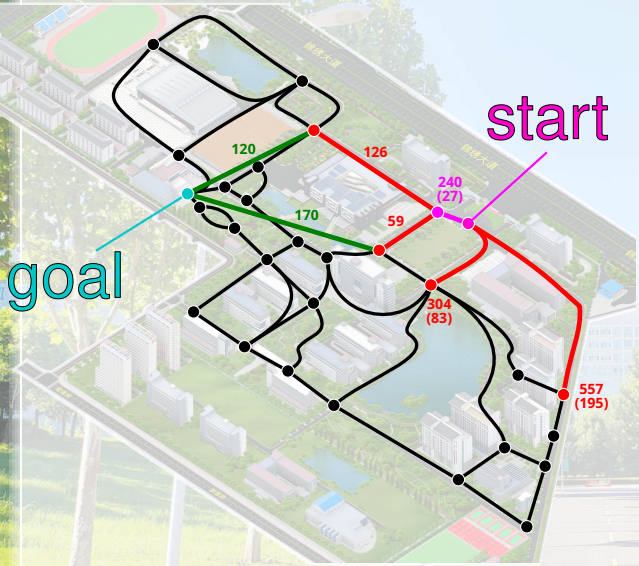
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- We have 2 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.
- From (a), we can go to (d) by walking for $126s$ or to (e) by walking for $59s$.

Find the Shortest Path from Start to Goal



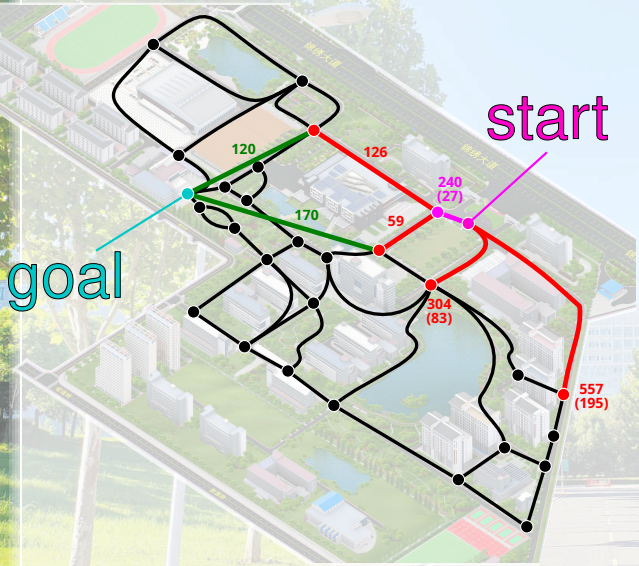
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- We have 2 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.
- From (a), we can go to (d) by walking for $126s$ or to (e) by walking for $59s$.
- For both we compute the distance as the crow flies.

Find the Shortest Path from Start to Goal



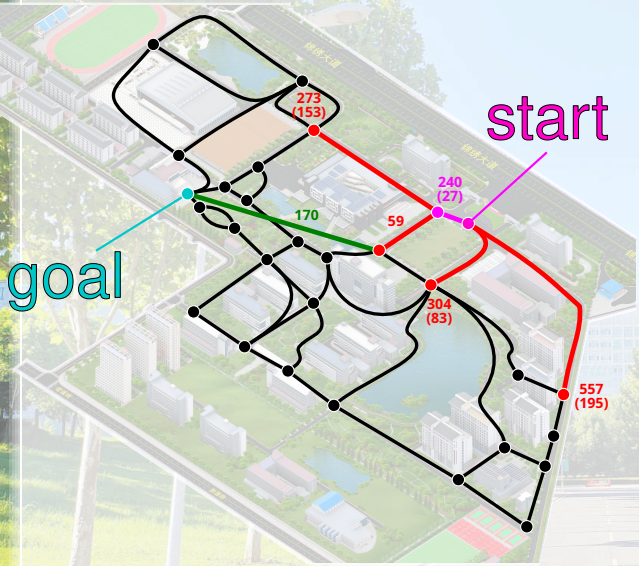
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- From (a), we can go to (d) by walking for $126s$ or to (e) by walking for $59s$.
- For both we compute the distance as the crow flies.

Find the Shortest Path from Start to Goal



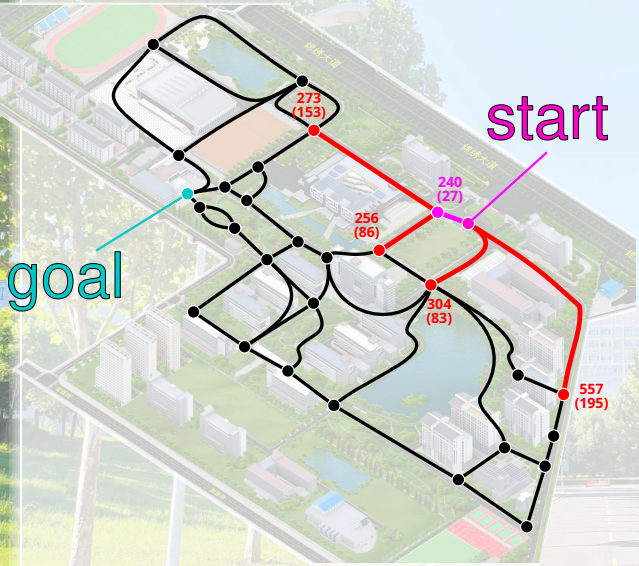
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- From (a), we can go to (d) by walking for $126s$ or to (e) by walking for $59s$.
- For both we compute the distance as the crow flies.
- We had 2 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$.

Find the Shortest Path from Start to Goal



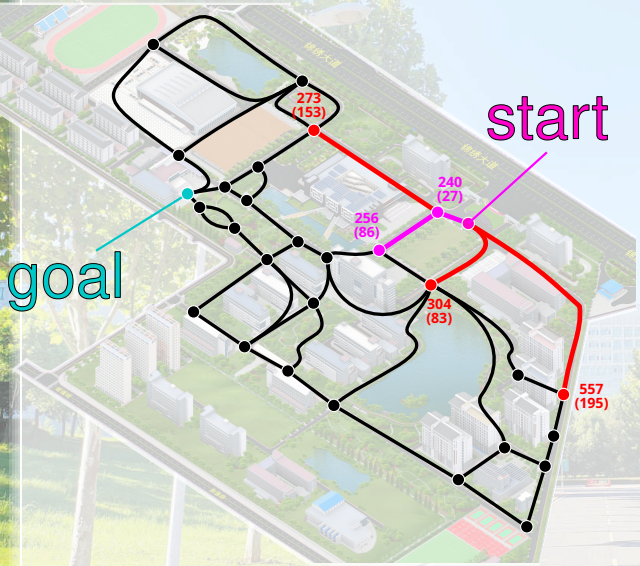
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- From (a), we can go to (d) by walking for $126s$ or to (e) by walking for $59s$.
- For both we compute the distance as the crow flies.
- We now have 3 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.

Find the Shortest Path from Start to Goal



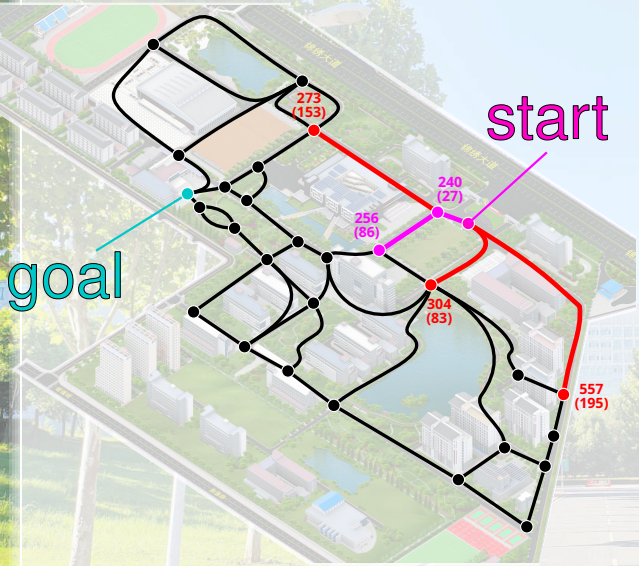
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- The most interesting candidate for the first step is clearly (a).
- From (a), we can go to (d) by walking for $126s$ or to (e) by walking for $59s$.
- For both we compute the distance as the crow flies.
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (e) $256s = 86s + 170s$.

Find the Shortest Path from Start to Goal



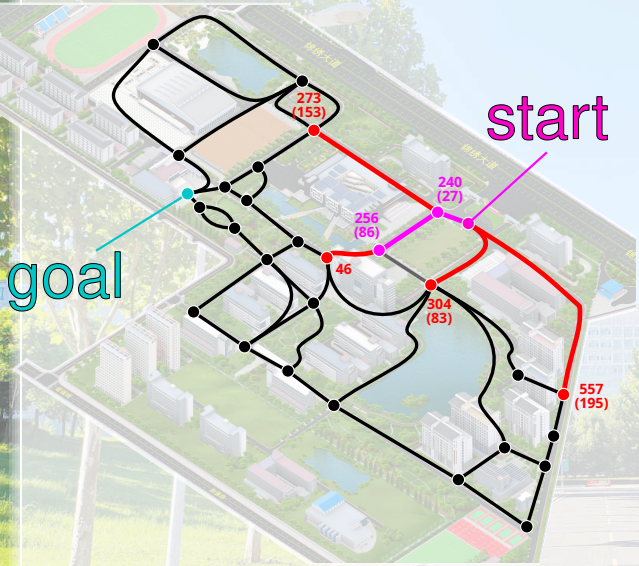
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (e) $256s = 86s + 170s$.
- (e) it is. . .

Find the Shortest Path from Start to Goal



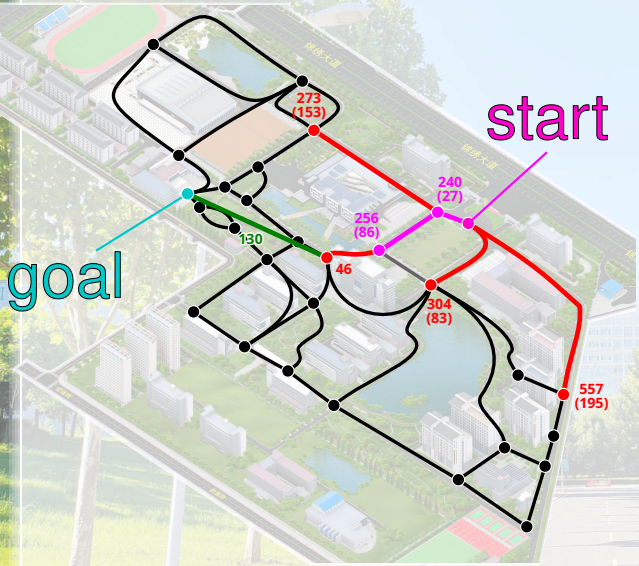
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (e) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.

Find the Shortest Path from Start to Goal



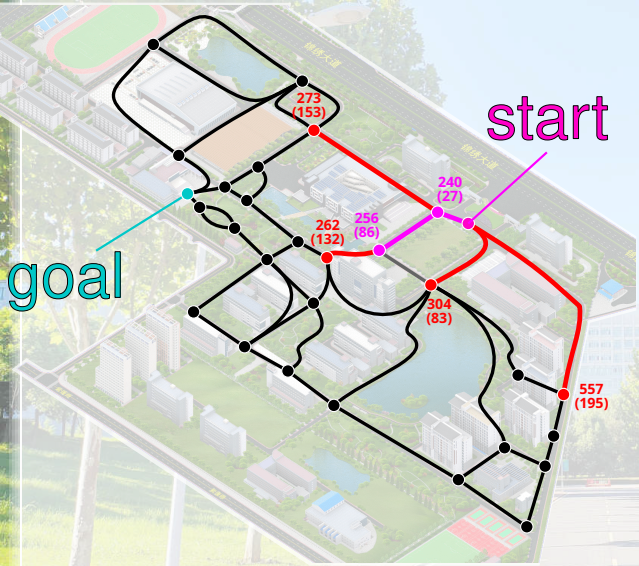
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From this point (e), we only have one reasonable choice where to go next.

Find the Shortest Path from Start to Goal



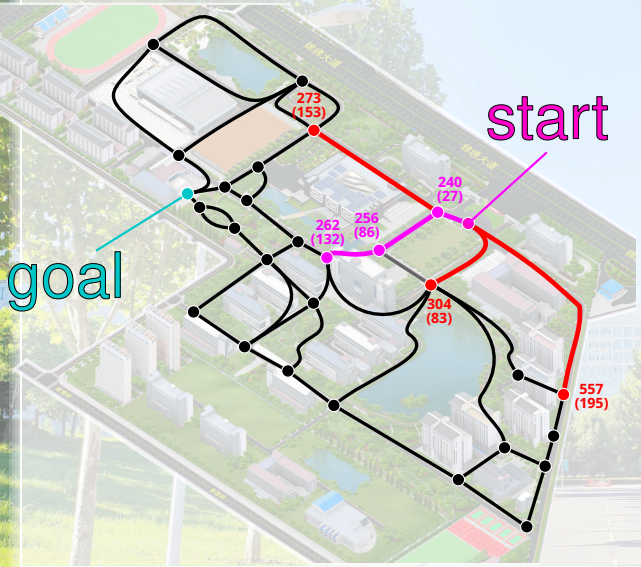
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From this point (e), we only have one reasonable choice where to go next.
- And we again compute the airline distance to the goal.

Find the Shortest Path from Start to Goal



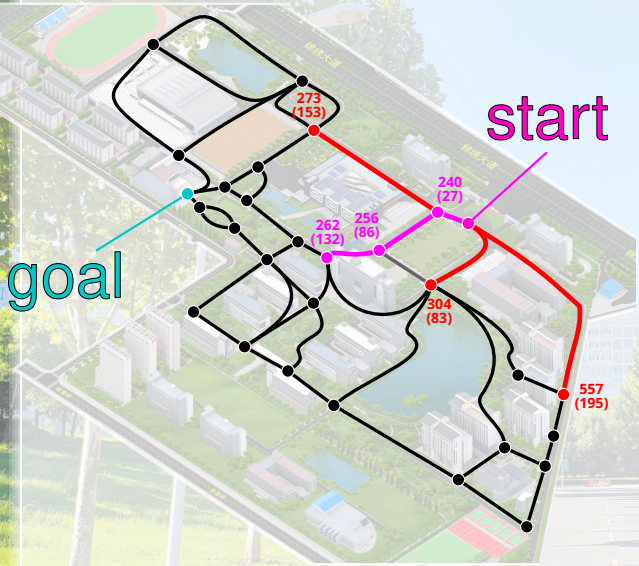
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- From this point (e), we only have one reasonable choice where to go next.
- And we again compute the airline distance to the goal.
- We now have 4 choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (f) $262s = 132s + 130s$.

Find the Shortest Path from Start to Goal



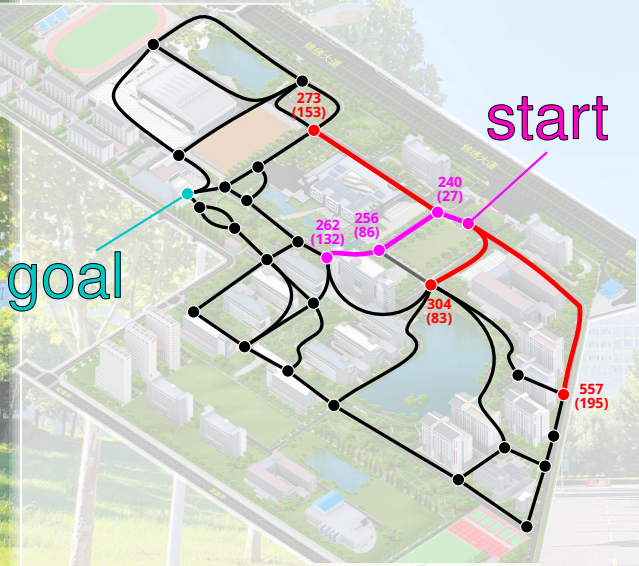
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- We now have 4 choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (f) $262s = 132s + 130s$.
- (f) it is. . .

Find the Shortest Path from Start to Goal



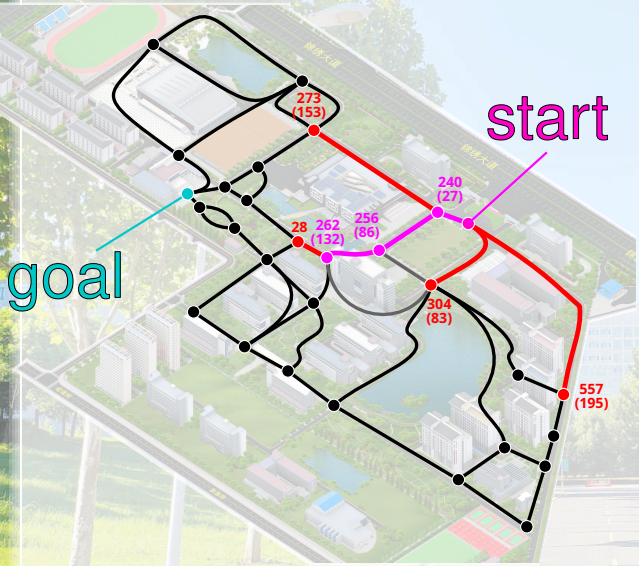
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.

Find the Shortest Path from Start to Goal



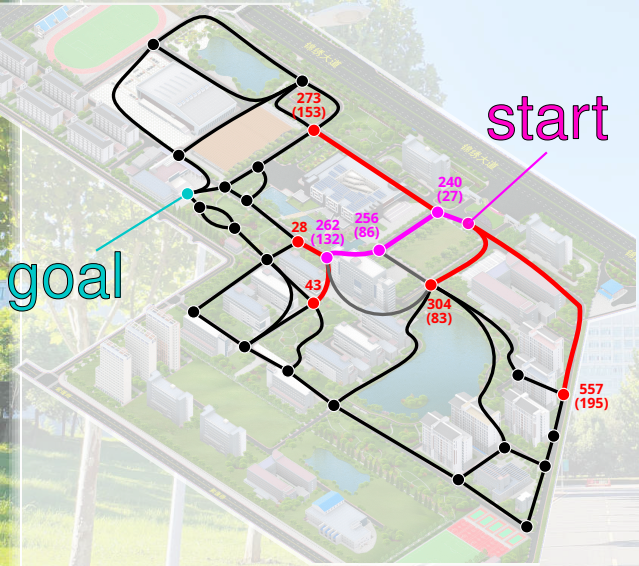
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From (f), we have two possible choices to continue.

Find the Shortest Path from Start to Goal



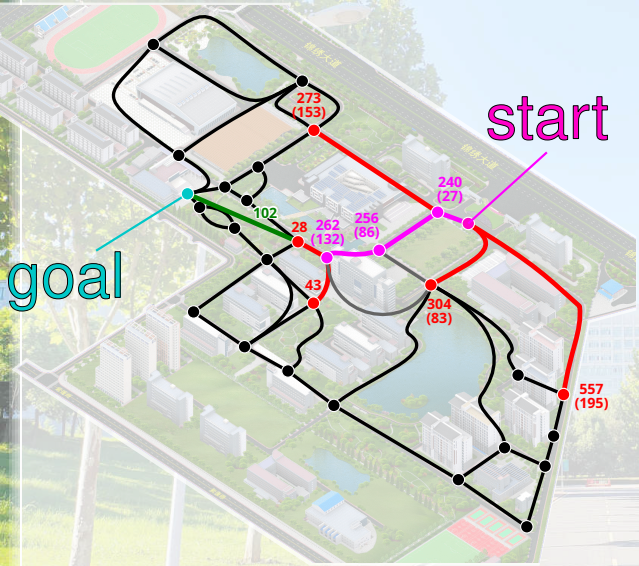
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From (f), we have two possible choices to continue.

Find the Shortest Path from Start to Goal



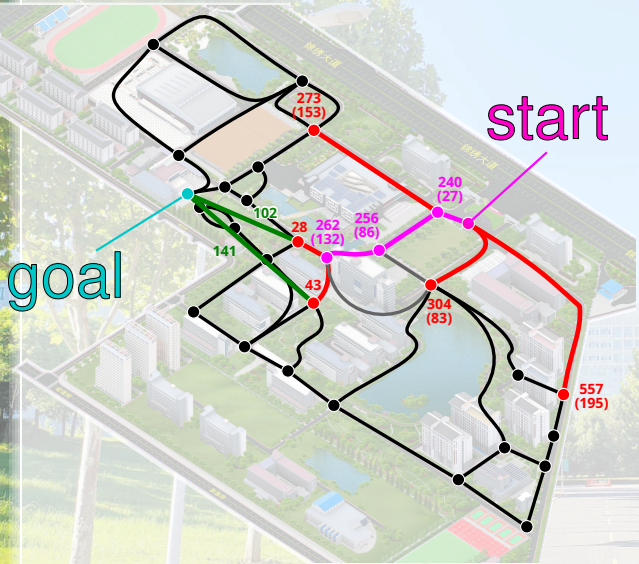
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From (f), we have two possible choices to continue.

Find the Shortest Path from Start to Goal



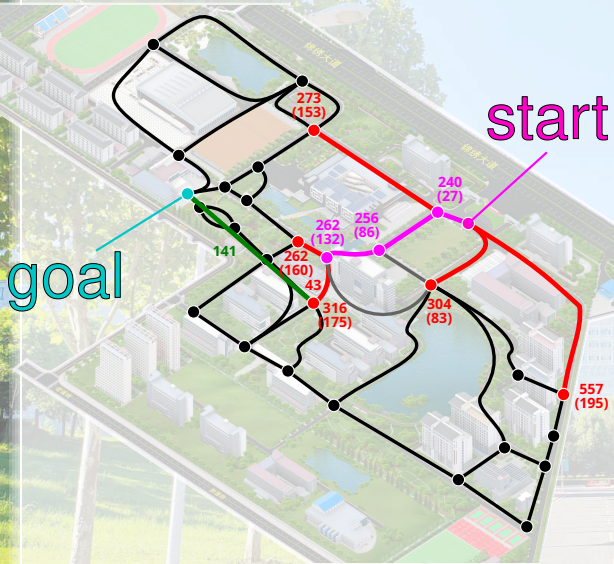
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From (f), we have two possible choices to continue.

Find the Shortest Path from Start to Goal



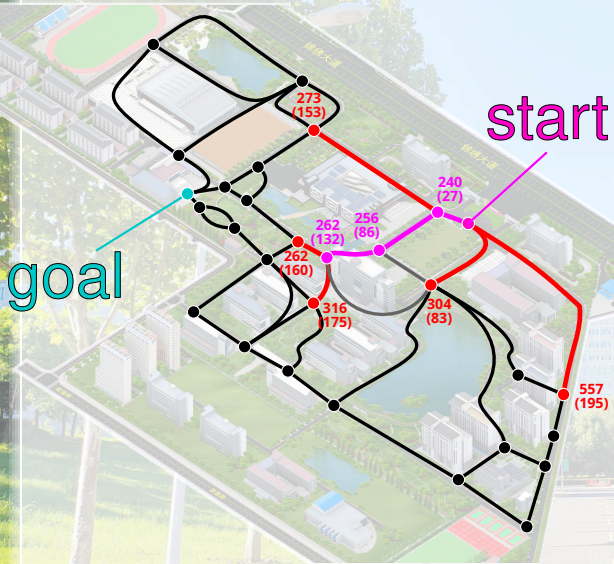
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- We have 3 remaining unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$.
- From (f), we have two possible choices to continue.

Find the Shortest Path from Start to Goal



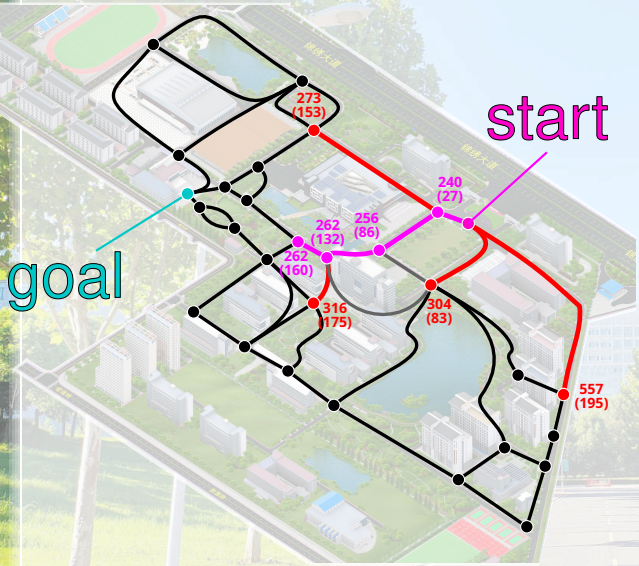
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- From (f), we have two possible choices to continue.
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (g) $262s = 160s + 102s$.

Find the Shortest Path from Start to Goal



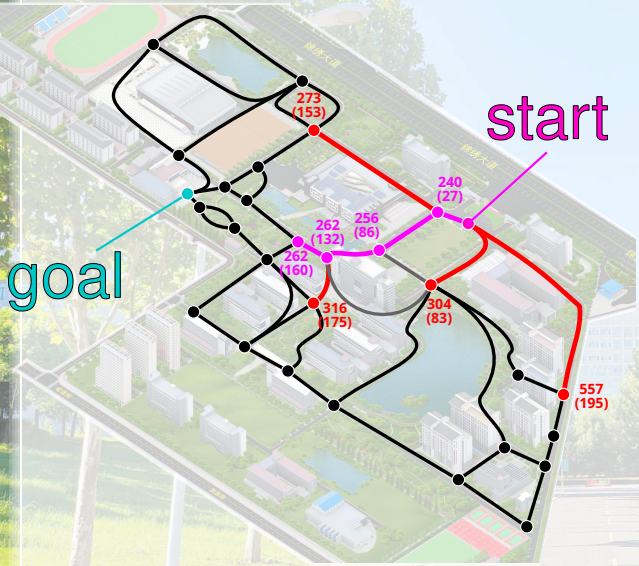
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- (f) it is...
- From (f), we have two possible choices to continue.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (g) $262s = 160s + 102s$,
  (h) $316s = 175s + 141s$.

Find the Shortest Path from Start to Goal



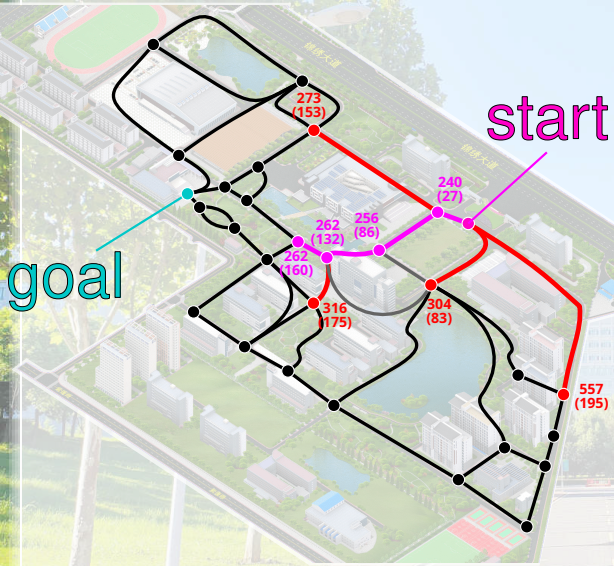
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we again compute the airline distance to the goal.
- From (f), we have two possible choices to continue.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (g) $262s = 160s + 102s$,
  (h) $316s = 175s + 141s$.
- (g) it is...

Find the Shortest Path from Start to Goal



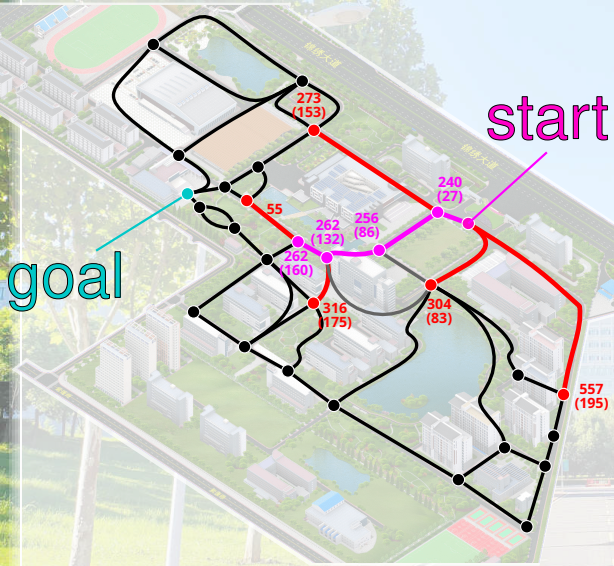
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (g) it is...
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$.

Find the Shortest Path from Start to Goal



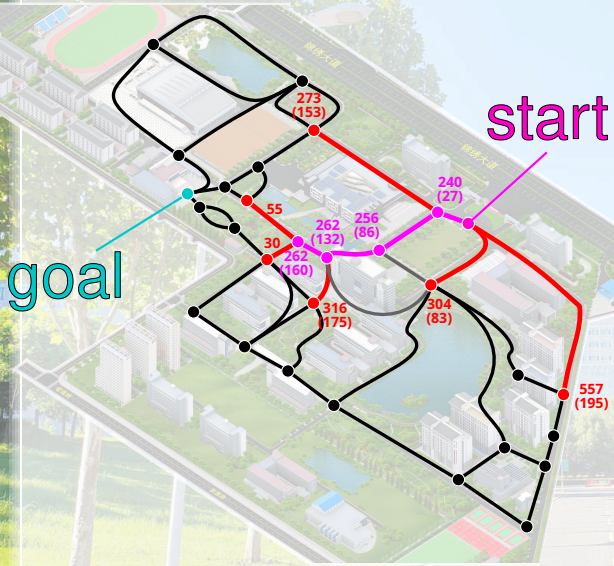
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (g) it is...
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$.
- And we get two new choices.

Find the Shortest Path from Start to Goal



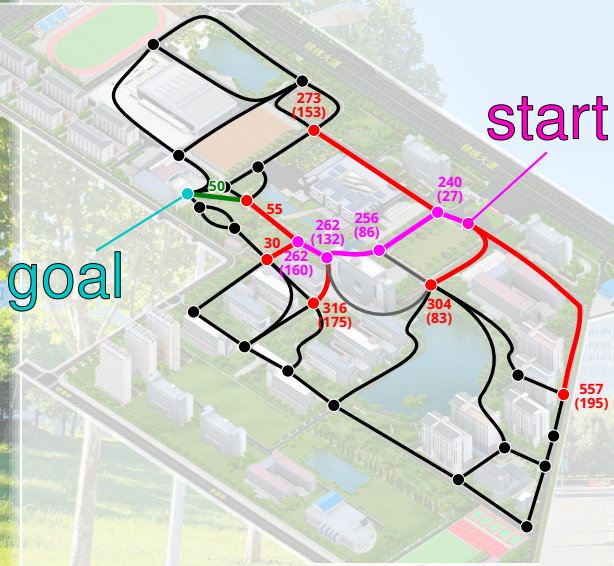
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (g) it is...
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$.
- And we get two new choices.

Find the Shortest Path from Start to Goal



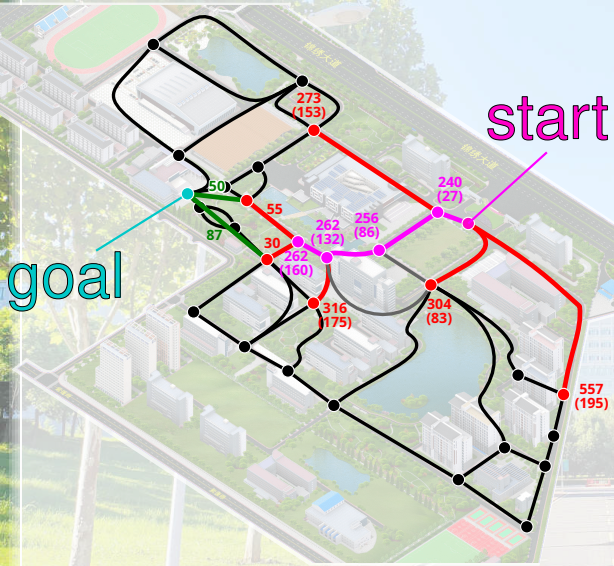
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (g) it is...
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$.
- And we get two new choices.

Find the Shortest Path from Start to Goal



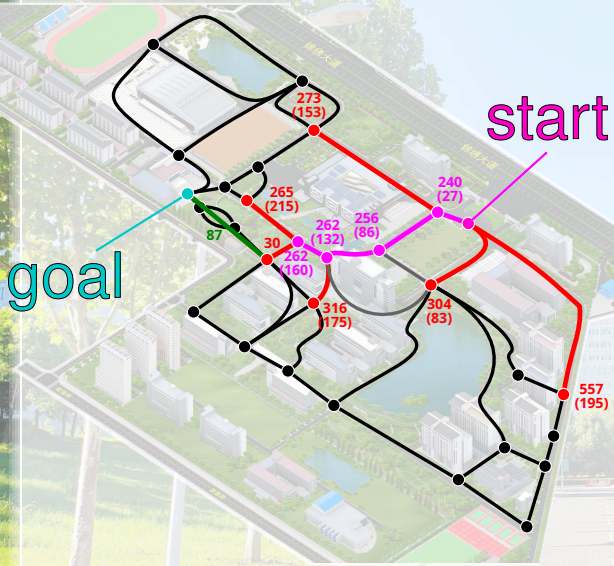
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (g) it is...
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$.
- And we get two new choices.

Find the Shortest Path from Start to Goal



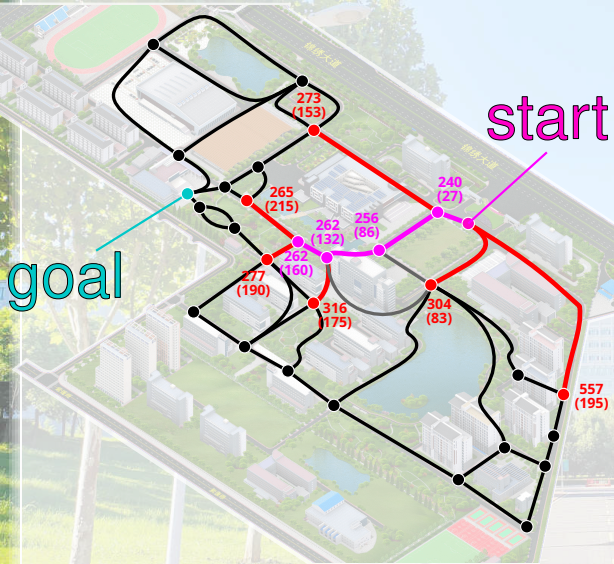
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- (g) it is...
- We now have 4 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$.
- And we get two new choices.

Find the Shortest Path from Start to Goal



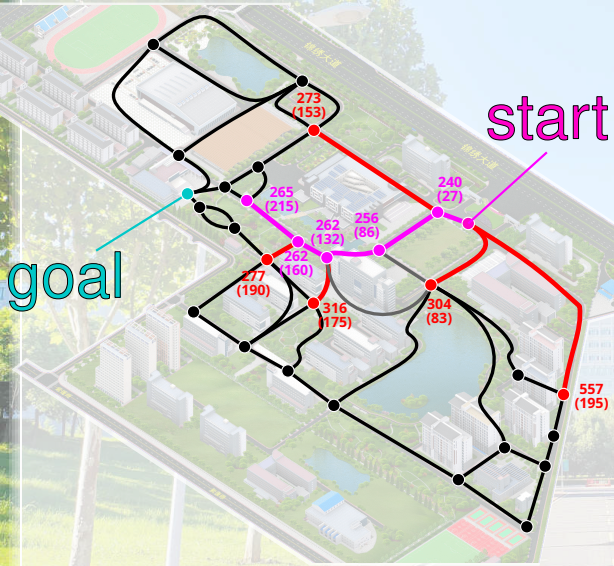
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we get two new choices.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (i) $265s = 215s + 50s$.

Find the Shortest Path from Start to Goal



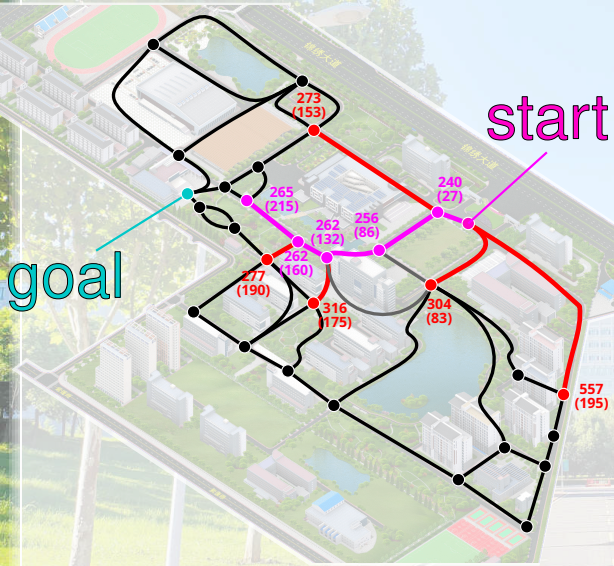
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we get two new choices.
- We now have 6 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (i) $265s = 215s + 50s$,
  - (j) $277s = 190s + 87s$.

Find the Shortest Path from Start to Goal



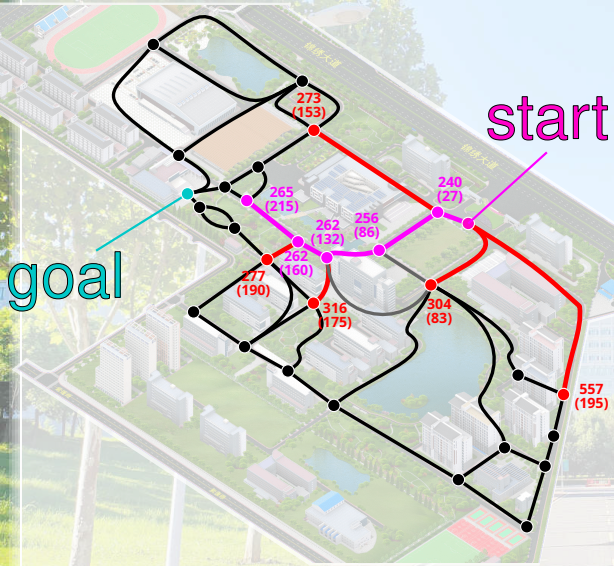
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- And we get two new choices.
- We now have 6 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (i) $265s = 215s + 50s$,
  - (j) $277s = 190s + 87s$.
- (i) it is. . .

Find the Shortest Path from Start to Goal



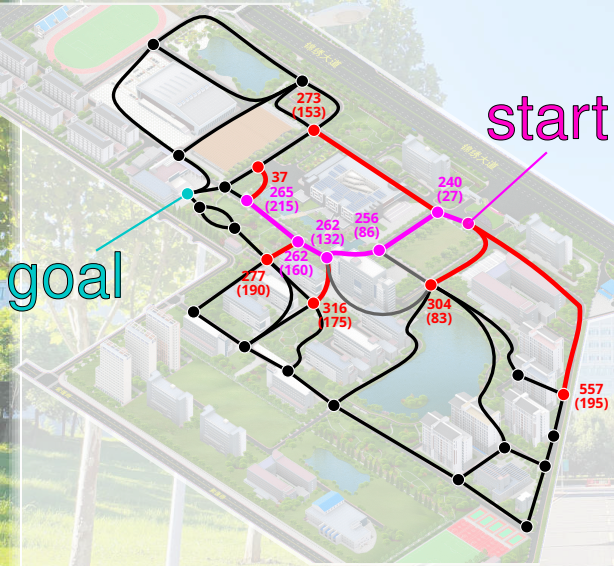
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$.

Find the Shortest Path from Start to Goal



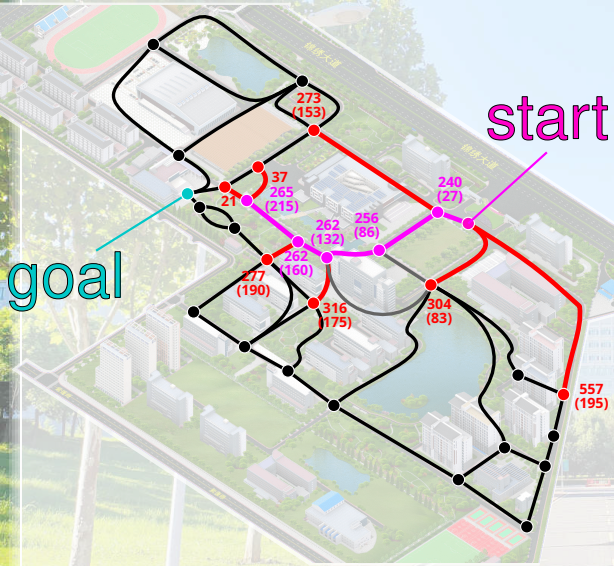
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$.
- From (i), we again got two new choices.

Find the Shortest Path from Start to Goal



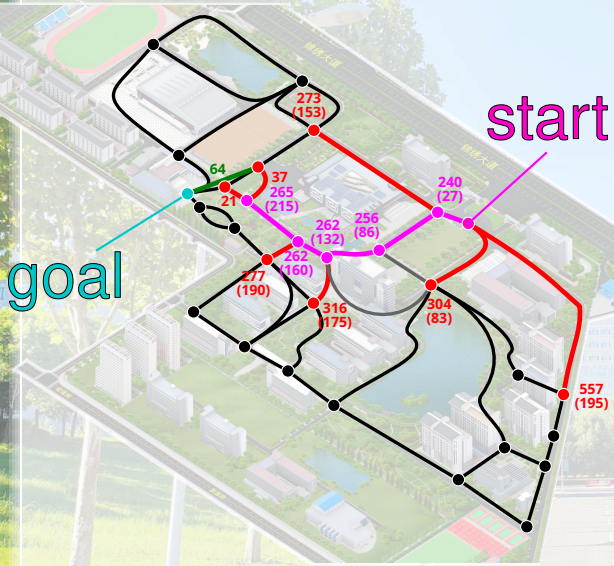
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$.
- From (i), we again got two new choices.

Find the Shortest Path from Start to Goal



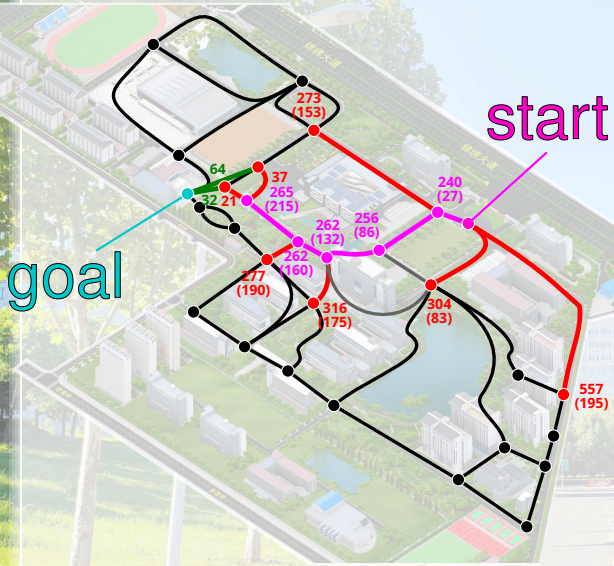
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$.
- From (i), we again got two new choices.

Find the Shortest Path from Start to Goal



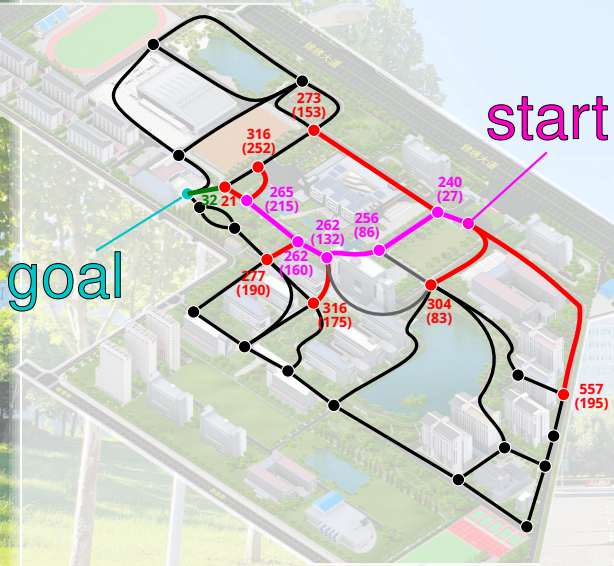
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$.
- From (i), we again got two new choices.

Find the Shortest Path from Start to Goal



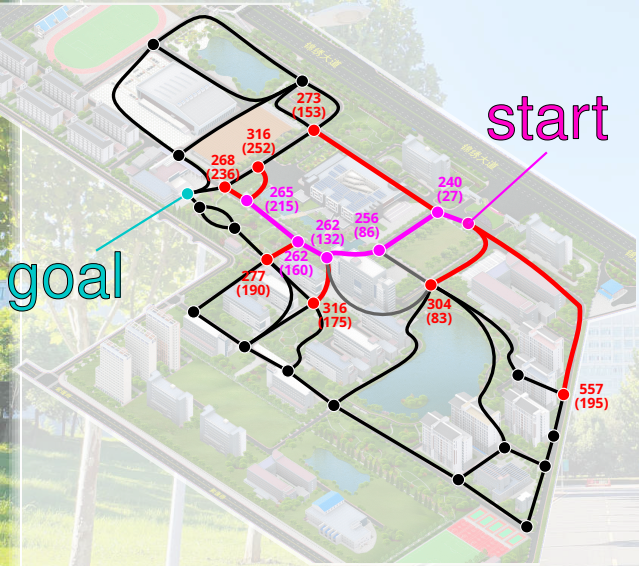
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 5 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$.
- From (i), we again got two new choices.

Find the Shortest Path from Start to Goal



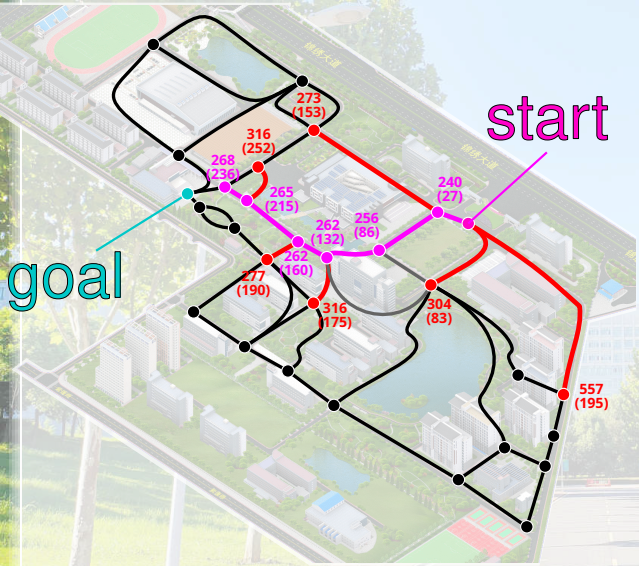
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- From (i), we again got two new choices.
- We now have 6 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.

Find the Shortest Path from Start to Goal



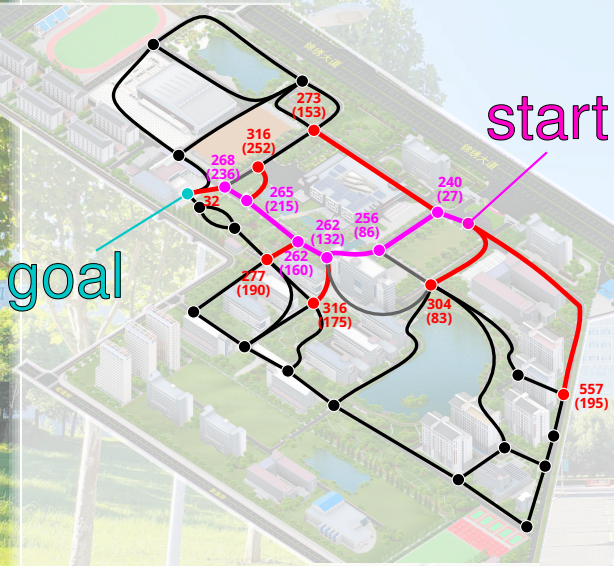
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- From (i), we again got two new choices.
- We now have 7 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$,
  - (l) $268s = 236s + 32s$.

Find the Shortest Path from Start to Goal



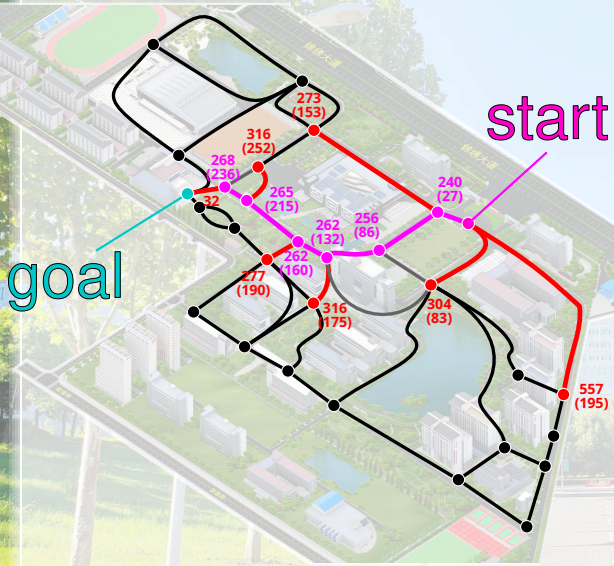
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- From (i), we again got two new choices.
- We now have 7 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$,
  - (l) $268s = 236s + 32s$.
- (l) it is...

Find the Shortest Path from Start to Goal



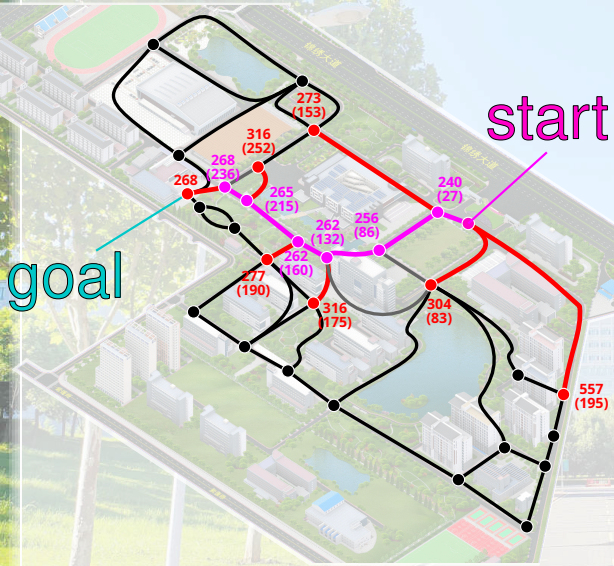
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 6 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$,
  (k) $316s = 252s + 64s$.
- (l) it is...
- There is only one choice to continue: go to the goal!

Find the Shortest Path from Start to Goal



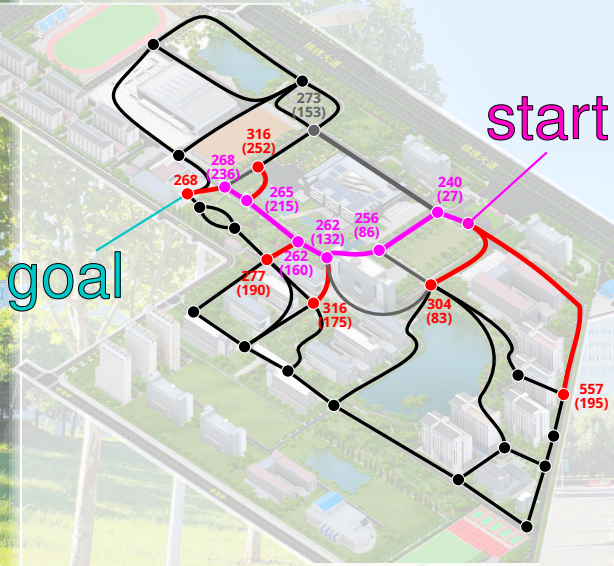
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 6 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$,
  (k) $316s = 252s + 64s$.
- (l) it is...
- There is only one choice to continue: go to the goal!
- We found a first complete path from start to goal.

Find the Shortest Path from Start to Goal



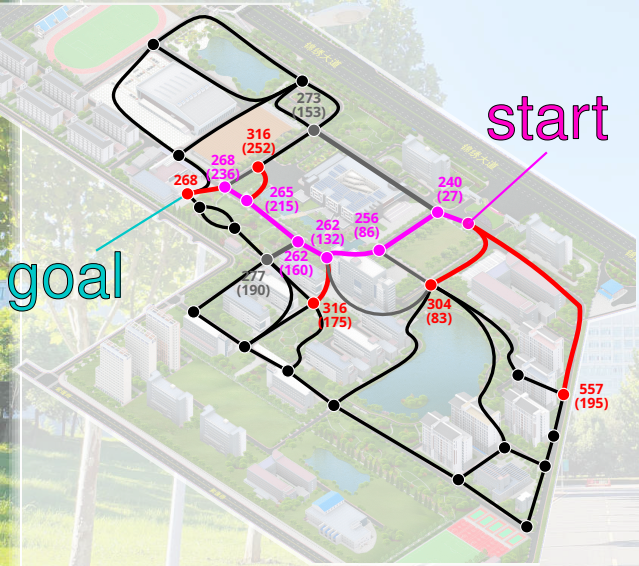
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 6 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$,
  (k) $316s = 252s + 64s$.
- There is only one choice to continue: go to the goal!
- We found a first complete path from start to goal.
- It has the total length $268s$.

Find the Shortest Path from Start to Goal



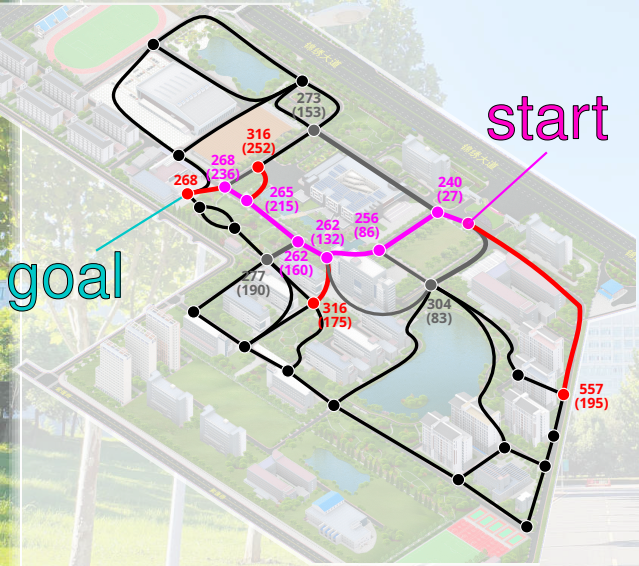
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 56 unexplored choices:
  (b) $304s = 83s + 221s$,
  (c) $557s = 195s + 362s$,
  (d) $273s = 153s + 120s$,
  (h) $316s = 175s + 141s$,
  (j) $277s = 190s + 87s$,
  (k) $316s = 252s + 64s$.
- (d) definitely will need more than $268s$.

Find the Shortest Path from Start to Goal



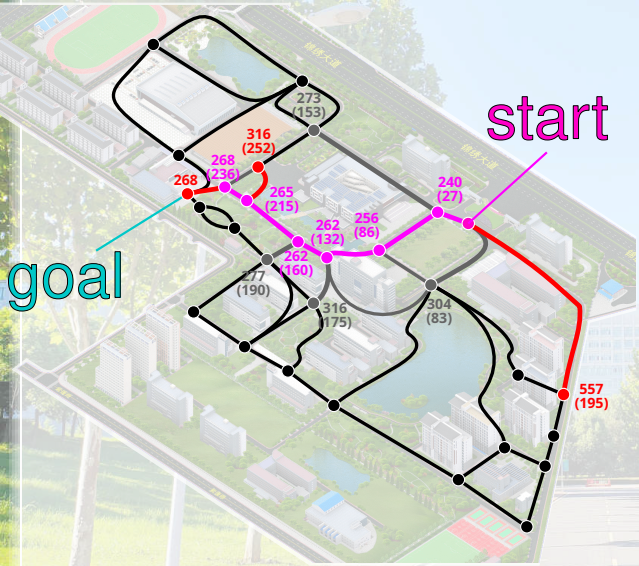
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 4 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- (j) definitely will need more than $268s$.

Find the Shortest Path from Start to Goal



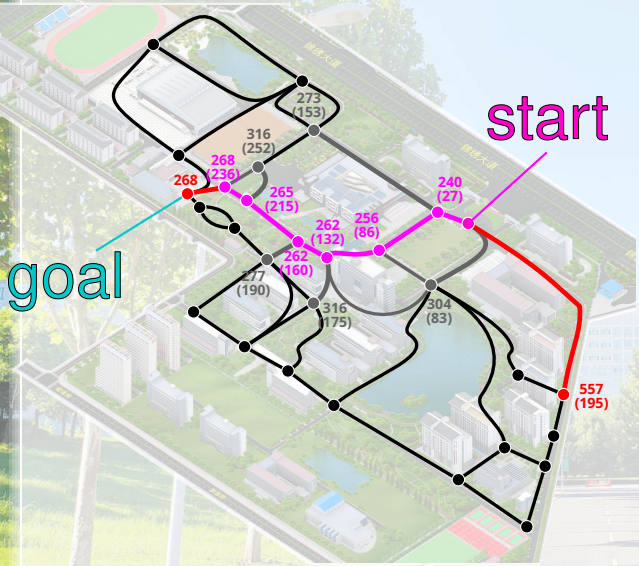
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 3 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- (b) definitely will need more than $268s$.

Find the Shortest Path from Start to Goal



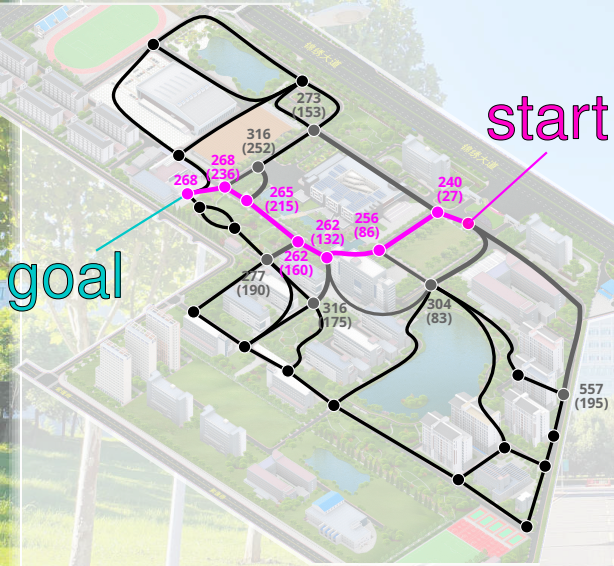
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 2 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- (h) definitely will need more than $268s$.

Find the Shortest Path from Start to Goal



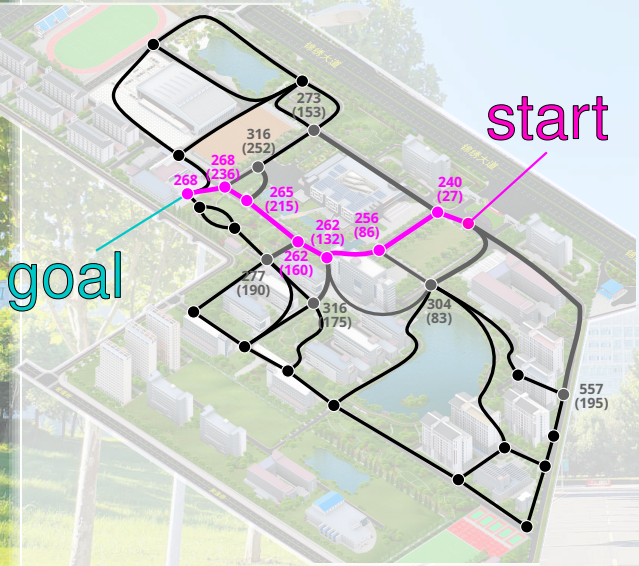
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 1 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- (k) definitely will need more than $268s$.

Find the Shortest Path from Start to Goal



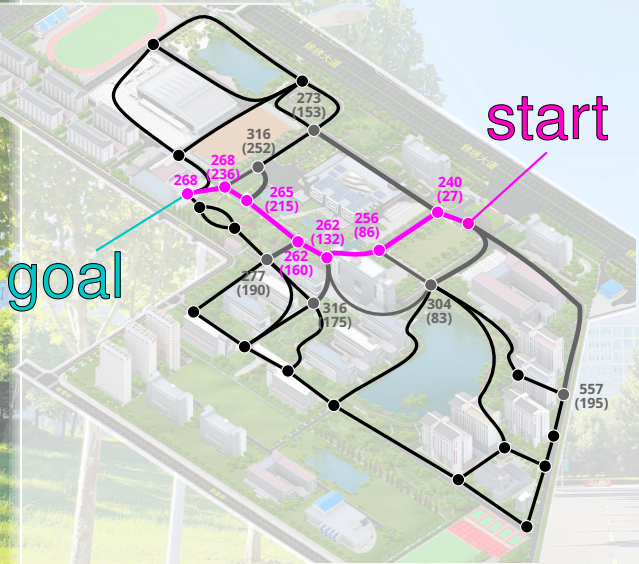
- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 0 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- (c) definitely will need more than $268s$.

Find the Shortest Path from Start to Goal



- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 0 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- We found the shortest possible path and it takes $268s$.

Find the Shortest Path from Start to Goal



- I am at the starting point 合肥大学南二区北大门.
- I want to go to the goal: 食堂.
- We now have 0 unexplored choices:
  - (b) $304s = 83s + 221s$,
  - (c) $557s = 195s + 362s$,
  - (d) $273s = 153s + 120s$,
  - (h) $316s = 175s + 141s$,
  - (j) $277s = 190s + 87s$,
  - (k) $316s = 252s + 64s$.
- We found the shortest possible path and it takes $268s$.
- **Solved.**

The A* Algorithmfor Finding Shortest Paths

- We just applied the A* Algorithm[23,39].

The A* Algorithmfor Finding Shortest Paths

- We just applied the A* Algorithm[23,39].
- The A* Algorithm iteratively constructs a solution.

The A* Algorithmfor Finding Shortest Paths

- We just applied the A* Algorithm[23,39].
- The A* Algorithm iteratively constructs a solution.
- It decides next step to test based on a combination of heuristic and cost.

The A* Algorithmfor Finding Shortest Paths

- We just applied the A* Algorithm[23,39].
- The A* Algorithm iteratively constructs a solution.
- It decides next step to test based on a combination of heuristic and cost.
- In the worst case, it needs to "look" at all intersections (nodes) and streets (edges) once.

The A* Algorithmfor Finding Shortest Paths

- We just applied the A* Algorithm[23,39].
- The A* Algorithm iteratively constructs a solution.
- It decides next step to test based on a combination of heuristic and cost.
- In the worst case, it needs to "look" at all intersections (nodes) and streets (edges) once.
- Usually, it is quite efficient to find shortest paths on maps.

The A* Algorithmfor Finding Shortest Paths

- We just applied the A* Algorithm[23,39].
- The A* Algorithm iteratively constructs a solution.
- It decides next step to test based on a combination of heuristic and cost.
- In the worst case, it needs to "look" at all intersections (nodes) and streets (edges) once.
- Usually, it is quite efficient to find shortest paths on maps.
- (If we look for shortest paths that do not visit any node twice, where the edge distances are not negative, where the number of choices per node is limited, and where the graph and path are not too big – then this algorithm is very efficient.)

Problems that Algorithms cannot Solve Efficiently *and* Exactly

Hard Problems (1)

- There exists a group of problems that are *hard*.

Hard Problems (1)

- There exists a group of problems that are *hard*.
- Actually, most of the problems I mention at the beginning of this talk belong to this group of hard problems.

Hard Problems (1)

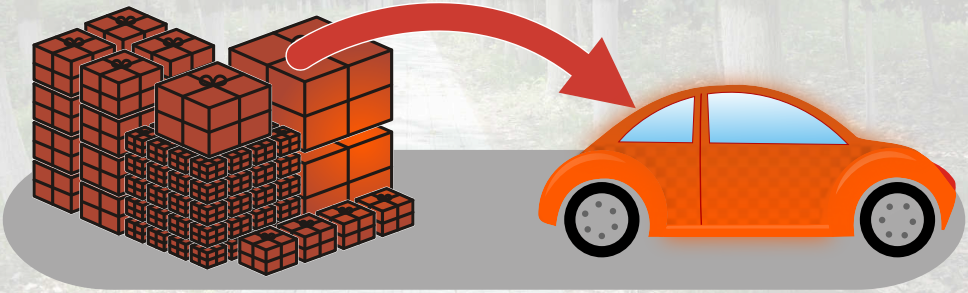
- There exists a group of problems that are *hard*.
- Actually, most of the problems I mention at the beginning of this talk belong to this group of hard problems.
- A hard problem cannot be solved both exactly and efficiently.

Hard Problems (1)

- There exists a group of problems that are *hard*.
- Actually, most of the problems I mention at the beginning of this talk belong to this group of hard problems.
- A hard problem cannot be solved both exactly and efficiently.
- Let‘s look at two quick examples.

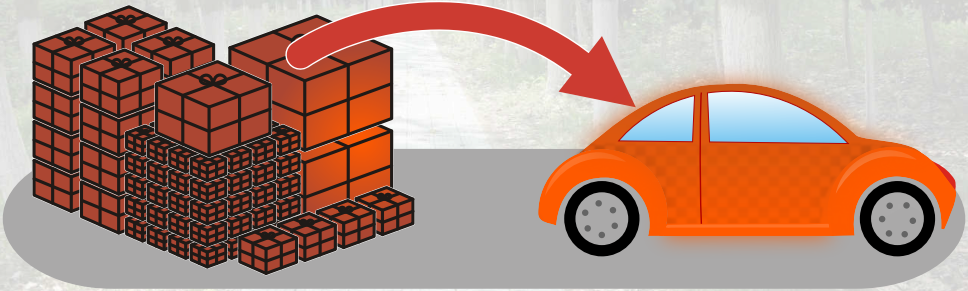
Bin Packing

- My car can carry Tkg of weight.



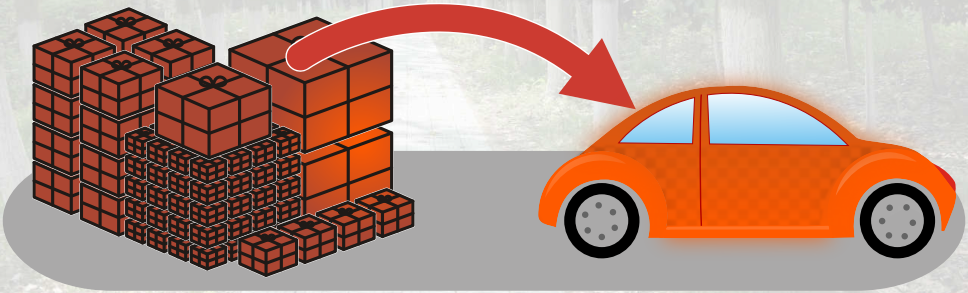
Bin Packing

- My car can carry Tkg of weight.
- I have n objects, each with weight wi for i in 1 to n.



Bin Packing

- My car can carry Tkg of weight.
- I have n objects, each with weight wi for i in 1 to n.
- How can I pack my car so that I can carry them from A to B with the fewest possible hauls?[56,57]



Traveling Salesperson Problem

- Find the shortest path that visits n locations and returns back to its origin.[1,29]

Hard Problems (2)

- These are just two examples from a huge family of problems called $\mathcal{NP}$-hard.

Hard Problems (2)

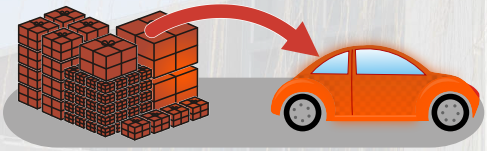
- These are just two examples from a huge family of problems called $\mathcal{NP}$-hard.
- What does that mean?

Hard Problems (2)

- These are just two examples from a huge family of problems called $\mathcal{NP}$-hard.
- What does that mean?

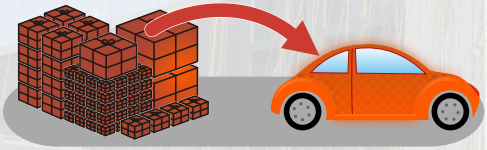
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the worst case.

Hard Problems (2)



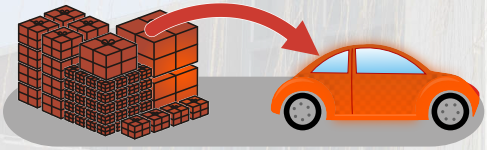
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the worst case.

Hard Problems (2)



If an algorithm guarantees to always find the **best-possible** solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the worst case.

Hard Problems (2)

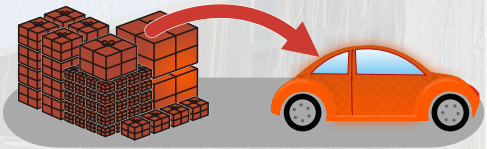


packing with fewest hauls

shortest round-trip tour

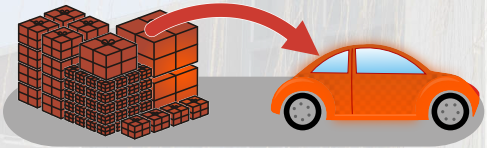
If an algorithm guarantees to always find the **best-possible** solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the worst case.

Hard Problems (2)



If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the **input size s** in the worst case.

Hard Problems (2)

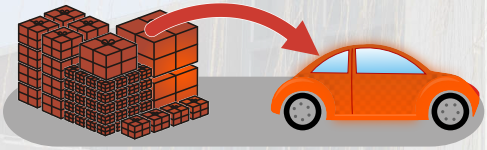


number of items to pack

number of cities to visit

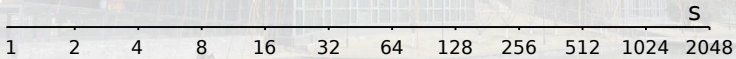
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the **input size s** in the worst case.

Hard Problems (2)



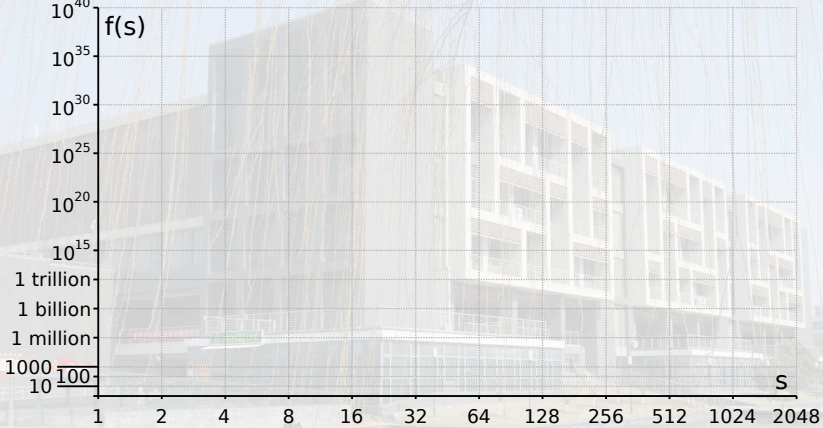
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



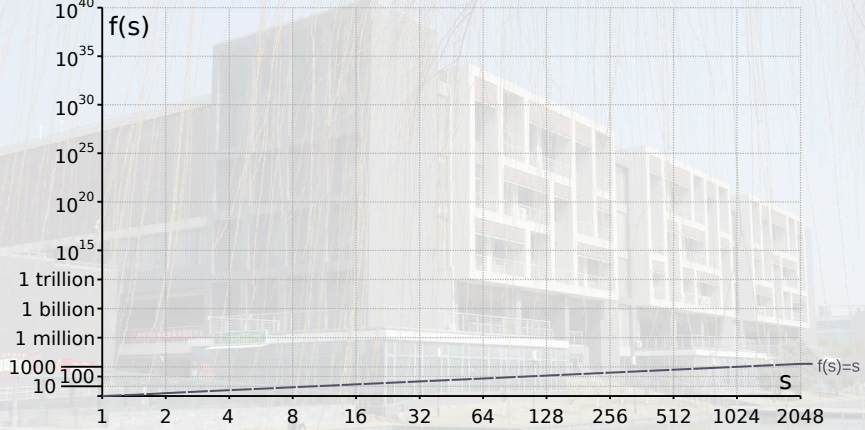
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



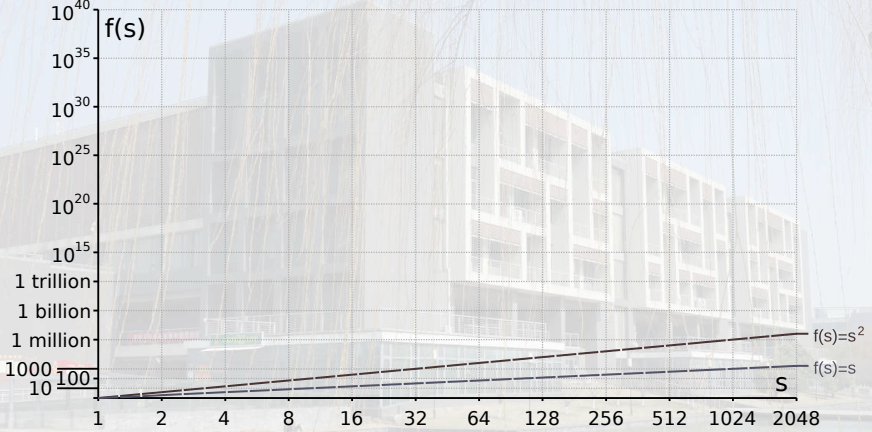
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



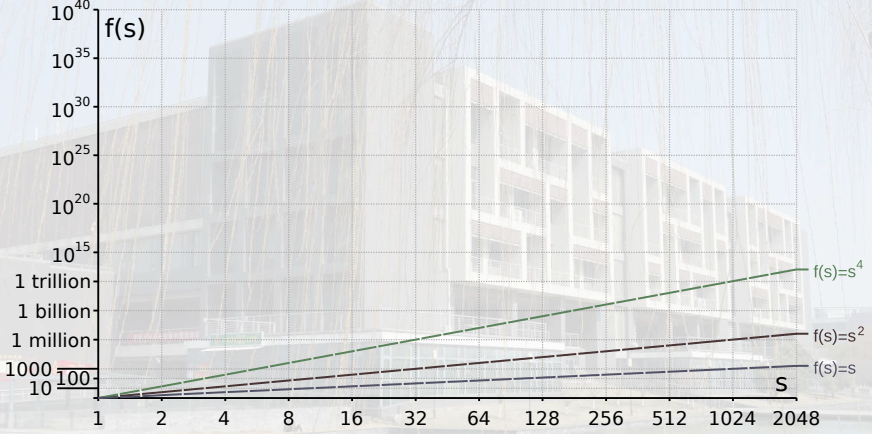
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



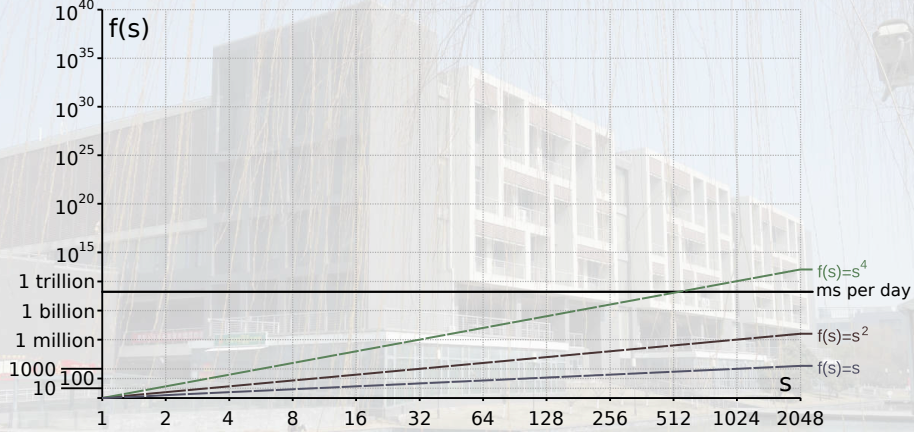
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



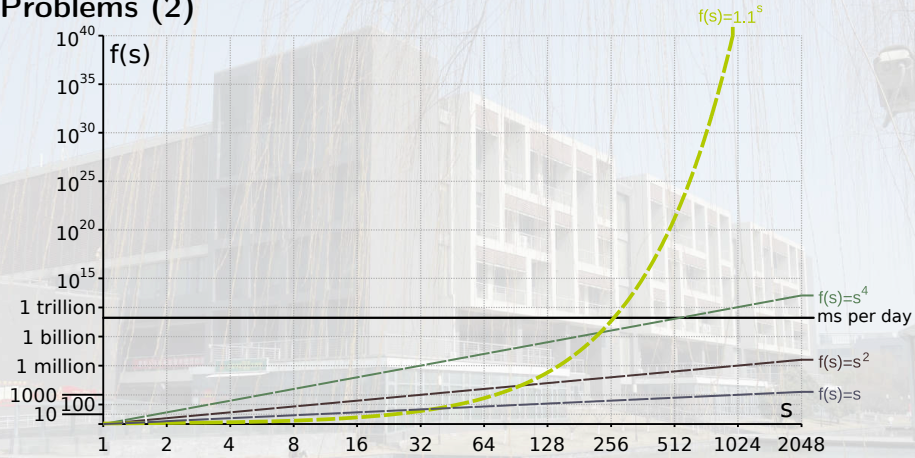
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



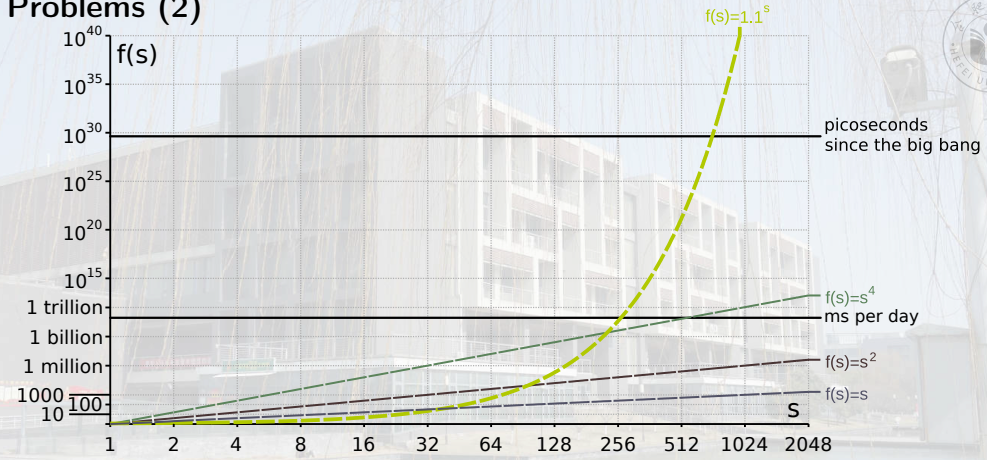
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



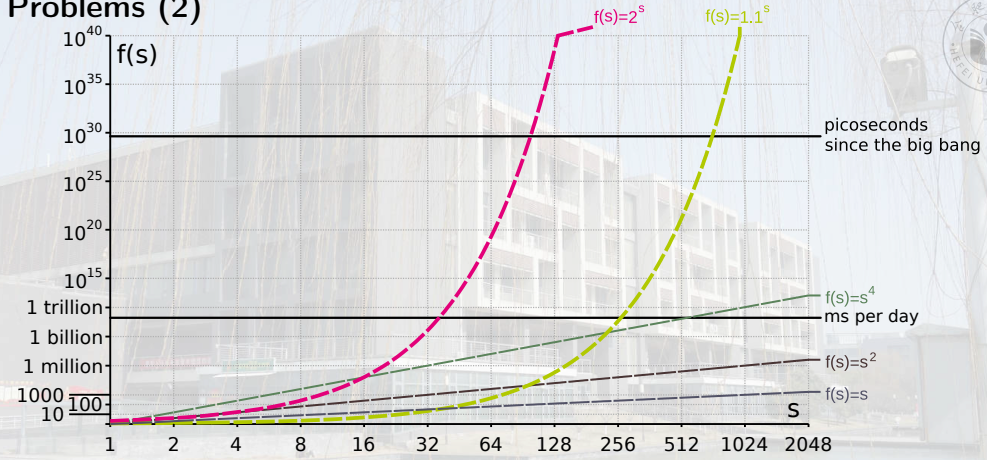
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



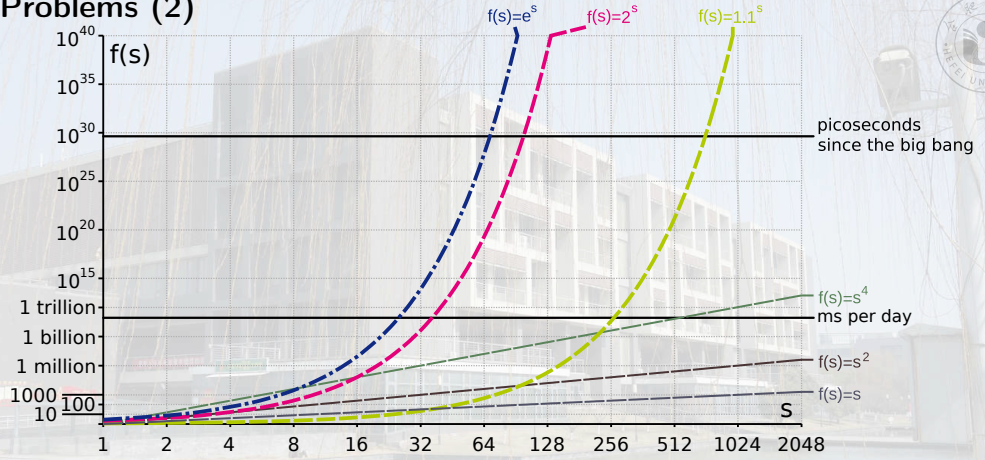
If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)



If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will **grow exponential** with the input size ***s*** in the worst case.

Hard Problems (2)

- These are just two examples from a huge family of problems called $\mathcal{NP}$-hard.
- What does that mean?

If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the worst case.

Hard Problems (2)

- These are just two examples from a huge family of problems called $\mathcal{NP}$-hard.
- What does that mean?

If we want to get the optimal solution, it will take too long.

If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the worst case.

Hard Problems (2)

- These are just two examples from a huge family of problems called $\mathcal{NP}$-hard.
- What does that mean?

If we want to get the optimal solution,
it will sometimes/often/always take too long.

If an algorithm guarantees to always find the best-possible solution for an $\mathcal{NP}$-hard problem, then its runtime will grow exponential with the input size s in the **worst case**.

Traveling Salesperson Problem

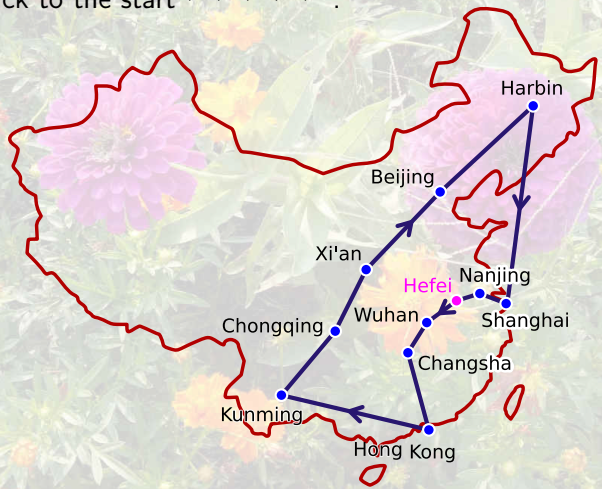
- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].

Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].

Traveling Salesperson Problem

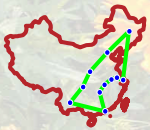
- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].



Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].

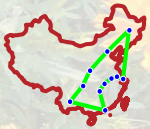
getting the optimal solution
for a TSP



Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].

getting the optimal solution
for a TSP may take too long

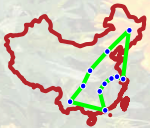


consumed runtime: very much / too (?) long

Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].

getting the optimal solution
for a TSP may take too long



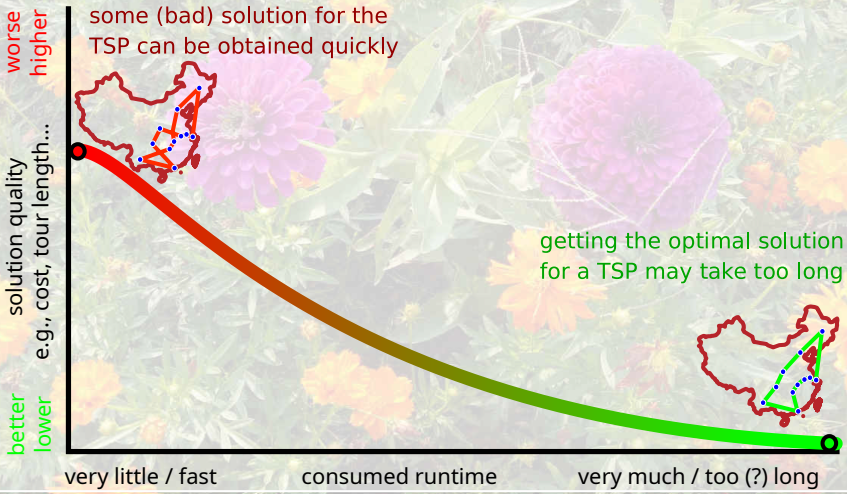
very little / fast — consumed runtime — very much / too (?) long

Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].

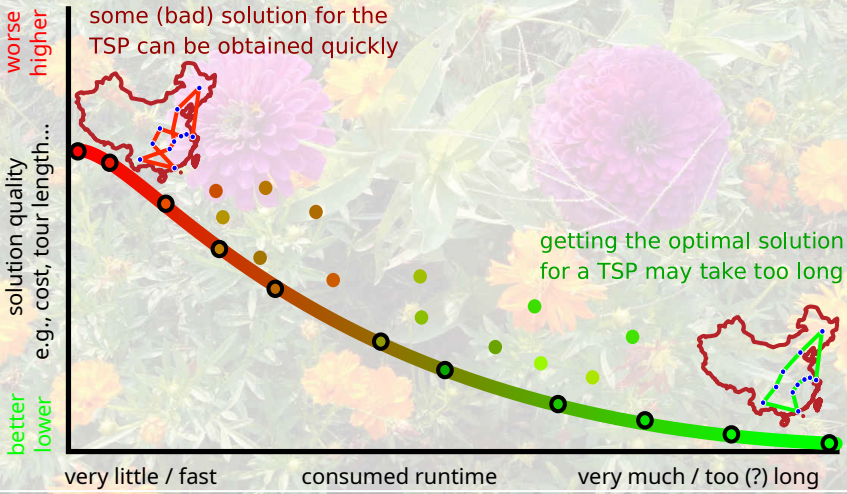
Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].



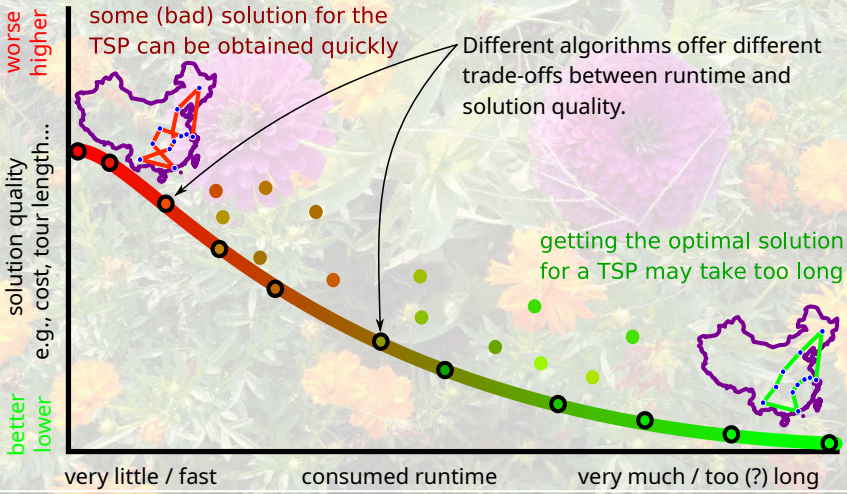
Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].



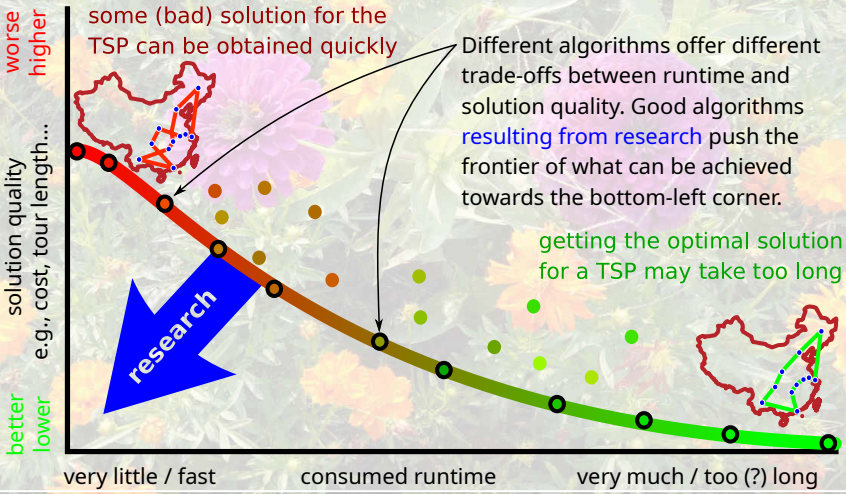
Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].



Traveling Salesperson Problem

- In an Traveling Salesperson Problem (TSP), we want to find the shortest path through n locations and back to the start[1,22,29,31,47,53].



Randomly Guessing Solutions

Almost Solving Hard Problems

- For many problems, it is very easy to "guess" a solution.

Almost Solving Hard Problems

- For many problems, it is very easy to "guess" a solution.
- This solution will (very likely) not be optimal.

Almost Solving Hard Problems

- For many problems, it is very easy to "guess" a solution.
- This solution will (very likely) not be optimal.
- It will very likely be very bad.

Almost Solving Hard Problems

- For many problems, it is very easy to "guess" a solution.
- This solution will (very likely) not be optimal.
- It will very likely be very bad.
- But it will be better than nothing.

Random Sampling for the TSP

- Back to the Traveling Salesperson Problem (TSP).

Random Sampling for the TSP

- Back to the Traveling Salesperson Problem (TSP).
- How do we "guess" a solution?

Random Sampling for the TSP

- Back to the Traveling Salesperson Problem (TSP).
- How do we "guess" a solution?
- We could write down the cities in a random order.

Random Sampling for the TSP

- We could write down the cities in a random order.

Random Sampling for the TSP

- We could write down the cities in a random order.

Random Sampling for the TSP

- We could write down the cities in a random order.

Random Sampling for the TSP

- We could write down the cities in a random order.

Random Sampling for the TSP

- We could write down the cities in a random order.

Random Sampling for the TSP

- We could write down the cities in a random order.
- If we do this, we will get a bad result.

Random Sampling for the TSP

- We could write down the cities in a random order.
- If we do this, we will get a bad result.
- But we will get it quickly.

Random Sampling for the TSP

- We could write down the cities in a random order.
- If we do this, we will get a bad result.
- But we will get it quickly.
- And we can do this often.

Random Sampling for the TSP

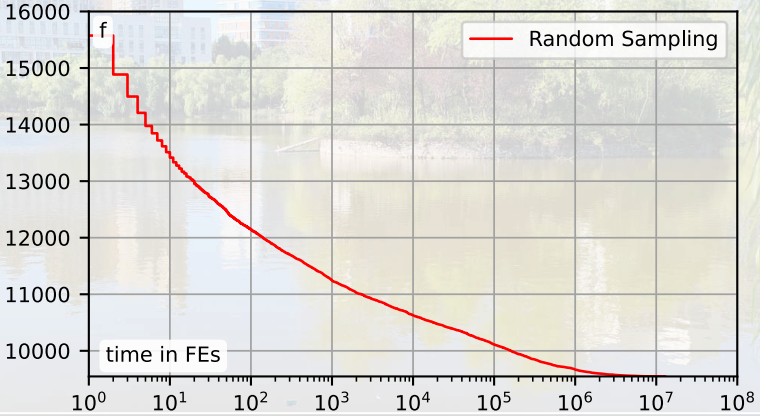
- We could write down the cities in a random order.
- If we do this, we will get a bad result.
- But we will get it quickly.
- And we can do this often.
- For this small problem, my computer can do this *millions* of times in a few seconds.

Random Sampling for the TSP

- We could write down the cities in a random order.
- If we do this, we will get a bad result.
- But we will get it quickly.
- And we can do this often.
- For this small problem, my computer can do this *millions* of times in a few seconds.
- We can let the algorithm run for as much time as we can wait ... and then simply take the best random tour we found.

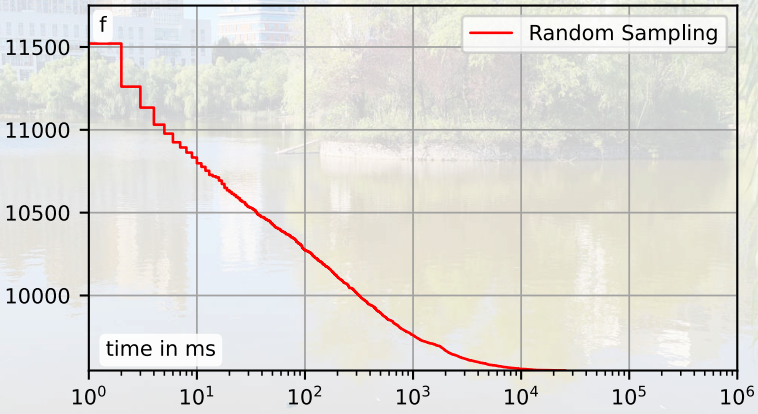
Random Sampling for the TSP

- We could write down the cities in a random order.
- This is the improvement of the best-so-far solution of this so-called Random Sampling algorithm over the number of solutions it has generated and tested (the so-called objective function evaluations (FEs)).



Random Sampling for the TSP

- We could write down the cities in a random order.
- This is the improvement of the best-so-far solution of this so-called Random Sampling algorithm over the consumed runtime in milliseconds.



Random Sampling for the TSP

- We could write down the cities in a random order.
- And for this small problem, this random sampling eventually finds the optimal tour.

Principle 1: We can randomly construct solutions.

Principle 1: We can randomly construct solutions.

Problem 1: Guessing an optimal (or even just good) solution randomly is very unlikely.

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?

1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25].

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?

1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25]. Which is the number of cities in the TSP...

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?

1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25]. Which is the number of cities in the TSP...
2. Also, it is a bit awkward: What are the heuristic and the goal state?

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?
- It will crash for any larger TSP.

1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25]. Which is the number of cities in the TSP...
2. Also, it is a bit awkward: What are the heuristic and the goal state?

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?
- It will crash for any larger TSP.
- But it can be done and used for our small TSP.

1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25]. Which is the number of cities in the TSP...
2. Also, it is a bit awkward: What are the heuristic and the goal state?

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?
- It will crash for any larger TSP.
- But it can be done and used for our small TSP.
- So let's do it.

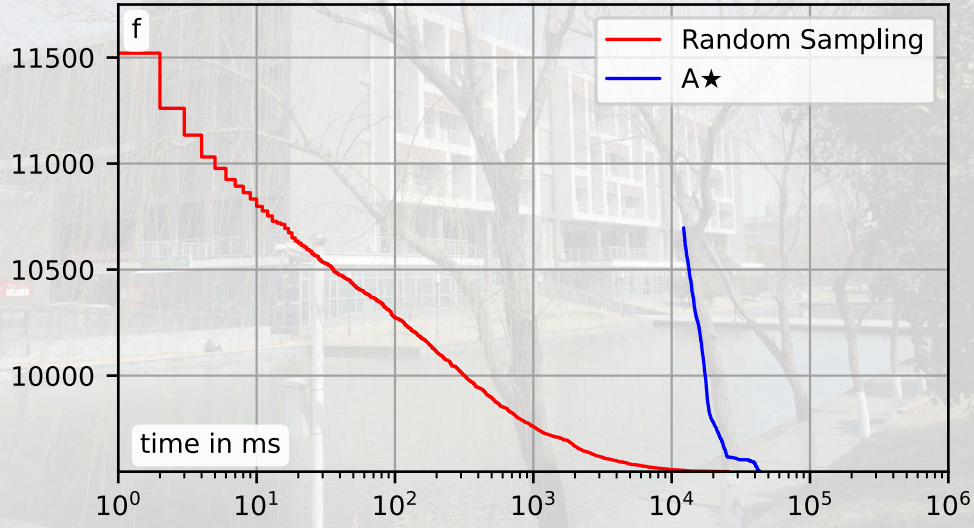
1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25]. Which is the number of cities in the TSP...
2. Also, it is a bit awkward: What are the heuristic and the goal state?

Why don't we use the A* Algorithm for the TSP?

- Before, we talked about the A* Algorithm for path finding.
- Isn't the TSP about finding a path?
- Why do we not use it for the TSP?
- It will crash for any larger TSP.
- But it can be done and used for our small TSP.
- So let's do it. (I spare you all details.)

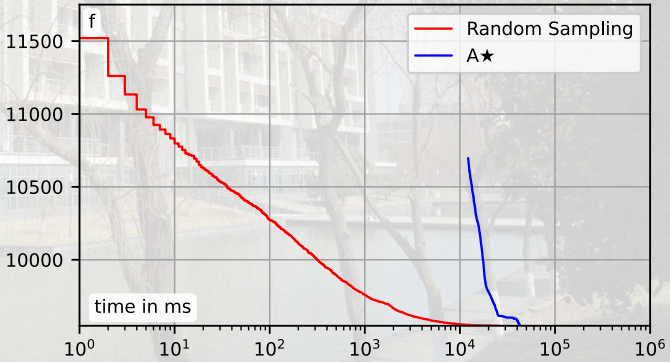
1. The A* Algorithm needs memory exponential in the number of nodes in the final solution[25]. Which is the number of cities in the TSP...
2. Also, it is a bit awkward: What are the heuristic and the goal state?

Why don't we use the A* Algorithm for the TSP?



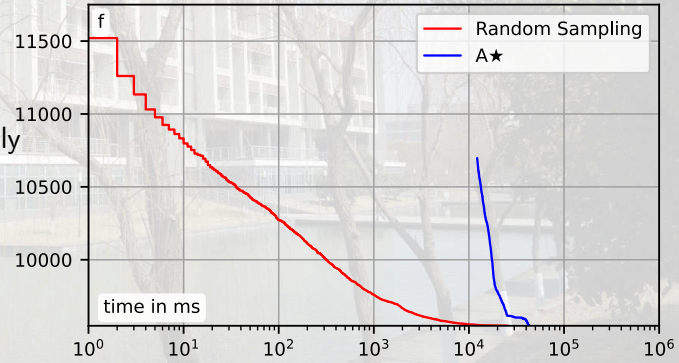
Why don't we use the A* Algorithm for the TSP?

- Because not only will it use too much memory for larger instances...



Why don't we use the A* Algorithm for the TSP?

- Because not only will it use too much memory for larger instances...
- ...it is also slower than randomly guessing on smaller instances.



Random Sampling still is a bad algorithm

- Random sampling is a bad algorithm for the TSP (and basically all other problems, too).

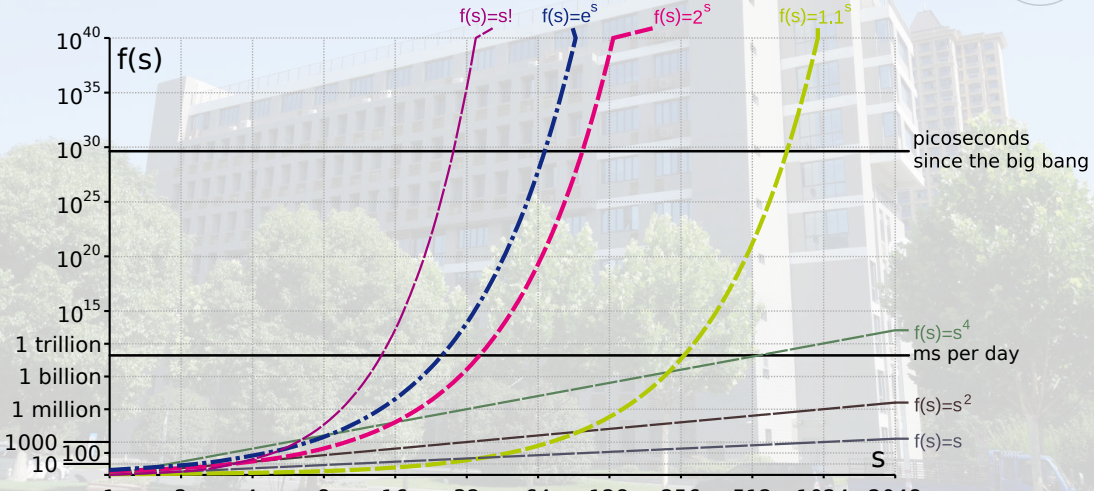
Random Sampling still is a bad algorithm

- Random sampling is a bad algorithm for the TSP (and basically all other problems, too).
- It relies on randomly guessing good tours.

Random Sampling still is a bad algorithm

- Random sampling is a bad algorithm for the TSP (and basically all other problems, too).
- It relies on randomly guessing good tours.
- If we have n cities, then there are $1 * 2 * 3 * \cdots * (n-1) * n = n!$ tours.

Random Sampling still is a bad algorithm



Random Sampling still is a bad algorithm

- Random sampling is a bad algorithm for the TSP (and basically all other problems, too).
- It relies on randomly guessing good tours.
- If we have n cities, then there are $1 * 2 * 3 * \cdots * (n-1) * n = n!$ tours.
- Factorial growth ($n!$) is even worse than exponential growth...

Random Sampling still is a bad algorithm

- Random sampling is a bad algorithm for the TSP (and basically all other problems, too).
- It relies on randomly guessing good tours.
- If we have n cities, then there are $1 * 2 * 3 * \cdots * (n-1) * n = n!$ tours.
- Factorial growth ($n!$) is even worse than exponential growth...
- So if we try to randomly sample the best possible tour, our chance is extremely small...

Local Search: Using Information

Random Sampling is a bad algorithm

- In each step, random sampling creates a completely random tour.

Random Sampling is a bad algorithm

- In each step, random sampling creates a completely random tour.
- It does not gather any information.

Random Sampling is a bad algorithm

- In each step, random sampling creates a completely random tour.
- It does not gather any information.
- It keeps the best tour, but does not use any information inside this tour.

Random Sampling is a bad algorithm

- In each step, random sampling creates a completely random tour.
- It does not gather any information.
- It keeps the best tour, but does not use any information inside this tour.
- Just guessing answers randomly is not a good method.

Random Sampling is a bad algorithm

- In each step, random sampling creates a completely random tour.
- It does not gather any information.
- It keeps the best tour, but does not use any information inside this tour.
- Just guessing answers randomly is not a good method.
- Clearly, seeing millions of tours, we should be able learn *something* and somehow use that to find better tours???

Randomized Local Search

- But what could we do?

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."
- neighborhood = solutions that are similar.

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."
- neighborhood $=$ solutions that are similar.
- If we have a tour x, we could randomly pick two cities in the tour and swap them.

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."
- neighborhood $=$ solutions that are similar.
- If we have a tour x, we could randomly pick two cities in the tour and swap them.
- If we do this, maybe we could get a better tour or a worse tour.

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."
- neighborhood $=$ solutions that are similar.
- If we have a tour x, we could randomly pick two cities in the tour and swap them.
- If we do this, maybe we could get a better tour or a worse tour.
- We could keep a better tour, but throw away a worse one.

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."
- neighborhood $=$ solutions that are similar.
- If we have a tour x, we could randomly pick two cities in the tour and swap them.
- If we do this, maybe we could get a better tour or a worse tour.
- We could keep a better tour, but throw away a worse one.
- Then we do the same with a better tour that we find, and with yet a better tour, and so on.

Randomized Local Search

- But what could we do?
- "In the neighborhood of a solution, there will probably be better or worse solutions."
- neighborhood $=$ solutions that are similar.
- If we have a tour x, we could randomly pick two cities in the tour and swap them.
- If we do this, maybe we could get a better tour or a worse tour.
- We could keep a better tour, but throw away a worse one.
- Then we do the same with a better tour that we find, and with yet a better tour, and so on.
- This method is called randomized local search (RLS).

Randomized Local Search for the TSP

15983 km

合肥, 北京, 西安, 南京, 哈尔滨,
香港, 长沙, 昆明, 武汉, 上海, 重庆

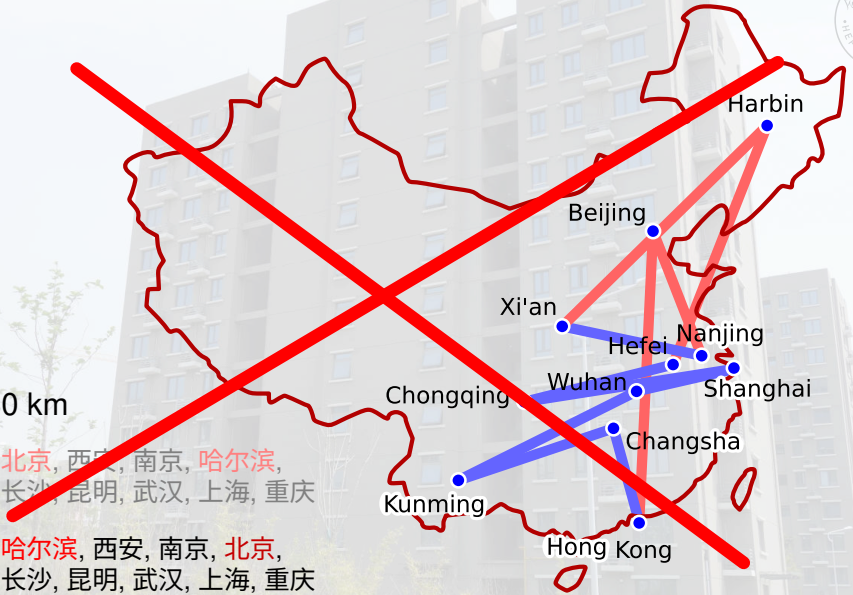
Randomized Local Search for the TSP

16080 km

合肥, 北京, 西安, 南京, 哈尔滨,
香港, 长沙, 昆明, 武汉, 上海, 重庆

合肥, 哈尔滨, 西安, 南京, 北京,
香港, 长沙, 昆明, 武汉, 上海, 重庆

Randomized Local Search for the TSP



16080 km

合肥, 北京, 西安, 南京, 哈尔滨,
香港, 长沙, 昆明, 武汉, 上海, 重庆

合肥, 哈尔滨, 西安, 南京, 北京,
香港, 长沙, 昆明, 武汉, 上海, 重庆

Randomized Local Search for the TSP

15983 km

合肥, 北京, 西安, 南京, 哈尔滨,
香港, 长沙, 昆明, 武汉, 上海, 重庆

Randomized Local Search for the TSP

15128 km

合肥, 北京, 西安, 南京, 哈尔滨,
香港, 长沙, 昆明, 武汉, 上海, 重庆

合肥, 北京, 西安, 南京, 上海,
香港, 长沙, 昆明, 武汉, 哈尔滨, 重庆

Randomized Local Search for the TSP

15128 km

合肥, 北京, 西安, 南京, 上海,
香港, 长沙, 昆明, 武汉, 哈尔滨, 重庆

Randomized Local Search for the TSP

16435 km

合肥, 北京, 西安, 南京, 上海,
香港, 长沙, 昆明, 武汉, 哈尔滨, 重庆

合肥, 北京, 西安, 南京, 香港,
上海, 长沙, 昆明, 武汉, 哈尔滨, 重庆

Randomized Local Search for the TSP

16435 km

合肥, 北京, 西安, 南京, 上海,
香港, 长沙, 昆明, 武汉, 哈尔滨, 重庆

合肥, 北京, 西安, 南京, 香港,
上海, 长沙, 昆明, 武汉, 哈尔滨, 重庆

Randomized Local Search for the TSP

15128 km

合肥, 北京, 西安, 南京, 上海,
香港, 长沙, 昆明, 武汉, 哈尔滨, 重庆

Randomized Local Search for the TSP

12432 km

合肥, 北京, 西安, 南京, 上海,
香港, 长沙, 昆明, 武汉, 哈尔滨, 重庆

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 武汉, 西安, 重庆

Randomized Local Search for the TSP

12432 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 武汉, 西安, 重庆

Randomized Local Search for the TSP

14036 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 武汉, 西安, 重庆

合肥, 北京, 哈尔滨, 武汉, 上海,
香港, 长沙, 昆明, 南京, 西安, 重庆

Randomized Local Search for the TSP

14036 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 武汉, 西安, 重庆

合肥, 北京, 哈尔滨, 武汉, 上海,
香港, 长沙, 昆明, 南京, 西安, 重庆

Randomized Local Search for the TSP

12432 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 武汉, 西安, 重庆

Randomized Local Search for the TSP

10857 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 武汉, 西安, 重庆

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 重庆, 西安, 武汉

Randomized Local Search for the TSP

10857 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 重庆, 西安, 武汉

Randomized Local Search for the TSP

10620 km

合肥, 北京, 哈尔滨, 南京, 上海,
香港, 长沙, 昆明, 重庆, 西安, 武汉

合肥, 北京, 哈尔滨, 南京, 上海,
长沙, 香港, 昆明, 重庆, 西安, 武汉

Randomized Local Search for the TSP

10620 km

合肥, 北京, 哈尔滨, 南京, 上海,
长沙, 香港, 昆明, 重庆, 西安, 武汉

Randomized Local Search for the TSP

10567 km

合肥, 北京, 哈尔滨, 南京, 上海,
长沙, 香港, 昆明, 重庆, 西安, 武汉

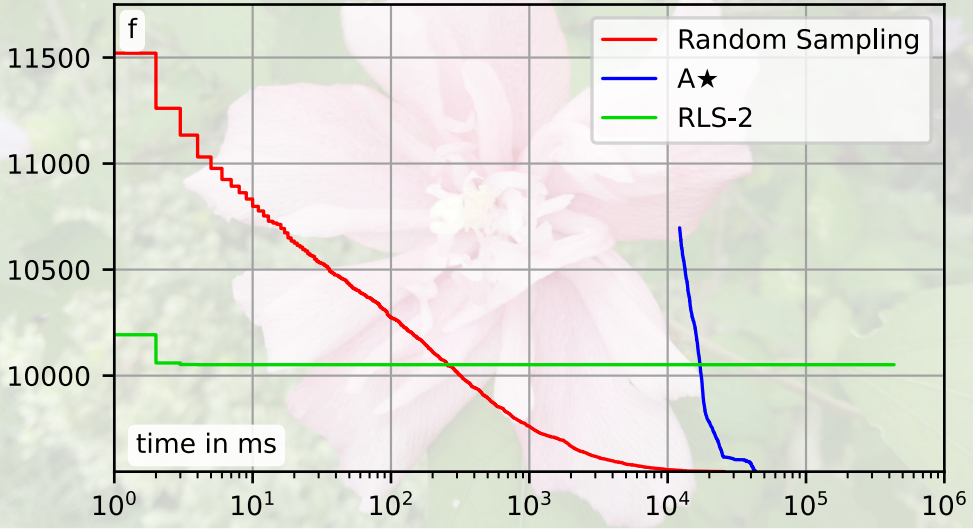
合肥, 北京, 哈尔滨, 上海, 南京,
长沙, 香港, 昆明, 重庆, 西安, 武汉

Randomized Local Search for the TSP

10567 km

合肥, 北京, 哈尔滨, 上海, 南京,
长沙, 香港, 昆明, 重庆, 西安, 武汉

Randomized Local Search for the TSP



Randomized Local Search for the TSP

- RLS is much faster than random sampling.

Randomized Local Search for the TSP

- RLS is much faster than random sampling.
- However...

Randomized Local Search for the TSP

- RLS is much faster than random sampling.
- However, if our algorithm is at a tour x, then it can only find other tours by swapping two cities.

Randomized Local Search for the TSP

- RLS is much faster than random sampling.
- However, if our algorithm is at a tour x, then it can only find other tours by swapping two cities.
- There are some tours that it cannot reach in one step.

Randomized Local Search for the TSP

- RLS is much faster than random sampling.
- However, if our algorithm is at a tour x, then it can only find other tours by swapping two cities.
- There are some tours that it cannot reach in one step.
- If all tours that we can reach in one step are worse than x, we will never leave x.

Randomized Local Search for the TSP

- RLS is much faster than random sampling.
- However, if our algorithm is at a tour x, then it can only find other tours by swapping two cities.
- There are some tours that it cannot reach in one step.
- If all tours that we can reach in one step are worse than x, we will never leave x.
- If x is not the optimum, then we are stuck and won't ever find the optimal/best tour.

Principle 2: Applying a small random change to an existing solution sometimes can give us a better solution.

Principle 2: Applying a small random change to an existing solution sometimes can give us a better solution. This way, we can step-by-step get better solutions.

Principle 2: Applying a small random change to an existing solution sometimes can give us a better solution. This way, we can step-by-step get better solutions.

Problem 2: Sometimes, a small change is not enough.

Principle 2: Applying a small random change to an existing solution sometimes can give us a better solution. This way, we can step-by-step get better solutions.

Problem 2: Sometimes, a small change is not enough. We can get stuck at a so-called *local* optimum.

Randomized Local Search for the TSP

- If all tours that we can reach in one step are worse than the current tour x, then our RLS will never leave x.

Randomized Local Search for the TSP

- If all tours that we can reach in one step are worse than the current tour x, then our RLS will never leave x.
- There are many things that we can do.

Randomized Local Search for the TSP

- If all tours that we can reach in one step are worse than the current tour x, then our RLS will never leave x.
- There are many things that we can do, for example:
  1. We could restart the algorithm at a new starting point after some time.

Randomized Local Search for the TSP

- If all tours that we can reach in one step are worse than the current tour x, then our RLS will never leave x.
- There are many things that we can do, for example:
  1. We could restart the algorithm at a new starting point after some time.
  2. Or we could sometimes allow it swap more cities in the current tour.

Randomized Local Search with a Larger Neighborhood

15151 km

合肥, 重庆, 武汉, 长沙, 北京, 上海,
南京, 昆明, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

15195 km

合肥, 重庆, 武汉, 长沙, 北京, 上海, 南京, 昆明, 香港, 西安, 哈尔滨

合肥, 重庆, 武汉, 长沙, 北京, 南京, 上海, 昆明, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

15195 km

合肥, 重庆, 武汉, 长沙, 北京, 上海, 南京, 昆明, 香港, 西安, 哈尔滨

合肥, 重庆, 武汉, 长沙, 北京, 南京, 上海, 昆明, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

15151 km

合肥, 重庆, 武汉, 长沙, 北京, 上海,
南京, 昆明, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

14539 km

合肥, 重庆, 武汉, 长沙, 北京, 上海,
南京, 昆明, 香港, 西安, 哈尔滨

合肥, 南京, 武汉, 长沙, 北京, 上海,
昆明, 重庆, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

14539 km

合肥, 南京, 武汉, 长沙, 北京, 上海,
昆明, 重庆, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

16624 km

合肥, 南京, 武汉, 长沙, 北京, 上海,
昆明, 重庆, 香港, 西安, 哈尔滨

昆明, 南京, 武汉, 长沙, 北京, 上海,
合肥, 重庆, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

16624 km

合肥, 南京, 武汉, 长沙, 北京, 上海,
昆明, 重庆, 香港, 西安, 哈尔滨

昆明, 南京, 武汉, 长沙, 北京, 上海,
合肥, 重庆, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

14539 km

合肥, 南京, 武汉, 长沙, 北京, 上海,
昆明, 重庆, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

13882 km

合肥, 南京, 武汉, 长沙, 北京, 上海, 昆明, 重庆, 香港, 西安, 哈尔滨

合肥, 南京, 武汉, 长沙, 北京, 上海, 重庆, 昆明, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

13882 km

合肥, 南京, 武汉, 长沙, 北京, 上海,
重庆, 昆明, 香港, 西安, 哈尔滨

Randomized Local Search with a Larger Neighborhood

10688 km

合肥, 南京, 武汉, 长沙, 北京, 上海,
重庆, 昆明, 香港, 西安, 哈尔滨

合肥, 南京, 武汉, 哈尔滨, 北京, 西安,
重庆, 昆明, 香港, 长沙, 上海

Randomized Local Search with a Larger Neighborhood

10688 km

合肥, 南京, 武汉, 哈尔滨, 北京, 西安,
重庆, 昆明, 香港, 长沙, 上海

Randomized Local Search with a Larger Neighborhood

12111 km

合肥, 南京, 武汉, 哈尔滨, 北京, 西安, 重庆, 昆明, 香港, 长沙, 上海

合肥, 南京, 哈尔滨, 武汉, 北京, 西安, 重庆, 昆明, 香港, 长沙, 上海

Randomized Local Search with a Larger Neighborhood

Randomized Local Search with a Larger Neighborhood

9547 km

合肥, 南京, 武汉, 哈尔滨, 北京, 西安,
重庆, 昆明, 香港, 长沙, 上海

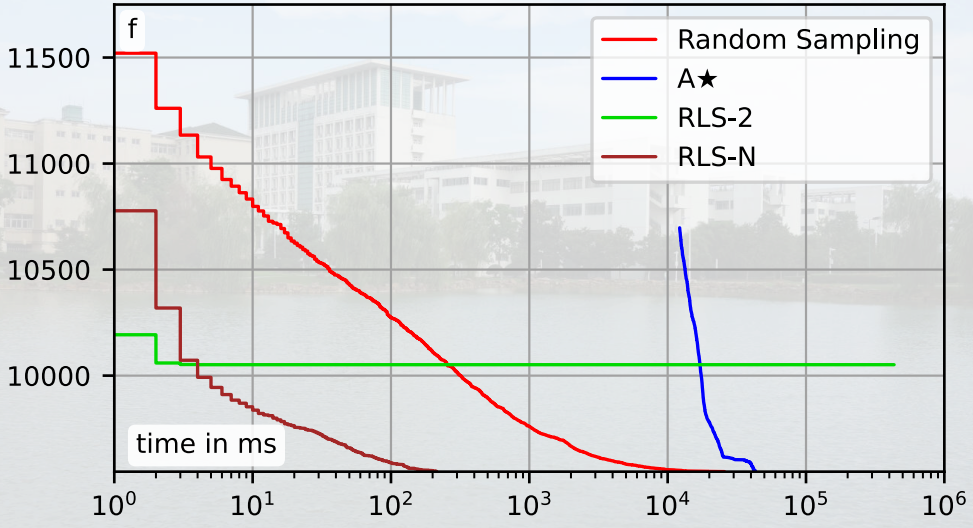
合肥, 南京, 上海, 哈尔滨, 北京, 西安,
重庆, 昆明, 香港, 长沙, 武汉

Randomized Local Search with a Larger Neighborhood

9547 km

合肥, 南京, 上海, 哈尔滨, 北京, 西安,
重庆, 昆明, 香港, 长沙, 武汉

Randomized Local Search with a Larger Neighborhood



Randomized Local Search with a Larger Neighborhood

- If all tours that we can reach in one step are worse than the current tour x, then our RLS will never leave x.
- There are many things that we can do, for example:
  1. We could restart the algorithm at a new starting point after some time.
  2. Or we could sometimes allow it swap more cities in the current tour.

Randomized Local Search with a Larger Neighborhood

- If all tours that we can reach in one step are worse than the current tour x, then our RLS will never leave x.
- There are many things that we can do, for example:
  1. We could restart the algorithm at a new starting point after some time.
  2. Or we could sometimes allow it swap more cities in the current tour.
- The larger neighborhood of solutions that can be reached in one step makes the search slower, but allows it to escape from local optima.

Principle 3: Larger changes can help us escape from local optima.

Principle 3: Larger changes can help us escape from local optima.

Problem 3: Larger changes are less likely to yield improvements (because the new solution is more different), so they slow down the search.

There are many more principles and problems surrounding (metaheuristic) optimization.

There are many more principles and problems surrounding (metaheuristic) optimization.

But we will leave it at what we have seen so far.

Summary

Summary on Algorithms

- We have learned some basic algorithmic principles.

Summary on Algorithms

- We have learned some basic algorithmic principles.
- Current research tries to improve both the speed of algorithms as well as the solution quality.

Summary on Algorithms

- We have learned some basic algorithmic principles.
- Current research tries to improve both the speed of algorithms as well as the solution quality.
- This requires carefully balancing the step-size of algorithms and to develop methods against getting stuck at local optima.

Summary on the TSP

- Today, we can actually solve TSPs with tens of thousands of cities to optimality[1,12].

Summary on the TSP

- Today, we can actually solve TSPs with tens of thousands of cities to optimality[1,12].
- For this, we actually use principles similar to the A* Algorithm, just a bit differently, in methods like branch and bound[1,13,34].

Summary on the TSP

- Today, we can actually solve TSPs with tens of thousands of cities to optimality[1,12].
- For this, we actually use principles similar to the A* Algorithm, just a bit differently, in methods like branch and bound[1,13,34].
- We can get close-to-optimal solutions for TSPs with millions of cities.

Summary on the TSP

- Today, we can actually solve TSPs with tens of thousands of cities to optimality[1,12].
- For this, we actually use principles similar to the A* Algorithm, just a bit differently, in methods like branch and bound[1,13,34].
- We can get close-to-optimal solutions for TSPs with millions of cities.
- For this, we use algorithms a bit similar to randomized local search (RLS), but with more targeted search steps, e.g., in the Lin-Kernighan-Helsgaun algorithm[24].

Summary on the TSP

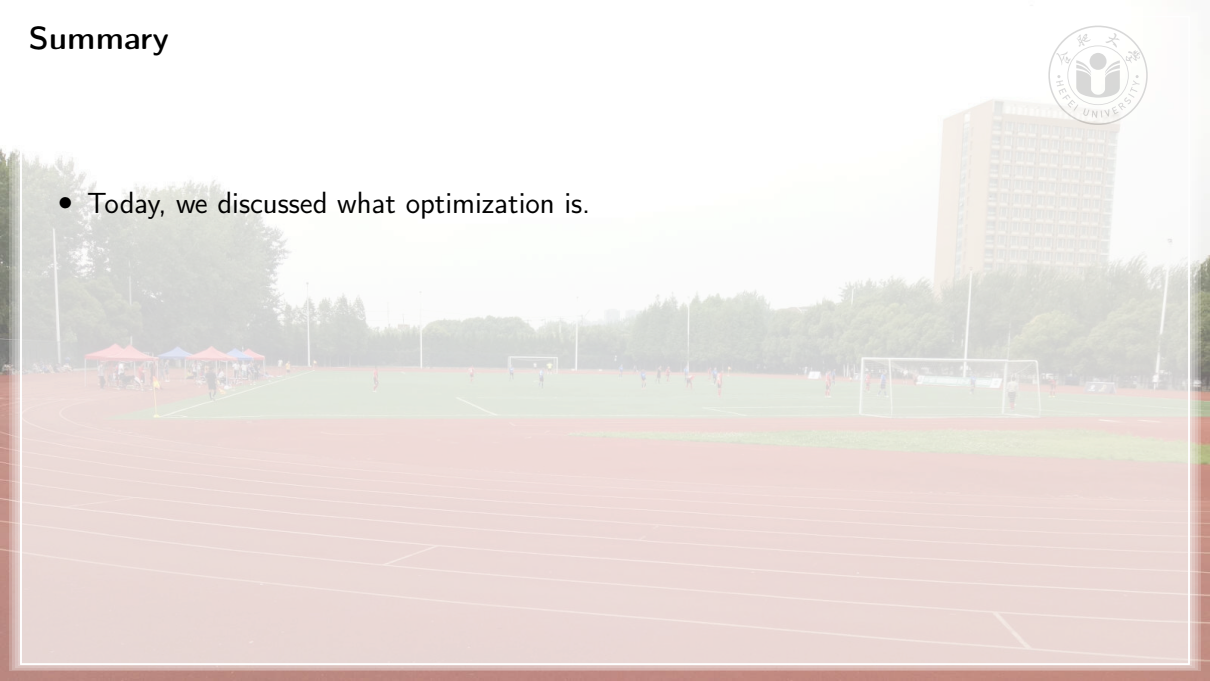
- Today, we can actually solve TSPs with tens of thousands of cities to optimality[1,12]. [but not all of them!]
- For this, we actually use principles similar to the A* Algorithm, just a bit differently, in methods like branch and bound[1,13,34].
- We can get close-to-optimal solutions for TSPs with millions of cities.

- For this, we use algorithms a bit similar to randomized local search (RLS), but with more targeted search steps, e.g., in the Lin-Kernighan-Helsgaun algorithm[24].

Summary on the TSP

- Today, we can actually solve TSPs with tens of thousands of cities to optimality[1,12]. [but not all of them!]
- For this, we actually use principles similar to the A* Algorithm, just a bit differently, in methods like branch and bound[1,13,34].
- We can get close-to-optimal solutions for TSPs with millions of cities. [close-to-optimal, but not optimal (at least not always)!]
- For this, we use algorithms a bit similar to randomized local search (RLS), but with more targeted search steps, e.g., in the Lin-Kernighan-Helsgaun algorithm[24].

Summary



- Today, we discussed what optimization is.

Summary

- Today, we discussed what optimization is.
- Today, you also saw some of the basic principles and methods that are inside of the algorithms used in optimization and operations research.

Summary

- Today, we discussed what optimization is.
- Today, you also saw some of the basic principles and methods that are inside of the algorithms used in optimization and operations research.
- You can imagine that the same principles that we tested on the TSP will work on many different kinds of problems.

Summary

- Today, we discussed what optimization is.
- Today, you also saw some of the basic principles and methods that are inside of the algorithms used in optimization and operations research.
- You can imagine that the same principles that we tested on the TSP will work on many different kinds of problems.
- For example: As long as we can randomly construct and randomly modify a solution, we can attack the problem with randomized local search (RLS).

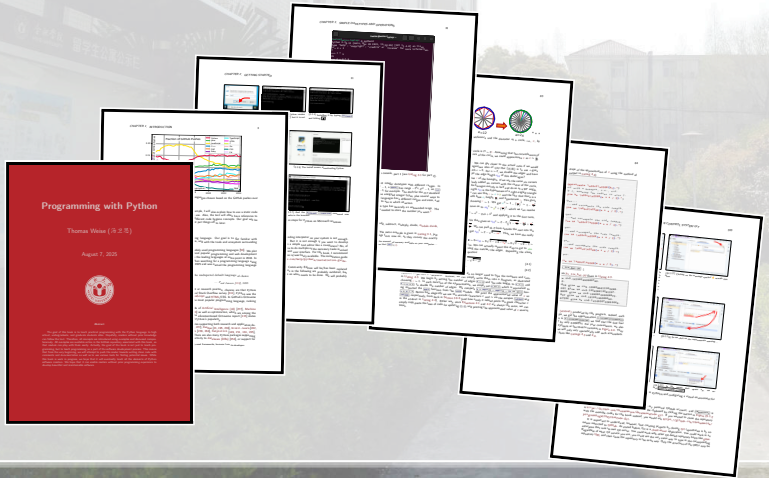
Summary

- Today, we discussed what optimization is.
- Today, you also saw some of the basic principles and methods that are inside of the algorithms used in optimization and operations research.
- You can imagine that the same principles that we tested on the TSP will work on many different kinds of problems.
- For example: As long as we can randomly construct and randomly modify a solution, we can attack the problem with randomized local search (RLS).
- Understanding the basic principles of optimization is not very hard.

Advertisement

Programming with Python

We have a freely available course book on *Programming with Python* at `https://thomasweise.github.io/programmingWithPython`, with focus on practical software development using the Python ecosystem of tools[46].

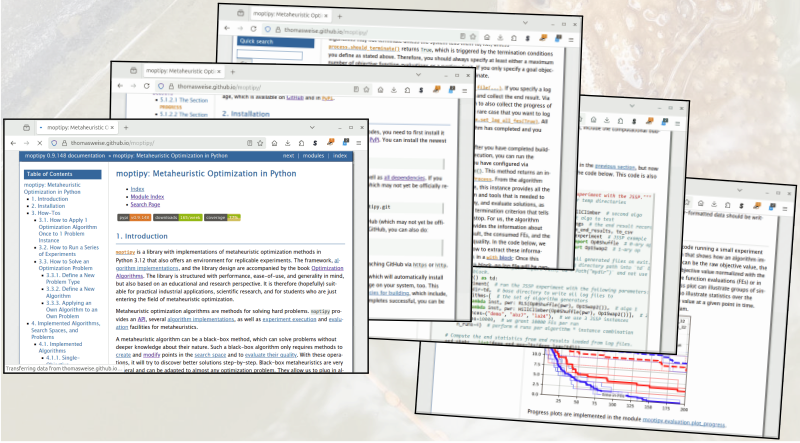


Databases

We have a freely available course book on *Databases* at `https://thomasweise.github.io/databases`, with actual practical examples using a real database management system (DBMS)[44].

Metaheuristic Optimization in Python: moptipy

We offer moptipy[52] a mature open source Python package for metaheuristic optimization, which implements several algorithms, can run self-documenting experiments in parallel and in a distributed fashion, and offers statistical evaluation tools.



谢谢您们！

Thank you!

Vielen Dank!

References I

[1] David Lee Applegate, Robert E. Bixby, Vašek Chvátal, and William John Cook. *The Traveling Salesman Problem: A Computational Study*. 2nd ed. Vol. 17 of Princeton Series in Applied Mathematics. Princeton, NJ, USA: Princeton University Press, 2007. ISBN: 978-0-691-12993-8 (cit. on pp. 169, 192–202, 322–327, 349).

[2] Thomas Bäck, David B. Fogel, and Zbigniew "Zbyszek" Michalewicz, eds. *Handbook of Evolutionary Computation*. Bristol, England, UK: IOP Publishing Ltd and Oxford, Oxfordshire, England, UK: Oxford University Press, 1997. ISBN: 978-0-7503-0392-7 (cit. on p. 347).

[3] Thomas Bartz-Beielstein, Carola Doerr, Daan van den Berg, Jakob Bossek, Sowmya Chandrasekaran, Tome Eftimov, Andreas Fischbach, Pascal Kerschke, William La Cava, Manuel López-Ibáñez, Katherine Mary Malan, Jason Hall Moore, Boris Naujoks, Patryk Orzechowski, Vanessa Volz, Markus Wagner, and Thomas Weise (汤卫思). "Benchmarking in Optimization: Best Practice and Open Issues". (abs/2007.03488), Dec. 18, 2020. doi:10.48550/arXiv.2007.03488. URL: https://arxiv.org/abs/2007.03488 (visited on 2025-07-25). arXiv:2007.03488v2 [cs.NE] 16 Dec 2020 (cit. on p. 348).

[4] Adam W. Bojanczyk and Richard P. Brent. "A Systolic Algorithm for Extended GCD Computation". *Computers & Mathematics with Applications* 14(4):233–238, 1987. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0898-1221 (cit. on pp. 70–76).

[5] Richard P. Brent. *Further Analysis of the Binary Euclidean Algorithm*. arXiv.org: Computing Research Repository (CoRR) abs/1303.2772. Ithaca, NY, USA: Cornell Universiy Library, Nov. 1999–Mar. 12, 2013. doi:10.48550/arXiv.1303.2772. URL: https://arxiv.org/abs/1303.2772 (visited on 2024-09-28). arXiv:1303.2772v1 [cs.DS] 12 Mar 2013. Report number PRG TR-7-99 of Oxford, Oxfordshire, England, UK: Oxford University Computing Laboratory, 11 1999, see https://maths-people.anu.edu.au/~brent/pd/rpb183tr.pdf (cit. on pp. 51–58, 70–76).

[6] Eduardo Carvalho Pinto and Carola Doerr. *Towards a More Practice-Aware Runtime Analysis of Evolutionary Algorithms*. arXiv.org: Computing Research Repository (CoRR) abs/1812.00493. Ithaca, NY, USA: Cornell Universiy Library, Dec. 3, 2018. doi:10.48550/arXiv.1812.00493. URL: https://arxiv.org/abs/1812.00493 (visited on 2025-08-08). arXiv:1812.00493v1 [cs.NE] 3 Dec 2018 (cit. on p. 347).

[7] Antonio Cavacini. "Is the CE/BCE notation becoming a standard in scholarly literature?" *Scientometrics* 102(2):1661–1668, July 2015. London, England, UK: Springer Nature Limited. ISSN: 0138-9130. doi:10.1007/s11192-014-1352-1 (cit. on pp. 347, 348).

References II

[8] Noureddine Chabini and Rachid Beguenane. "FPGA-Based Designs of the Factorial Function". In: *IEEE Canadian Conference on Electrical and Computer Engineering (CCECE'2022)*. Sept. 18–20, 2022, Halifax, NS, Canada. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022, pp. 16–20. ISBN: 978-1-6654-8432-9. doi:10.1109/CCECE49351.2022.9918302 (cit. on p. 349).

[9] Bo Chen, Chris N. Potts, and Gerhard J. Woeginger. "A Review of Machine Scheduling: Complexity, Algorithms and Approximability". In: *Handbook of Combinatorial Optimization*. Ed. by Panos Miltiades Pardalos, Ding-Zhu Du, and Ronald Lewis Graham. 1st ed. Boston, MA, USA: Springer, 1998, pp. 1493–1641. ISBN: 978-1-4613-7987-4. doi:10.1007/978-1-4613-0303-9_25. See also pages 21–169 in volume 3/3 by Norwell, MA, USA: Kluwer Academic Publishers. (Cit. on p. 349).

[10] Jiayang Chen (陈嘉阳), Zhize Wu (吴志泽), Sarah Louise Thomson, and Thomas Weise (汤卫思). "Frequency Fitness Assignment: Optimization Without Bias for Good Solution Outperforms Randomized Local Search on the Quadratic Assignment Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 27–37. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888600003837 (cit. on p. 348).

[11] Stephen Arthur Cook. "The Complexity of Theorem-Proving Procedures". In: *Third Annual ACM Symposium on Theory of Computing (STOC'1971)*. May 3–5, 1971, Shaker Heights, OH, USA. Ed. by Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman. New York, NY, USA: Association for Computing Machinery (ACM), 1971, pp. 151–158. ISBN: 978-1-4503-7464-4. doi:10.1145/800157.805047 (cit. on p. 349).

[12] William John Cook. *World TSP*. Waterloo, ON, Canada: University of Waterloo, 2013–Oct. 25, 2025. URL: http://www.math.uwaterloo.ca/tsp/world (visited on 2026-01-03) (cit. on pp. 322–327).

[13] William John Cook, Daniel G. Espinoza, and Marcos Goycoolea. *Computing with Domino-Parity Inequalities for the TSP*. Tech. rep. Atlanta, GA, USA: Georgia Institute of TechnologyIndustrial and Systems Engineering, 2005. URL: https://www.math.uwaterloo.ca/~bico/papers/DP_paper.pdf (visited on 2026-01-03) (cit. on pp. 322–327).

[14] Christopher Cullen. "Learning from Liu Hui? A Different Way to Do Mathematics". *Notices of the American Mathematical Society* 49(7):783–790, Aug. 2002. Providence, RI, USA: American Mathematical Society (AMS). ISSN: 1088-9477. URL: https://www.ams.org/notices/200207/comm-cullen.pdf (visited on 2024-08-09) (cit. on pp. 70–76).

References III

[15] Joseph W. Dauben. "Archimedes and Liu Hui on Circles and Spheres". *Ontology Studies (Cuadernos de Ontología)* 10:21–38, 2010. Leioa, Bizkaia, Spain: Universidad del País Vasco / Euskal Herriko Unibertsitatea. ISSN: 1576-2270. URL: https://ddd.uab.cat/pub/ontstu/15762270n10/15762270n10p21.pdf (visited on 2024-08-10) (cit. on pp. 70–76).

[16] *Definition of Operations Research*. University of Western Ontario, London, ON, Canada: International Federation of Operational Research Societies (IFORS), 2020. URL: https://www.ifors.org/what-is-or (visited on 2026-01-01) (cit. on p. 348).

[17] Stefan Droste, Thomas Jansen, and Ingo Wegener. "On the Analysis of the $(1 + 1)$ Evolutionary Algorithm". *Theoretical Computer Science* 276(1-2):51–81, Apr. 2002. Amsterdam, The Netherlands: Elsevier B.V. ISSN: 0304-3975. doi:10.1016/S0304-3975(01)00182-7 (cit. on p. 347).

[18] Jacques Dutka. "The Early History of the Factorial Function". *Archive for History of Exact Sciences* 43(3):225–249, Sept. 1991. Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany. ISSN: 0003-9519. doi:10.1007/BF00389433. Communicated by Umberto Bottazzini (cit. on p. 349).

[19] Euclid of Alexandria (*Εὐκλείδης*). *Euclid's Elements of Geometry (Στοιχεῖα). The Greek Text of J.L. Heiberg (1883-1885) from Euclidis Elementa, Edidit et Latine Interpretatus est I.L. Heiberg in Aedibus B. G. Teubneri, 1883-1885. Edited, and provided with a modern English translation, by Richard Fitzpatrick*. Vol. 7. *Elementary Number Theory*. Ed. by Richard Fitzpatrick. Trans. by Johan Ludvig Heiberg. revised and corrected. Austin, TX, USA: The University of Texas at Austin, 2008. ISBN: 978-0-615-17984-1. URL: https://farside.ph.utexas.edu/Books/Euclid/Elements.pdf (visited on 2024-09-30) (cit. on pp. 51–58).

[20] Toru Fujita, Koji Nakano, and Yasuaki Ito. "Bulk Execution of Euclidean Algorithms on the CUDA-Enabled GPU". *International Journal of Networking and Computing (IJNC)* 6(1):42–63, Jan. 2016. Higashi-Hiroshima, Japan: Department of Information Engineering, Hiroshima University. ISSN: 2185-2839. URL: http://www.ijnc.org (visited on 2024-09-28) (cit. on pp. 51–58).

[21] Michael T. Goodrich. *A Gentle Introduction to NP-Completeness*. Irvine, CA, USA: University of California, Irvine, Apr. 2022. URL: https://ics.uci.edu/~goodrich/teach/cs165/notes/NPComplete.pdf (visited on 2025-08-01) (cit. on p. 349).

[22] Gregory Z. Gutin and Abraham P. Punnen, eds. *The Traveling Salesman Problem and its Variations*. Vol. 12 of Combinatorial Optimization (COOP). New York, NY, USA: Springer New York, May 2002. ISSN: 1388-3011. doi:10.1007/b101971 (cit. on pp. 192–202, 349).

References IV

[23] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". *IEEE Transactions on Systems Science and Cybernetics* 4:100–107, July 31, 1968. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 0536-1567. doi:10.1109/TSSC.1968.300136 (cit. on pp. 8–14, 155–160, 347).

[24] Keld Helsgaun. "General k-opt Submoves for the Lin–Kernighan TSP Heuristic". *Mathematical Programming Computation* 1(2-3):119–163, July 2009. London, England, UK: Springer Nature Limited. ISSN: 1867-2949. doi:10.1007/s12532-009-0004-6 (cit. on pp. 322–327).

[25] ."Why is the Complexity of A* Exponential in Memory?" In: *Stack Overflow*. Ed. by Paul. New York, NY, USA: Stack Exchange Inc., Nov. 11, 2009–July 17, 2013. URL: https://stackoverflow.com/questions/1715401 (visited on 2026-01-03) (cit. on pp. 226–235).

[26] John Hunt. *A Beginners Guide to Python 3 Programming*. 2nd ed. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2023. ISBN: 978-3-031-35121-1. doi:10.1007/978-3-031-35122-8 (cit. on p. 348).

[27] Shen Kangshen, John Newsome Crossley, and Anthony W.-C. Lun. *The Nine Chapters on the Mathematical Art: Companion and Commentary*. Oxford, Oxfordshire, England, UK: Oxford University Press, Oct. 7, 1999. ISBN: 978-0-19-853936-0. doi:10.1093/oso/9780198539360.001.0001 (cit. on pp. 70–76).

[28] Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. "Sequencing and Scheduling: Algorithms and Complexity". In: *Production Planning and Inventory*. Ed. by Stephen C. Graves, Alexander Hendrik George Rinnooy Kan, and Paul H. Zipkin. Vol. IV of Handbooks of Operations Research and Management Science. Amsterdam, The Netherlands: Elsevier B.V., 1993. Chap. 9, pp. 445–522. ISSN: 0927-0507. ISBN: 978-0-444-87472-6. doi:10.1016/S0927-0507(05)80189-6. URL: http://alexandria.tue.nl/repository/books/339776.pdf (visited on 2023-12-06) (cit. on p. 349).

[29] Eugene Leighton Lawler, Jan Karel Lenstra, Alexander Hendrik George Rinnooy Kan, and David B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. Estimation, Simulation, and Control – Wiley-Interscience Series in Discrete Mathematics and Optimization. Chichester, West Sussex, England, UK: Wiley Interscience, Sept. 1985. ISSN: 0277-2698. ISBN: 978-0-471-90413-7 (cit. on pp. 169, 192–202, 349).

[30] Kent D. Lee and Steve Hubbard. *Data Structures and Algorithms with Python*. Undergraduate Topics in Computer Science (UTICS). Cham, Switzerland: Springer, 2015. ISBN: 978-3-319-13071-2. doi:10.1007/978-3-319-13072-9 (cit. on p. 348).

References V

[31] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Jörg Lässig, Daan van den Berg, Sarah Louise Thomson, and Thomas Weise (汤卫思). "Addressing the Traveling Salesperson Problem with Frequency Fitness Assignment and Hybrid Algorithms". *Soft Computing* 28(17-18):9495–9508, July 2024. London, England, UK: Springer Nature Limited. ISSN: 1432-7643. doi:10.1007/S00500-024-09718-8 (cit. on pp. 192–202, 348, 349).

[32] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Jörg Lässig, Daan van den Berg, and Thomas Weise (汤卫思). "Solving the Traveling Salesperson Problem using Frequency Fitness Assignment". In: *IEEE Symposium Series on Computational Intelligence (SSCI'2022)*. Dec. 4–7, 2022, Singapore. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE), 2022. ISBN: 978-1-6654-8769-6. doi:10.1109/SSCI51031.2022.10022296 (cit. on p. 348).

[33] Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Matthias Thürer, Markus Wagner, and Thomas Weise (汤卫思). "Generating Small Instances with Interesting Features for the Traveling Salesperson Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 173–180. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888800003837 (cit. on p. 348).

[34] John D. C. Little, Katta G. Murty, Dura W. Sweeny, and Caroline Karel. *An Algorithm for the Traveling Salesman Problem*. Sloan Working Papers 07-63. Massachusetts Institute of Technology (MIT): Massachusetts Institute of Technology (MIT), Sloan School of Management, Mar. 1963. URL: http://dspace.mit.edu/bitstream/handle/1721.1/46828/algorithmfortrav00litt.pdf (visited on 2026-01-03) (cit. on pp. 322–327).

[35] Peter Luschny. *A New Kind of Factorial Function*. Highland Park, NJ, USA: The OEIS Foundation Inc., Oct. 4, 2015. URL: https://oeis.org/A000142/a000142.pdf (visited on 2024-09-29) (cit. on p. 349).

[36] Mark Lutz. *Learning Python*. 6th ed. Sebastopol, CA, USA: O'Reilly Media, Inc., Mar. 2025. ISBN: 978-1-0981-7130-8 (cit. on p. 348).

[37] Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe, eds. *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0000195000003837.

[38] John J. O'Connor and Edmund F. Robertson. *Liu Hui*. St Andrews, Scotland, UK: University of St Andrews, School of Mathematics and Statistics, Dec. 2003. URL: https://mathshistory.st-andrews.ac.uk/Biographies/Liu_Hui (visited on 2024-08-10) (cit. on pp. 70–76).

References VI

[39] Amit Patel. *Amit's A* Pages*. Stanford, CA, USA: Stanford University, 1997–2025. URL: https://theory.stanford.edu/~amitp/GameProgramming (visited on 2026-01-02) (cit. on pp. 8–14, 155–160, 347).

[40] Alexander Podlich, Thomas Weise (汤卫思), Manfred Menze, and Christian Gorldt. "Intelligente Wechselbrückensteuerung für die Logistik von Morgen". *Electronic Communications of the EASST (ECEASST)* 17(Communication in Distributed Systems), Feb. 27, 2009. The Netherlands: European Association of Software Science and Technology (EASST). ISSN: 1863-2122. doi:10.14279/tuj.eceasst.17.205. In the proceedings of *Workshops der Wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen 2009* (WowKiVS 2009), 3 6, 2009, Kassel, Hessen, Germany (cit. on pp. 8–14).

[41] Syamal K. Sen and Ravi P. Agarwal. "Existence of year zero in astronomical counting is advantageous and preserves compatibility with significance of AD, BC, CE, and BCE". In: *Zero – A Landmark Discovery, the Dreadful Void, and the Ultimate Mind*. Amsterdam, The Netherlands: Elsevier B.V., 2016. Chap. 5.5, pp. 94–95. ISBN: 978-0-08-100774-7. doi:10.1016/C2015-0-02299-7 (cit. on pp. 347, 348).

[42] Philip D. Straffin Jr. "Liu Hui and the First Golden Age of Chinese Mathematics". *Mathematics Magazine* 71(3):163–181, June 1998. London, England, UK: Taylor and Francis Ltd. ISSN: 0025-570X. doi:10.2307/2691200. URL: https://www.researchgate.net/publication/237334342 (visited on 2024-08-10) (cit. on pp. 70–76).

[43] Sarah Louise Thomson, Gabriela Ochoa, Daan van den Berg, Tianyu Liang (梁天宇), and Thomas Weise (汤卫思). "Entropy, Search Trajectories, and Explainability for Frequency Fitness Assignment". In: *Parallel Problem Solving from Nature (PPSN XVIII)*. Vol. 1. Sept. 14–18, 2024, Hagenberg, Mühlkreis, Austria. Ed. by Michael Affenzeller, Stephan M. Winkler, Anna V. Kononova, Heike Trautmann, Tea Tušar, Penousal Machado, and Thomas Bäck. Vol. 15148 of Lecture Notes in Computer Science (LNCS). Cham, Switzerland: Springer. ISSN: 0302-9743. ISBN: 978-3-031-70054-5. doi:10.1007/978-3-031-70055-2_23 (cit. on p. 348).

[44] Thomas Weise (汤卫思). *Databases*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2025. URL: https://thomasweise.github.io/databases (visited on 2025-01-05) (cit. on pp. 335, 348).

[45] Thomas Weise (汤卫思). *Global Optimization Algorithms – Theory and Application*. self-published, 2009. URL: https://www.researchgate.net/publication/200622167 (visited on 2025-07-25) (cit. on p. 347).

References VII

[46] Thomas Weise (汤卫思). *Programming with Python*. Hefei, Anhui, China (中国安徽省合肥市): Hefei University (合肥大学), School of Artificial Intelligence and Big Data (人工智能与大数据学院), Institute of Applied Optimization (应用优化研究所, IAO), 2024–2025. URL: https://thomasweise.github.io/programmingWithPython (visited on 2025-01-05) (cit. on pp. 334, 348).

[47] Thomas Weise (汤卫思), Raymond Chiong, Jörg Lässig, Ke Tang (唐珂), Shigeyoshi Tsutsui, Wenxiang Chen (陈文祥), Zbigniew "Zbyszek" Michalewicz, and Xin Yao (姚新). "Benchmarking Optimization Algorithms: An Open Source Framework for the Traveling Salesman Problem". *IEEE Computational Intelligence Magazine (CIM)* 9(3):40–52, Aug. 2014. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers (IEEE). ISSN: 1556-603X. doi:10.1109/MCI.2014.2326101 (cit. on pp. 192–202, 348, 349).

[48] Thomas Weise (汤卫思), Li Niu (牛力), and Ke Tang (唐珂). "AOAB – Automated Optimization Algorithm Benchmarking". In: *12th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO'2010)*. July 7–11, 2010, Portland, OR, USA. Ed. by Martin Pelikan and Jürgen Branke. New York, NY, USA: Association for Computing Machinery (ACM), 2010, pp. 1479–1486. ISBN: 978-1-4503-0073-5. doi:10.1145/1830761.1830763 (cit. on p. 348).

[49] Thomas Weise (汤卫思), Alexander Podlich, and Christian Gorldt. "Solving Real-World Vehicle Routing Problems with Evolutionary Algorithms". In: *Natural Intelligence for Scheduling, Planning and Packing Problems*. Ed. by Raymond Chiong and Sandeep Dhakal. Vol. 250 of Studies in Computational Intelligence (SCI). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, Sept. 2009. Chap. 2, pp. 29–53. ISSN: 1860-949X. ISBN: 978-3-642-04038-2. doi:10.1007/978-3-642-04039-9_2 (cit. on pp. 8–14).

[50] Thomas Weise (汤卫思), Alexander Podlich, Manfred Menze, and Christian Gorldt. "Optimierte Güterverkehrsplanung mit Evolutionären Algorithmen". *Industrie Management – Zeitschrift für industrielle Geschäftsprozesse* 10(3):37–40, June 2009. Berlin, Germany: GITO mbH Verlag (cit. on pp. 8–14).

[51] Thomas Weise (汤卫思), Alexander Podlich, Kai Reinhard, Christian Gorldt, and Kurt Geihs. "Evolutionary Freight Transportation Planning". In: *Applications of Evolutionary Computing (EvoWorkshops 2009): Proceedings of EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG*. Apr. 15–17, 2009, Tübingen, Baden-Württemberg, Germany. Ed. by Mario Giacobini, Anthony Brabazon, Stefano Cagnoni, Gianni A. Caro, Anikó Ekárt, Anna Isabel Esparcia-Alcázar, Muddassar Farooq, Andreas Fink, and Penousal Machado. Vol. 5484 of Theoretical Computer Science and General Issues (LNTCS), sub-series of Lecture Notes in Computer Science (LNCS). Berlin/Heidelberg, Germany: Springer-Verlag GmbH Germany, Apr. 2009, pp. 768–777. ISSN: 2512-2010. doi:10.1007/978-3-642-01129-0_87 (cit. on pp. 8–14).

References VIII

[52] Thomas Weise (汤卫思) and Zhize Wu (吴志泽). "Replicable Self-Documenting Experiments with Arbitrary Search Spaces and Algorithms". In: *Conference on Genetic and Evolutionary Computation (GECCO'2023), Companion Volume*. July 15–19, 2023, Lisbon, Portugal. Ed. by Sara Silva and Luís Paquete. New York, NY, USA: Association for Computing Machinery (ACM), 2023, pp. 1891–1899. ISBN: 979-8-4007-0120-7. doi:10.1145/3583133.3596306 (cit. on pp. 336, 348).

[53] Thomas Weise (汤卫思), Yuezhong Wu (吴越钟), Raymond Chiong, Ke Tang (唐珂), and Jörg Lässig. "Global versus Local Search: The Impact of Population Sizes on Evolutionary Algorithm Performance". *Journal of Global Optimization* 66(3):511–534, Feb. 2016. London, England, UK: Springer Nature Limited. ISSN: 0925-5001. doi:10.1007/s10898-016-0417-5 (cit. on pp. 192–202, 349).

[54] CAO Xiang (曹翔), Zhize Wu (吴志泽), Daan van den Berg, and Thomas Weise (汤卫思). "Randomized Local Search vs. NSGA-II vs. Frequency Fitness Assignment on The Traveling Tournament Problem". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 38–49. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012891500003837 (cit. on p. 348).

[55] Kinza Yasar and Craig S. Mullins. *Definition: Database Management System (DBMS)*. Newton, MA, USA: TechTarget, Inc., June 2024. URL: https://www.techtarget.com/searchdatamanagement/definition/database-management-system (visited on 2025-01-11) (cit. on p. 348).

[56] Rui Zhao (赵睿), Tianyu Liang (梁天宇), Zhize Wu (吴志泽), Daan van den Berg, Matthias Thürer, and Thomas Weise (汤卫思). "Randomized Local Search on the 2D Rectangular Bin Packing Problem with Item Rotation". In: *Genetic and Evolutionary Computation Conference (GECCO'2024)*. July 14–18, 2024, Melbourne, VIC, Australia. Ed. by Xiaodong Li and Julia Handl. New York, NY, USA: Association for Computing Machinery (ACM), 2024, pp. 235–238. ISBN: 979-8-4007-0494-9. doi:10.1145/3638530.3654139 (cit. on pp. 8–14, 166–168, 348).

[57] Rui Zhao (赵睿), Zhize Wu (吴志泽), Daan van den Berg, Matthias Thürer, Tianyu Liang (梁天宇), Ming Tan (檀明), and Thomas Weise (汤卫思). "Randomized Local Search for Two-Dimensional Bin Packing and a Negative Result for Frequency Fitness Assignment". In: *16th International Joint Conference on Computational Intelligence (IJCCI'24)*. Nov. 20–22, 2024, Porto, Portugal. Ed. by Francesco Marcelloni, Kurosh Madani, Niki van Stein, and Joaquim Filipe. Porto, Portugal: SciTePress: Science and Technology Publications, Lda, 2024, pp. 15–26. ISSN: 2184-3236. ISBN: 978-989-758-721-4. doi:10.5220/0012888500003837 (cit. on pp. 8–14, 166–168, 348).

References IX

[58] Nicola Abdo Ziadeh, Michael B. Rowton, A. Geoffrey Woodhead, Wolfgang Helck, Jean L.A. Filliozat, Hiroyuki Momo, Eric Thompson, E.J. Wiesenberg, and Shih-ch'ang Wu. "Chronology – Christian History, Dates, Events". In: *Encyclopaedia Britannica*. Ed. by The Editors of Encyclopaedia Britannica. Chicago, IL, USA: Encyclopædia Britannica, Inc., July 26, 1999–Mar. 20, 2024. URL: https://www.britannica.com/topic/chronology/Christian (visited on 2025-08-27) (cit. on pp. 347, 348).

Glossary I

$(1+1)$ EA The $(1+1)$ EA is a local search algorithm that retains the best solution x_c discovered so far during the search[6,17]. In each step, it applies a unary search operator to this best-so-far solution x_c and derives a new solution x_n. If the new solution x_n is *better or equally good* when compared with x_c, i.e., not worse, then it replaces it, i.e., is stored as the new x_c. If the search space are bit strings of length n, then the $(1+1)$ EA uses a unary search operator that flips each bit independently with probability m/n, where usually $m = 1$. This operator is the main difference to randomized local search (RLS). The $(1+1)$ EA is a special case of the $(\mu+\lambda)$ evolutionary algorithm ($(\mu+\lambda)$ EA) where $\mu = \lambda = 1$.

EA An *evolutionary algorithm* is a metaheuristic optimization method that maintains a population of candidate solutions, which undergo selection (where better solutions are chosen with higher probability) and reproduction (where mutation and recombination create a new candidate solution from one or two existing ones, repectively)[2,45].

$(\mu+\lambda)$ EA The $(\mu+\lambda)$ EA is an evolutionary algorithm (EA) where, in each generation, λ offspring solutions are generated from the current population of μ parent solutions. The offspring and parent populations are merged, yielding $\mu+\lambda$ solutions, from which then the best μ solutions are ratained to form the parent population of the next generation. If the search space is the bit strings of length n, then this algorithm usually applies a mutation operator flipping each bit independently with probability $1/n$.

A* Algorithm The A* Algorithm is an a greedy best-first-first search for finding the shortest path between two locations[23,39]. The algorithm iteratively constructs the shortest path. It maintains a list of current candidate paths. For choosing the next candidate path whose end node should be expanded, it computes a value $f(p)$ of these paths p. $f(p)$ is the sum of the cost $g(p)$ so far incurred by the path as well as a heuristic $h(p)$ predicting the cost from the current end of the path to the goal. $h(p)$ must never overestimate that cost.

BCE The time notation *before Common Era* is a non-religious but chronological equivalent alternative to the traditional *Before Christ (BC)* notation, which refers to the years *before* the birth of Jesus Christ[7]. The years BCE are counted down, i.e., the larger the year, the farther in the past. The year 1 BCE comes directly before the year 1 CE[41,58].

Glossary II

CE The time notation *Common Era* is a non-religious but chronological equivalent alternative to the traditional *Anno Domini (AD)* notation, which refers to the years *after* the birth of Jesus Christ[7]. The years CE are counted upwards, i.e., the smaller they are, the farther they are in the past. The year 1 CE comes directly after the year 1 BCE[41,58].

DB A *database* is an organized collection of structured information or data, typically stored electronically in a computer system. Databases are discussed in our book *Databases*[44].

DBMS A *database management system* is the software layer located between the user or application and the database (DB). The DBMS allows the user/application to create, read, write, update, delete, and otherwise manipulate the data in the DB[55].

FE *Objective function evaluations* are an implementation-independent measure of runtime for optimization algorithms[3,47,48,52]. 1 FE equals to one evaluated candidate solution during the optimization process.

moptipy is the *Metaheuristic Optimization in Python* library[52]. It has been used in several different research works, including[10,31–33,43,54,56,57]. Learn more at https://thomasweise.github.io/moptipy and https://thomasweise.github.io/moptipyapps.

OR Operations Research (or Operational Research) is the application of sciences such as mathematics and computer science to the management and organization of systems, organizations, enterprises, factories, or projects. It encompasses the development and application of problem-solving methods and techniques (such as mathematical optimization, simulation, queueing theory and other stochastic models) with the goal to improve decision-making and efficiency[16].

Python The Python programming language[26,30,36,46], i.e., what you will learn about in our book[46]. Learn more at https://python.org.

Glossary III

RLS Randomized local search retains the best solution x_c discovered so far during the search and, in each step, it applies a unary search operator to this best-so-far solution x_c and derives a new solution x_n. If the new solution x_n is *better or equally good* when compared with x_c, i.e., not worse, then it replaces it, i.e., is stored as the new x_c. If the search space are bit strings of length n, then RLS uses a unary search operator that flips exactly one bit. This operator is the main difference to $(1+1)$ evolutionary algorithm ($(1+1)$ EA).

TSP In an instance of the *Traveling Salesperson Problem*, also known as *Traveling Salesman Problem*, a set of n cities or locations as well as the distances between them are defined[1,22,29,31,47,53]. The goal is to find the shortest round-trip tour that starts at one city, visits all the other cities one time each, and returns to the origin. The TSP is one of the most well-known $\mathcal{NP}$-hard combinatorial optimization problems[22].

$i!$ The factorial $a!$ of a natural number $a \in \mathbb{N}_1$ is the product of all positive natural numbers less than or equal to a, i.e., $a! = 1 * 2 * 3 * 4 * \cdots * (a-1) * a$[8,18,35].

$\mathbb{N}_1$ the set of the natural numbers *excluding* 0, i.e., 1, 2, 3, 4, and so on. It holds that $\mathbb{N}_1 \subset \mathbb{Z}$.

$\mathcal{NP}$ is the class of computational problems that can be solved in polynomial time by a non-deterministic machine and can be verified in polynomial time by a deterministic machine (such as a normal computer)[21].

$\mathcal{NP}$-hard Algorithms that guarantee to find the correct solutions of $\mathcal{NP}$-hard problems[9,11,28] need a runtime that is exponential in the problem scale in the worst case. A problem is $\mathcal{NP}$-hard if all problems in $\mathcal{NP}$ are reducible to it in polynomial time[21].

$\mathbb{R}$ the set of the real numbers.

$\mathbb{Z}$ the set of the integers numbers including positive and negative numbers and 0, i.e., . . . , -3, -2, -1, 0, 1, 2, 3, . . . , and so on. It holds that $\mathbb{Z} \subset \mathbb{R}$.