The Travelling Salesman Problem

October 2, 2020

The Travelling Salesman Problem

Imagine that you're a salesperson. You start from home. Along the way, you have to make a number of stops before you end up back home. However, the order in which you make the stops matters. The problem of finding the optimal order here is what's known as **the travelling salesman problem**.

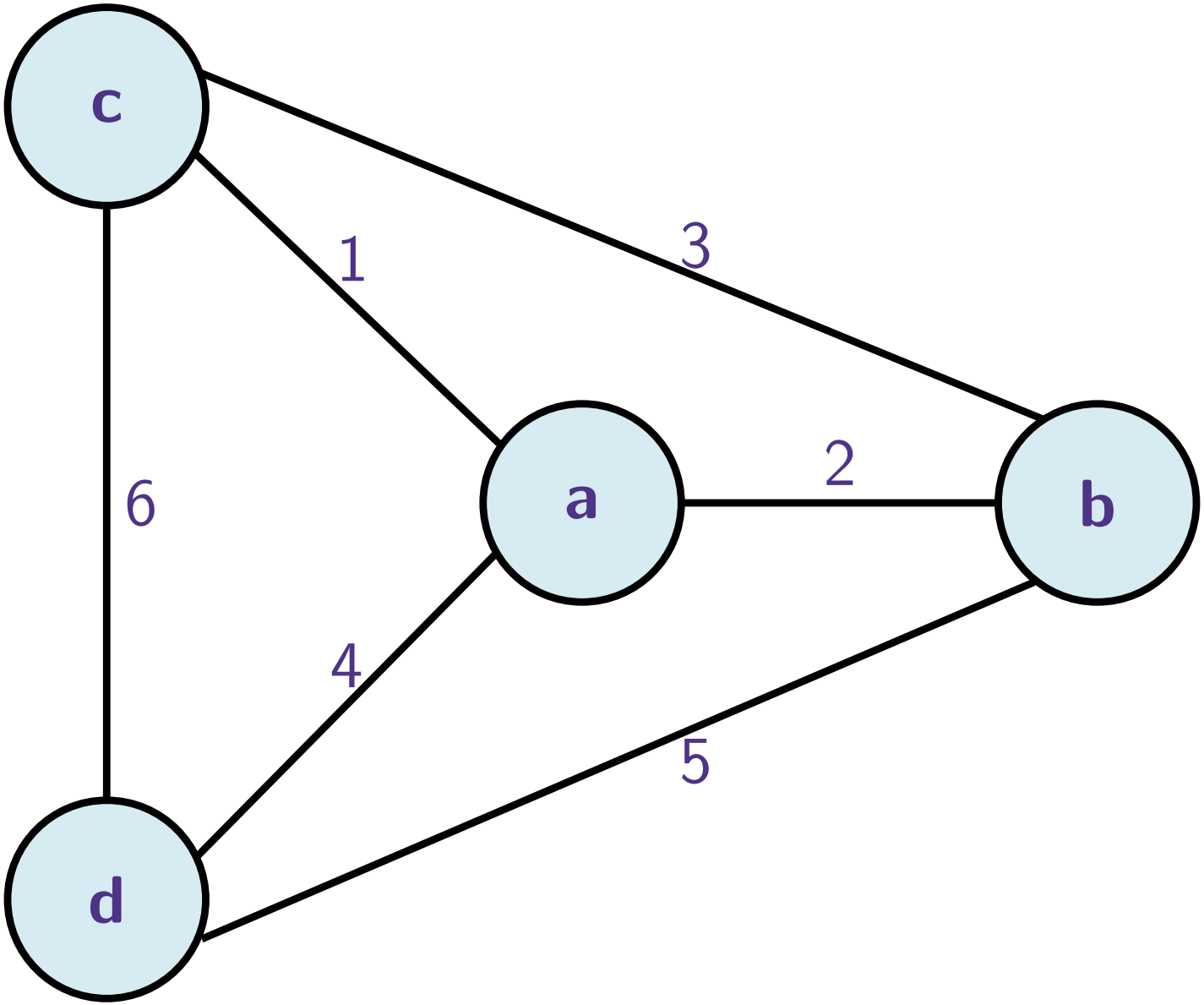
Problem Definition

- **Input** : A complete undirected graph with non-negative edge costs.
- **Output** : A minimum cost tour i.e. a cycle that visits all the vertices exactly once.

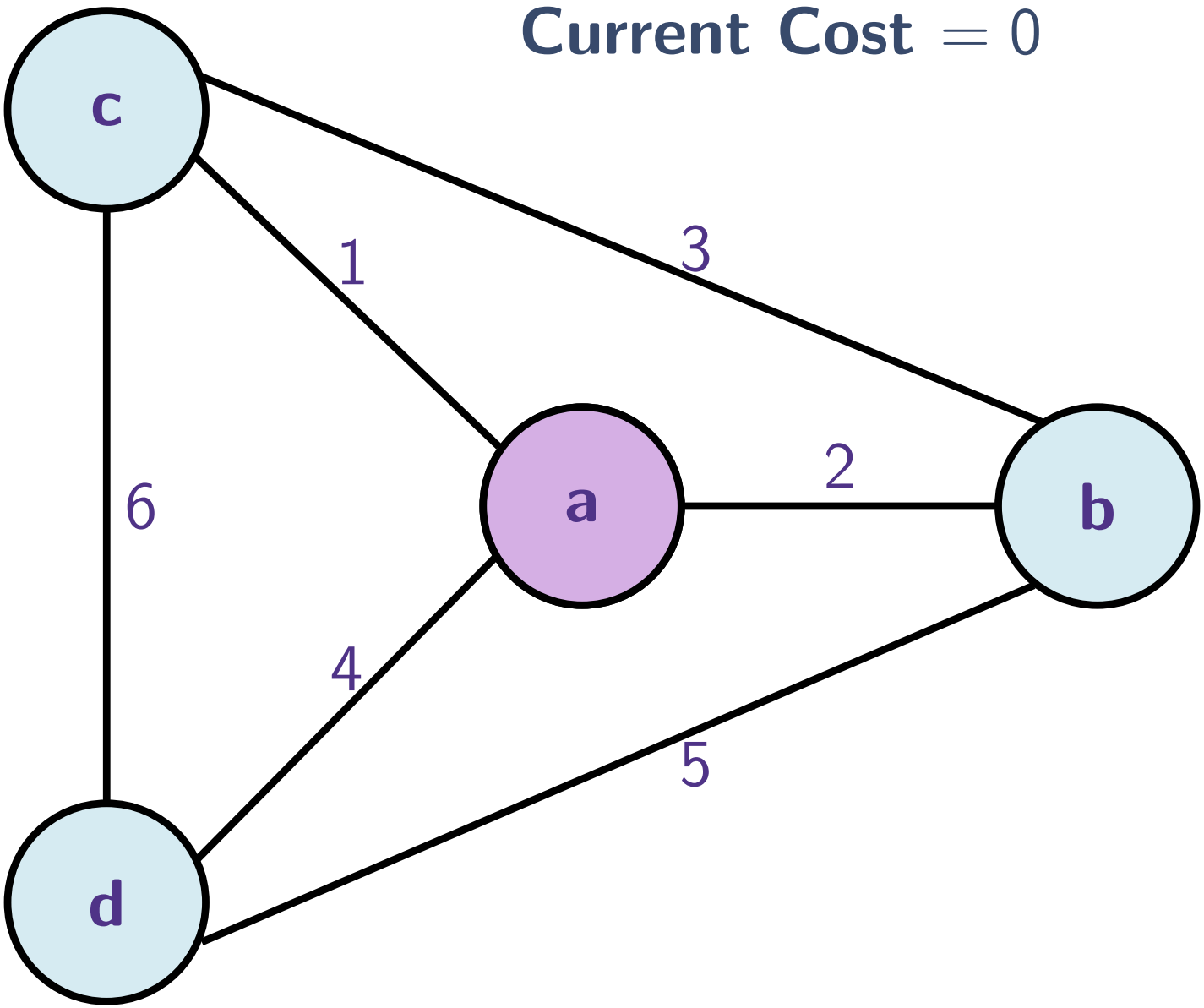
Problem Definition

- **Input** : A complete undirected graph with non-negative edge costs.
- **Output** : A minimum cost tour i.e. a cycle that visits all the vertices exactly once.

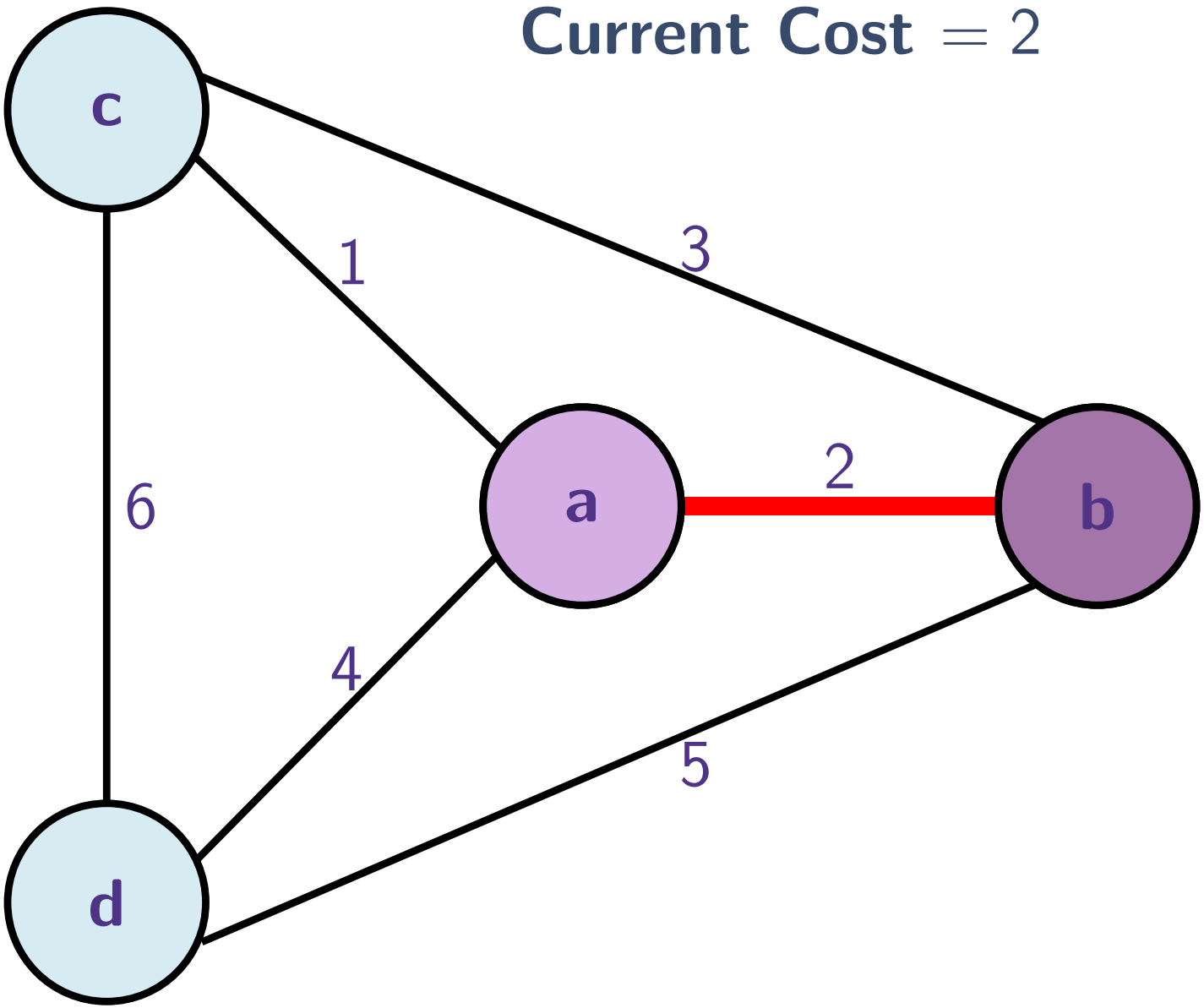
Naive Algorithm



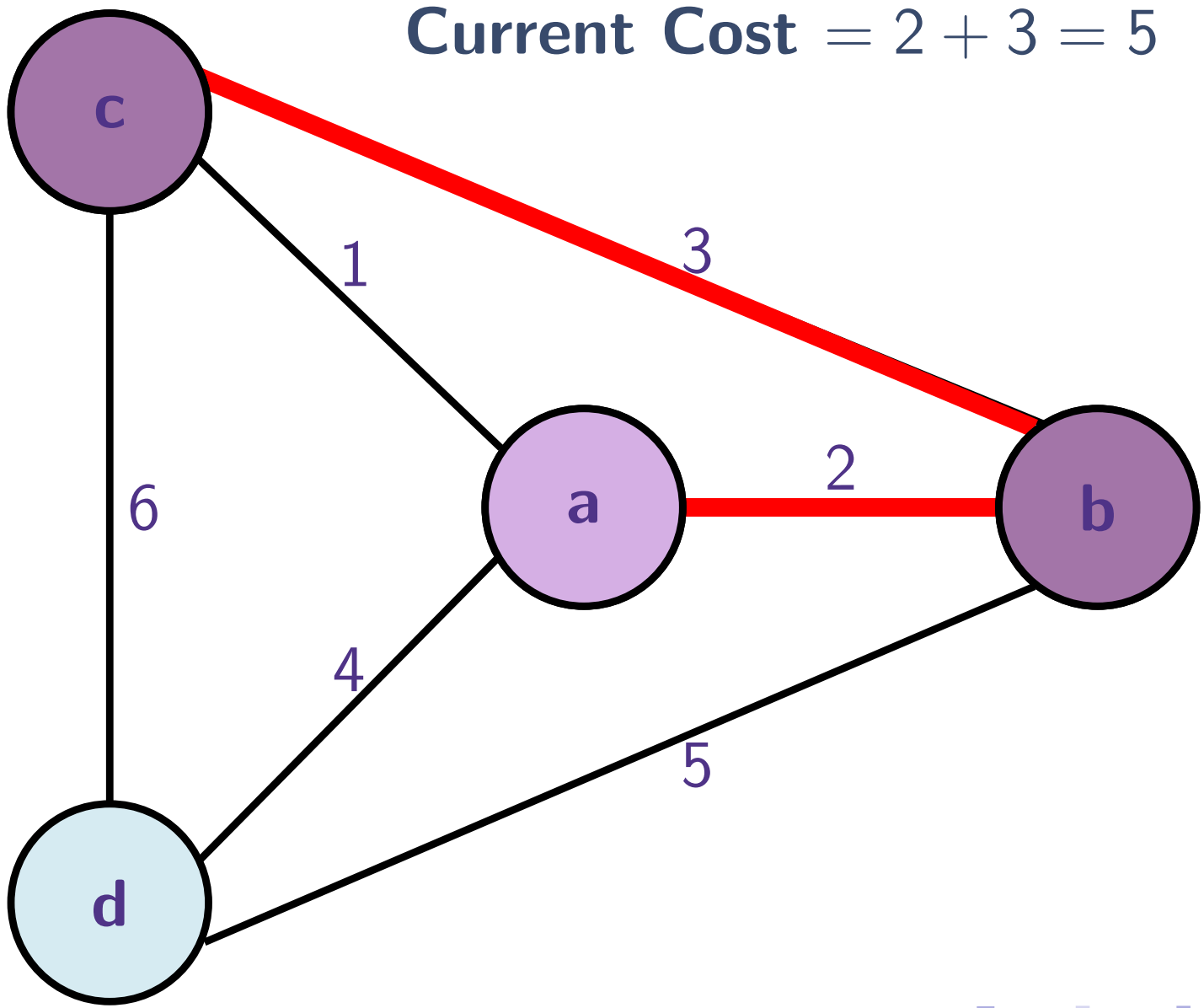
Naive Algorithm



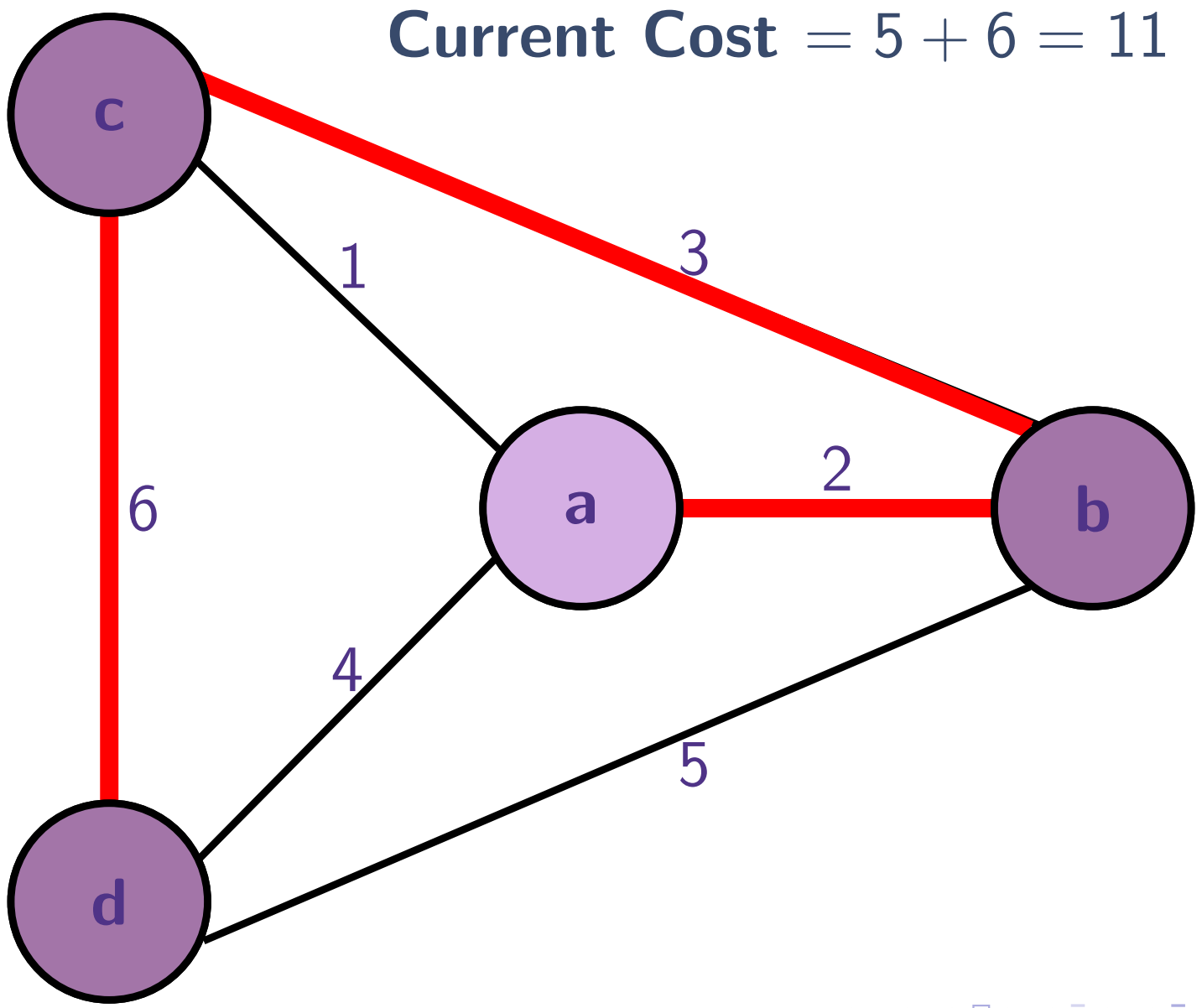
Naive Algorithm



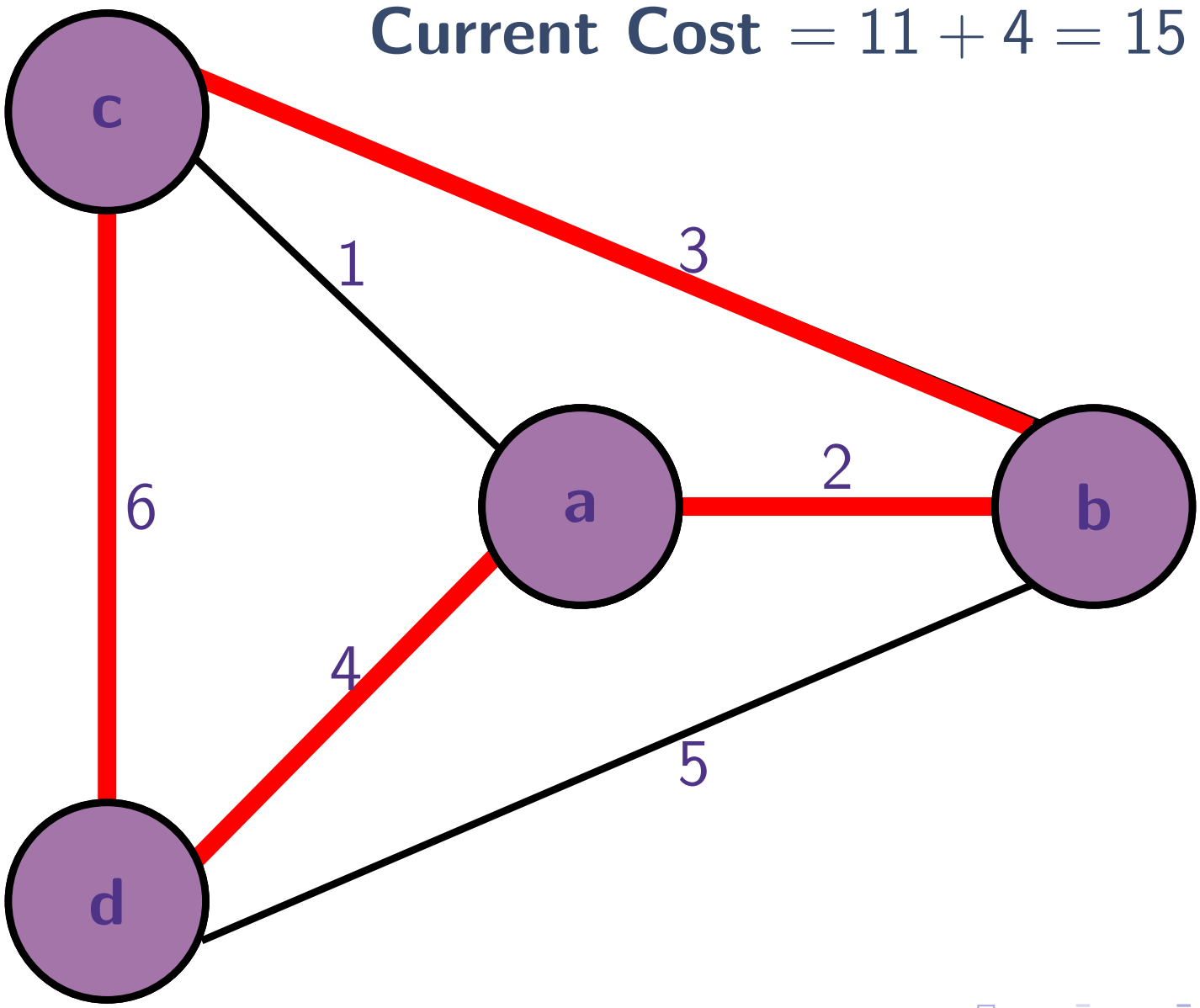
Naive Algorithm



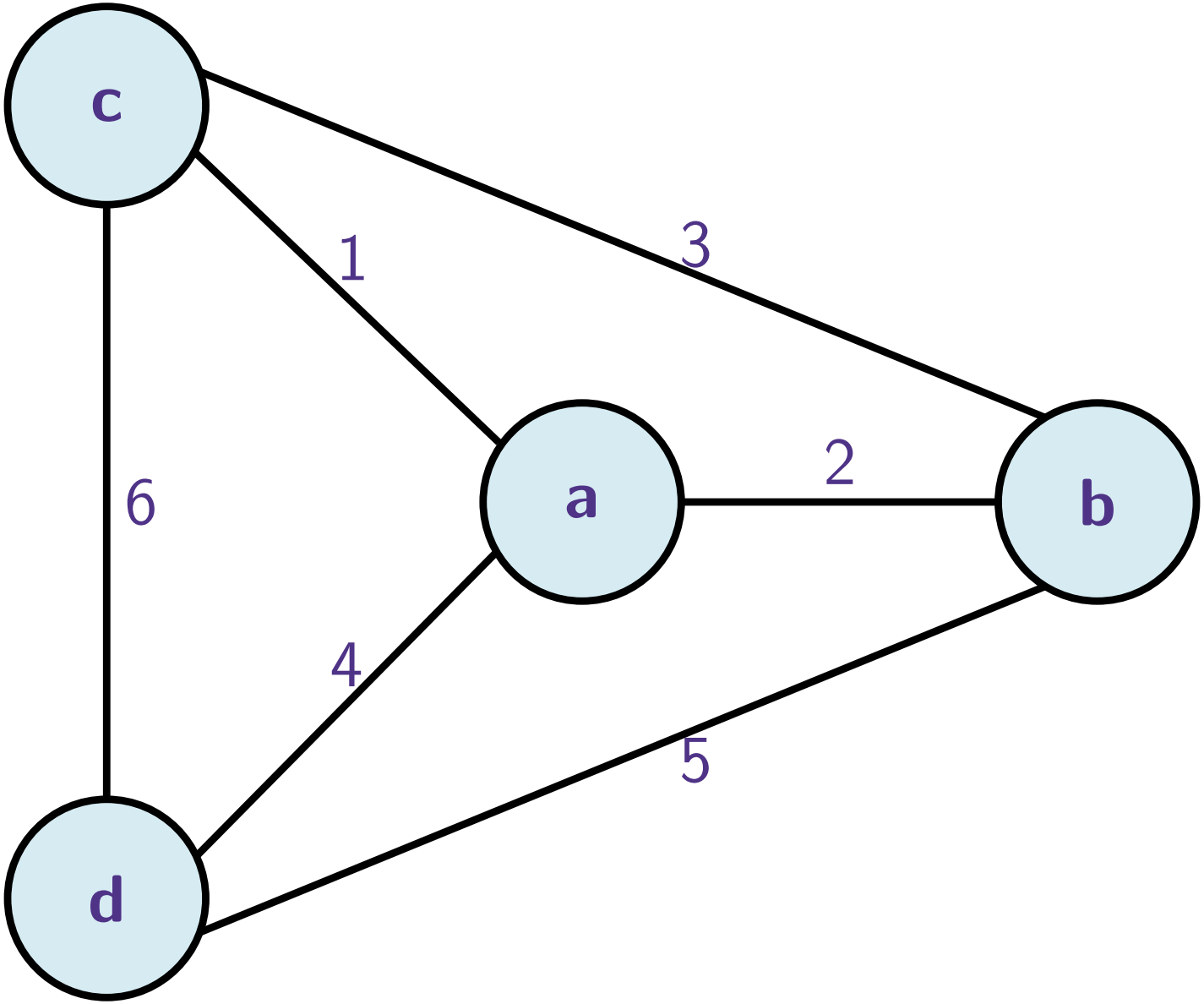
Naive Algorithm



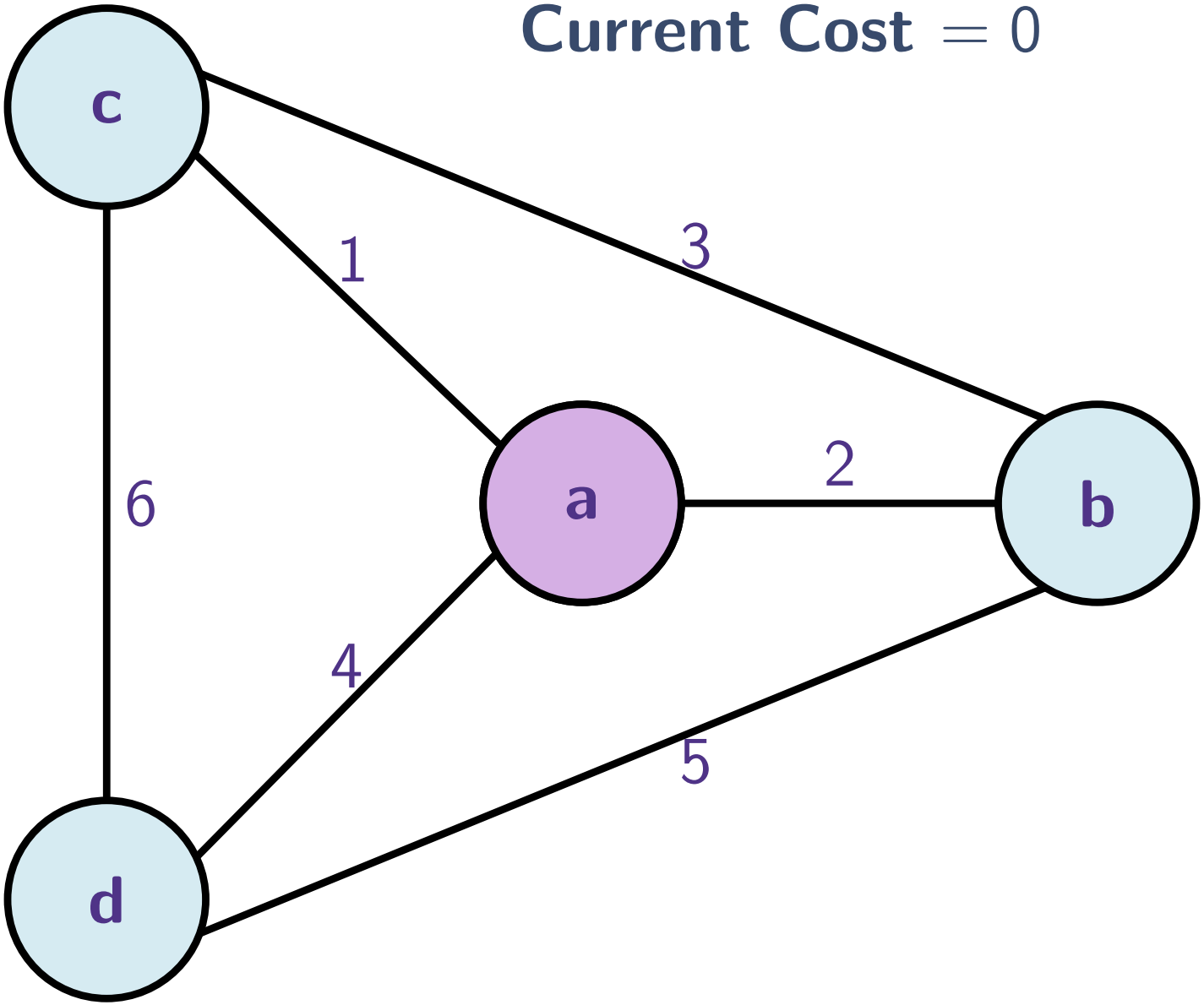
Naive Algorithm



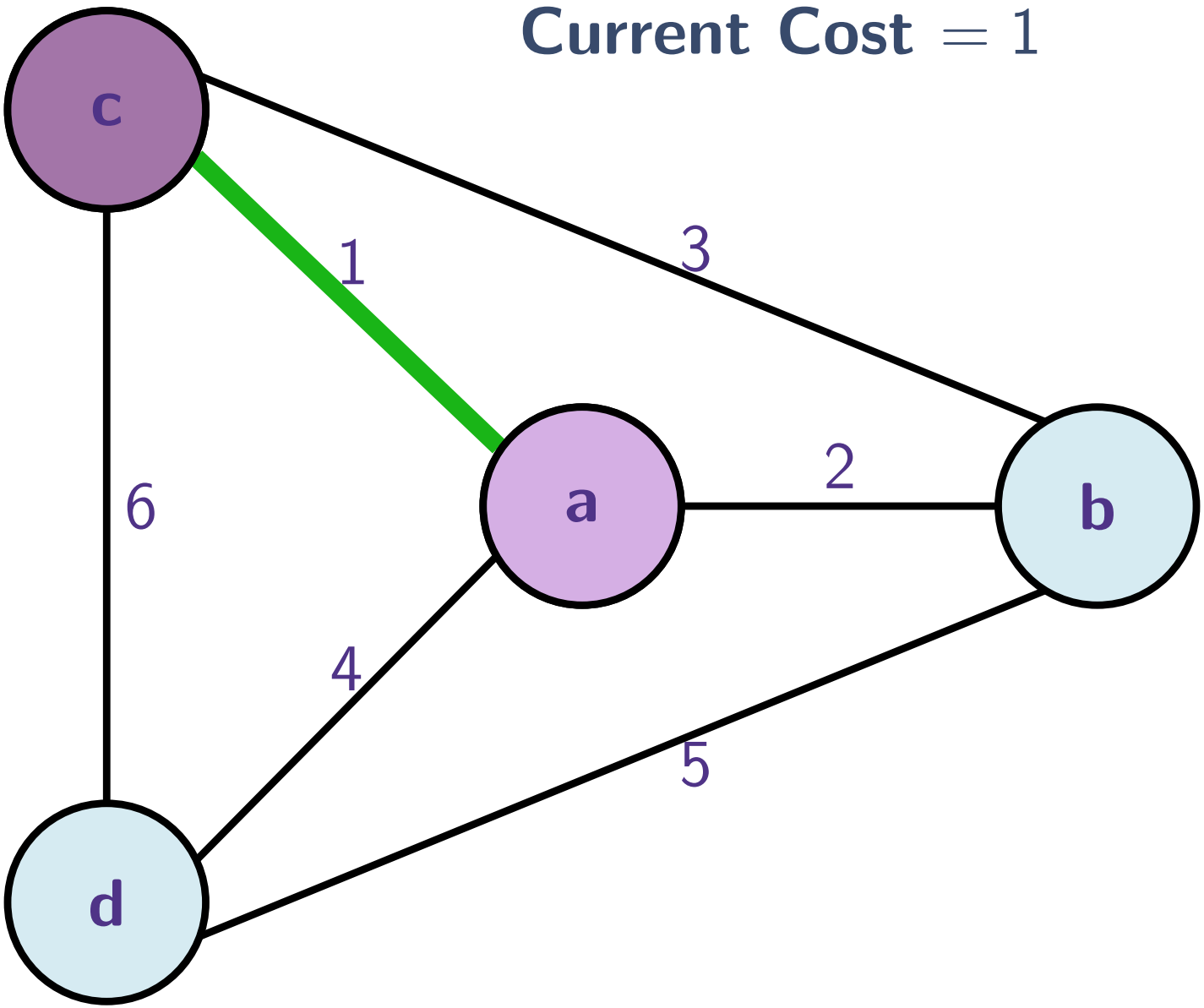
Another Tour



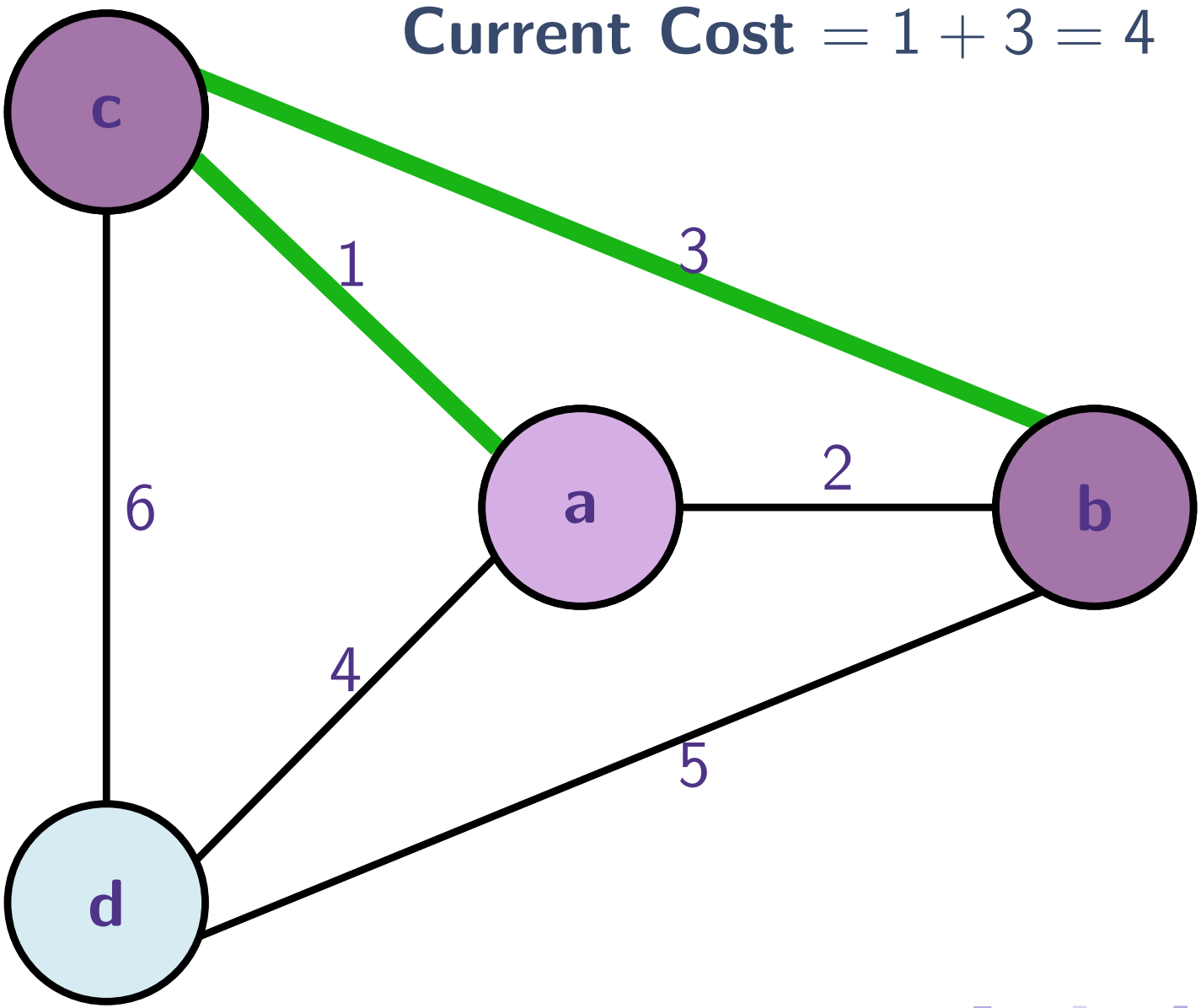
Another Tour



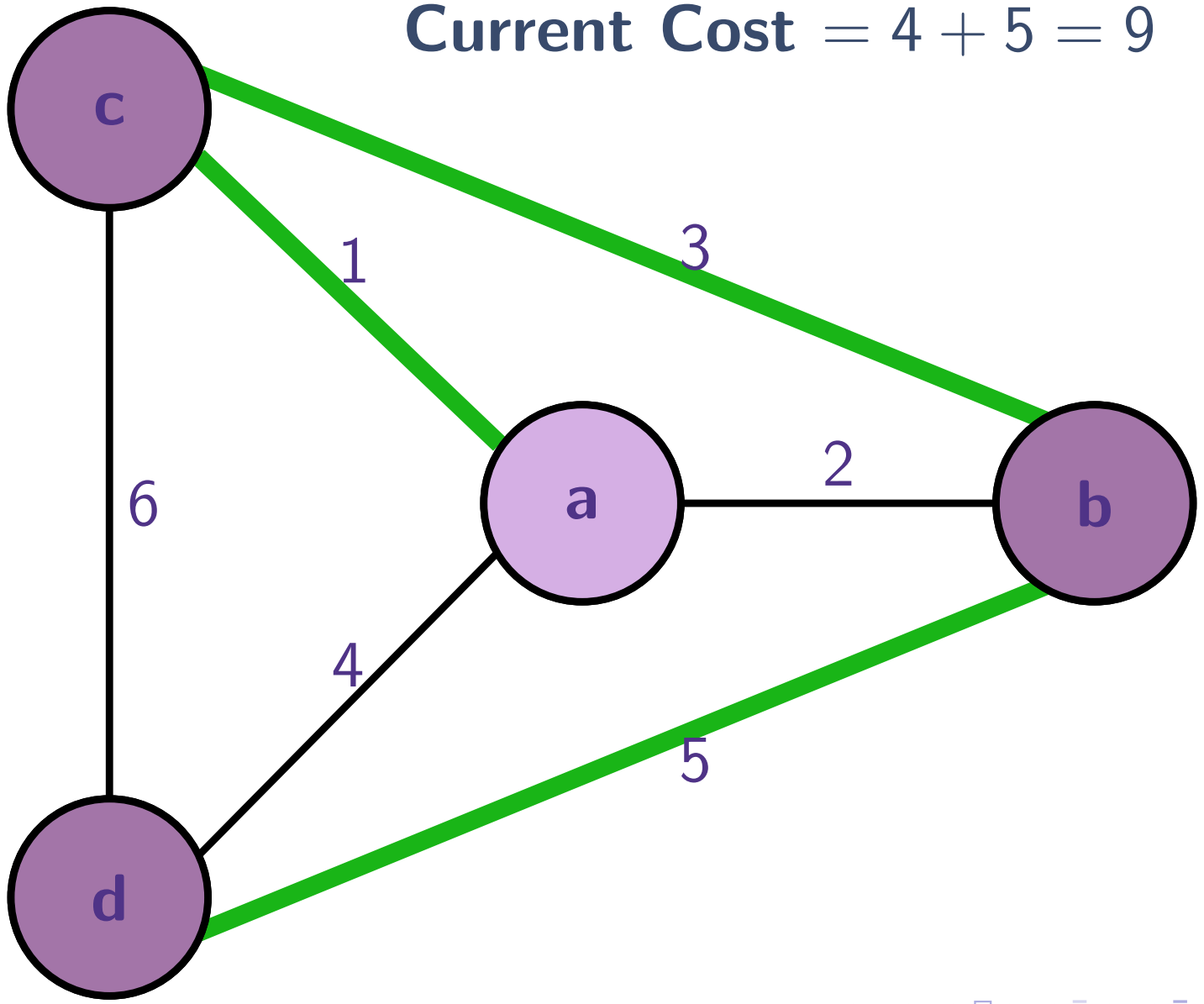
Another Tour



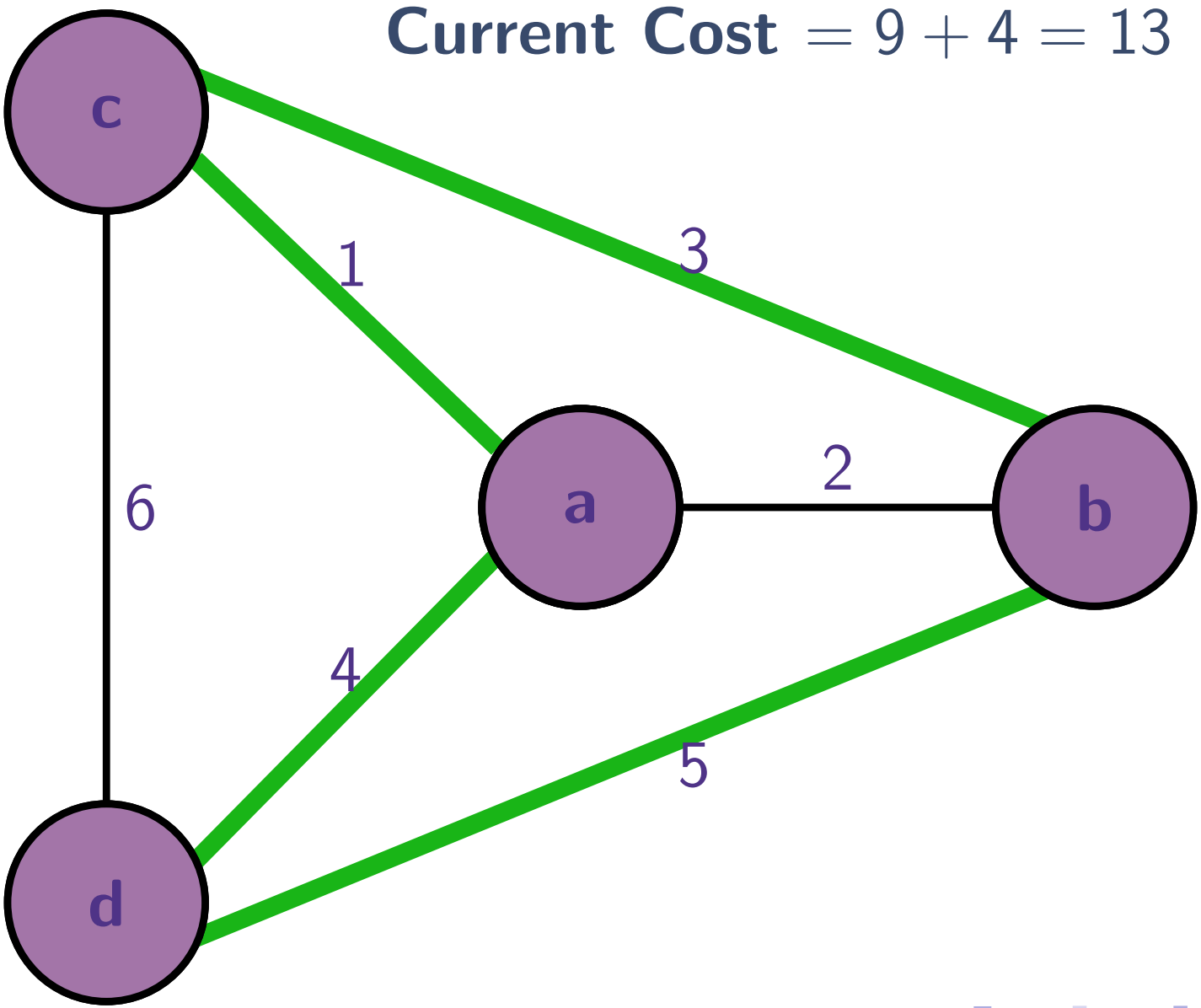
Another Tour



Another Tour



Another Tour



Travelling Salesman Problem

A Question

How do you find such a tour?

Brute Force Algorithm

The Brute Force approach:

- Look at all possible tours in the graph.
- Compute their costs.
- Pick the minimum from them.

Brute Force Algorithm

The Brute Force approach:

- Look at all possible tours in the graph.
- Compute their costs.
- Pick the minimum from them.

Brute Force Algorithm

The Brute Force approach:

- Look at all possible tours in the graph.
- Compute their costs.
- Pick the minimum from them.

Brute Force Algorithm

The Brute Force approach:

- Look at all possible tours in the graph.
- Compute their costs.
- Pick the minimum from them.

Brute Force Algorithm Complexity

However

- In a graph on n vertices, there are (n-1)! such tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  number of tours $\times$ cost of computing one tour
  $= (n-1)! \times O(n) = O(n!)$

Brute Force Algorithm Complexity

However

- In a graph on n vertices, there are (n-1)! such tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  number of tours $\times$ cost of computing one tour
  $= (n-1)! \times O(n) = O(n!)$

Brute Force Algorithm Complexity

However

- In a graph on n vertices, there are (n-1)! such tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  number of tours $\times$ cost of computing one tour
  $= (n-1)! \times O(n) = O(n!)$

Brute Force Algorithm Complexity

However

- In a graph on n vertices, there are (n-1)! such tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  number of tours $\times$ cost of computing one tour
  $= (n-1)! \times O(n) = O(n!)$

An Efficient Algorithm?

A Question

Can we do better?

An Efficient Algorithm?

We can. But a polynomial time algorithm doesn't seem likely.

An Efficient Algorithm?

We can. But a polynomial time algorithm doesn't seem likely.

- *The Traveling Salesman Problem* has been studied since the 1950s.No amount of significant progress has been made so far.

An Efficient Algorithm?

We can. But a polynomial time algorithm doesn't seem likely.

- *The Traveling Salesman Problem* has been studied since the 1950s.No amount of significant progress has been made so far.
- **Edmonds' Conjecture (1965):** there is no polynomial time algorithm for the *Traveling Salesman Problem*.

An Efficient Algorithm?

We can. But a polynomial time algorithm doesn't seem likely.

- *The Traveling Salesman Problem* has been studied since the 1950s.No amount of significant progress has been made so far.
- **Edmonds' Conjecture (1965):** there is no polynomial time algorithm for the *Traveling Salesman Problem*.
- Edmonds' Conjecture equivalent to $P \neq NP$.

An Efficient Algorithm?

We can. But a polynomial time algorithm doesn't seem likely.

- *The Traveling Salesman Problem* has been studied since the 1950s.No amount of significant progress has been made so far.
- **Edmonds' Conjecture (1965):** there is no polynomial time algorithm for the *Traveling Salesman Problem*.
- Edmonds' Conjecture equivalent to $P \neq NP$.
- *The Traveling Salesman Problem* is NP-Complete!

Coping with NP-Completeness

We can

- Solve TSP exactly, but take a really long time for it.
- Solve it only approximately, but do it fast.
- Solve it exactly, but for really special cases.

Coping with NP-Completeness

We can

- Solve TSP exactly, but take a really long time for it.
- Solve it only approximately, but do it fast.
- Solve it exactly, but for really special cases.

Coping with NP-Completeness

We can

- Solve TSP exactly, but take a really long time for it.
- Solve it only approximately, but do it fast.
- Solve it exactly, but for really special cases.

Coping with NP-Completeness

We can

- Solve TSP exactly, but take a really long time for it.
- Solve it only approximately, but do it fast.
- Solve it exactly, but for really special cases.

Coping with NP-Completeness

We can

- Solve TSP exactly, but take a really long time for it.
- Solve it only approximately, but do it fast.
- Solve it exactly, but for really special cases.

Hard to even approximate

There is a catch.

Inapproximability of TSP

Unless P $=$ NP, there does not exist a polynomial time α approximation algorithm for travelling salesman problem.

Coping with NP-Completeness

We can

- Solve TSP exactly, but take a really long time for it.
- Solve it only approximately, but do it fast.
- Solve it exactly, but for really special cases.

Metric TSP

Triangle inequality holds shortest path between vertices = the one hop path between them.

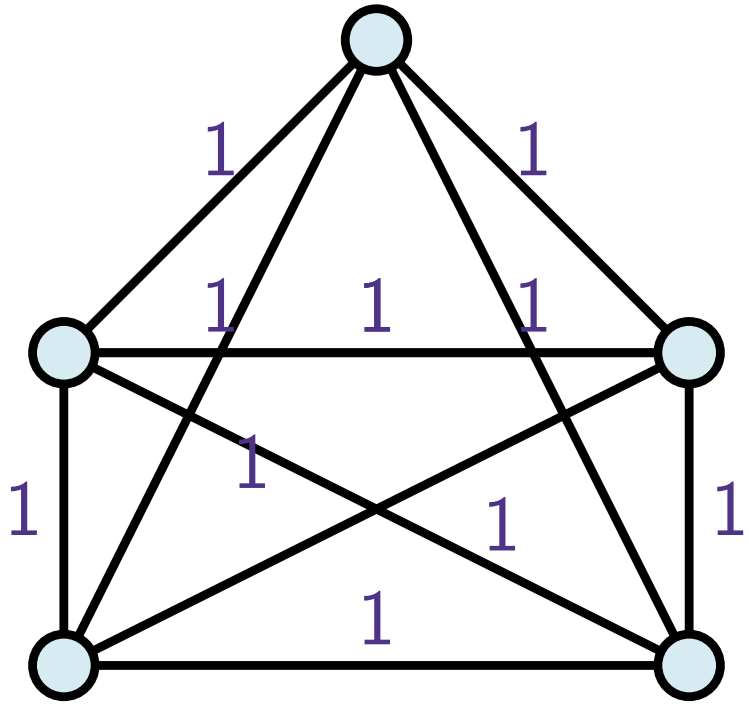


Figure: A Metric TSP instance.

Approximation algorithm for metric TSP

- Still a NP-Complete.
- But there are good approximation algorithms.
  - The MST Heuristic
  - Christofide's Algorithm

Acknowledgement

- Still a NP-Complete.
- But there are good approximation algorithms.