The Traveling Salesman Problem

Zahin Wahab   Mursalin Habib

Department of Computer Science & Engineering
Bangladesh University of Engineering and Technology

July 16, 2018

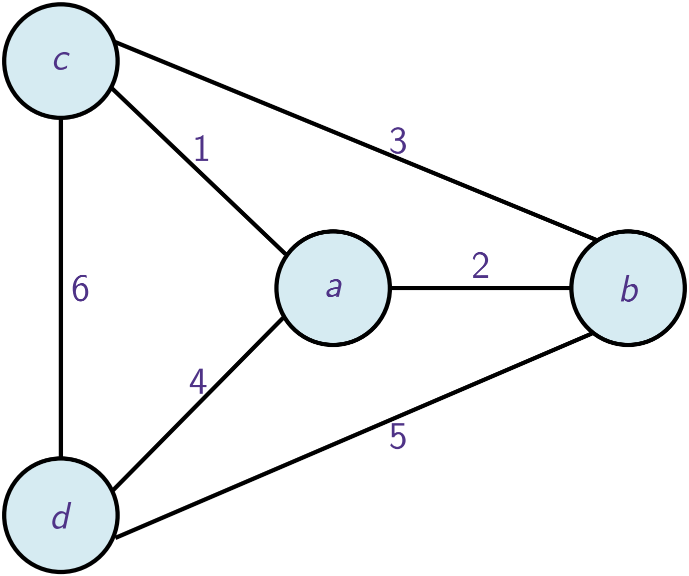
Problem Statement

- **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 Statement

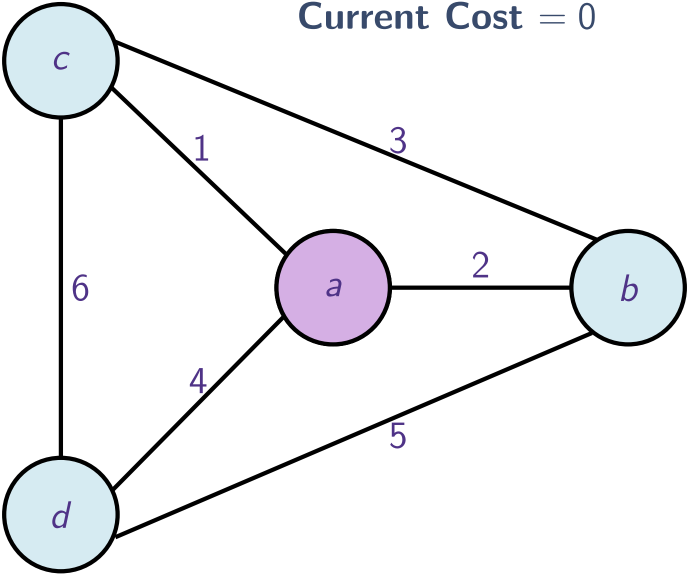
- **Input** : A complete undirected graph with non-negative edge costs.
- **Output** : A minimum cost tour i.e. a cycle that visits all the vertices exactly once.

An Example

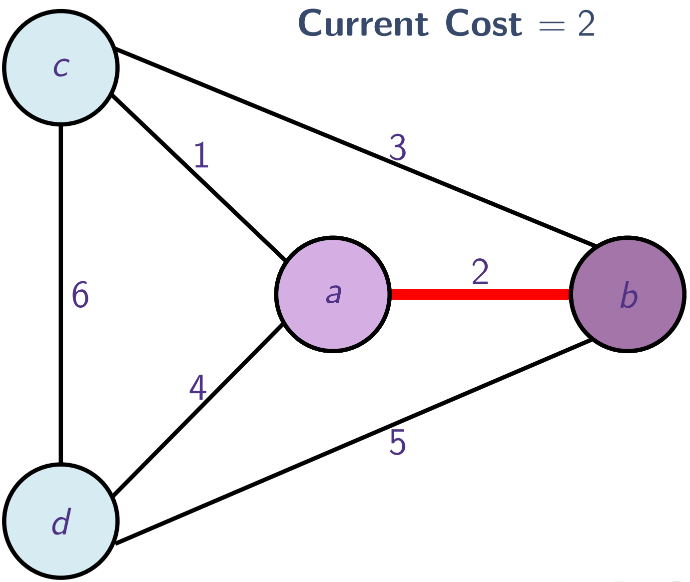


An Example

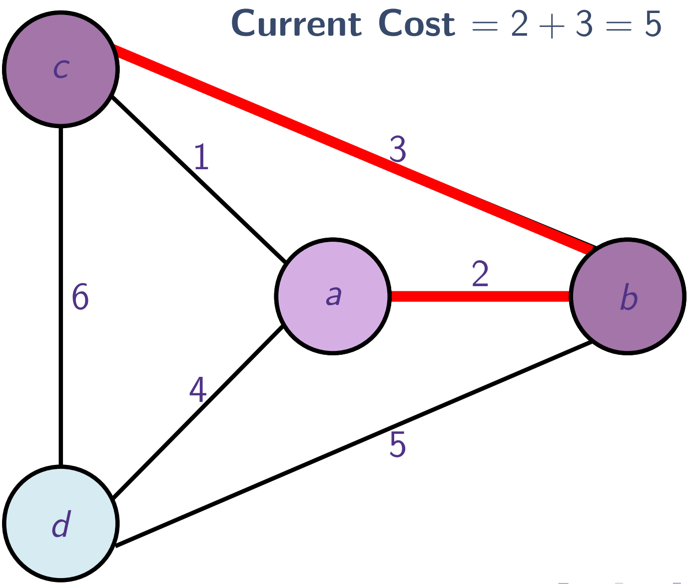
Current Cost $= 0$



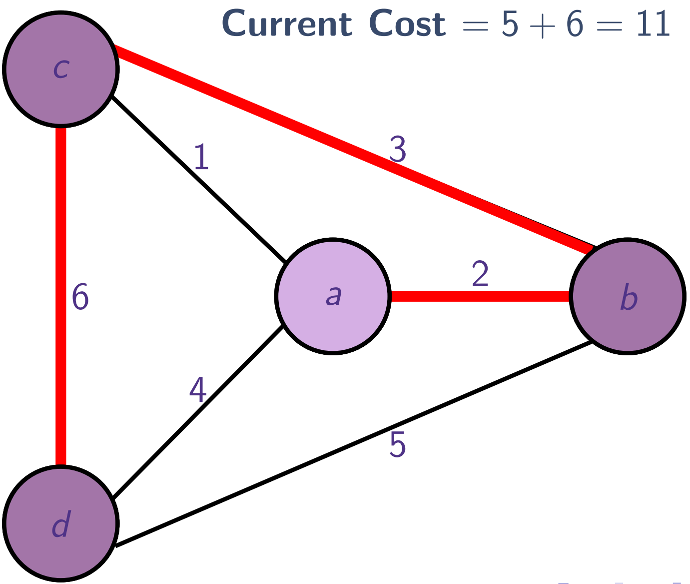
An Example



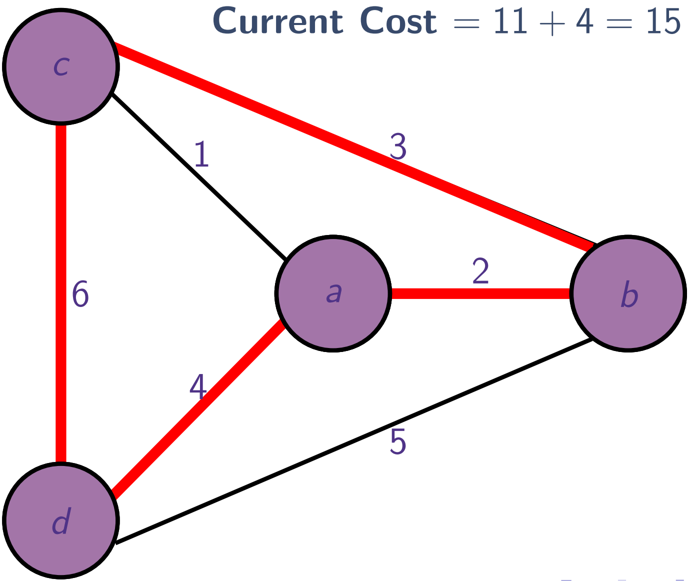
An Example



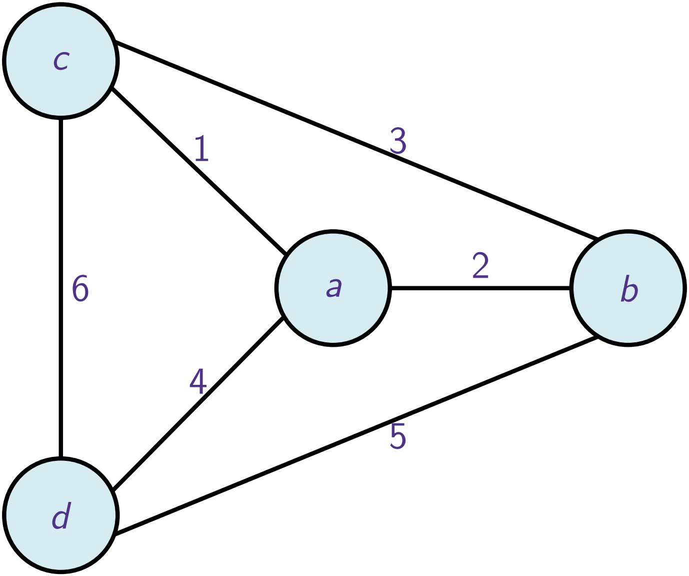
An Example



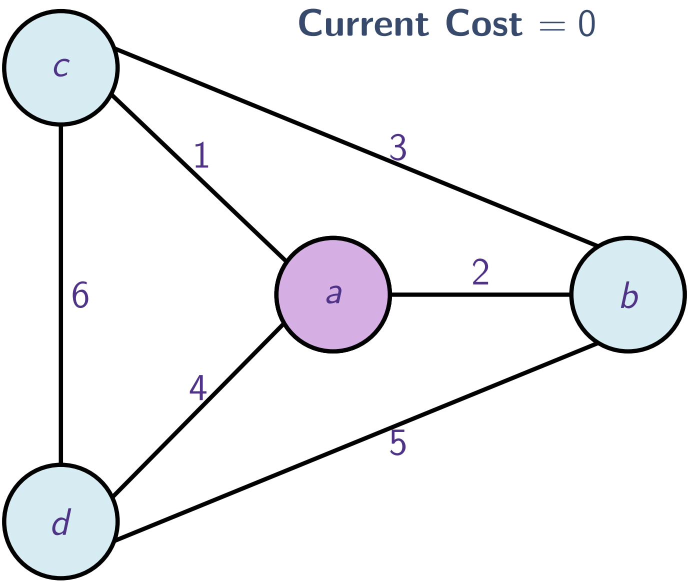
An Example



An Example (Continued)

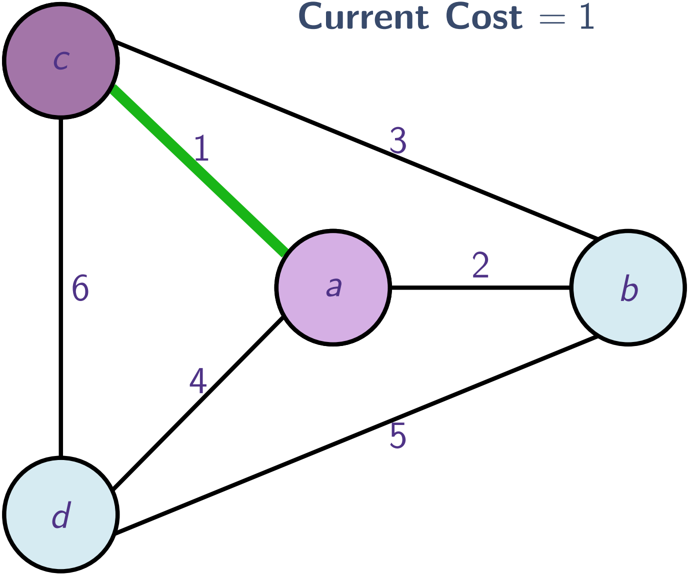


An Example (Continued)

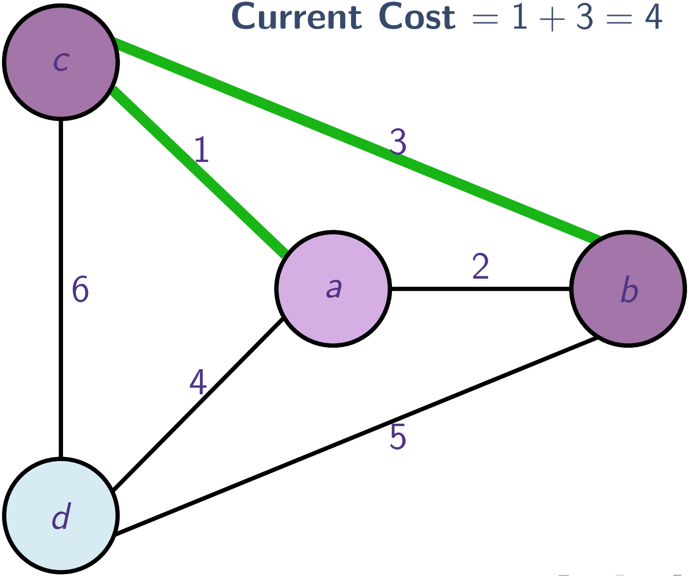


An Example (Continued)

Current Cost $= 1$

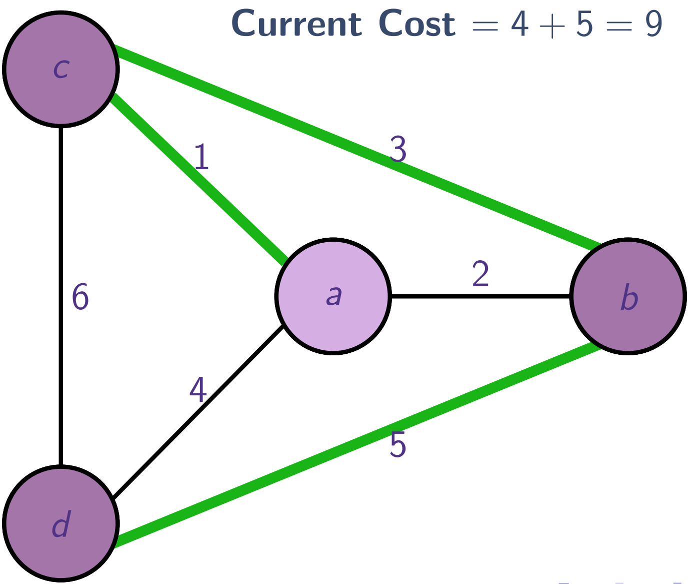


An Example (Continued)



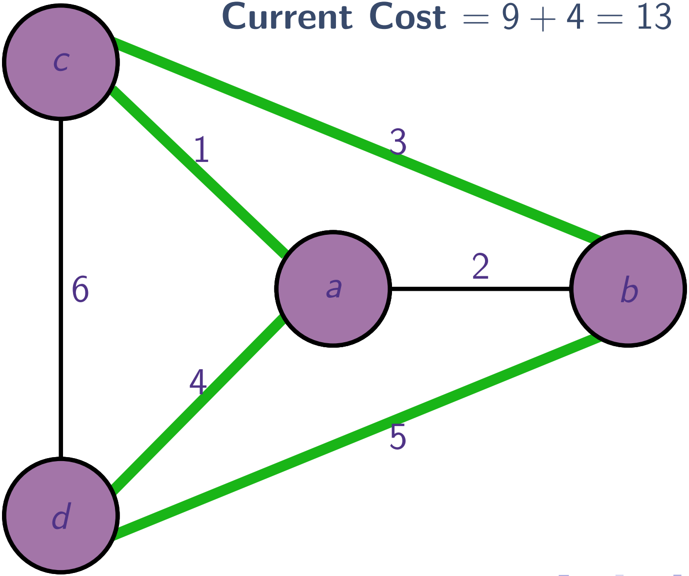
Current Cost $= 1 + 3 = 4$

An Example (Continued)



An Example (Continued)

Current Cost $= 9 + 4 = 13$



Question of the Day

How do we find an optimal tour?

The Brute Force Algorithm

The Brute Force approach:

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

The Brute Force Algorithm

The Brute Force approach:

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

The Brute Force Algorithm

The Brute Force approach:

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

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 $(n-1)!$ TSP tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  \# 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 $(n-1)!$ TSP tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  $\#$ 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 $(n-1)!$ TSP tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  $\#$ 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 $(n-1)!$ TSP tours.
- Computing the cost of a tour takes linear time.
- **Brute-Force Algorithm running time:**
  $\#$ of tours $\times$ cost of computing one tour $= (n-1)! \times O(n) = O(n!)$

An Efficient Algorithm?

A Question We Should be Asking Everyday

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!

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.

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.

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.

Theorem

Unless $P = NP$, there does not exist a polynomial time α- approximation algorithm for the Traveling 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

Edge costs satisfy the triangle inequality i.e. the shortest path between vertices = the one-hop path between them.

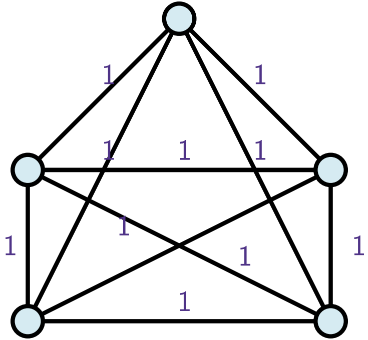


Figure: A Metric TSP instance.

Approximation Algorithms for Metric TSP

- Still NP-Complete!
- But there are good approximation algorithms.
  - **The MST Heuristic** (a 2-approximation algorithm)
  - **Christofides's Algorithm (1976)** (a $\frac{3}{2}$-approximation algorithm)

To Summarize

- The *Traveling Salesman Problem* is interesting.
- *The Traveling Salesman Problem* is hard!
- Approximation algorithms for NP-Complete Problems are still an active area of research.

Acknowledgements I

Tim Roughgarden.
Stanford CS261 Lecture Notes.
2016.

Jack Edmonds.
Paths, trees, and flowers.
Can. J. Math. 17: 449467, 1965.