Travelling Salesman Problem

M.M. Nayem[1] Kazi Jayed Haider[2] Iffat Bin Hossain[3]

[1]2005078

[2]2005081

[3]2005087

February 20, 2024

Table of Contents

Introduction

What is TSP?

Travelling salesman problem (TSP), *asks the following question: "Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?" [3]*

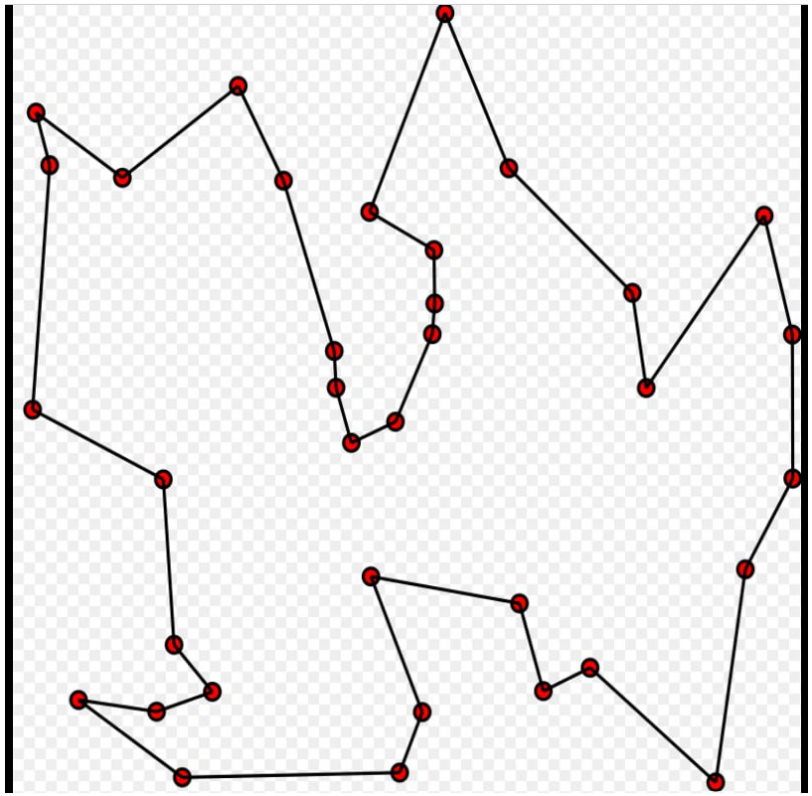


Figure: TSP

Why TSP is Interesting?

- It's NP-hard problem in complexity theory
- Algorithmic Challenges
- Benchmark Problem
- Real-world Applications

Real-world Applications

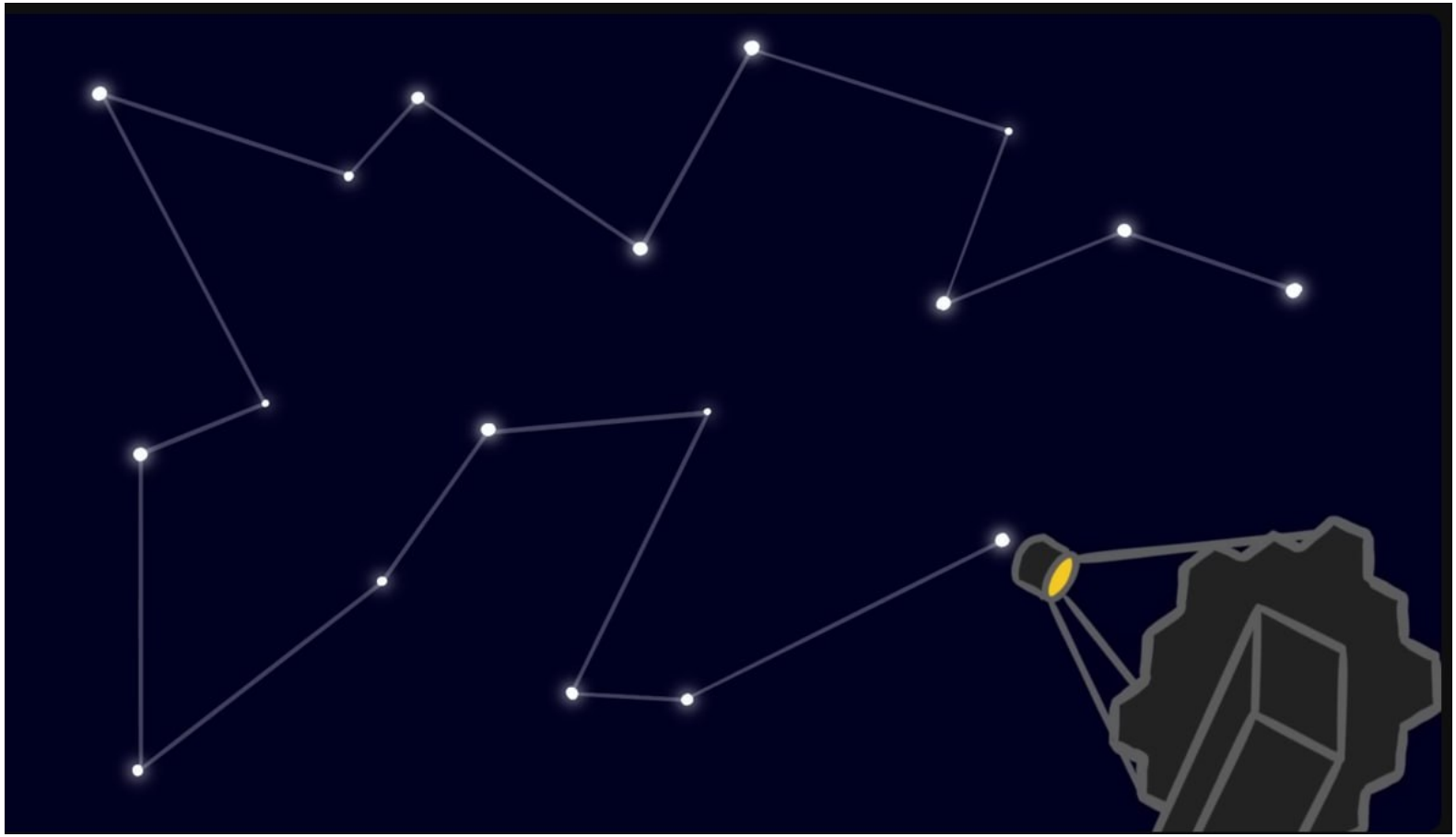


Figure: In Hubble Telescope

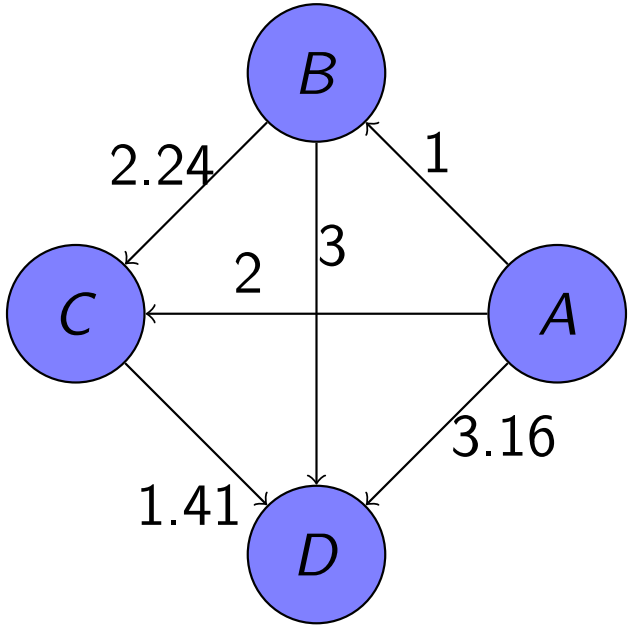
Why TSP can't be solved using Greedy?

- Can be proved Mathematically That TSP can't be solved using Greedy approach

Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

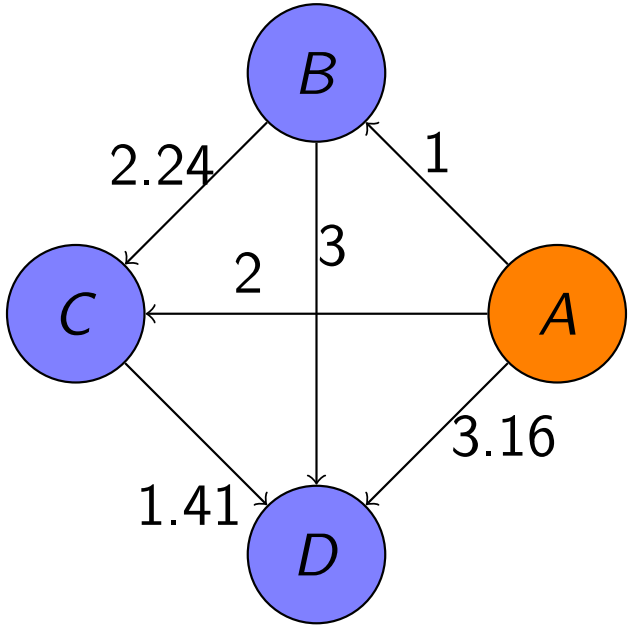
Why TSP can't be solved using Greedy



Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

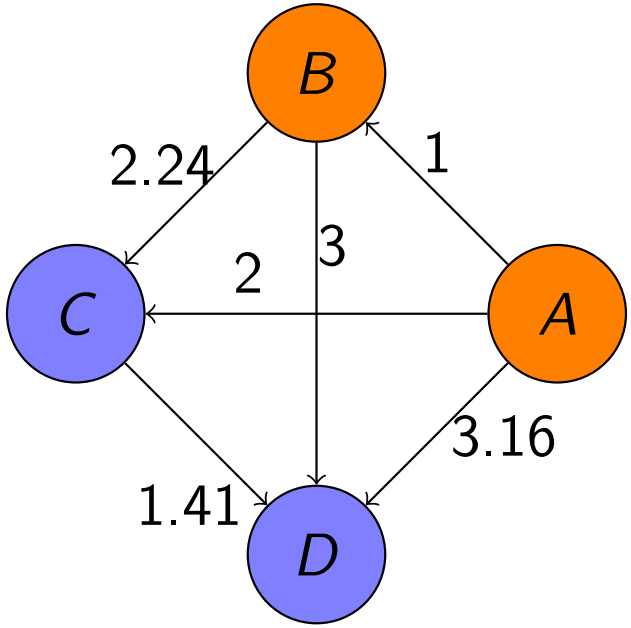
Why TSP can't be solved using Greedy



Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

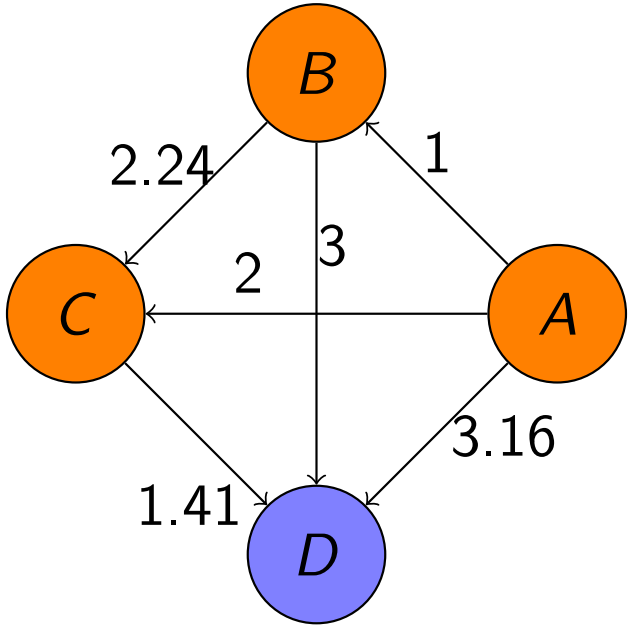
Why TSP can't be solved using Greedy



Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

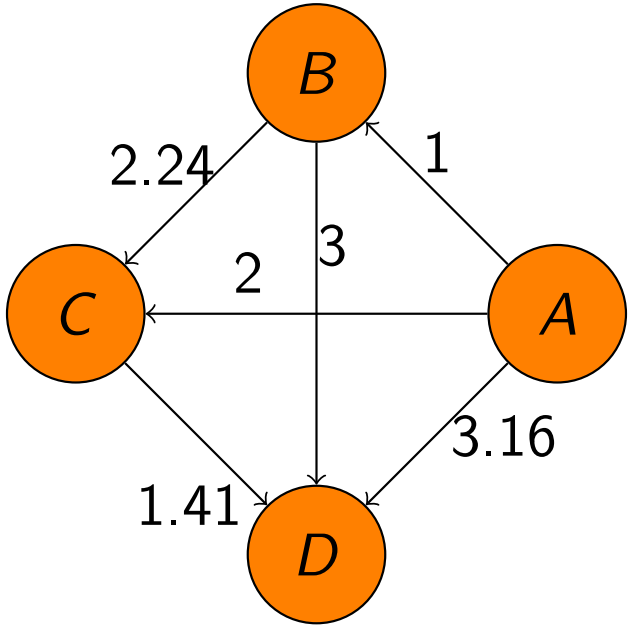
Why TSP can't be solved using Greedy



Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

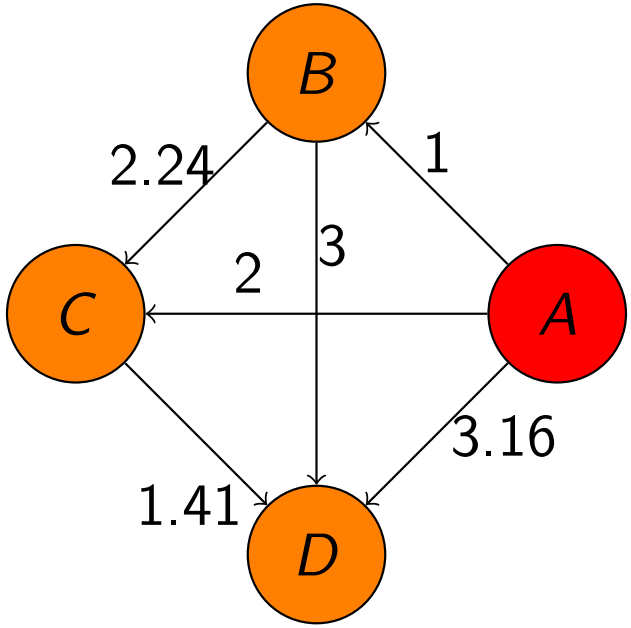
Why TSP can't be solved using Greedy



Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

Why TSP can't be solved using Greedy



Examples

Consider a complete graph with 4 vertices A(0,0),B(0,1),C(2,0),D(3,1) A Salesman starts in A,here B is 1 unit away,C is 2 unit away,D is 3.16 unit away The salesman goes to B which is closest,then C is 2.24 unit away and D is 3 away The salesman goes to C which is closest,then to D which is the last unvisited city then back to A . The total trip A-B-C-D-A is 7.81 long,while the trip A-B-D-C-A is 7.41 unit long.Thus Greedy fails.

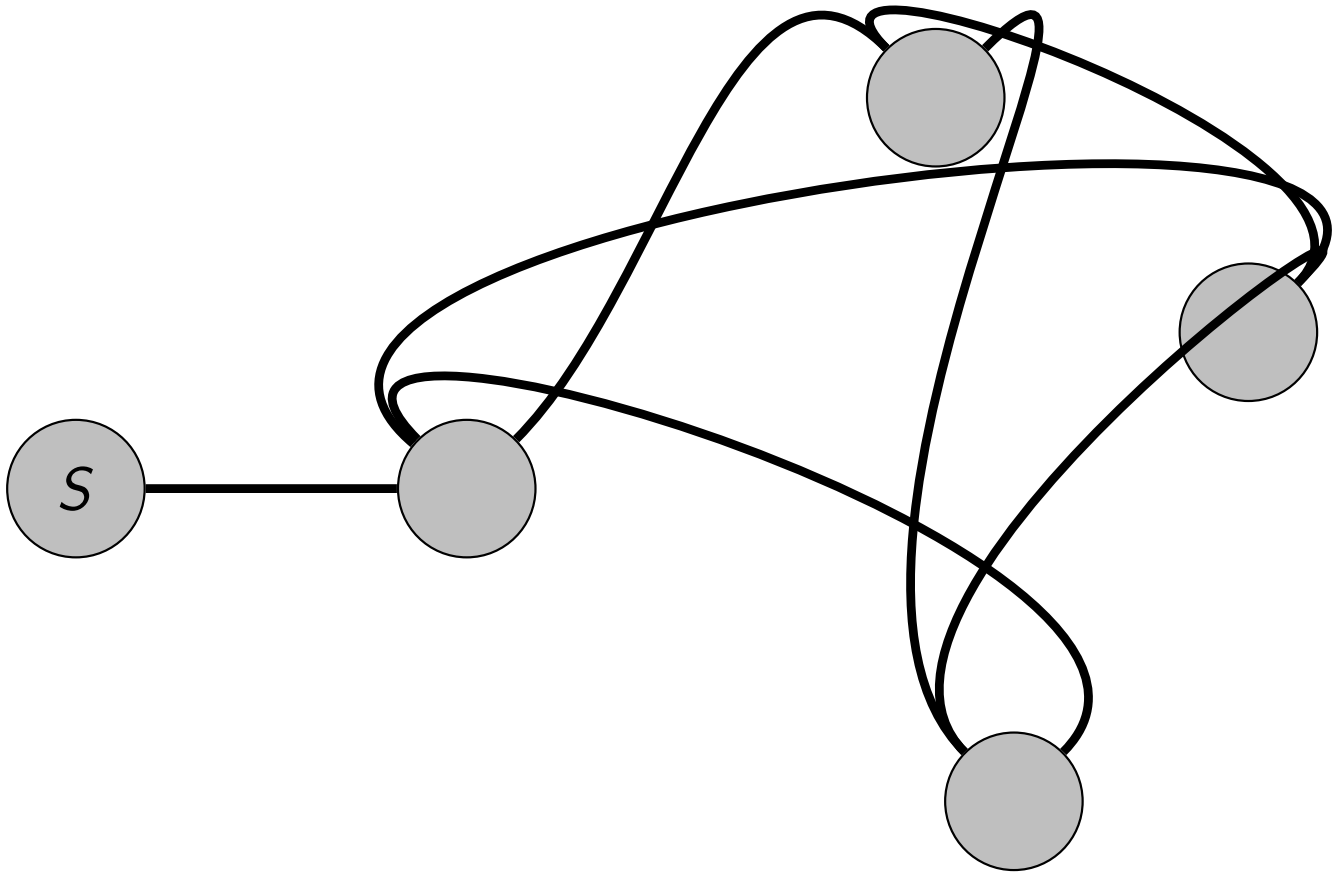
TSP solution method

There are several algorithm to solve **Travelling Salesman Problem**.

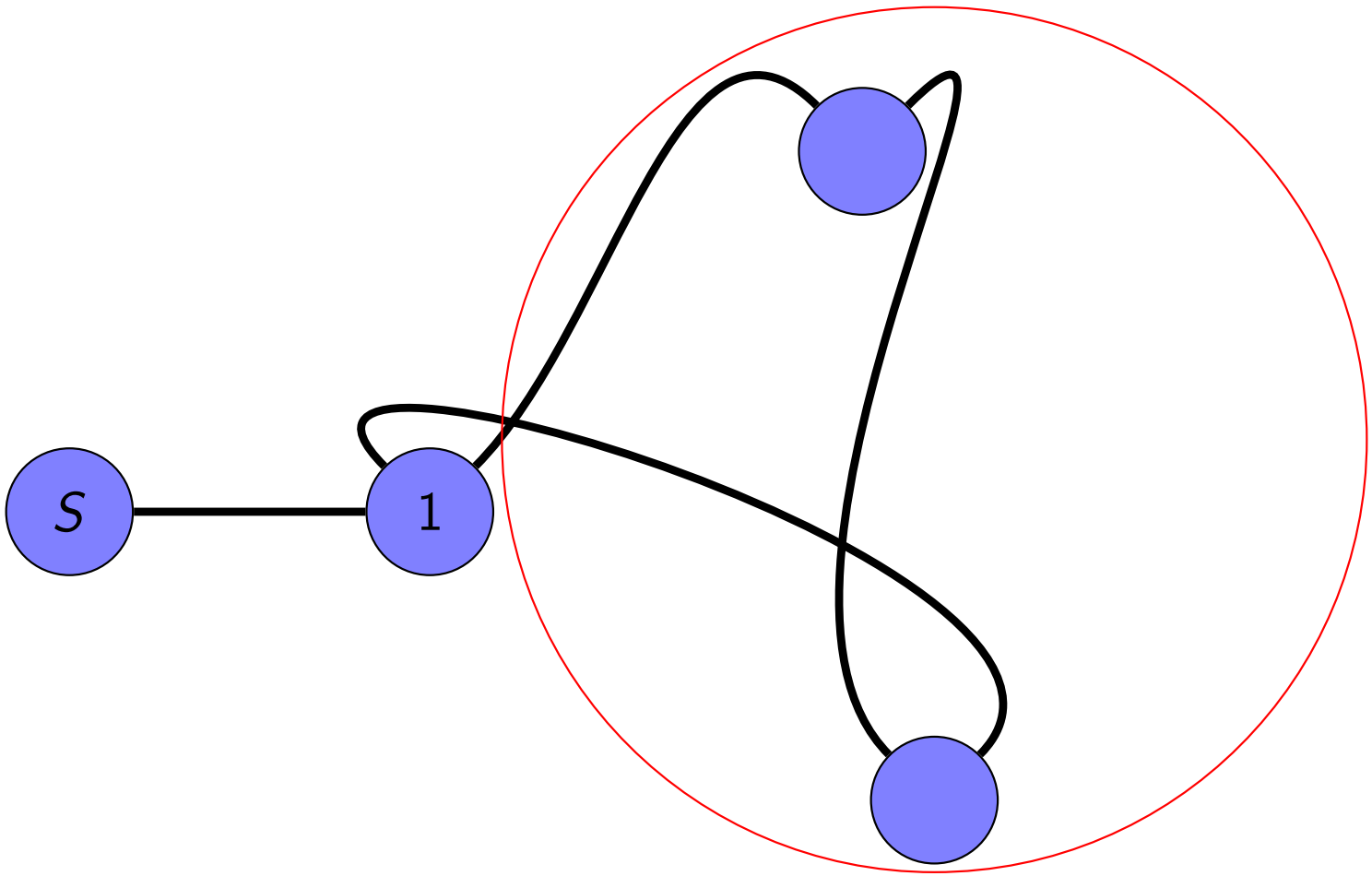
- TSP with Brute Force
- TSP with Dynamic Programming(using Bitmask)
- TSP with Approximation Algorithm
- TSP with Branch & Bound
- TSP with Genetic Algorithm

TSP with Dynamic Programming(using Bitmask)

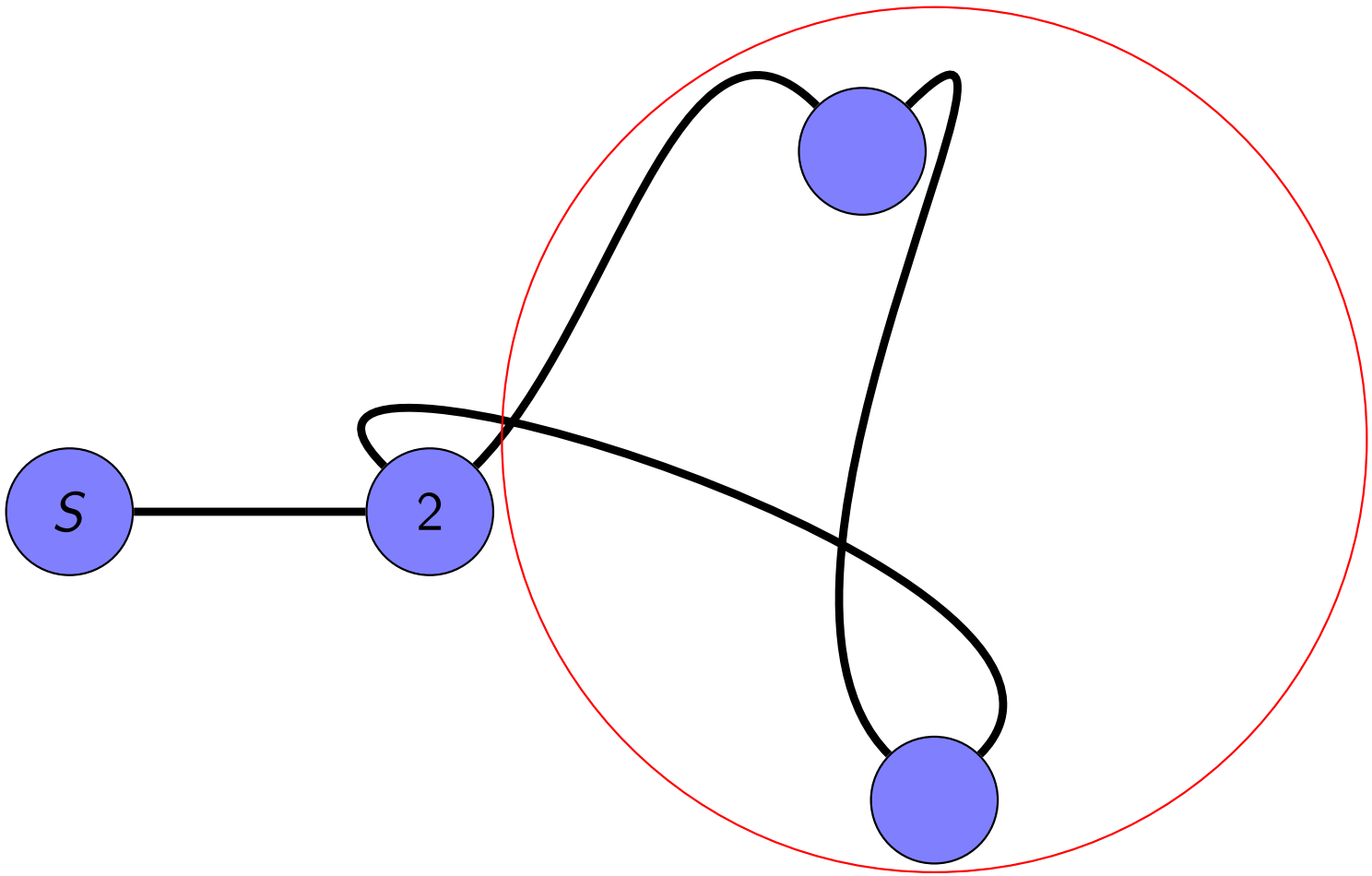
Let us consider a graph with source S containg n nodes



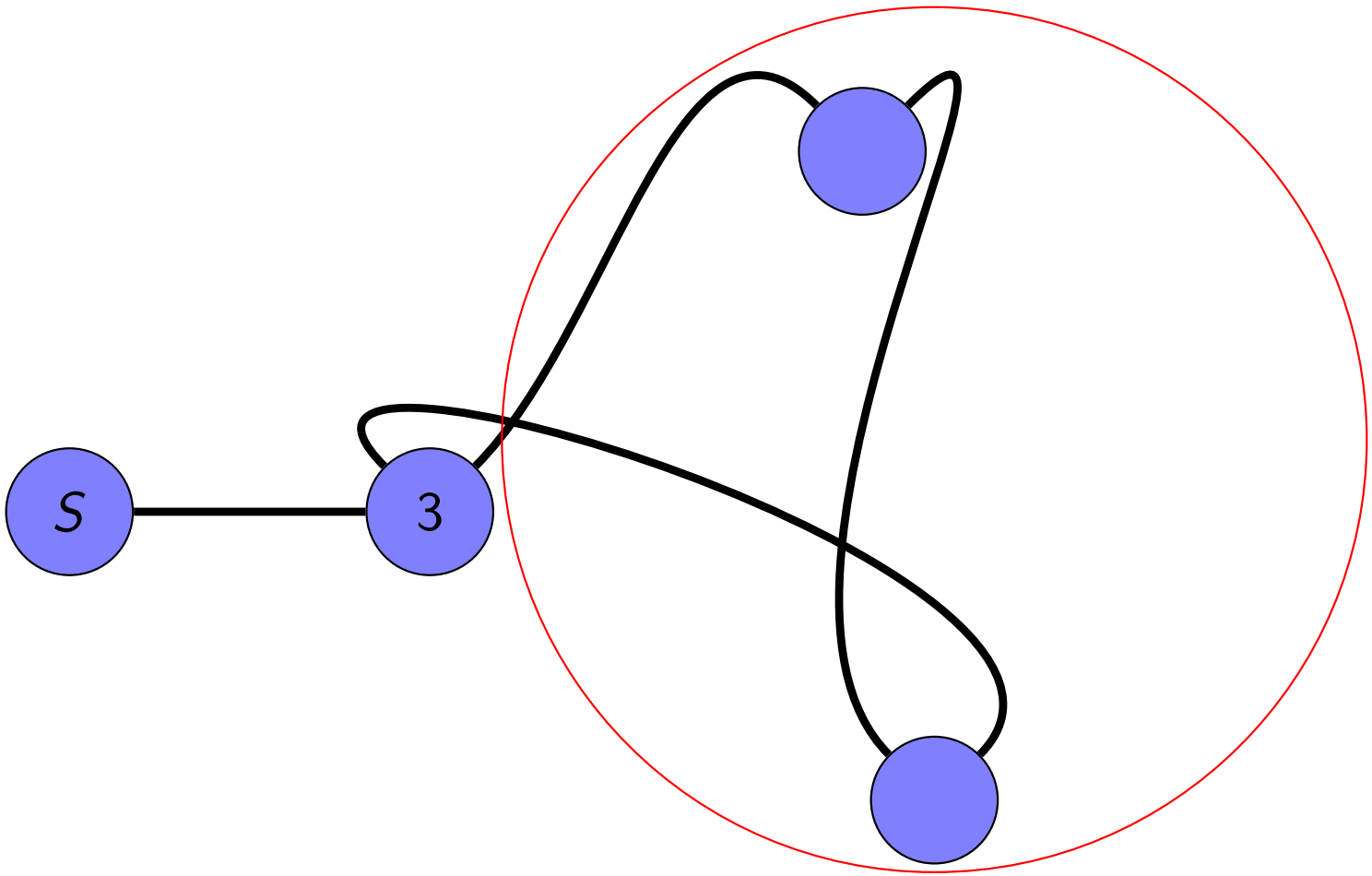
TSP with Dynamic Programming(using Bitmask)



TSP with Dynamic Programming(using Bitmask)



TSP with Dynamic Programming(using Bitmask)



TSP with Dynamic Programming(using Bitmask)

Which path should we choose?

We will choose the node so that $distance[S][i] + Cost_{remaining}$ *becomes minimum*

Our Concern!!!

How to find the optimal path from remaining portion?

TSP with Dynamic Programming(using Bitmask)

Solution

Here Comes Recursion!

$$f(i, 2^{n-1}) = dis[i][0]$$

$$f(i, mask) = min(f(j, turnOn(mask, j) + w(i, j)) \; where(i, j) \in E$$

[1]

TSP with Dynamic Programming(using Bitmask)

Steps to perform Dynamic Programming(using Bitmask)

- *Initialize a DP table to store the optimal solution for subproblems. The DP table can be a 2D array where dp[mask][i] represents the minimum cost of visiting all vertices in the subset represented by the bitmask mask and ending at vertex i.*
- *Initialize the DP table for the base case where only one vertex is visited (i.e., dp[1 « i][i] = 0 for all vertices i).*
- *terate over all possible subsets of vertices represented by bitmasks and all possible ending vertices. For each subset and ending vertex, compute the optimal cost based on the previously computed values.*
- *he final answer will be the minimum value among all values dp[(1 « n) - 1][i], where i varies from 0 to n-1*

Time Complexity

Time Complexity

$O(n^2 \cdot 2^n)$ where $O(n \cdot 2^n)$ are maximum number of unique subproblems/states and $O(n)$ for transition (through for loop as in code) in every states.

TSP with Approximation Algorithm

Approximation Algorithm

- *An approximation algorithm is a way of dealing with NP-completeness for an optimization problem.*
- *This technique does not guarantee the best solution.*
- *The approximate algorithms work only if the problem instance satisfies* ***Triangle-Inequality***.

TSP with Approximation Algorithm

Triangle-Inequality

The least distant path to reach a vertex j from i is always to reach j directly from i, rather than through some other vertex k (or vertices), i.e., dis(i, j) is always less than or equal to dis(i, k) + dist(k, j). [2]

Why Triangle-Inequality is a necessary condition?

When the cost function satisfies the triangle inequality, we can design an approximate algorithm for TSP that returns a tour whose cost is never more than twice the cost of an optimal tour.

TSP with Approximation Algorithm

In the traveling salesperson problem, the optimization problem is to find the **shortest cycle**, and the approximation problem is to find a short cycle.

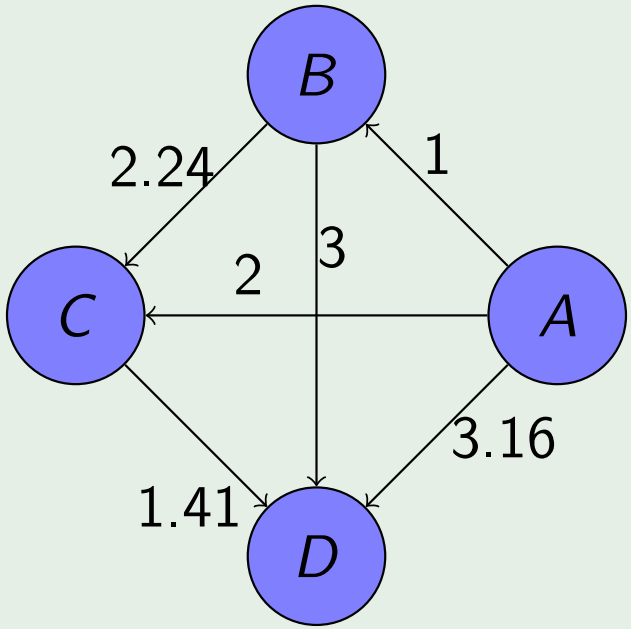
Steps to perform Approximation Algorithm

- *Let any node as starting node of the given graph*
- *Construct MST from with starting node as root.*
  Two ways to find MST
  - *Prim's Algorithm*
  - *Kruskal's Algorithm*
- *List vertices visited in preorder walk(DFS) of the constructed MST and add starting node at the end.*

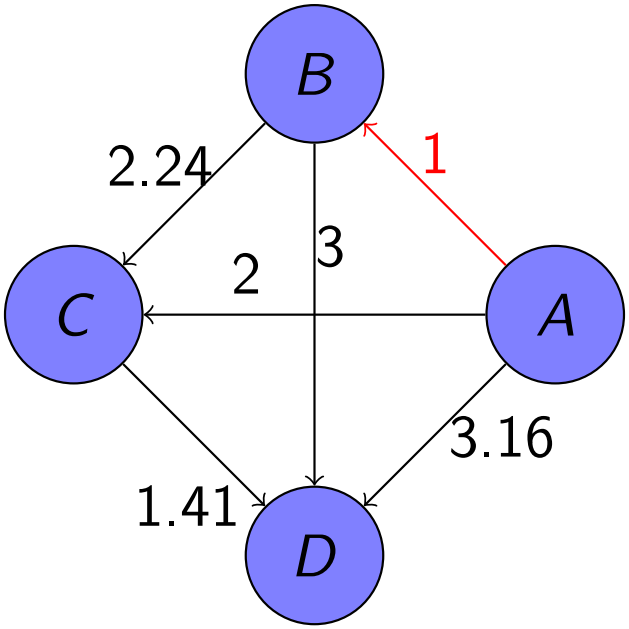
TSP with Approximation Algorithm

Let's take an example.
Consider A as starting node

Example

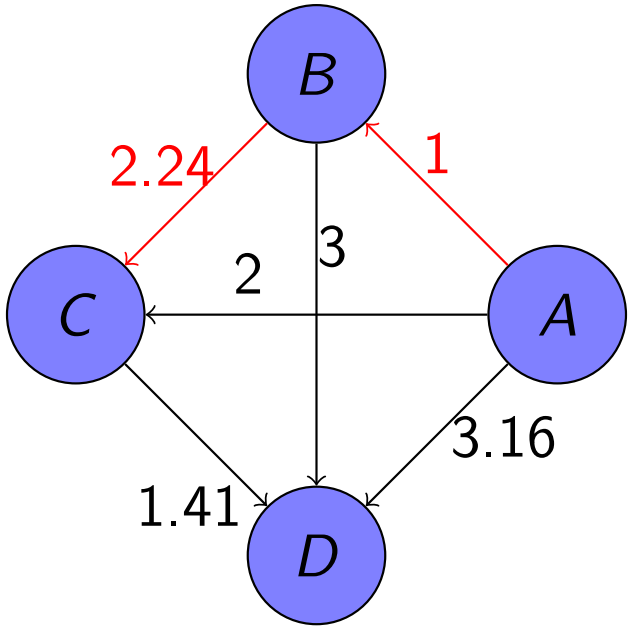


TSP with Approximation Algorithm



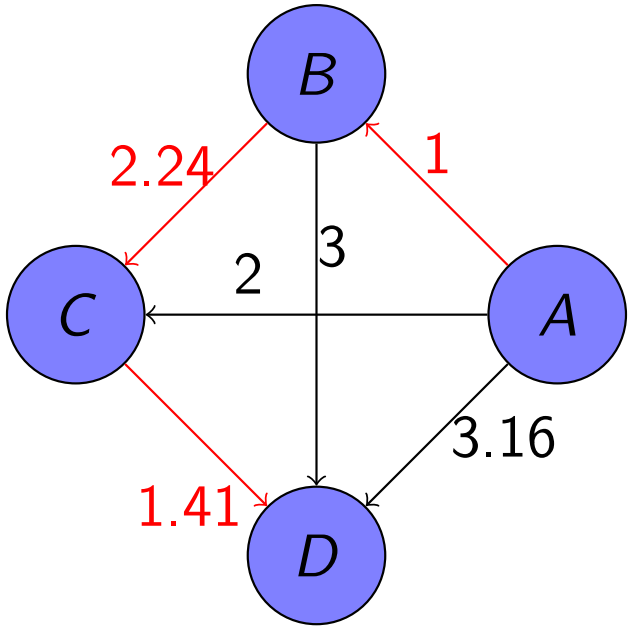
Caption: Constructing MST

TSP with Approximation Algorithm



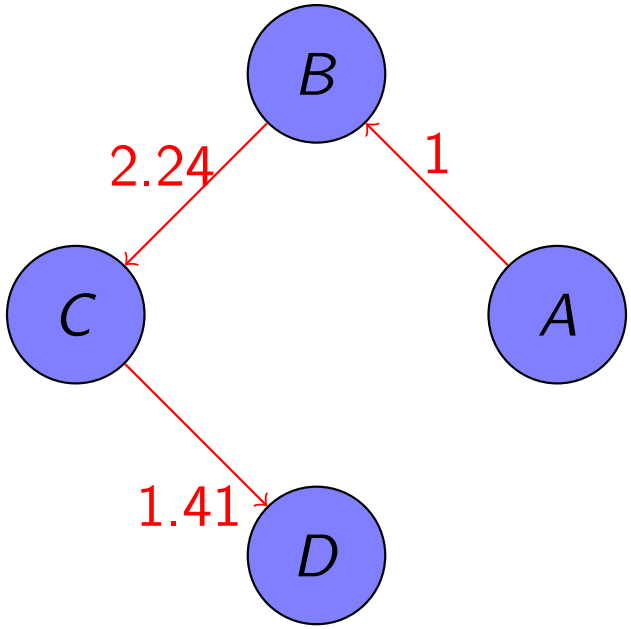
Caption: Constructing MST

TSP with Approximation Algorithm



Caption: Constructing MST

TSP with Approximation Algorithm

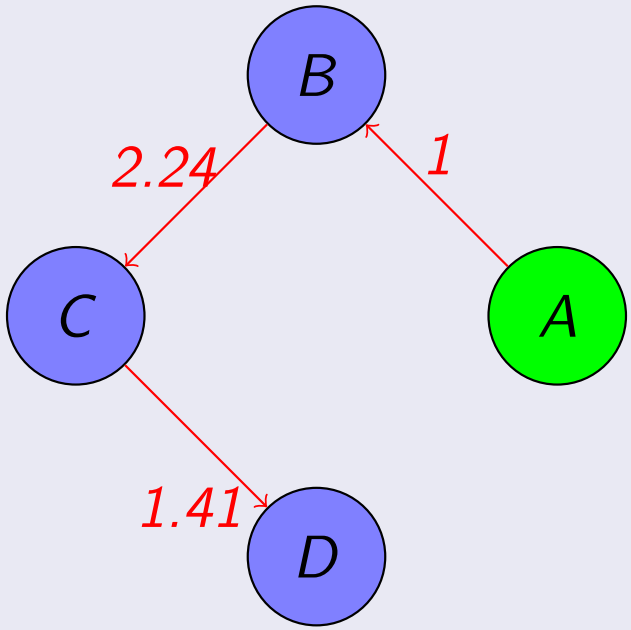


Caption: Final MST

TSP with Approximation Algorithm

Now Perform DFS traversal on generated MST

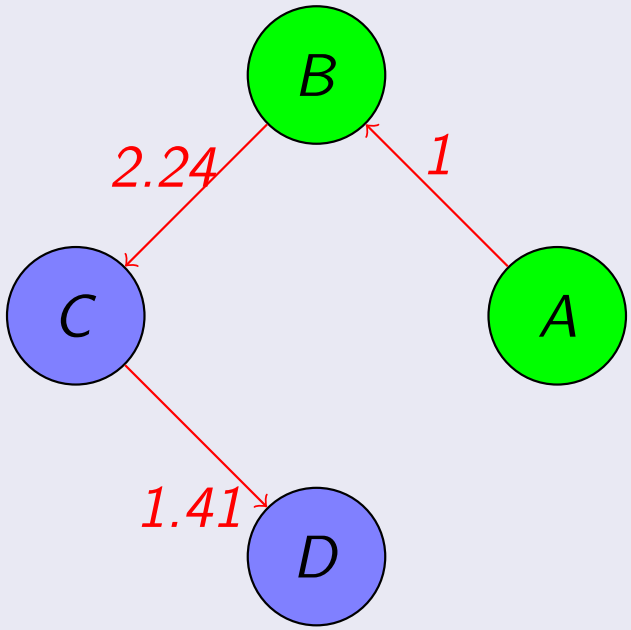
DFS traversal



TSP with Approximation Algorithm

MST under DFS traversal

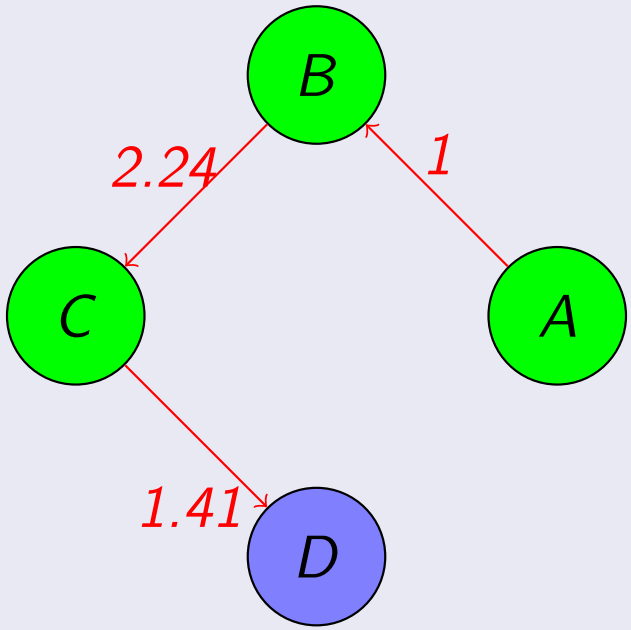
DFS traversal



TSP with Approximation Algorithm

MST under DFS traversal

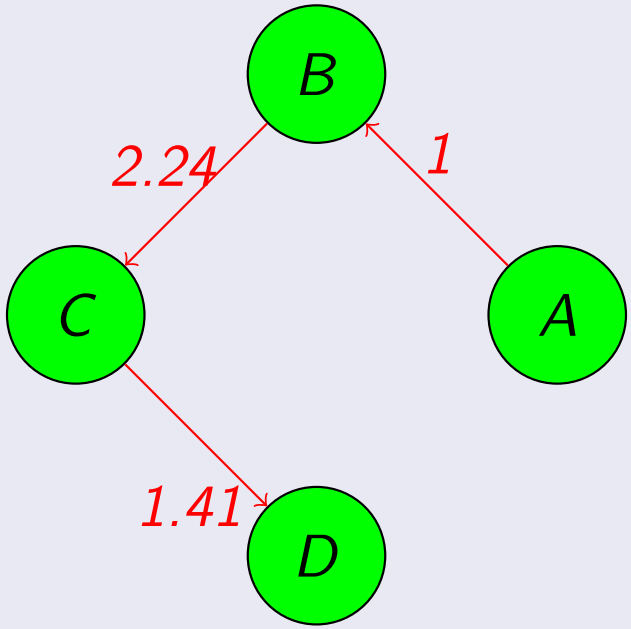
DFS traversal



TSP with Approximation Algorithm

MST under DFS traversal

DFS traversal



The full walk of above tree A $\rightarrow B \rightarrow C \rightarrow D$

$$Cost_{\text{MST}} = Weight_{AB} + Weight_{BC} + Weight_{CD}$$
$$= 1 + 2.24 + 1.41$$
$$= 4.65$$

TSP with Approximation Algorithm

Facts

- $Cost_{\textit{Best possible Travelling Salesman tour}} \geq Cost_{MST}$.
- *The definition of MST says, it is a minimum cost tree that connects all vertices.*
- $Cost_{\textit{Best possible Travelling Salesman tour}} \leq 2 \cdot Cost_{MST}$.
- *Every edge of MST is visited at most twice.*

TSP with Approximation Algorithm

Time Complexity

Time Complexity

Overall time complexity is dominated by the MST construction step, which is typically O(E log V)

Comparison between Exact and Approximate Solution

Exact solution from DP

$Result_{DP} = 4.65$

Approximate solution from AA

$Result_{AA} = 2 \cdot Cost_{MST} = 9.3$

Ratio of two Sol^n

$Ratio = \frac{Result_{AA}}{Result_{DP}} = 2$


Shafaet Ashraf.
Dynamic programming: From novice to advanced.
Website, 2024.


GeeksforGeeks.
Approximate solution for travelling salesman problem using mst.
GeeksforGeeks, 2024.


Wikipedia.
Traveling salesman problem, 2024.

THE END