TA Contest
INFO 6205 PSA

Lesson 1: Greedy Algorithms II

Akshit Kallepalli
002771603

Expectations for the Role of TA

The primary focus of this class being problem-solving, I aim to provide engaging problems for quizzes as part of my TA responsibilities.

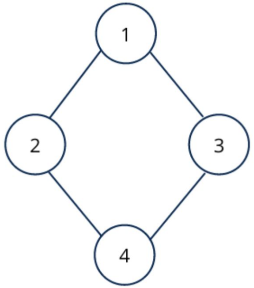
I aim to maintain a professional and punctual approach during TA hours.

I aim to ensure proper explanations for every deducted mark during the assignment grading process.

If I'm not proficient in a particular topic, my aim is to attend the class to ensure I can assist students effectively.

Spanning Trees

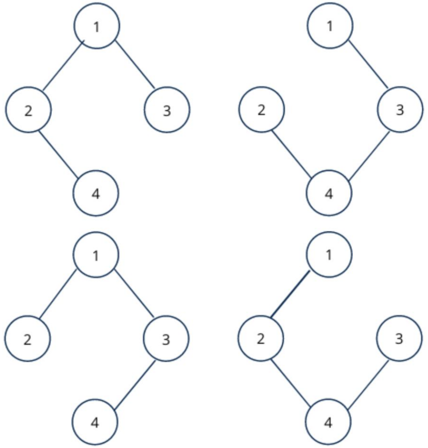
Spanning Tree



Definition:
A spanning tree of a connected graph is a subgraph that is a tree and spans all the vertices of the original graph.

Properties:
- Minimum Number of Edges: (V - 1) edges for a graph with V vertices.
- No Cycles: A spanning tree doesn't contain any cycles.

Finding a Spanning Tree:
- Depth-First Search (DFS): Traverse the graph and add edges to form a tree.
- Breadth-First Search (BFS): Traverse level by level, adding edges.
- Prim's Algorithm and Kruskal's Algorithm: Greedy approaches for finding spanning trees.



Importance in Network Design:
- Efficient communication and resource utilization.
- Redundancy for fault tolerance.

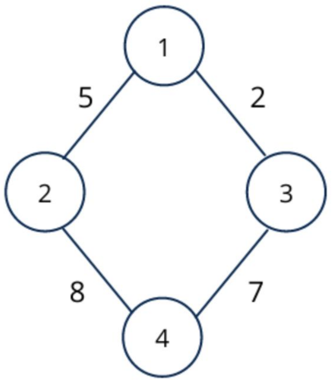
Minimum Spanning Tree

Definition:
- MST in a weighted graph is a tree with the least total edge weight.

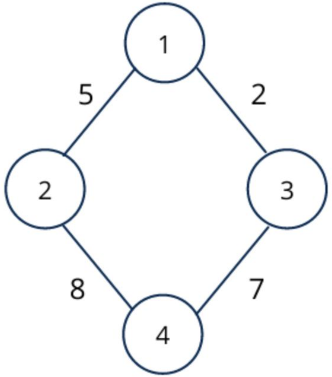
Characteristics:
- Connects all vertices while minimizing total weight.
- No cycles, maintains a tree structure. Properties:
- Contains (V - 1) edges for V vertices

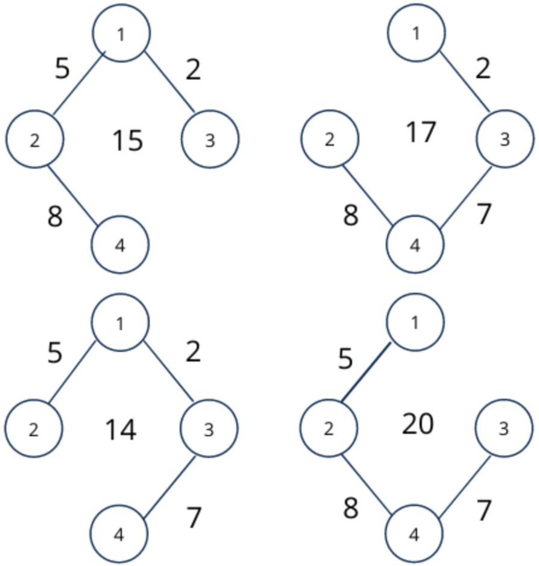
Finding MST:
- Algorithms: Kruskal's and Prim's.
- Kruskal's: Sorts edges, adds by weight without cycles.
- Prim's: Grows from vertex, adds min-weight edges.

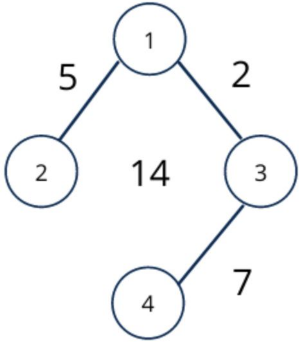
Importance:
- Efficient Networks: Optimizes communication, resource use.
- Infrastructure: Efficient links, resource saving.
- Applications: Networks, circuits, transport.



Minimum Spanning Tree







Greedy Algorithms

Introduction:

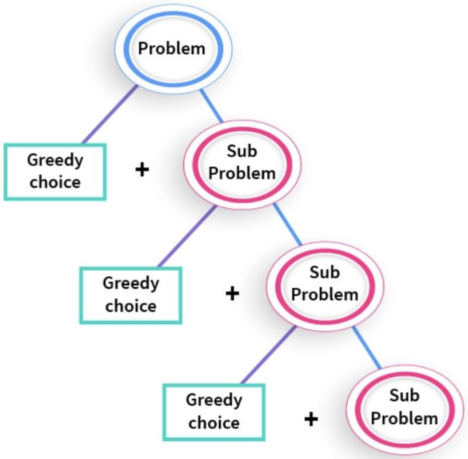
- Greedy algorithms make locally optimal choices at each step to achieve a global optimum.
- Often used for optimization problems.

Characteristics:

- Greedy Choice Property: A global optimum can be reached by selecting a locally optimum choice at each step.
- Optimal Substructure: An optimal solution to the problem contains optimal solutions to its subproblems.

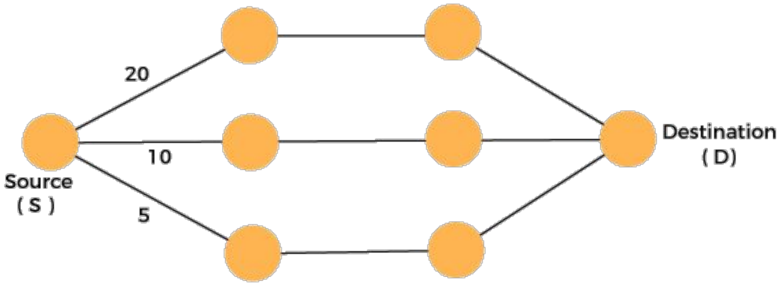
Pros and Cons of Greedy Algorithms:

- Pros: Simplicity, efficiency, and can work well for certain problems.
- Cons: Might not always provide the optimal solution for all problems.



Types of Greedy Algorithms

- **Prim's Algorithm**
- **Kruskal's Algorithm**
- **Dijkstra's Algorithm**
- Activity Selection
- Fractional Knapsack
- Huffman Coding
- Interval Scheduling
- Coin Change
- Greedy Coloring



Prim’s Algorithm

Prim's Algorithm

History

- Prim's Algorithm independently created by Vojtěch Jarník and Robert C. Prim.

- Developed as MST solution in 1930 and 1957, respectively.

- Originated for efficient electrical networks, later applied widely.

- Employs greedy approach, selecting smallest-weight edges step by step.

- Holds key role in graph theory education, aids network planning and optimization.

Prim's Algorithm

Introduction

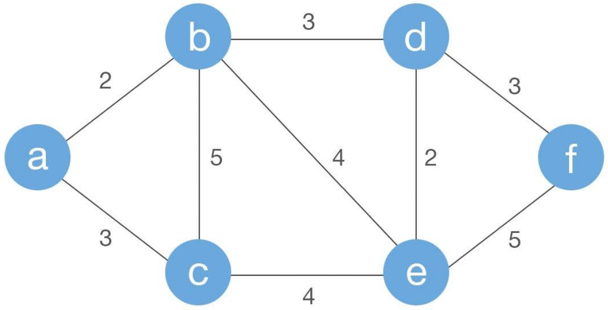
- Prim's Algorithm: Greedy approach for global optimum.

- Guarantees MST in connected, weighted, undirected graphs with non-negative edge weights.

- Used in network design, transportation planning, and operations research.

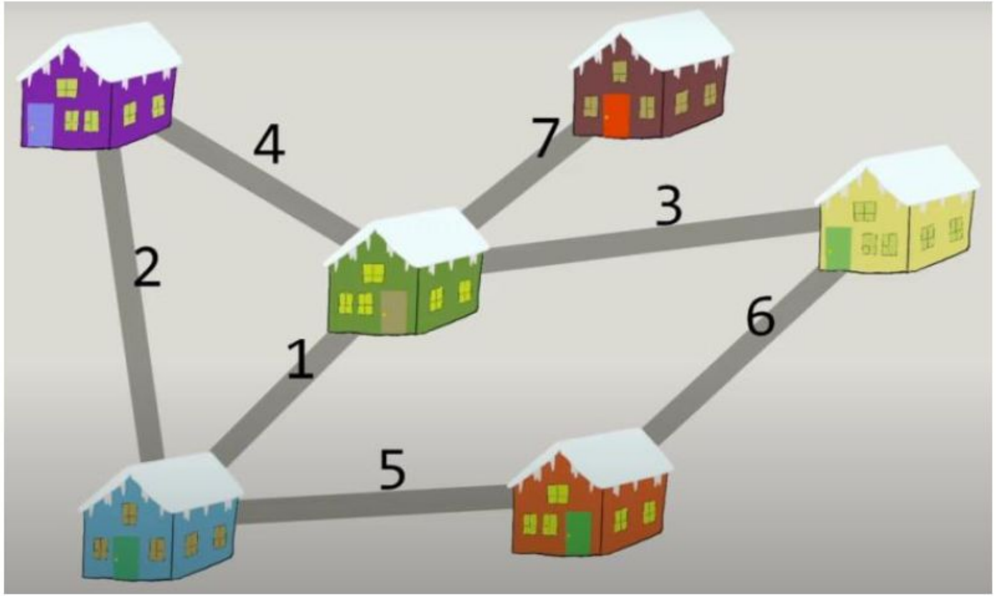
- Starts from any vertex, adds nearest unvisited vertex to MST.

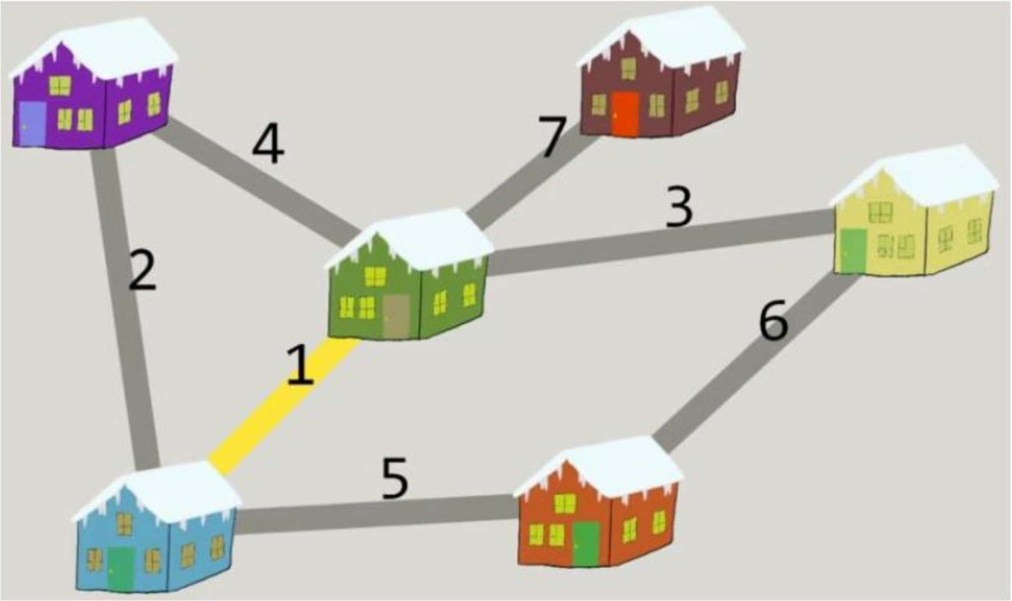
- Chooses smallest-weight edge, iterates until all vertices in MST.

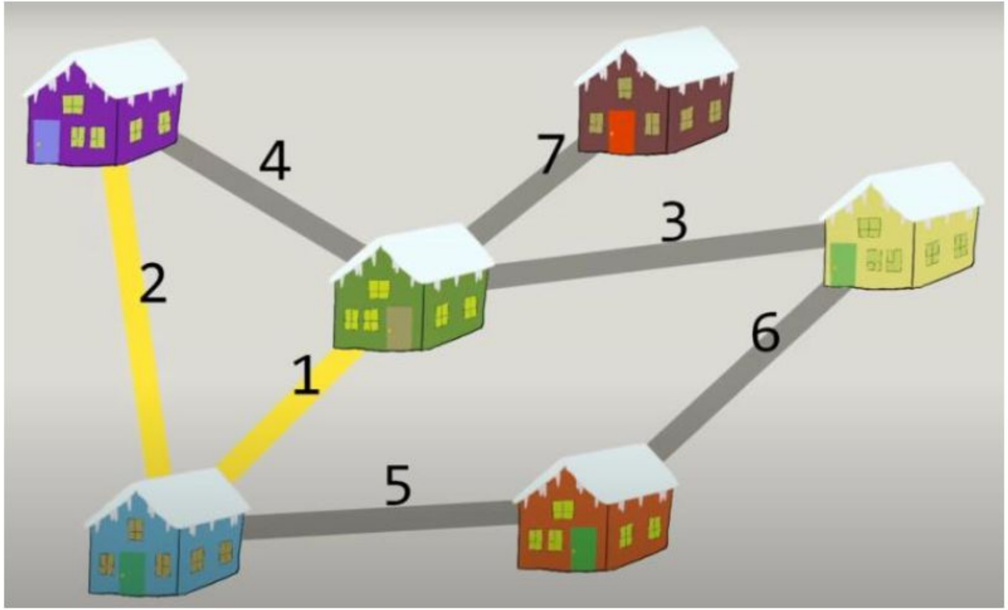


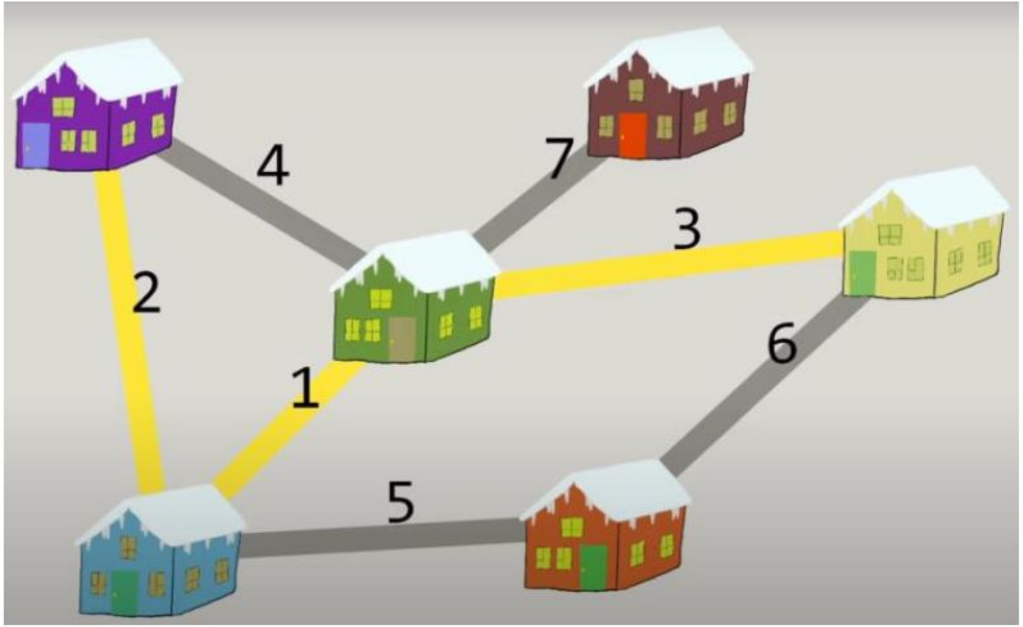
Prim's Algorithm

Implementation

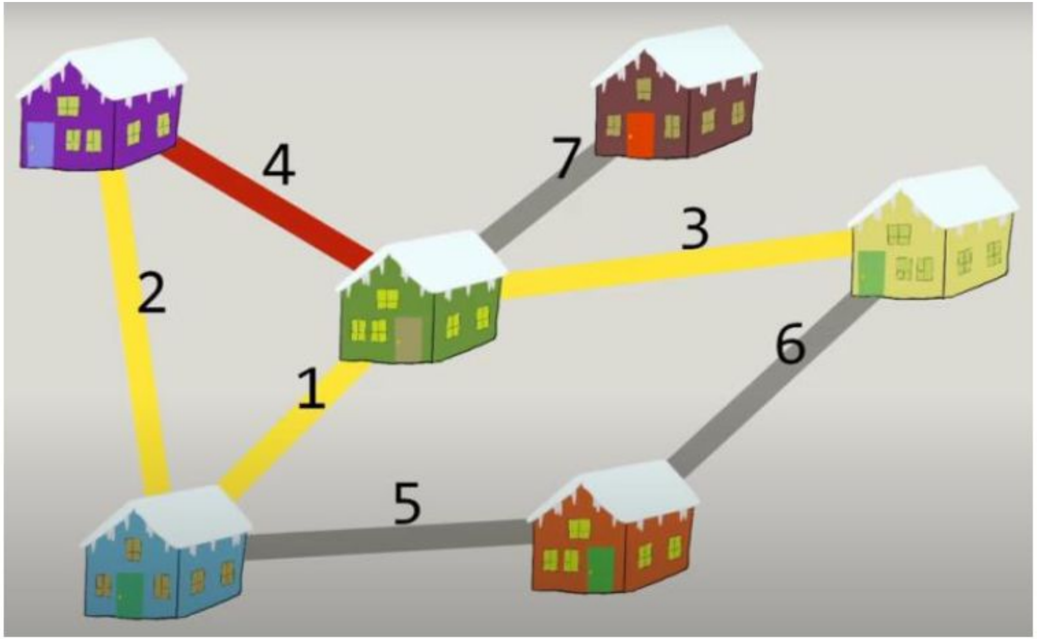


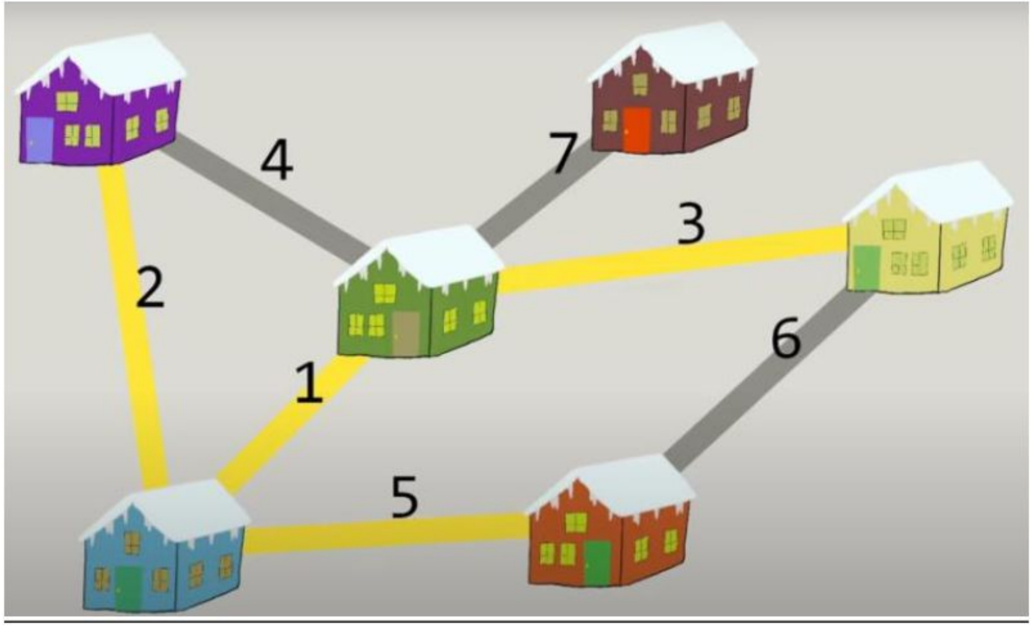

4
7
3
2
6
1
5

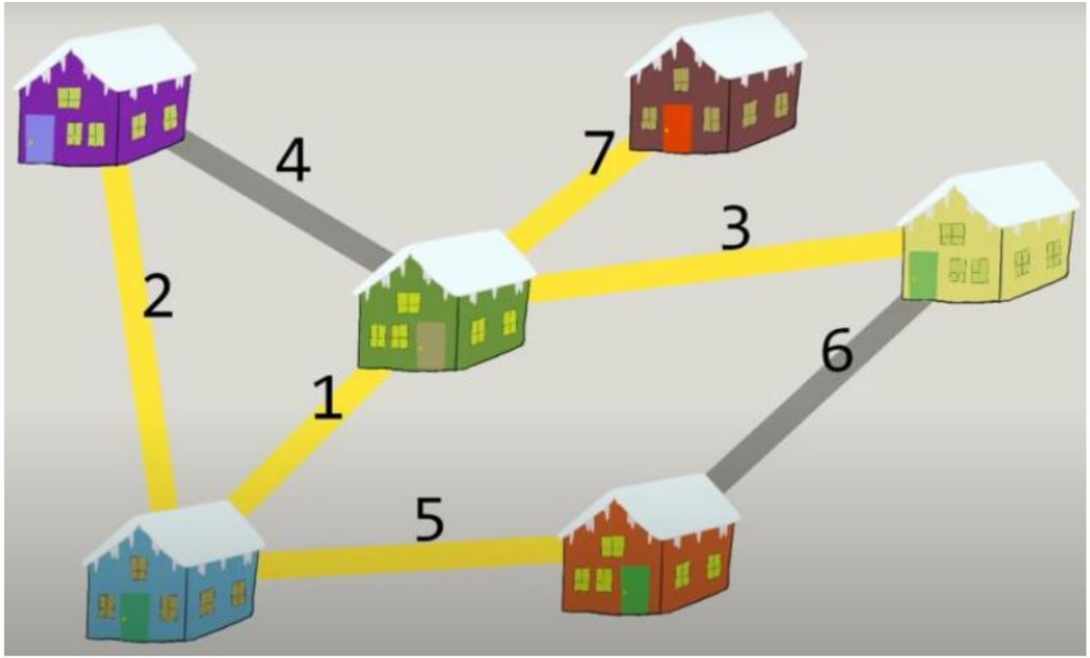

4
7
3
2
1
6
5



4
7
3
2
6
1
5

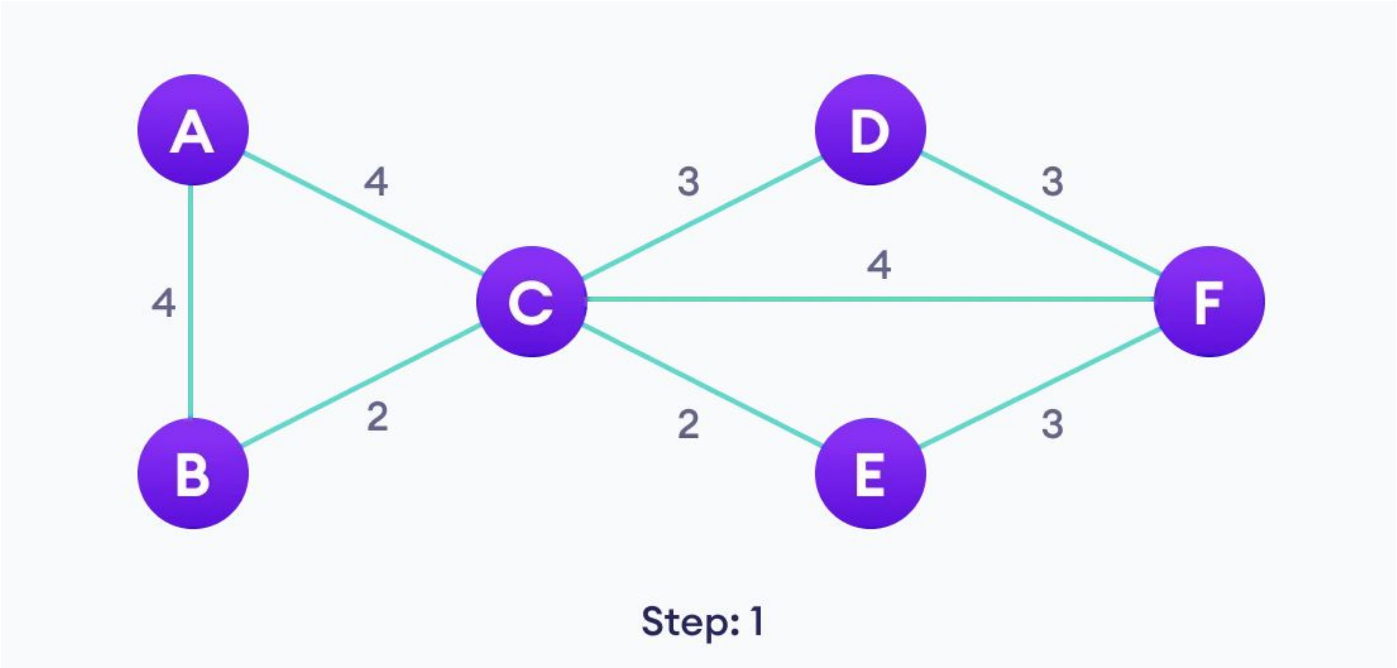

4
7
3
2
6
1
5


4
7
3
2
6
1
5

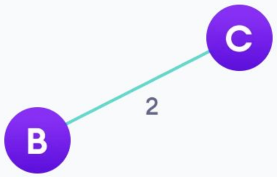

4
7
3
2
1
6
5

Prim's Algorithm

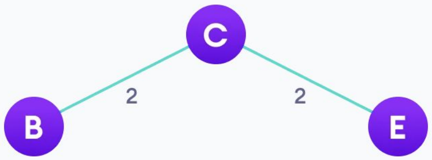
Example



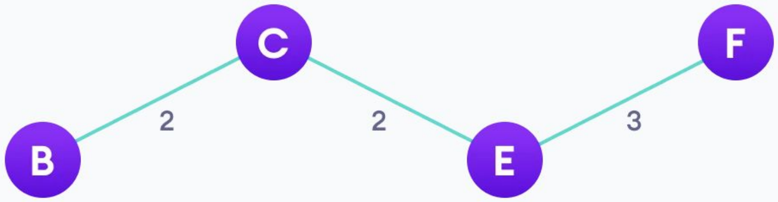
Step: 2



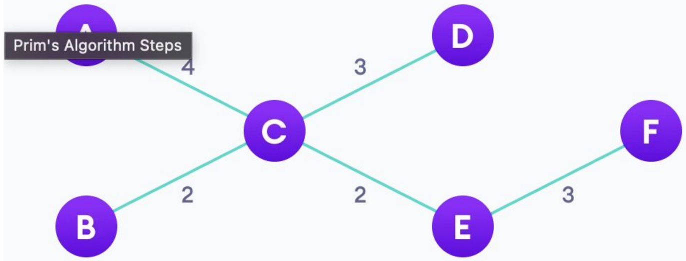
Step: 3



Step: 4



Step: 5



Step: 6

Prim's Algorithm

Complexity

Time Complexity:

O(E Log V) where, E is the number of edges and V is the number of vertices.

Space Complexity:

O(E+V) where, V is the number of vertices and E is the number of edges.

Prim's Algorithm

Real World applications

- Network infrastructure planning: Designs efficient and cost-effective communication networks, minimizing connection costs.

- Transportation network optimization: Plans road, rail, and public transport systems for reduced travel expenses.

- Resource allocation and supply chains: Optimizes distribution, minimizing resource usage and associated costs.

- Urban planning and utility lines: Lays out utility lines in cities, reducing construction expenses and disruptions.

- Travelling salesman problem: Determines the most efficient route for a salesman to visit multiple locations, minimizing the total distance traveled and optimizing the path.

Kruskal’s Algorithm

Kruskal’s Algorithm

History

- Kruskal's Algorithm independently created by Joseph Kruskal and George L. Miller.

- Developed as MST solution in 1956 by Joseph Kruskal and 1957 by George L. Miller.

- Initially applied in transportation planning and circuit design.

- Utilizes a greedy strategy, sorting edges by weight and incorporating them gradually.

- Significant in graph theory education and contributes to efficient network design.

Kruskal’s Algorithm

Introduction

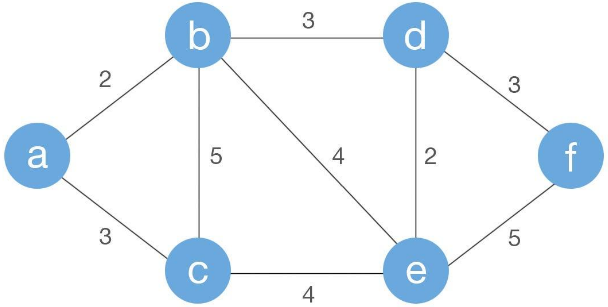
- Kruskal's Algorithm: Greedy approach for finding minimum spanning trees (MST).

- Applicable to connected, weighted, undirected graphs with non-negative edge weights.

- Begins with isolated vertices as individual MST components.

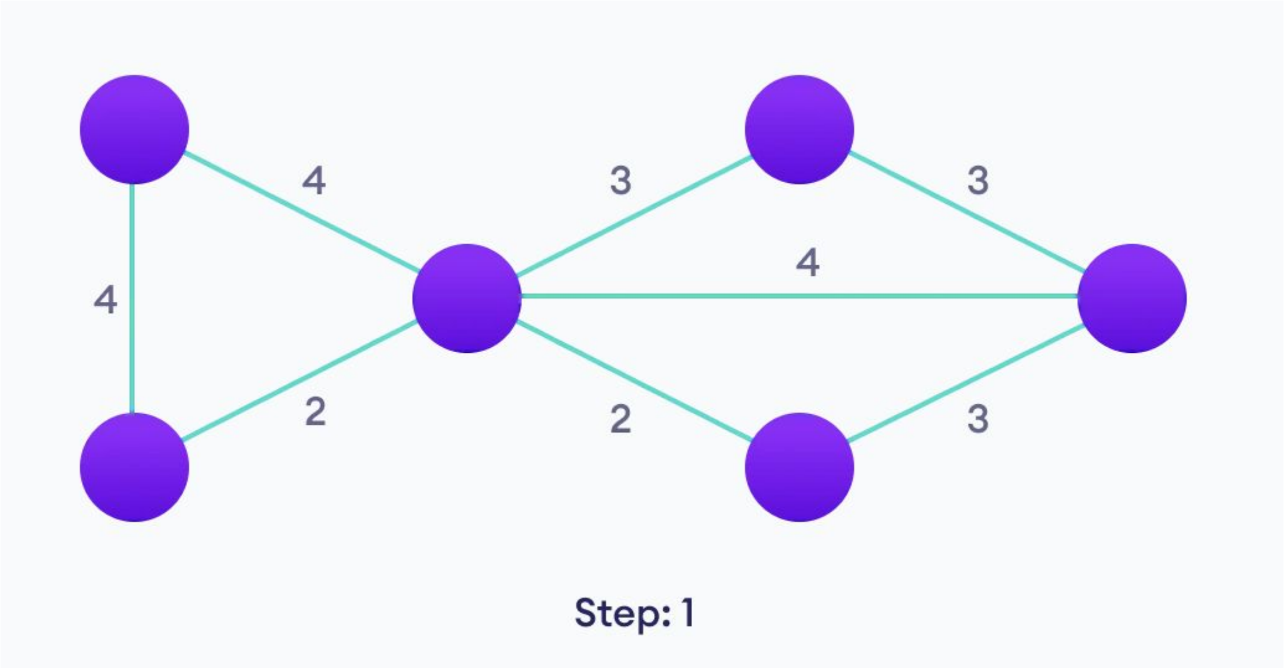
- Sorts edges by weight and iterates, adding edges to MST if they connect separate components.

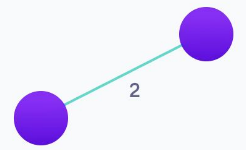
- Selects smallest-weight edges that don't create cycles until all vertices are included in MST.



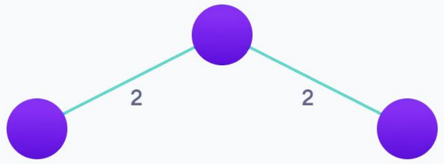
Kruskal's Algorithm

Implementation

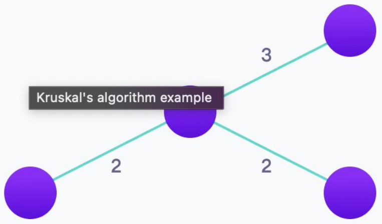




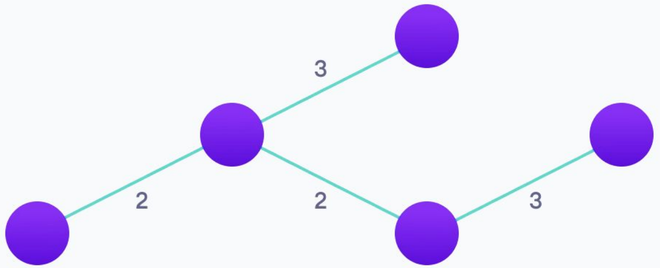
Step: 2



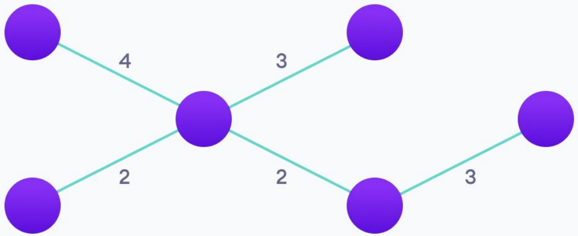
Step: 3



Step: 4



Step: 5



Step: 6

Kruskal's Algorithm

Complexity

Time Complexity:

O(E Log E) where, E is the number of edges.

Space Complexity:

O(E+V) where, V is the number of vertices and E is the number of edges.

Kruskal’s Algorithm

Real World applications

- Telecommunications network design: Optimal pathways for efficient data transmission.

- Network infrastructure planning: Efficient and cost-effective communication networks.

- Transportation network optimization: Reduced travel expenses for road, rail, and public transport systems.

- Resource allocation and supply chains: Minimized resource usage and distribution costs.

- Urban planning and utility lines: Reduced construction expenses and disruptions in cities.

Dijkstra's Algorithm

Dijkstra’s Algorithm

History

- Dijkstra's Algorithm created by Dutch computer scientist Edsger W. Dijkstra in 1956.

- Initially designed for solving the shortest path problem in networks of arbitrary lengths.

- Later became a fundamental concept in various applications, including routing in computer networks.

- Utilizes a greedy strategy, iteratively selecting the vertex with the smallest tentative distance.

- Significantly contributes to graph theory education and real-world pathfinding challenges.

Dijkstra's Algorithm

Introduction

- Dijkstra's Algorithm: Greedy approach for finding shortest paths.

- Guarantees shortest path in graphs with non-negative edge weights.

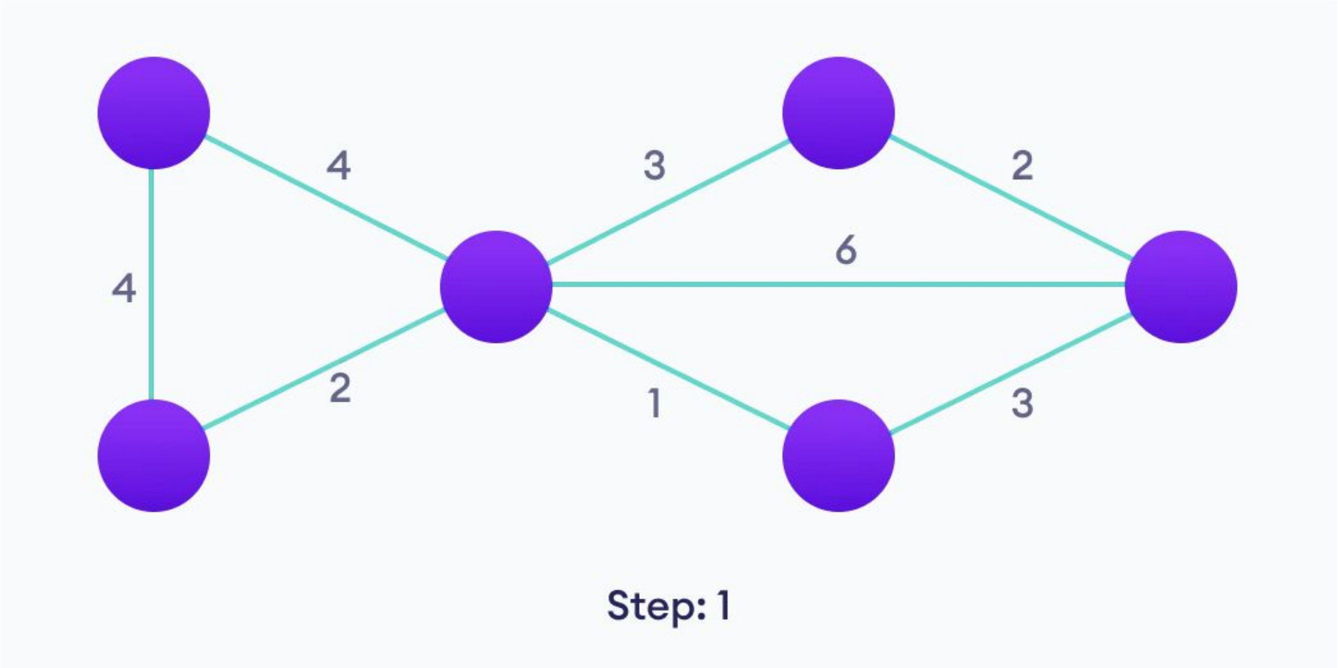
- Used in routing, navigation systems, and network protocols.

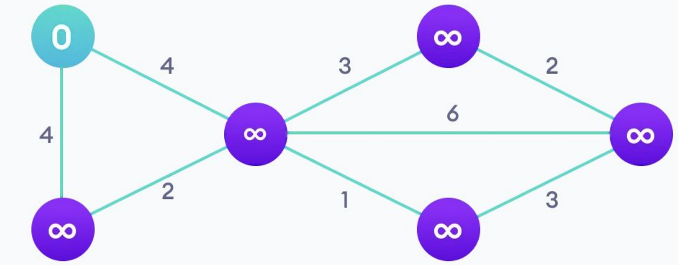
- Starts from a source vertex, gradually explores outward to find shortest paths.

- Chooses vertex with smallest tentative distance, iterates until destination reached.

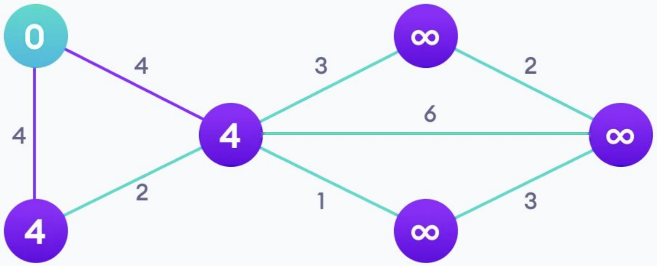
Dijkstra’s Algorithm

Implementation

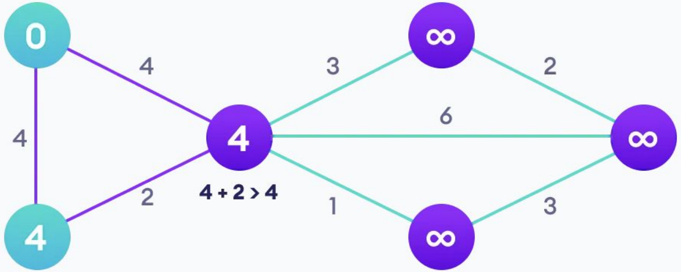




Step: 2

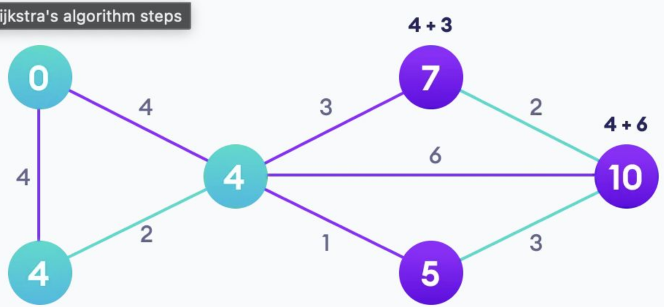


Step: 3

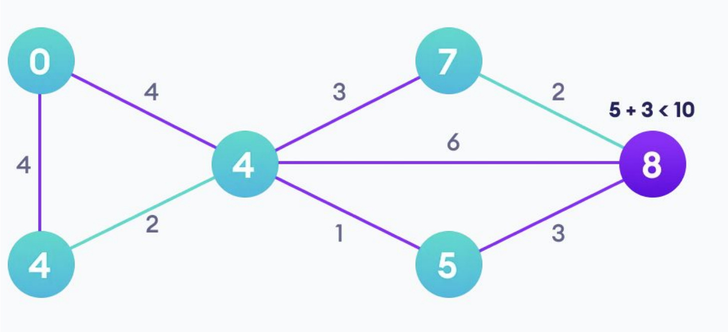


Step: 4

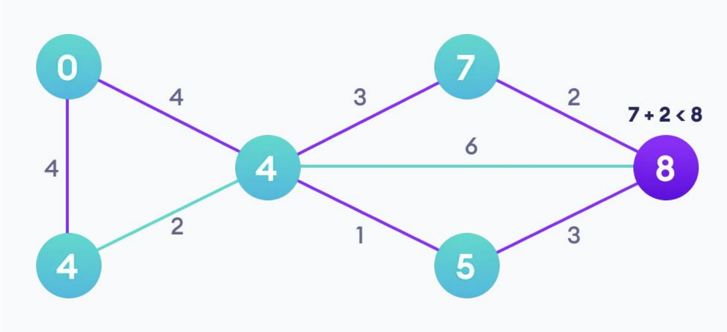
Dijkstra's algorithm steps



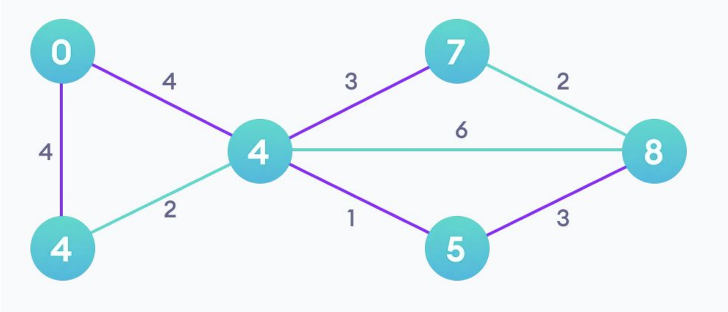
Step: 5



Step: 6



Step: 7



Step: 8

Dijkstra's Algorithm

Complexity

Time Complexity:

O(E Log V) where, E is the number of edges and V is the number of vertices.

Space Complexity:

O(V) where, V is the number of vertices.

Dijkstra's Algorithm

Real World applications

1. Digital Mapping : Finds shortest routes between locations in mapping services like Google Maps.

2. Social Networking: Suggests friends efficiently based on shortest paths in large social networks.

3. Telephone Network: Determines optimal paths for data transmission in telephone networks.

4. IP Routing (OSPF): Used in routers to find best paths for data packets in networks.

5. Flight Agenda: Calculates earliest arrival time for flights between airports.

6. LAN File Server: Minimizes hops to designate a file server in local networks.

7. Robotic Path Planning: Guides drones and robots along shortest paths for efficient navigation.

Quiz Questions

1) Which of the following statements about Minimum Spanning Trees (MST) is/are correct?

A. In a graph with distinct edge weights, the MST is unique.

B. Kruskal's algorithm can be used to find the MST of a weighted, connected graph.

C. Prim's algorithm always selects the edge with the highest weight at each step.

D. Dijkstra's algorithm is a variation of Prim's algorithm for finding the MST.

Answers: A, B

2) Which of the following statements about Dijkstra's algorithm is/are true?

A. It can handle graphs with negative edge weights.

B. It is used to find the shortest path between two specific vertices.

C. Dijkstra's algorithm may fail to produce correct results when used on a graph with negative edge weights.

D. It guarantees finding the shortest paths even in graphs with cycles.

Answers: B, C

3) In the context of graphs and algorithms, which of the following statements are true?

A. Dijkstra's algorithm can be used to find a Minimum Spanning Tree (MST).

B. Prim's algorithm is more suitable than Dijkstra's algorithm for finding the shortest path between two specific vertices

C. Dijkstra's algorithm modifies the edge weights during its execution.

D. Prim's algorithm always maintains a forest of trees during its execution.

Answers: B, D

4) If all edge weights in a graph are equal, which algorithm would be most efficient for finding the minimum spanning tree?

A. Prim's algorithm

B. Kruskal's algorithm

C. Dijkstra's algorithm

D. Bellman-Ford algorithm

Answer: A

5) Which of the following statements about Prim's algorithm for finding a minimum spanning tree is/are correct?

A. Prim's algorithm is a greedy algorithm.

B. It can be used to find the shortest path in a graph.

C. The algorithm starts with a single vertex and adds the closest edge in each step.

D. Prim's algorithm guarantees a minimum spanning tree for both directed and undirected graphs.

Answers: A, C

References

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press. ISBN 978-0-262-03384-8.

2. Spanning Tree. (2019, November 24). How Do You Calculate a Minimum Spanning Tree? [Video]. YouTube. https://www.youtube.com/watch?v=Yldkh0aOEcg&ab_channel=SpanningTree

3. Python heapq Module Documentation. (n.d.). Retrieved from https://docs.python.org/3/library/heapq.html

4. Dijkstra's Algorithm - Programiz. (n.d.). Retrieved from https://www.programiz.com/dsa/dijkstra-algorithm

5. Prim's Algorithm - Programiz. (n.d.). Retrieved from https://www.programiz.com/dsa/prim-algorithm

6. Kruskal's Algorithm - Programiz. (n.d.). Retrieved from https://www.programiz.com/dsa/kruskal-algorithm

Thank you