Dushimire Aine
Class:Y2D
DSA Homework

1.Meaning of Greedy Algorthim

A greedy algorithm is an algorithmic approach that makes the locally optimal choice at each step with the hope of finding a global optimum. It solves problems by selecting the best option available at the moment without reconsidering previous choices, often leading to efficient but not always optimal solutions.

Examples of greedy algorithms:
1.kruskal's algorithm
2.prim's algorithm
3.Dijkstra's algorithm
4.Huffman Coding
5.Fractional Knapsack Problem

2.Meaning of Spanning Tree

Spanning tree of a connected,undirected graph is a subgraph that is acyclic and connected and includes all the vertices of the original graph with a subset of its edges. For a graph with n vertices,a spanning tree has exactly n-1 edges

3.Meaning of Minimum Spanning Tree (MST) of Graph

It is a a weighted,connected,undirected graph with the minimum total edge weight .it connects all vertices with the least possible sum of edges weights,ensuring no cycles.

4.Meaning of Shortest Path in Graph and differentiate with MST

It is a graph between two vertices with the minimum total weight of the edges traversed eg: Dijkstra's algorithm finds the shortest path from a source vertex to all other vertices in a weighted graph with non-negative edge weights.

Difference Between Shortest Path and MST:

Objective:

- MST: Minimizes the total weight of edges to connect all vertices, forming a tree with n−1 n-1 n−1 edges.

- Shortest Path: Minimizes the total weight of edges to travel from one specific vertex to another.

Structure:

- MST: Always a tree (acyclic, connects all vertices).
- Shortest Path: Can be a path (may include cycles in the graph, focuses on two vertices).

Algorithm:

- MST: Uses Kruskal's or Prim's algorithm.
- Shortest Path: Uses Dijkstra's or Bellman-Ford algorithm.

Application:

- MST: Network design (e.g., minimizing cable length).
- Shortest Path: Routing (e.g., GPS navigation).

5.Kruskal's Algorithm

Kruskal's algorithm finds the minimum spanning tree of a connected,undirected,weighted graph it works by:

1.sorting all edges by weight in non-decreasing order

2.iteratively selecting the smallest edge that does not form a cycle with the edges already included.

3.using a disjoint-set (union-find) data structure to detect cycles efficiently

4.continuing until n-1 edges are selected ,where n is the number of vertices

Pseudocode

Algorithm Kruskal(G):

Input: Weighted undirected graph G = (V, E)

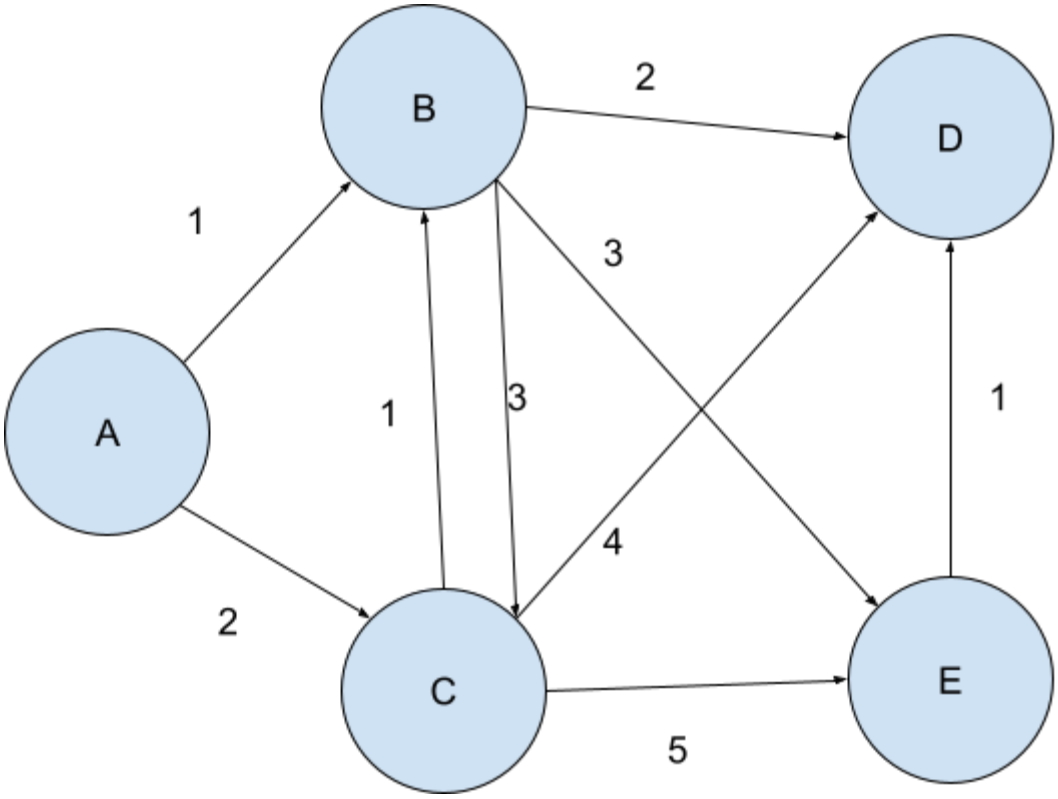
Output: Minimum Spanning Tree T

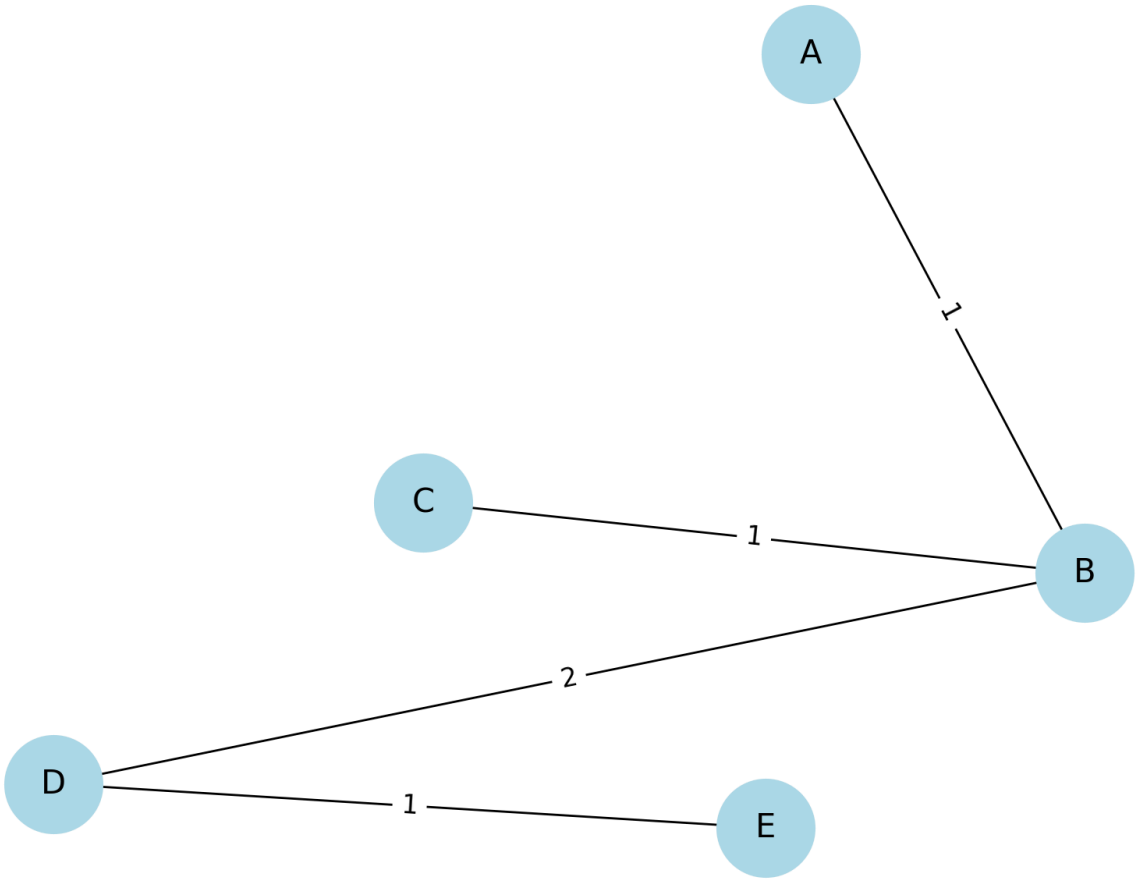
T = ∅ // Initialize empty MST

Sort edges E by weight in non-decreasing order

```
Initialize disjoint-set for all vertices


for each edge (u, v, weight) in sorted E:
    if find(u) != find(v):  // u and v are in different sets (no cycle)
        Add (u, v) to T
        Union(u, v)  // Merge sets of u and v
return T
```


B
2
D
1
3
A
1
3
1
4
2
C
E
5



6. Prim’s Algorithm

Description (Natural Language)

Prim’s algorithm finds the Minimum Spanning Tree (MST) of a connected, undirected, weighted graph. It works by:

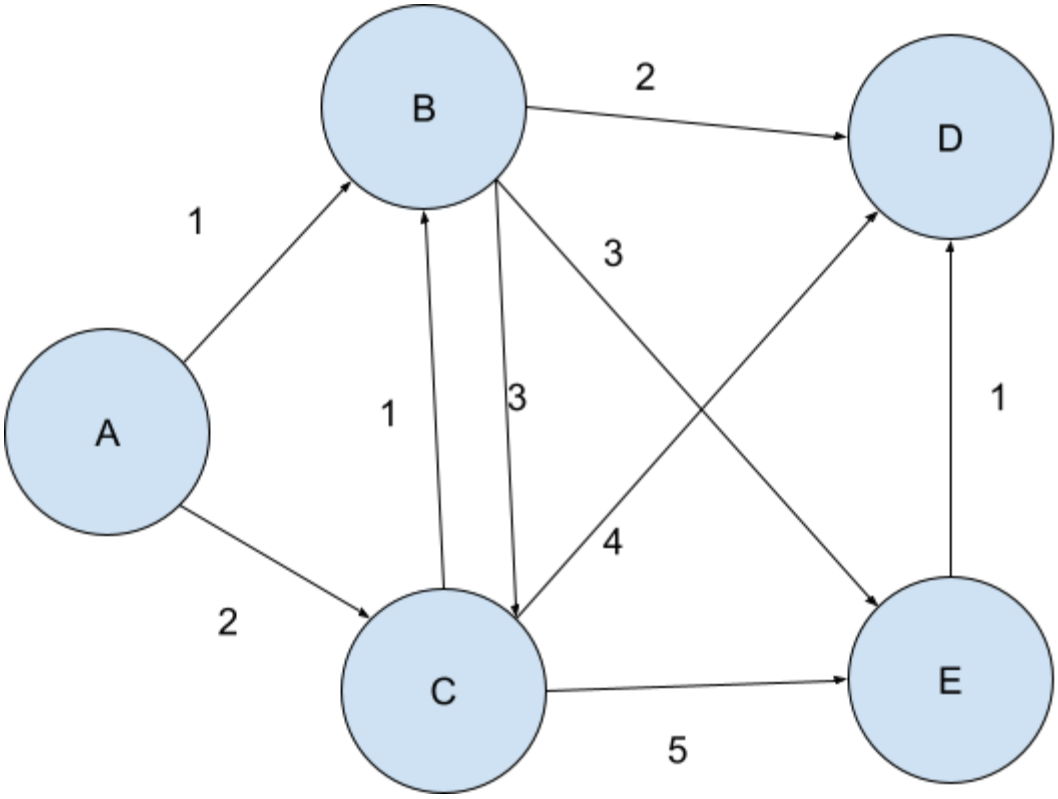
1. Starting from an arbitrary vertex and adding it to the MST.
2. Maintaining a priority queue of edges connected to the MST, prioritized by weight.
3. Repeatedly selecting the smallest edge connecting an MST vertex to a non-MST vertex.
4. Adding the new vertex and edge to the MST until all vertices are included.

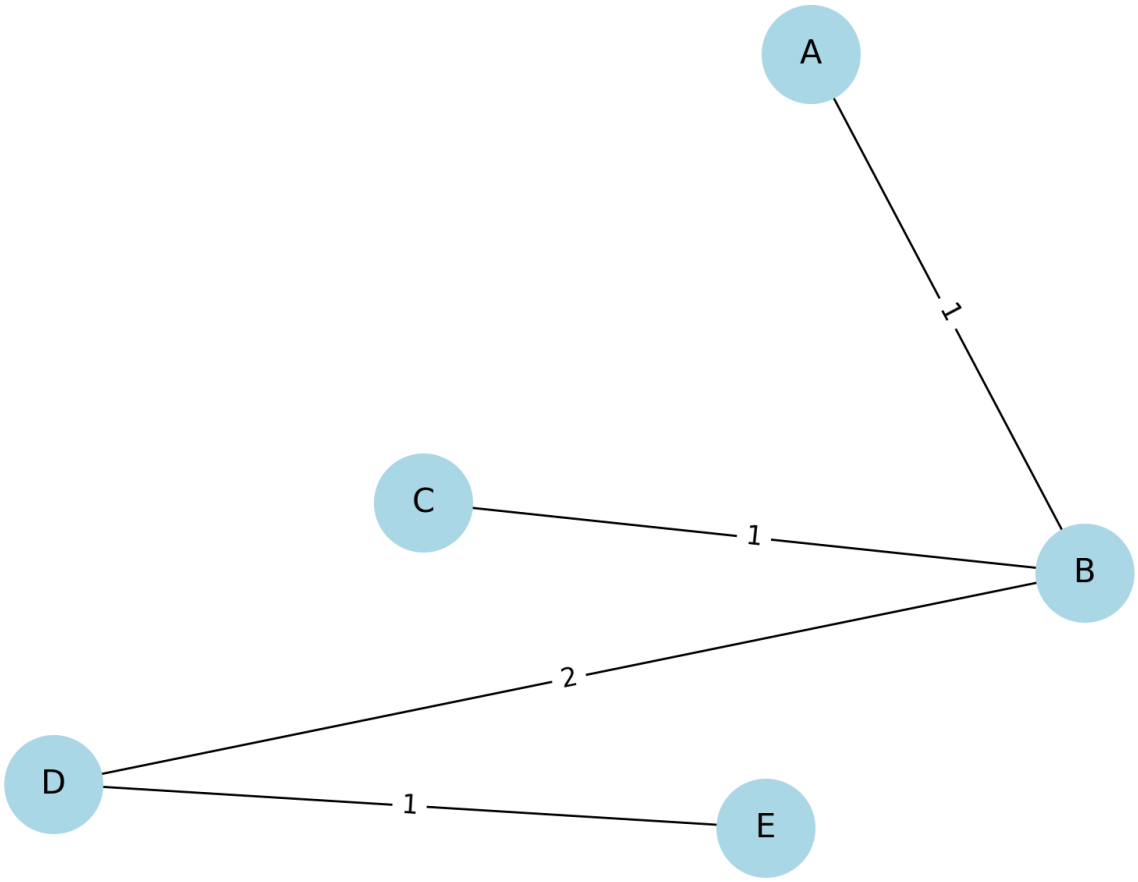
Pseudocode

Algorithm Prim(G):
  Input: Weighted undirected graph G = (V, E)
  Output: Minimum Spanning Tree T

```
T = ∅  // Initialize empty MST
Initialize priority queue PQ of edges
visited = ∅  // Set of visited vertices
Select arbitrary vertex s, mark as visited
Add all edges from s to PQ

while PQ is not empty and |T| < V-1:
    (u, v, weight) = PQ.extract_min()
    if v is not visited:
        Add (u, v) to T
        Mark v as visited
        Add all unvisited edges from v to PQ
return T
```


B
2
D
1
3
A
1
3
1
4
2
C
E
5



3. Dijkstra's Algorithm

Description (Natural Language)

Dijkstra's algorithm finds the shortest path from a single source vertex to all other vertices in a weighted graph with non-negative edge weights. It works by:

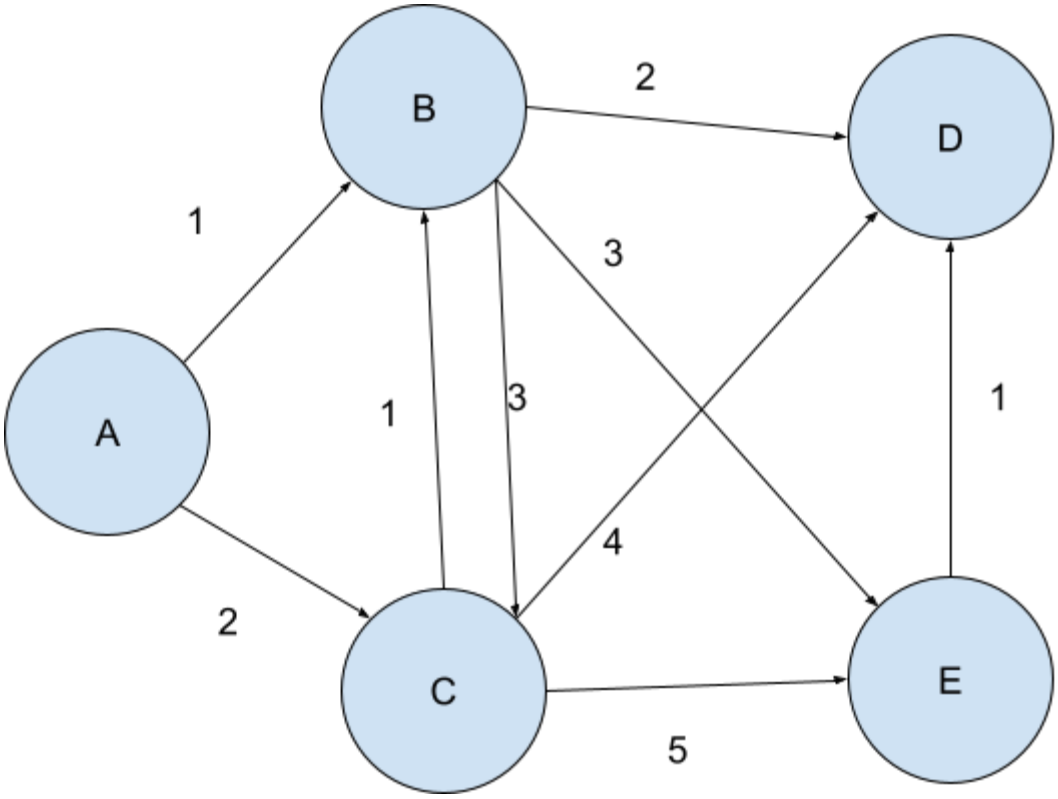
1. Maintaining a priority queue of vertices, prioritized by their tentative distance from the source.
2. Starting with the source vertex at distance 0 and all others at infinity.
3. Repeatedly selecting the vertex with the smallest tentative distance, updating distances to its neighbors if a shorter path is found.
4. Continuing until all vertices are processed or the target is reached.

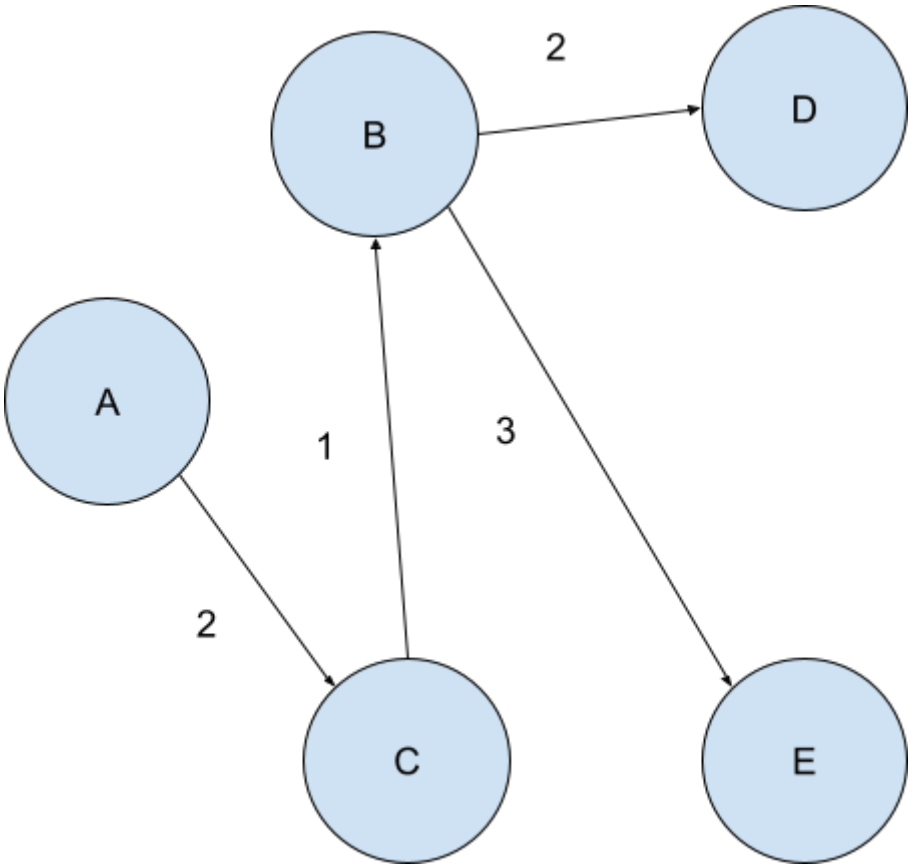
Pseudocode

```
Algorithm Dijkstra(G, source):
    Input: Weighted graph G = (V, E), source vertex
    Output: Shortest distances and paths from source

    dist[source] = 0, dist[other] = ∞
    prev[v] = undefined for all v
    PQ = priority queue of all vertices, prioritized by dist
    visited = ∅

    while PQ is not empty:
        u = PQ.extract_min()
        if u in visited: continue
        Add u to visited
        for each neighbor v of u:
            alt = dist[u] + weight(u, v)
            if alt < dist[v]:
                dist[v] = alt
                prev[v] = u
                Update PQ with new dist[v]
    return dist, prev
```


B
2
D
1
3
A
1
3
1
4
2
C
E
5


2
B
D
A
1
3
2
C
E