Practical Work number 4:

Problem number 6, Manea Robert-Petrisor Group 914.

Prim's algorithm is a popular algorithm used to find the minimum spanning tree (MST) of a weighted undirected graph. The minimum spanning tree is a tree that spans all the vertices of the graph while minimizing the total weight or cost of the tree.

```
def prim(graph,start_vertex):
        visited = {start_vertex}
        edges = []
        current_node = start_vertex

        while len(visited) < graph.numberOfVertices():
            for neighbor in graph.Outbound_of_vertex(current_node):
                weight = graph.Costs_of_vertices(current_node, neighbor)
                if neighbor not in visited:
                    heappush(edges, (weight, current_node, neighbor))

            while edges:
                weight, u, v = heappop(edges)
                if v not in visited:
                    visited.add(v)
                    yield u, v, weight
                    current_node = v
                    break
```

Start with a variable **start_vertex** (this is the vertex inputted by the user) and add it to the set of visited vertices.

Create an empty list named **edges** to store potential edges for expanding the MST.

While there are unvisited vertices:

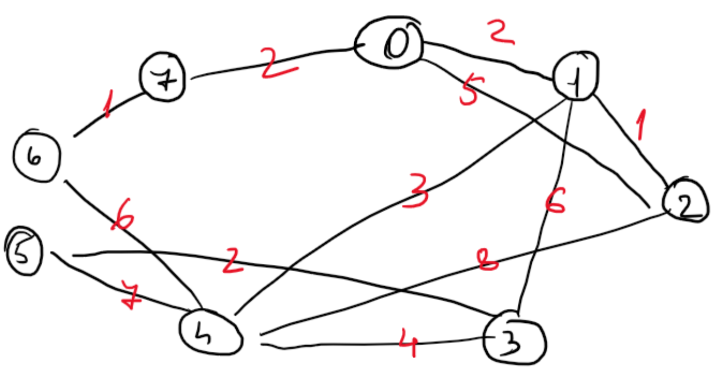
- For each neighbor of the current vertex, calculate the weight of the edge connecting them.
- If the neighbor is not visited, add the edge to the **edges** list.
- Choose the edge with the minimum weight from the **edges** list.
- If the destination vertex of the chosen edge is not visited, add it to the set of visited vertices and yield the edge as part of the MST.
- Set the current vertex to the destination vertex of the chosen edge.

Repeat until all vertices are visited.

In summary, Prim's algorithm starts from a specified vertex and repeatedly selects the minimum-weight edge that connects visited vertices to unvisited vertices. It grows the minimum spanning tree by adding these edges until all vertices are included.

Manual Executions:

Graph :
Vertices : 0 -> 7
Edges : (first vertex, second vertex, weight) :
(0,1,2), (0,2,5), (1,2,1), (1,3,6), (1,4,3), (2,4,8),
(3,4,4), (3,5,2), (4,5,7), (4,6,6), (6,7,1), (7,0,2)

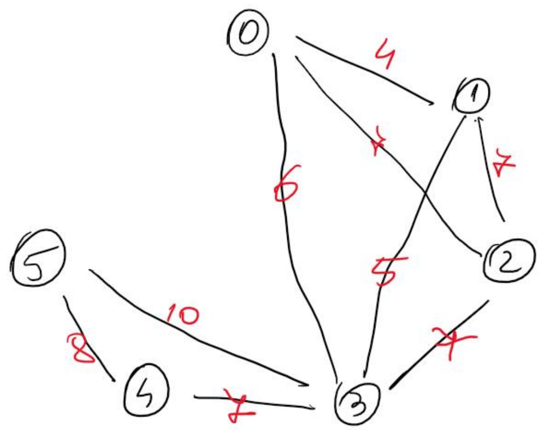


Iteration	Visited Vertices	Current node	Edges in heap	Adding edges to heap	Selected Edge
1	0	0	[ ]	[(2,0,1),(5,0,2),(2,0,7)]	(0,1,2)
2	0,1	1	[(2,0,7),(5,0,2)]	[(1,1,2),(3,1,4),(2,0,7),(6,1,3),(5,0,2)]	(1,2,1)
3	0,1,2	2	[(2,0,7),(3,1,4),(5,0,2),(6,1,3)]	[(2,0,7),(3,1,4),(5,0,2),(6,1,3),(8,2,4)]	(0,7,2)
4	0,1,2,7	7	[(3,1,4)(6,1,3),(5,0,2),(8,2,4)]	[(1,7,6),(3,1,4),(5,0,2),(8,2,4),(6,1,3)]	(7,6,1)

5	0, 1, 2, 6, 7	6	[(7, 1, 4), (6, 1, 3), (5, 0, 2), (8, 2, 4)]	[(3, 4, 4), (6, 1, 3), (5, 0, 2), (8, 2, 4), (6, 6, 4)]	(4, 4, 3)
6	{0, 1, 2, 4, 6, 7}	4	[(5, 0, 2), (6, 1, 3), (6, 6, 4), (8, 2, 4)]	[(4, 4, 3), (5, 0, 2), (6, 6, 4), (8, 2, 4), (6, 1, 3), (7, 4, 5)]	(4, 3, 4)
7	{0, 1, 2, 3, 4, 6, 7}	3	[(5, 0, 2), (6, 1, 3), (6, 6, 4), (8, 2, 4), (7, 4, 5)]	[(2, 3, 5), (6, 1, 3), (5, 0, 2), (8, 2, 4), (7, 4, 5), (6, 6, 4)]	(3, 5, 2)

Edges + Costs

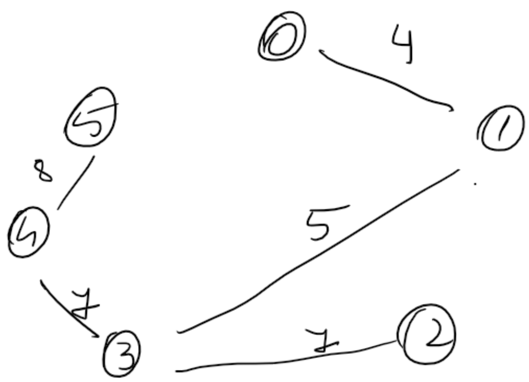
(0, 1) : 4
(0, 2) : 7
(1, 2) : 7
(1, 3) : 5
(0, 3) : 6
(2, 3) : 7
(3, 4) : 7
(4, 5) : 8
(5, 3) : 10



1st MST :

Total cost :

31



2nd MST :

Total cost :

31