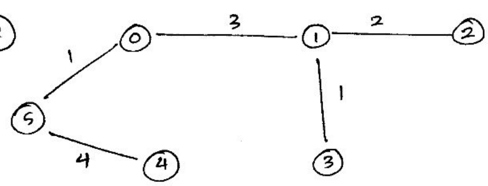
200650U – Thilakarathne D.L.J - CS2023 – Lab 11

GitHub link - https://github.com/LasithaJananjaya/CS2023-Data-Structures-and-Algorithms---Workspace/tree/main/in-class-lab11

Q1

	0	1	2	3	4	5
0	0	3	0	0	0	1
1	3	0	2	1	10	0
2	0	2	0	3	0	5
3	0	1	3	0	5	0
4	0	10	0	5	0	4
5	1	0	5	0	4	0

Q2



Cost = 11

main.cpp

```cpp
// This program demonstrates Prim's algorithm for finding the
    Minimum Spanning Tree (MST) of a graph.

#include <iostream>
#include <vector>
#include <climits>

using namespace std;

// Function to find the vertex with the minimum key value
    among the vertices not yet included in MST
int findMinKey(vector<int> &key, vector<bool> &mst_set, int V)
{
int minimum_key = INT_MAX;
int mininum_index = -1;

// Iterate through all vertices to find the minimum key value
for (int i = 0; i < V; ++i)
{
    if (!mst_set[i] && key[i] < minimum_key)
    {
```

Output

```
/tmp/RUad7WRFGG.o
Enter the number of vertices: 6
Enter the adjacency matrix:
0 3 0 0 0 1
3 0 2 1 10 0
0 2 0 3 0 5
0 1 3 0 5 0
0 10 0 5 0 4
1 0 5 0 4 0
Edge    Weight
0 - 1   3
1 - 2   2
1 - 3   1
5 - 4   4
0 - 5   1
```

(Q4) Yes, It is essential that all the weights within the graph ~~exh~~ exhibit uniqueness, indicating that each individual edge possesses a distinct & exclusive weight.

(Q5)

Prim's Algorithm	Kruskal's Algorithm
Time complexity depends on the data structure used	Time complexity depends on the sorting of edges. ~~based on their costs~~
$O(V^2) \rightarrow$ Adjacency matrix	$O(E \log E)$ or $O(E \log V)$
$O((V+E)\log V) \rightarrow$ binary heap	Performs union-find operations to detect and merge disjoint sets.

Typically, Prim's algorithm tends to exhibit higher time complexity compared to Kruskal's algorithm. (Particularly in dense graphs.). Kruskal's algorithm proves to be efficient for sparse graphs, where the number of edges are comparatively lower.