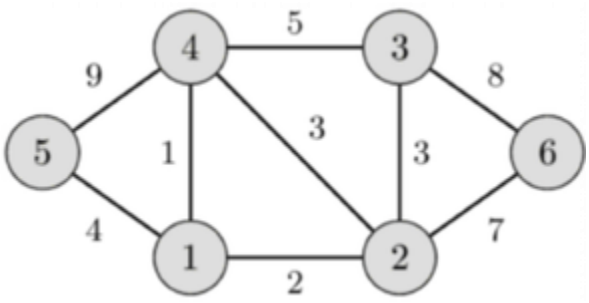
EXERCISE 11

A H K Swarun

21BPS1252

QUESTION 1:

Consider the below graph and Implement the Minimum Cost spanning tree (any algorithm) and Dijikstra's Algorithm.



Minimum Cost Spanning Tree using Krushkal Algorithm:

```c
#include <stdio.h>
#define MAX 30
typedef struct edge {
  int u, v, w;
} edge;

typedef struct edge_list {
  edge data[MAX];
  int n;
} edge_list;

edge_list elist;

int Graph[MAX][MAX], n;
edge_list spanlist;

void kruskalAlgo();
int find(int belongs[], int vertexno);
void applyUnion(int belongs[], int c1, int c2);
```

```
void sort();
void print();

// Applying Krushkal Algo
void kruskalAlgo() {
  int belongs[MAX], i, j, cno1, cno2;
  elist.n = 0;

  for (i = 1; i < n; i++)
    for (j = 0; j < i; j++) {
      if (Graph[i][j] != 0) {
        elist.data[elist.n].u = i;
        elist.data[elist.n].v = j;
        elist.data[elist.n].w = Graph[i][j];
        elist.n++;
      }
    }

  sort();

  for (i = 0; i < n; i++)
    belongs[i] = i;

  spanlist.n = 0;

  for (i = 0; i < elist.n; i++) {
    cno1 = find(belongs, elist.data[i].u);
    cno2 = find(belongs, elist.data[i].v);

    if (cno1 != cno2) {
      spanlist.data[spanlist.n] = elist.data[i];
      spanlist.n = spanlist.n + 1;
      applyUnion(belongs, cno1, cno2);
    }
  }
}

int find(int belongs[], int vertexno) {
  return (belongs[vertexno]);
}

void applyUnion(int belongs[], int c1, int c2) {
  int i;

  for (i = 0; i < n; i++)
```

```
    if (belongs[i] == c2)
      belongs[i] = c1;
}

// Sorting algo
void sort() {
  int i, j;
  edge temp;

  for (i = 1; i < elist.n; i++)
    for (j = 0; j < elist.n - 1; j++)
      if (elist.data[j].w > elist.data[j + 1].w) {
        temp = elist.data[j];
        elist.data[j] = elist.data[j + 1];
        elist.data[j + 1] = temp;
      }
}

// Printing the result
void print() {
  int i, cost = 0;

  for (i = 0; i < spanlist.n; i++) {
    printf("\n%d - %d : %d", spanlist.data[i].u, spanlist.data[i].v, span-
list.data[i].w);
    cost = cost + spanlist.data[i].w;
  }

  printf("\nSpanning tree cost: %d", cost);
}

int main() {
  int i, j, total_cost;

  n = 7;
  for(i=1;i<n;i++){
    for(j=1;j<n;j++){
        scanf("%d",&Graph[i][j]);
    }
  }

  kruskalAlgo();
  print();
}
```

EXECUTION and OUTPUT:

```
PS C:\Users\ahks4> cd Desktop
PS C:\Users\ahks4\Desktop> gcc exe11.c
PS C:\Users\ahks4\Desktop> .\a.exe
0 2 0 1 4 0
2 0 3 3 0 7
0 3 0 5 0 8
1 3 5 0 9 0
4 0 0 9 0 0
0 7 8 0 0 0

4 - 1 : 1
2 - 1 : 2
3 - 2 : 3
5 - 1 : 4
6 - 2 : 7
Spanning tree cost: 17
```

Dijkstra's Algorithm:

CODE:

```c
#include<stdio.h>
#include<conio.h>
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX], int n, int startnode);
int main() {
  int G[MAX][MAX], i, j, n, u;
  printf("Enter no. of vertices:");
  scanf("%d", & n);
  printf("\nEnter the adjacency matrix:\n");
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      scanf("%d", & G[i][j]);
  printf("\nEnter the starting node:");
  scanf("%d", & u);
```

```
  dijkstra(G, n, u);
  return 0;
}

void dijkstra(int G[MAX][MAX], int n, int startnode) {
  int cost[MAX][MAX], distance[MAX], pred[MAX];
  int visited[MAX], count, mindistance, nextnode, i, j;
  for (i = 0; i < n; i++)
    for (j = 0; j < n; j++)
      if (G[i][j] == 0)
        cost[i][j] = INFINITY;
      else
        cost[i][j] = G[i][j];
          for (i = 0; i < n; i++) {
    distance[i] = cost[startnode][i];
    pred[i] = startnode;
    visited[i] = 0;
  }
  distance[startnode] = 0;
  visited[startnode] = 1;
  count = 1;
  while (count < n - 1) {
    mindistance = INFINITY;
    for (i = 0; i < n; i++)
      if (distance[i] < mindistance && !visited[i]) {
        mindistance = distance[i];
        nextnode = i;
      }
    visited[nextnode] = 1;
    for (i = 0; i < n; i++)
      if (!visited[i])
        if (mindistance + cost[nextnode][i] < distance[i]) {
          distance[i] = mindistance + cost[nextnode][i];
          pred[i] = nextnode;
        }
    count++;
  }

  for (i = 0; i < n; i++)
    if (i != startnode) {
      printf("\nDistance of node%d=%d", i, distance[i]);
      printf("\nPath=%d", i);
      j = i;
      do {
        j = pred[j];
```

```
        printf("<-%d", j);
      } while (j != startnode);
    }
}
```

EXECUTION and OUTPUT:

```
PS C:\Users\ahks4> cd Desktop
PS C:\Users\ahks4\Desktop> gcc exe11.c
PS C:\Users\ahks4\Desktop> .\a.exe
Enter no. of vertices:6

Enter the adjacency matrix:
0 2 0 1 4 0
2 0 3 3 0 7
0 3 0 5 0 8
1 3 5 0 9 0
4 0 0 9 0 0
0 7 8 0 0 0

Enter the starting node:0

Distance of node1=2
Path=1<-0
Distance of node2=5
Path=2<-1<-0
Distance of node3=1
Path=3<-0
Distance of node4=4
Path=4<-0
Distance of node5=9
Path=5<-1<-0
PS C:\Users\ahks4\Desktop>
```