DAA Prac 09

Name: Vijiyant Tanaji Shejwalkar

Reg no: 2020BIT057

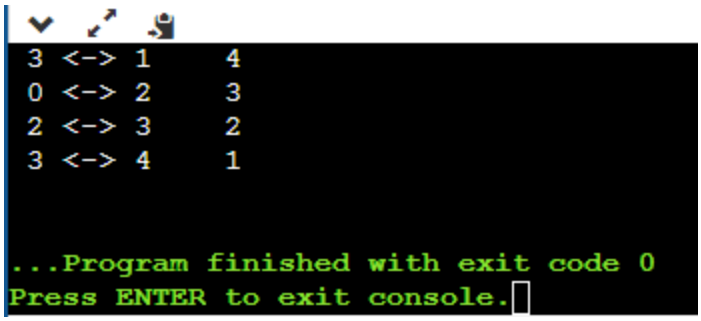
1. Prims algoritham

```
#include <stdio.h>
#include <limits.h>
#define vertices 5  /*Define the number of vertices in the graph*/
/* create minimum_key() method for finding the vertex that has minimum key-value
and that is not added in MST yet */
int minimum_key(int k[], int mst[])
{
   int minimum  = INT_MAX, min,i;

   /*iterate over all vertices to find the vertex with minimum key-value*/
   for (i = 0; i < vertices; i++)
      if (mst[i] == 0 && k[i] < minimum )
         minimum = k[i], min = i;
   return min;
}
/* create prim() method for constructing and printing the MST.
The g[vertices][vertices] is an adjacency matrix that defines the graph for MST.*/
void prim(int g[vertices][vertices])
{
   /* create array of size equal to total number of vertices for storing the MST*/
   int parent[vertices];
   /* create k[vertices] array for selecting an edge having minimum weight*/
   int k[vertices];
   int mst[vertices];
   int i, count,edge,v; /*Here 'v' is the vertex*/
   for (i = 0; i < vertices; i++)
   {
      k[i] = INT_MAX;
```

```
    mst[i] = 0;
  }
  k[0] = 0; /*It select as first vertex*/
  parent[0] = -1;   /* set first value of parent[] array to -1 to make it root of MST*/
  for (count = 0; count < vertices-1; count++)
  {
    /*select the vertex having minimum key and that is not added in the MST yet
from the set of vertices*/
    edge = minimum_key(k, mst);
    mst[edge] = 1;
    for (v = 0; v < vertices; v++)
    {
      if (g[edge][v] && mst[v] == 0 && g[edge][v] <  k[v])
      {
        parent[v]  = edge, k[v] = g[edge][v];
      }
    }
   }
   /*Print the constructed Minimum spanning tree*/
   printf("\n Edge \t  Weight\n");
   for (i = 1; i < vertices; i++)
   printf(" %d <-> %d    %d \n", parent[i], i, g[i][parent[i]]);

}
int main()
{
  int g[vertices][vertices] = {{0, 0, 3, 0, 0},
                        {0, 0, 10, 4, 0},
                        {3, 10, 0, 2, 6},
                        {0, 4, 2, 0, 1},
                        {0, 0, 6, 1, 0},
                        };
  prim(g);
  return 0;
}
```

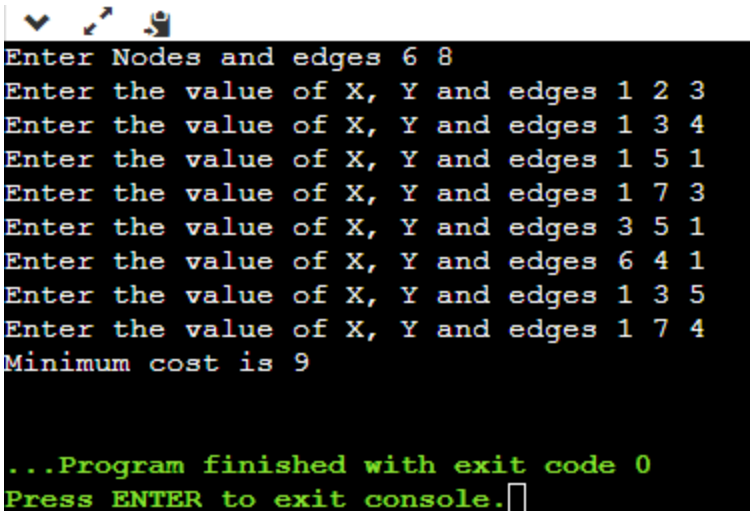


2. Kruskal's algorithm

```
#include <iostream>
#include <algorithm>
using namespace std;
const int MAX = 1e4 + 5;
int id[MAX], nodes, edges;
pair <long long, pair<int, int> > p[MAX];
void init()
{
   for(int i = 0;i < MAX;++i)
      id[i] = i;
}
int root(int x)
{
   while(id[x] != x)
   {
      id[x] = id[id[x]];
```

```
        x = id[x];
    }
    return x;
}
void union1(int x, int y)
{
    int p = root(x);
    int q = root(y);
    id[p] = id[q];
}
long long kruskal(pair<long long, pair<int, int> > p[])
{
    int x, y;
    long long cost, minimumCost = 0;
    for(int i = 0;i < edges;++i)
    {
        x = p[i].second.first;
        y = p[i].second.second;
        cost = p[i].first;
        if(root(x) != root(y))
        {
            minimumCost += cost;
            union1(x, y);
        }
```

```
    }
    return minimumCost;
}
int main()
{
    int x, y;
    long long weight, cost, minimumCost;
    init();
    cout <<"Enter Nodes and edges";
    cin >> nodes >> edges;
    for(int i = 0;i < edges;++i)
    {
        cout<<"Enter the value of X, Y and edges";
    cin >> x >> y >> weight;
        p[i] = make_pair(weight, make_pair(x, y));
    }
    sort(p, p + edges);
    minimumCost = kruskal(p);
    cout <<"Minimum cost is "<< minimumCost << endl;
    return 0;
}
```


Enter Nodes and edges 6 8
Enter the value of X, Y and edges 1 2 3
Enter the value of X, Y and edges 1 3 4
Enter the value of X, Y and edges 1 5 1
Enter the value of X, Y and edges 1 7 3
Enter the value of X, Y and edges 3 5 1
Enter the value of X, Y and edges 6 4 1
Enter the value of X, Y and edges 1 3 5
Enter the value of X, Y and edges 1 7 4
Minimum cost is 9
...Program finished with exit code 0
Press ENTER to exit console.