UNIVERSITY OF INFORMATION TECHNOLOGY AND SCIENCES (UITS)

LAB REPORT - 6

IT-214 : ALGORITHM LAB

Kruskal's Algorithm And Prim's Algorithm

Submitted To:

Sumaiya Akhtar Mitu
Lecturer,
Department of IT, UITS

Submitted By:

Name: Nazmul Zaman
Student ID:2014755055
Department of IT UITS

24 MAY 2022

Contents

1 Abstrac

In mathematics and computer science, an algorithm is a finite sequence of well-defined instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing.

2 Introduction

In this task we are going to learn about Minimum spainning tree and there types are : .

Kruskal's Algorithm :An algorithm to construct a Minimum Spanning Tree for a connected weighted graph. It is a Greedy Algorithm. The Greedy Choice is to put the smallest weight edge that does not because a cycle in the MST constructed so far.
Time comeplexity : O(ElogV))
Space complexity : O(E+V).

Prim's algorithm (also known as Jarník's algorithm) is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph..
Time comeplexity : O(V^2))
$Spacecomplexity : O(E + V)$.

3 Procedure

Kruskal's Algorithm :
Step 1: Sort all edges in increasing order of their edge weights.
Step 2: Pick the smallest edge.
Step 3: Check if the new edge creates a cycle or loop in a spanning tree.
Step 4: If it doesn't form the cycle, then include that edge in MST. Otherwise, discard it.
Step 5: Repeat from step 2 until it includes —V— - 1 edges in MST.

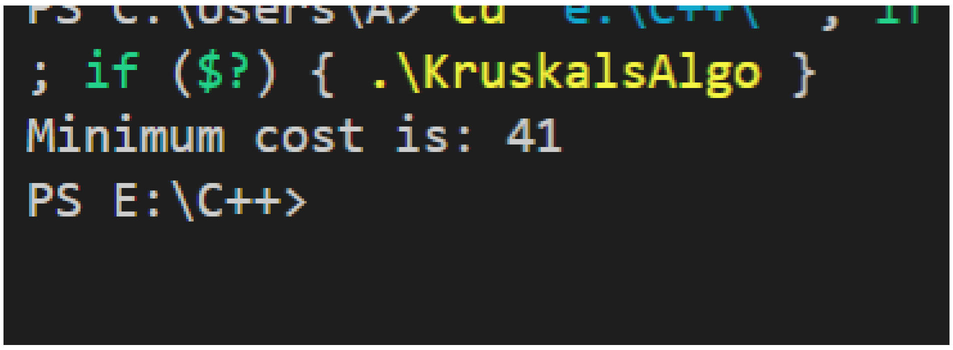
Prim's algorithm :
1.Initialize the minimum spanning tree with a vertex chosen at random.
2.Find all the edges that connect the tree to new vertices, find the minimum and add it to the tree
3.Keep repeating step 2 until we get a minimum spanning tre

4 Working methods

4.1 (Kruskals Algorithm

```

#define NAZMUL_ ZAMAN Bsc in IT

#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
const int MAX = 1e6-1;
int root[MAX];
const int nodes = 4, edges = 5;
pair <long long, pair<int, int> > p[MAX];

int parent(int a)
        //find the parent of the given node
{
    while(root[a] != a)
    {
        root[a] = root[root[a]];
        a = root[a];
    }
    return a;
}

void union_find(int a, int b)
      //check if the given two vertices are in the sa[U+FFFD]union[U+FFFD]
    or not
{
    int d = parent(a);
    int e = parent(b);
    root[d] = root[e];
}

long long kruskal()
{
    int a, b;
    long long cost, minCost = 0;
    for(int i = 0 ; i < edges ; ++i)
    {
        a = p[i].second.first;
        b = p[i].second.second;
        cost = p[i].first;
        if(parent(a) != parent(b))
    //only select edge if it does not create a cycle (ie the two
   nodes forming it have different root nodes)
        {
            minCost += cost;
            union_find(a, b);
        }
    }
    return minCost;
}

int main()
{
    int x, y;
    long long weight, cost, minCost;
    for(int i = 0;i < MAX;++i)
     //initialize the array groups
    {
        root[i] = i;
    }
    p[0] = make_pair(10, make_pair(0, 1));
    p[1] = make_pair(18, make_pair(1, 2));
    p[2] = make_pair(13, make_pair(2, 3));
    p[3] = make_pair(21, make_pair(0, 2));
    p[4] = make_pair(22, make_pair(1, 3));
    sort(p, p + edges);
    //sort the array of edges
    minCost = kruskal();
    cout << "Minimum cost is: "<< minCost << endl;
    return 0;
}
```

?



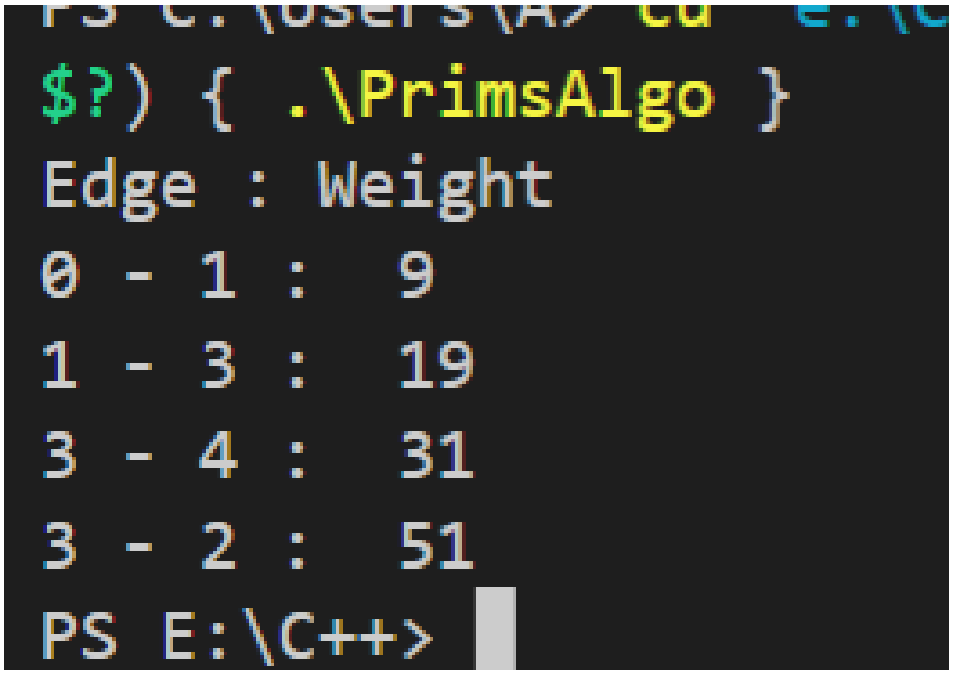
```
PS C:\Users\A> cd "e:\C++\" ; if
; if ($?) { .\KruskalsAlgo }
Minimum cost is: 41
PS E:\C++>
```

Figure 1: OUTPUT

4.2 Prims Algorithm)

```cpp
// Prim's Algorithm in C++
#define NAZMUL_ ZAMAN Bsc in IT

#include <cstring>
#include <iostream>
using namespace std;

#define INF 9999999

// number of vertices in grapj
#define V 5

// create a 2d array of size 5x5
//for adjacency matrix to represent graph

int G[V][V] = {
  {0, 9, 75, 0, 0},
  {9, 0, 95, 19, 42},
  {75, 95, 0, 51, 66},
  {0, 19, 51, 0, 31},
  {0, 42, 66, 31, 0}};

int main() {
  int no_edge;  // number of edge

  // create a array to track selected vertex
  // selected will become true otherwise false
  int selected[V];

  // set selected false initially
  memset(selected, false, sizeof(selected));

  // set number of edge to 0
  no_edge = 0;


  // choose 0th vertex and make it true
  selected[0] = true;

  int x;  //  row number
  int y;  //  col number

  // print for edge and weight
  cout << "Edge"
     << " : "
     << "Weight";
  cout << endl;
  while (no_edge < V - 1) {

    int min = INF;
    x = 0;
    y = 0;

    for (int i = 0; i < V; i++) {
      if (selected[i]) {
        for (int j = 0; j < V; j++) {
          if (!selected[j] && G[i][j]) {  // not in selected and there is an edge
            if (min > G[i][j]) {
              min = G[i][j];
              x = i;
              y = j;
            }
          }
        }
      }
    }
    cout << x << " - " << y << " :  " << G[x][y];
    cout << endl;
    selected[y] = true;
    no_edge++;
  }

  return 0;
}
```

?



```
$?) { .\PrimsAlgo }
Edge : Weight
0 - 1 :  9
1 - 3 :  19
3 - 4 :  31
3 - 2 :  51
PS E:\C++>
```

Figure 2: OUTPUT

5 Conclusion

In this lab report we learn about different type of greedy algorithm and how does it's work with deeply knowledge about kruskals ans prims algorithm and there implementation also different types of uses.We learn about real life example prims algo like Network for roads and Rail tracks connecting all the cities and kruskals algo like Landing cables,,TV Network,Tour Operation.

6 References

1.https://www.w3schools.com/algorithm/algo$_intro.asp$

2.https://www.javatpoint.com/algo$_intro.asp$

3.https://github.com/ZamanNazmul