TITLE OF THE MINI PROJECT

A MINI PROJECT REPORT

18CSC204J -Design and Analysis of Algorithms Laboratory

Submitted by

KOTHOLLA JASWANTH REDDY [RA2011028010132]

MARLA SAI RUTHWIK [RA2011028010124]

Under the guidance of

Dr. B Yamini

Assistant Professor, Department of Networking and Communication

In Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE ENGINEERING with specialization in CLOUD COMPUTING

DEPARTMENT OF NETWORKING AND COMMUNICATIONS

COLLEGE OF ENGINEERING AND TECHNOLOGY SRM

INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR- 603 203

JUNE - 2022

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this mini project report titled “Road Traffic Management System” is the bonafide work done by Kotholla Jaswanth Reddy RA2011028010132 and Marla Sai Ruthwik RA2011028010124 who carried out the mini project work and Laboratory exercises under my supervision for **18CSC204J -Design and Analysis of Algorithms Laboratory**. Certified further, that to the best of my knowledge the work reported herein does not form part of any other work.

Dr. B Yamini
ASSISTANT PROFESSOR
18CSC204J -Design and Analysis of Algorithms
Course Faculty
Department of Networking and Communications

Signature of the Internal Examiner-I **Signature of the Internal Examiner-II**

ABSTRACT

The dynamic road traffic management is based on the dynamic vehicle routing during peak hour traffic. The real time traffic management has become a hectic problem in daily life due to the increasing traffic, sudden accidents, and bottle necks due to various reasons. In Dynamic Vehicle Navigation System (DVNS), the real time traffic junctions are mapped as nodes and the traffic rate between the signals is considered as the link weight for the selection of routes from source to destination. The selection of the route depends on various parameters such as traffic rate, speed of the vehicle, shortest path etc. The dynamic route selection is implemented using the krushkal's algorithm based on different parameters. The end-user selects the source and destination and sends the information to our system using Google maps; our system suggests the optimal route to the end-user based on the dynamic routing table. The traffic in-charge personnel have to update the traffic rates using their mobile PDA. The Dynamic routing table is updated periodically to accomplish the optimal solution.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATION

SYMBOLS/ ABBREVIATION	MEANING / EXPANSION
NP	Non deterministic polynomial time
MST	Minimum Spanning Tree

CHAPTER-1

PROBLEM DEFINITION

Nowadays, there is an urgent need for the robust and reliable traffic assistance system to improve traffic control and management to solve the problem of increasing urban traffic. Vehicle detection technique appears to be the weakest in traffic guidance and control. Many traffic state parameters can be detected through traffic guidance system, including traffic flow density, the length of queue, average traffic speed and total vehicle in fixed time interval. To achieve these goals, in past decades, there have been many approaches proposed for handling the related problems. And also the road traffic has been increase from the few years and there is also no certain traffic assistance nowadays

Aim:- The aim of this project is to decrease road traffic by finding dynamic routes.

CHAPTER-2

PROBLEM EXPLANATION

In dynamic vehicle navigation system, the real time traffic junctions are mapped as nodes and the traffic rate between the signals is considered as the link weight for the selection of routes from source to destination.

With the help of the map which is generated by the kruskal's algorithm , the traffic in-charge has to update dynamic traffic information like traffic rates , traffic diversions in routing table. Routing table consists of nodes and weights between the nodes , the rate at which vehicle rates during peak hours. We represent 1 as low, 3 as medium and 5 as high traffic rate.

CHAPTER-3

DESIGN TECHNIQUES

The design technique used to solve this problem is greedy method.

Greedy is an algorithmic paradigm that builds up a solution piece by piece, always choosing the next piece that offers the most obvious and immediate benefit. Problems were choosing locally optimal also leads to global solution are best fit for greedy.

The greedy algorithm is a top-down approach that never goes back to its previous decision, even if it does not give the best answer for the overall problem.

The greedy technique does not always provide the best solution to all the problems.

The greedy technique does not always provide the best solution to all the problems.

Advantages:-

- Greedy approach is easy to implement.
- Typically have less time complexities.
- Greedy algorithms can be used for optimization purposes or finding close to optimization in case of NP Hard problems.
- Greedy algorithm is easier to describe.

Disadvantages:-

- The local optimal solution may not always be global optimal.
- Never goes back to it's previous decision, even when the choice is wrong.

CHAPTER 4

ALGORITHM FOR THE PROBLEM

The algorithm used to find dynamic road route is Kruskal's Algorithm. Kruskal's Algorithm to find minimum cost spanning tree uses greedy approach. This algorithm treats the graph as a forest and every node in it as an individual tree. A tree connects to another only if it has least cost among all available options and doesn't violate MST properties.

Steps in krushkal's Algorithm:
Step1 : T = ∅;
Step 2 : perform until v ∈ V MakeSet(v);
Step 3: sort E by increasing edge weight w
Step 4 : perform until (u,v) ∈ E (in sorted order)
Step 6: if FindSet(u) ≠ FindSet(v) goto step 7
Step 7: T = T U {{u,v}};
Step 8: Union(FindSet(u), FindSet(v));
Here the u, v are two nodes from X= {x1, x2,…, xi} which uses the weight W= { w11,w12, …. wnn}, SP= { sp11,sp12, …. spnn} and the traffic rate R= {r11,r12, …., rnn} are used to generate the map which will be shown to the traffic incharge.

CHAPTER-5

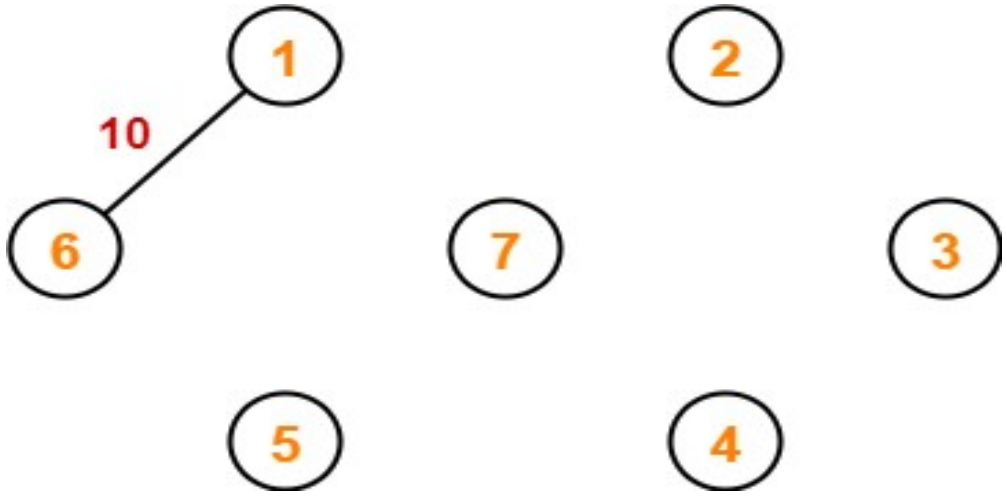
EXPLANATION OF ALGORITHM

Step-1:- Sort all the edges in non-decreasing order of their weight.

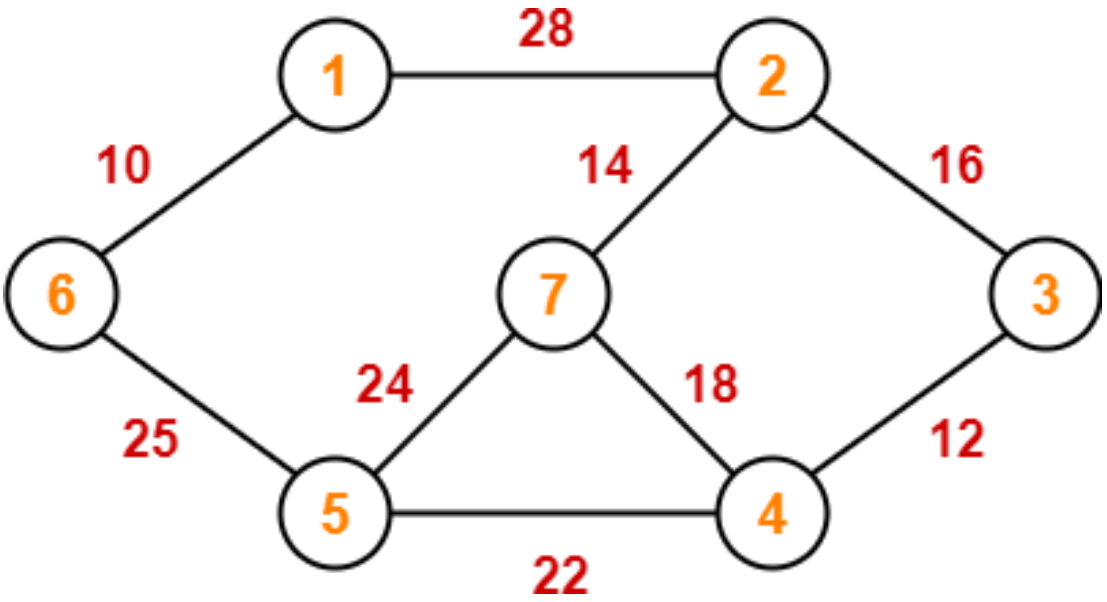
Step-2:- Pick the smallest edge. Check if it forms a cycle with the spanning tree formed so far. If cycle is not formed, include this edge Else, discard it.

Step-3:- Repeat Step#2 until there are (V-1) edges in the spanning tree.

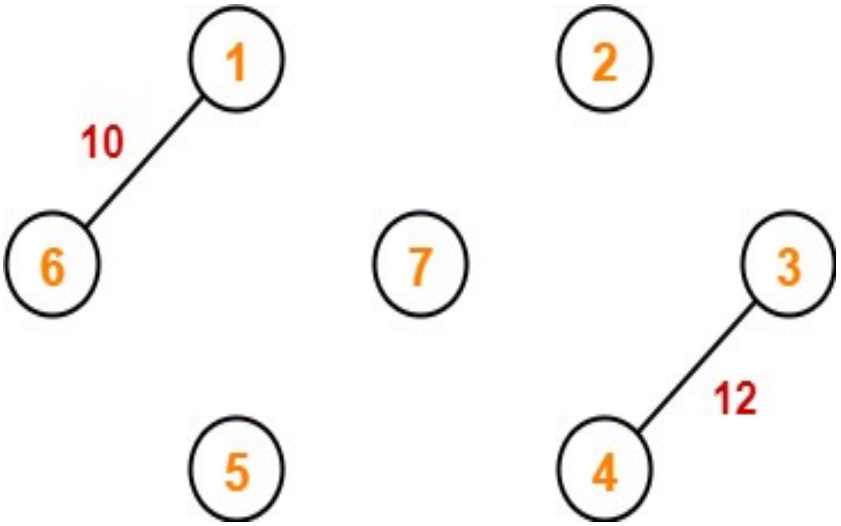
Example:-



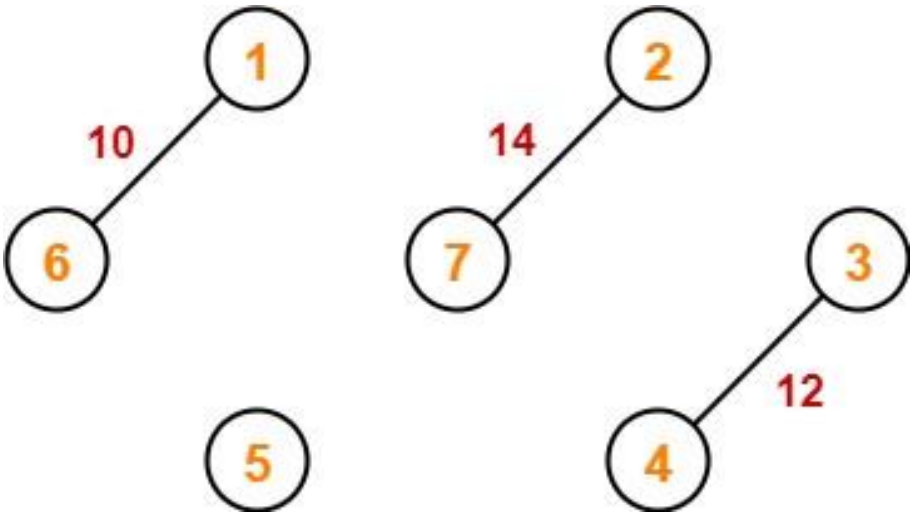
Step 1:_



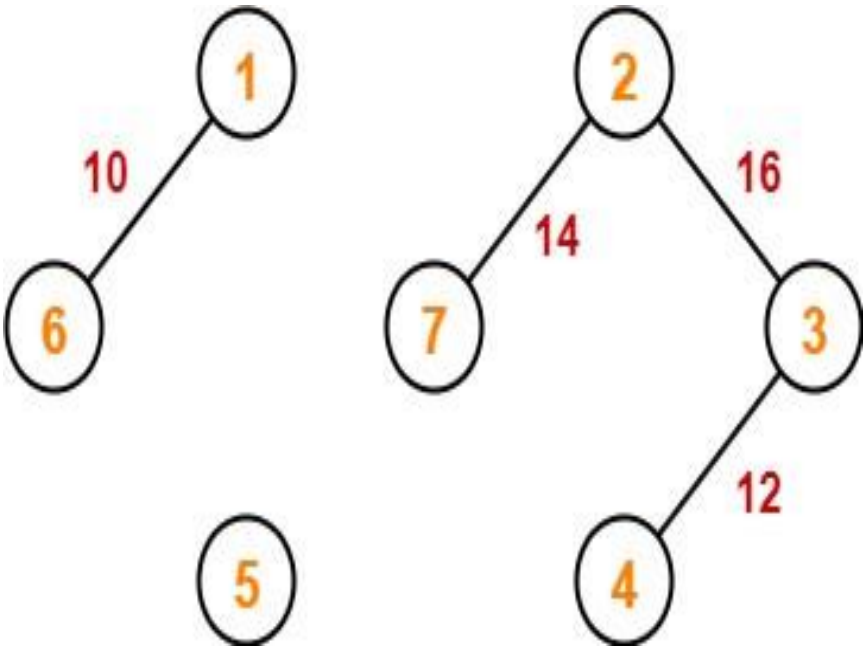
Srep-2:-



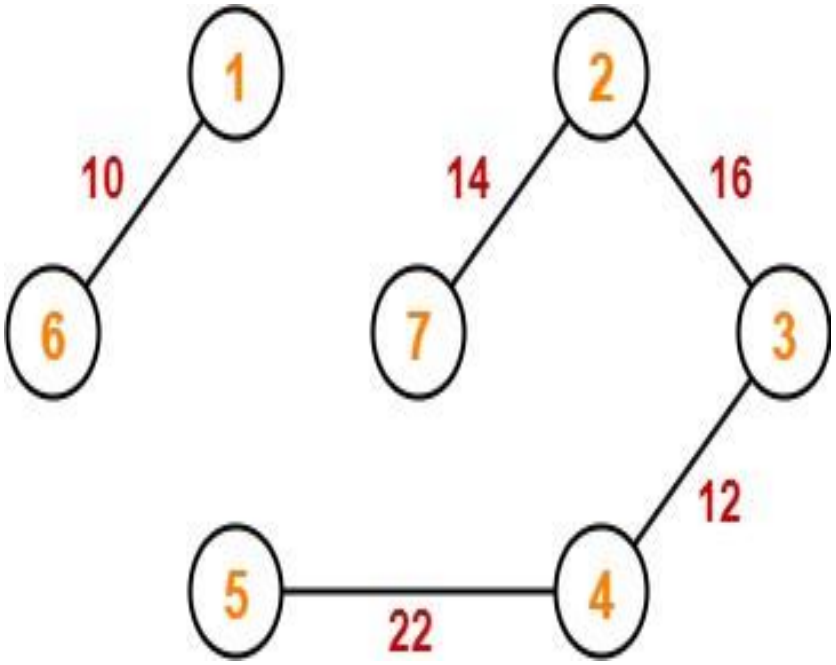
Step-3:-



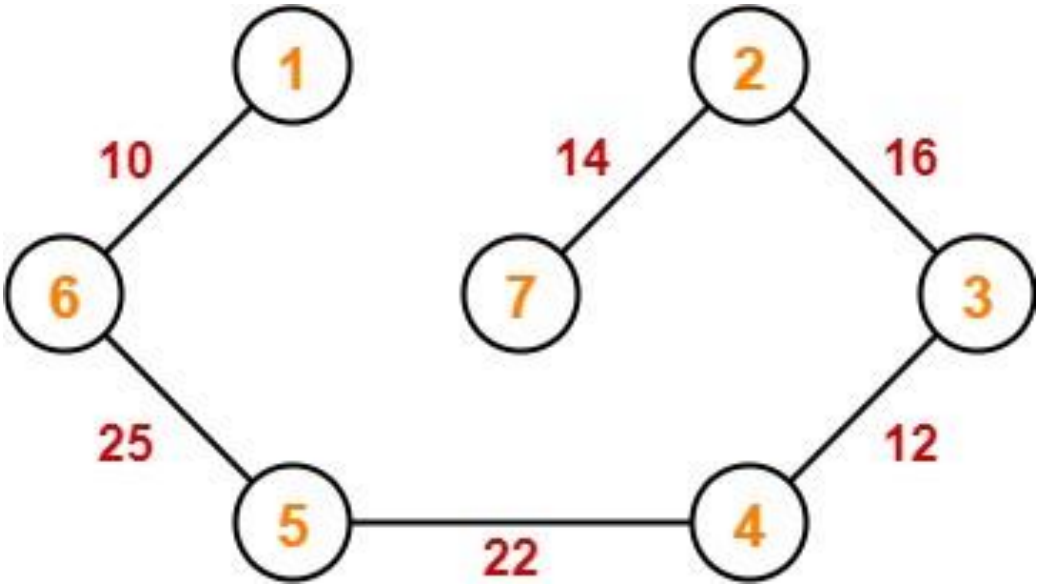
Step-4:-



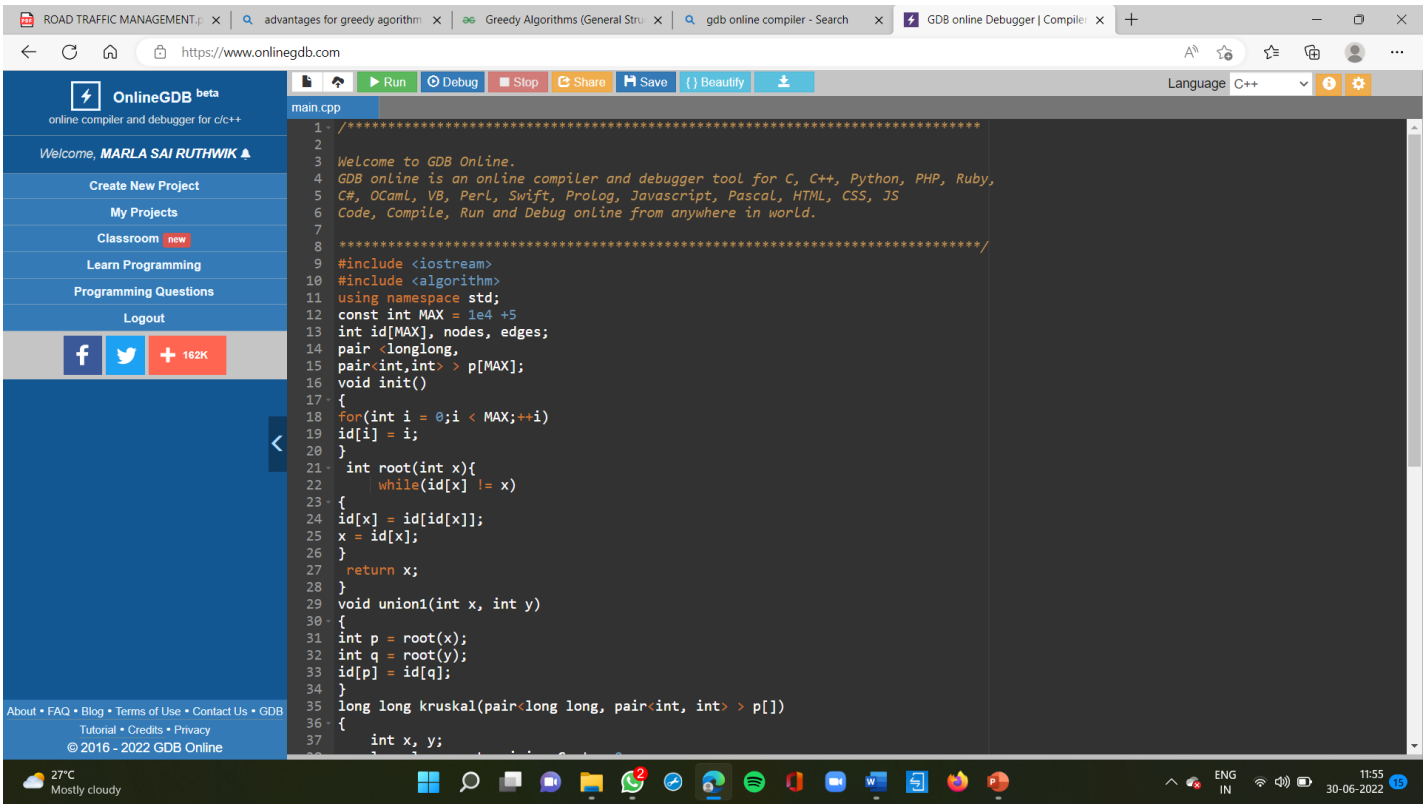
Step-5:-

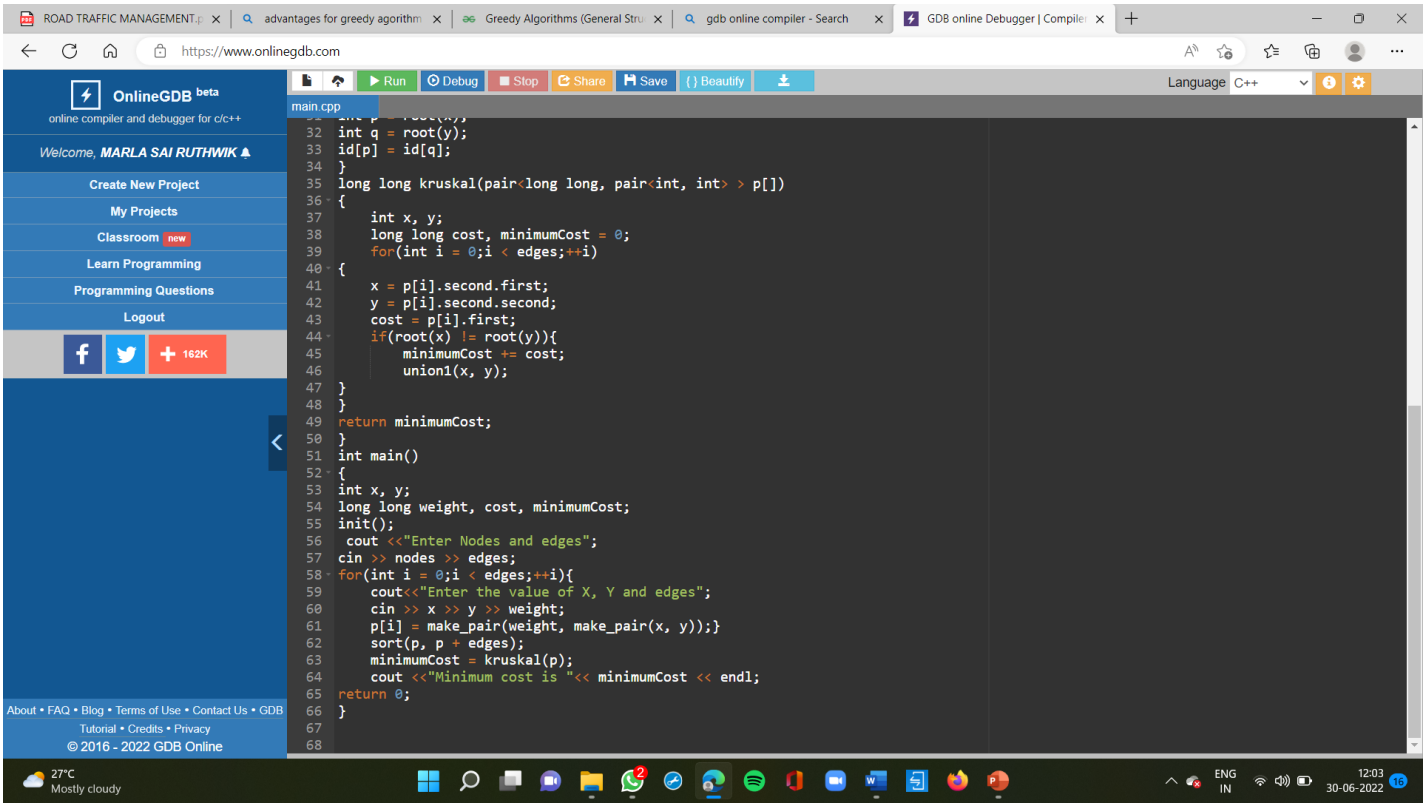


STEP 6

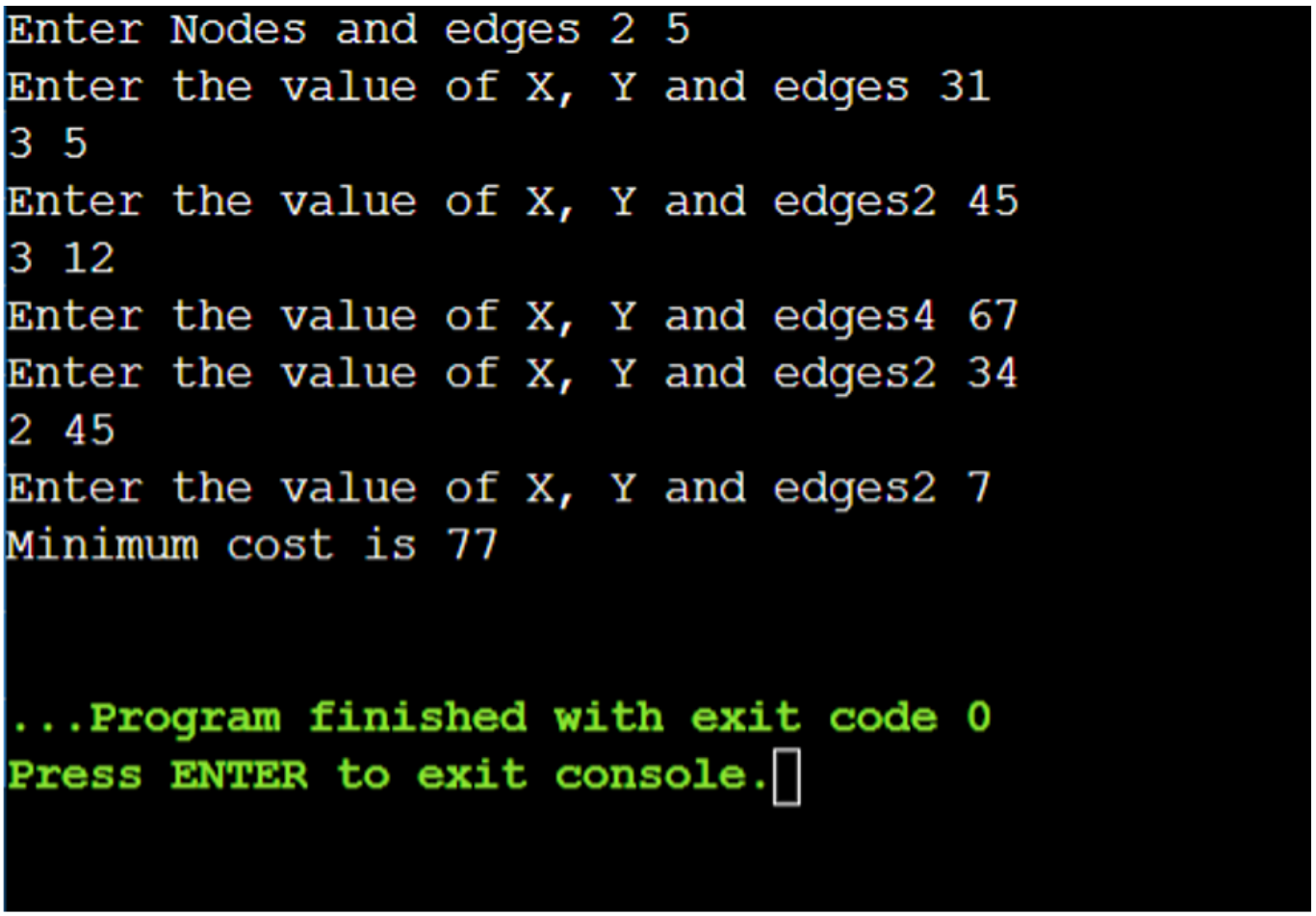


Input:-


ROAD TRAFFIC MANAGEMENT.p
advantages for greedy agorithm
Greedy Algorithms (General Stru
gdb online compiler - Search
GDB online Debugger | Compile
https://www.onlinegdb.com
OnlineGDB beta
online compiler and debugger for c/c++
Welcome, MARLA SAI RUTHWIK
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Logout
162K
About • FAQ • Blog • Terms of Use • Contact Us • GDB
Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
/*
Welcome to GDB Online.
GDB online is an online compiler and debugger tool for C, C++, Python, PHP, Ruby,
C#, OCaml, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
Code, Compile, Run and Debug online from anywhere in world.
*/
#include <iostream>
#include <algorithm>
using namespace std;
const int MAX = 1e4 +5
int id[MAX], nodes, edges;
pair <longlong,
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;
27°C
Mostly cloudy
ENG
IN
11:55
30-06-2022


ROAD TRAFFIC MANAGEMENT.p
advantages for greedy agorithm
Greedy Algorithms (General Stru
gdb online compiler - Search
GDB online Debugger | Compile
https://www.onlinegdb.com
OnlineGDB beta
online compiler and debugger for c/c++
Welcome, MARLA SAI RUTHWIK
Create New Project
My Projects
Classroom new
Learn Programming
Programming Questions
Logout
162K
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2022 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C++
main.cpp
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;
}
27°C
Mostly cloudy
ENG
IN
12:03
30-06-2022

Output:-


Enter Nodes and edges 2 5
Enter the value of X, Y and edges 31
3 5
Enter the value of X, Y and edges2 45
3 12
Enter the value of X, Y and edges4 67
Enter the value of X, Y and edges2 34
2 45
Enter the value of X, Y and edges2 7
Minimum cost is 77
...Program finished with exit code 0
Press ENTER to exit console.

CHAPTER-6

COMPLEXITY ANALYSIS

Time Complexity:-

Time Complexity for Kruskal's Algorithm is O(E LogE) or O(E LogV).

Sorting of Edges takes O(ELogE). After sorting we iterate through all edges and apply the find-union algorithm. The find and union algorithms can take atmost O(LogV) time. So, overall Time Complexity is O(E LogE + E LogV) time. The value of V can be atmost O(V^2).So, O(Log V) and O(Log V) are the same. Therefore, the overall time complexity is O(ELogE)

Space Complexity:-

Space Complexity for Kruskal's Algorithm is O(Log(E)).

CHAPTER-7

CONCLUSION

1) **Greedy algorithm is an approach for solving a problem by selecting te best option available at the moment. The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach.**
2) **The greedy algorithm is easier to describe.**
3) **Greedy algorithm is very useful in solving optimization problems.**

REFERENCES

- **Geeks for Geeks**
- **JavaTPoint**
- **Aeron-Thomas A, Jacobs GD, Sexton B, Gururaj G & Rahman F (2004), *The Impact of Crashes on the Poor*. Study commissioned from TRL by GRSP with funding from the Swedish International Development Cooperation Agency (Sida) and TRL, Crowthorne.**
- **AfDB (2012), *Road Safety in Africa: an overview*, African Development Bank, MDBs Training Initiative Global Road Safety Facility, Washington DC.**
- **Belin M-A, Tillgren P & Vedung E (2012), *Vision zero – a road safety policy innovation,* International**

Journal of Injury Control and Safety Promotion Volume 19, Issue 2, 2012, pages 171-179

- **Bhalla K, Shahraz S, Naghavl M, & Murray C (2008), *Estimating the potential impact of safety policies on road traffic death rates in developing countries*, 9th World Conference on Injury Prevention and Safety Promotion, Merida, Mexico**
- **Bliss T (2011), *Global Directions in Road Safety,* Strategic Road Safety Forum, Monash University Accident Research Centre, Melbourne**
- **Bliss T & Breen J (2011), *Improving Road Safety Performance: Lessons From International Experience a resource paper prepared for the World Bank*, Washington DC for the National Transport Development Policy Committee (NTDPC), Government of India, Delhi. (From following link, go to 'Papers received from the World Bank as technical assistance', and then go to 'WB Papers on the Highway Sector by Clell Harral, See Resource Paper**
- **Breen J, Humphries R & Melibaeva S (2013), *Mainstreaming road safety***

Commission of the European Communities (CEC) (2003), *European Road Safety Action Programme. Halving the number of road accident victims in the European Union by 2010: A shared responsibility* COM(2003) 311 final, 2.6.2003, Brussels.

- **DaCoTA (2012a), *Cost-benefit analysis*, Deliverable 4.8d of the EC FP7 project DaCoTA, Brussels.**

- DaCoTA (2012b), ***Work-related road safety***, Deliverable 4.8v of the EC FP7 project DaCoTA, Brussels.
- DaCoTa (2012c), ***Road Safety Management***, Deliverable 4.8p of the EC

APPENDIX

CODE

1. CODE

```
#include
<iostream>
#include
<algorithm>
using namespace
```

```
std;     const int
MAX = 1e4 +5 int
id[MAX], nodes,
edges;     pair
<longlong,
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.seco
nd;
   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;
}
```