project on : Project On: Ordering a Cake recipe.

lab code:CSE-206

Language: c++

Submitted to:
Md. Solaiman Mia,
Assistant Professor,
Dept. of CSE,
Green University of Bangladesh.

Submitted By:
Md. Sobuj Mia -191902048
Shaiful Islam -191902051
Dept. of CSE
Green University of Bangladesh.

Date of Submission:
30/12/2020

- **Why I have selected the project?**

  Because we see a cake maker face this problem. Then we learn a algorithm named topological sort in our lab class that solve this problem easily.

- **what is the implementation of our project?**

  *** what we need to collect for make a cake.**

  *** Ordering a cake recipe.**

  *** Shortest path from source to destination.**

- **What algorithm we need ?**

  1) Topological sort.
  2) Dijkstra .
  3) dfs.

- **source code:**

```
#include<bits/stdc++.h>
using namespace std;
vector<int> g1[100];
vector<int>cost[100];
int dis[100];
vector<int>g[100];
vector<int>g2[100];
vector<string>top;
vector<string>o;
int t=1,m=0,adj,vis[100],f=0;
int edg[100][100],par[100],t1=0,ar[100][100],t2=1;
pair<int, int>T[100];
```

```
void dfs(int s)
{
   vis[s]=1;
   cout<<o[s-1]<<" ";
   T[s].first=t++;

   for(int i=0; i<g[s].size(); i++)

   {
      adj=g[s][i];
      if(vis[adj]==0)
      {
         edg[s][adj]=1;

         dfs(adj);
      }
      else if(vis[adj]==1)
      {
         if(T[adj].second==0)
         {
            edg[s][adj]=2;
            m++;
         }
      }
   }
   top.push_back(o[s-1]);
   T[s].second=t++;
}


void dijk(int s1)
{
   fill(dis, dis+100, 1e9);
   dis[s1]=0;
   priority_queue<pair<int, int>>Q;
```

```
    Q.push(make_pair(-0, s1));
    while(!Q.empty())
    {
        int node=Q.top().second;
        Q.pop();
        for(int i=0; i<g1[node].size(); i++)
        {
            int c=cost[node][i];
            int v=g1[node][i];
            if(dis[v]>dis[node]+c)
            {
                dis[v]=dis[node]+c;
                g2[v].push_back(node);

                Q.push(make_pair(-dis[v],v));
            }
        }
    }
}


int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    cout.tie(NULL);
    int V,E,a,b,L;
    string k;
    freopen("Cake.txt","r",stdin);
    cin>>V>>E;

    for(int i=1; i<=E; i++)
    {
        cin>>a>>b;
        g[a].push_back(b);
```

```
}

for(int i=1; i<=V; i++)
{
   cin>>k;
   o.push_back(k);
}
cout<<"Collected product for Cake recipe:\n\n";

for(int i=1; i<=V; i++)
{
   if(vis[i]==0)
   {
      dfs(i);

   }
}
cout<<"\n\n";

for(int i=1; i<=V; i++)
{
   cout<<"Node "<<o[i-1]<<": "<<T[i].first<<" " <<T[i].second<<"\n";
}
cout<<"\n";
reverse(top.begin(), top.end());
cout<<"\n";
if(m)
   cout<<"This graph is not DAG\n";
else
{
   cout<<"\n\nOrdering of Cake Recipe:\n\n";
   cout<<"\n";

   for(int i=0; i<V; i++)
   {
```

```
        cout<<top[i]<<" ";
        if(i==9)
            continue;
        else
            cout<<"-> ";
    }
}
cout<<"\n";
cout<<"\nEnter the Source: ";
int V1,E1,a1,b1,c1;
freopen("input.txt","r",stdin);
cin>>V1>>E1;
for(int i=1; i<=E1; i++)
{
    cin>>a1>>b1>>c1;
    g1[a1].push_back(b1);
    cost[a1].push_back(c1);
}
cout<<"\n";
int s1;
cin>>s1;
dijk(s1);
for(int i=1; i<=V1; i++)
{
    if(dis[i]==1e9)
        cout<<"Distance of "<<i<<" from source: infinity\n";
    else
        cout<<"Distance of "<<i<<" from source: "<<dis[i]<<"\n";
}

cout<<"\n";

for(int i=1; i<=V1; i++)
{
    int d=i;
```

```
        par[t1++]=i;

        for(int j=g2[d].size(); j=g2[d].size(); j--)
        {
            par[t1++]=g2[d][j-1];
            d=g2[d][j-1];

        }
       cout<<"For node "<<i<<" shortest path from source:";

        for(int l=t1-1; l>=0; l--)
        {
            cout<<par[l]<<" ";
        }
        cout<<"\n";
        t1=0;
        fill(par,par+V1,0);
    }

    return 0;
}
```

- **output:**

```
Collected product for Cake recipe:

Salt Mix Container Heat Packet Flur Sugar Water Egg Oil

Node Salt: 1 10
Node Mix: 2 9
Node Flur: 11 12
Node Sugar: 13 14
Node Water: 15 16
Node Egg: 17 18
Node Oil: 19 20
Node Container: 3 8
Node Heat: 4 7
Node Packet: 5 6



Ordering of Cake Recipe:


Oil -> Egg -> Water -> Sugar -> Flur -> Salt -> Mix -> Container -> Heat -> Packet

Enter the Source:
Distance of 1 from source: 80
Distance of 2 from source: 0
Distance of 3 from source: 50
Distance of 4 from source: 120

For node 1 shortest path from source :2 3 1
For node 2 shortest path from source :2
For node 3 shortest path from source :2 3
For node 4 shortest path from source :2 3 1 4
```