Heaven's Light is Our Guide

Rajshahi University of Engineering and Technology
Department of Computer Science and Engineering

Course No: CSE.2202
Course Title: Sessional based on CSE.2201 (Computer Algorithms)

Lab Report No: 09
Lab Report On: Dijkstra Algorithm, Kruskal's Algorithm and Longest Common Subsequence.

Submitted By
Md. Ariful Islam
Roll No: 1803046
Section: A
Department: CSE

Submitted To
Biprodip Pal
Assistant Professor
Dept. of CSE,RUET

Date: 14-08-2021

Dijkstra Algorithm

❖ Source Code:

```
#include<bits/stdc++.h>
using namespace std;
using namespace std::chrono;

typedef long long ll;

int adjmat[505][505];
vector<pair<int,int>>adjlist[505];
vector<int>usa;
map<pair<int,int>,int>mp;
int cost[505];
int parent[505];
int binhp[700];
int hps;

void initialize(){
   hps=0;
   for(int i=0;i<502;i++){
      cost[i]=INT_MAX;
      parent[i]=0;
   }
}

void binhp_add(int z){
      binhp[hps]=z;
      for(int i=hps;i>0;i=i/2){
         int par = (i-1)/2;
         if(cost[binhp[par]]>cost[binhp[i]]){
            swap(binhp[par],binhp[i]);
         }
      }
}

void binhp_pop(){
   int i=0;
   while(((i*2)+1)<=hps){
      if((i*2)+2<=hps){
         if(binhp[i]>binhp[(i*2)+2] &&
binhp[(i*2)+1]>binhp[(i*2)+2]){
            binhp[i]=binhp[(i*2)+2];
            i=(i*2)+2;
         }
```

```
            else if(binhp[i]>binhp[(i*2)+1] &&
binhp[(i*2)+2]>binhp[(i*2)+1]){
                binhp[i]=binhp[(i*2)+1];
                i=(i*2)+1;
            }
            else{
                break;
            }
        }
        else{
            if(binhp[i]>binhp[(i*2)+1])
                binhp[i]=binhp[(i*2)+1];
                i=(i*2)+1;
            }
    }
}

int binhp_adjmat(int x, int y){
    for(int i=1;i<=101;i++){
        if(adjmat[x][i]!=-1){
            if(cost[i]>(cost[x]+adjmat[x][i])){
                parent[i]=x;
                cost[i]=cost[x]+adjmat[x][i];
                binhp_add(i);
                hps+=1;
            }
        }
    }

    while(hps>0){
        x=binhp[0];
        binhp_pop();
        hps-=1;
        binhp_adjmat(x,y);
    }
}


int binhp_adjlist(int x, int y){
    for(int i=0;i<adjlist[x].size();i++){
        pair<int,int>pp;
        pp = adjlist[x][i];

        if(cost[pp.first]>(cost[x]+pp.second)){
                parent[pp.first]=x;
                cost[pp.first]=cost[x]+pp.second;
                binhp_add(pp.first);
```

```
            hps+=1;
        }
    }

    while(hps>0){
        x=binhp[0];
        binhp_pop();
        hps-=1;
        binhp_adjmat(x,y);
    }
}


int usa_adjmat(int x, int y){
    for(int i=1;i<=101;i++){
        if(adjmat[x][i]!=-1){
            if(cost[i]>(cost[x]+adjmat[x][i])){
                parent[i]=x;
                cost[i]=cost[x]+adjmat[x][i];
                usa.push_back(i);
                hps+=1;
            }
        }
    }

    while(usa.size()>0){
        x=cost[usa[0]];
        int z=0;
        for(int i=1;i<usa.size();i++){
            if(cost[usa[i]]<x){
                x=cost[usa[i]];
                z=i;
            }
        }
        x=usa[z];
        usa.erase(usa.begin()+z);

        hps-=1;
        usa_adjmat(x,y);
    }
}

int usa_adjlist(int x, int y){
    for(int i=0;i<adjlist[x].size();i++){
        pair<int,int>pp;
        pp = adjlist[x][i];
```

```
        if(cost[pp.first]>(cost[x]+pp.second)){
            parent[pp.first]=x;
            cost[pp.first]=cost[x]+pp.second;
            usa.push_back(pp.first);
            hps+=1;
        }
    }

    while(usa.size()>0){
        x=cost[usa[0]];
        int z=0;
        for(int i=1;i<usa.size();i++){
            if(cost[usa[i]]<x){
                x=cost[usa[i]];
                z=i;
            }
        }
        x=usa[z];
        usa.erase(usa.begin()+z);

        hps-=1;
        usa_adjlist(x,y);
    }
}




int main(){
    memset(adjmat,-1,sizeof(adjmat));

    int n,e,a,b,c;

    cout<<"Enter N & E: ";
    cin>>n>>e;


    for(int i=0;i<e;i++){
        //srand(time(0));
        a=(rand()%n)+1;
        //srand(time(0));
        b=(rand()%n)+1;
        if(mp[{a,b}]!=0 || mp[{b,a}]!=0 || a==b){
            i-=1;
            continue;
        }
        mp[{a,b}]=1;
```

```
        mp[{b,a}]=1;
        //srand(time(0));
        c=(rand()%7)+3;
        cout<<"Enter E to E & C: ";
        cout<<a<<" "<<b<<" "<<c<<endl;
        adjmat[a][b]=c;
        adjmat[b][a]=c;
        adjlist[a].push_back(make_pair(b,c));
        adjlist[b].push_back(make_pair(a,c));
    }

    while(1){
        cout<<"Enter Nodes(0 0 to exit): ";
        cin>>a>>b;
        if(a<1 || a>n || b<1 ||b>n){
            cout<<"Exiting..."<<endl;
            break;
        }


        initialize();
        cost[a]=0;
        auto start = high_resolution_clock::now();
        binhp_adjmat(a,n);
        auto stop = high_resolution_clock::now();

        auto duration = duration_cast<microseconds>(stop - start);
        cout<<"Binary Heap AdjMat: "<<duration.count()<<"
Microseconds"<<endl;
        cout<<cost[b]<<endl;

        initialize();
        cost[a]=0;
        start = high_resolution_clock::now();
        binhp_adjlist(a,n);
        stop = high_resolution_clock::now();
        duration = duration_cast<microseconds>(stop - start);
        cout<<"Binary Heap AdjList: "<<duration.count()<<"
Microseconds"<<endl;
        cout<<cost[b]<<endl;

        initialize();
        cost[a]=0;
        start = high_resolution_clock::now();
        usa_adjmat(a,n);
        stop = high_resolution_clock::now();
        duration = duration_cast<microseconds>(stop - start);
```

```
        cout<<"UnSorted Array AdjMat: "<<duration.count()<<"
Microseconds"<<endl;
        cout<<cost[b]<<endl;

        initialize();
        cost[a]=0;
        start = high_resolution_clock::now();
        usa_adjlist(a,n);
        stop = high_resolution_clock::now();
        duration = duration_cast<microseconds>(stop - start);
        cout<<"UnSorted Array AdjList: "<<duration.count()<<"
Microseconds"<<endl;
        cout<<cost[b]<<endl;
    }


    return 0;
}
```

❖ Output:

Time to find shortest path		
Priority queue data structure	**Adjacency list**	**Adjacency matrix**
Binary Heap	**0**	**0**
Unsorted Array	**0**	**91 micro sec.**

Kruskal's Algorithm

❖ Source Code:

```
#include<bits/stdc++.h>
using namespace std;
using namespace std::chrono;

typedef long long ll;
typedef pair<int,pair<int,int>> pp;

priority_queue<pp, vector<pp>, greater<pp> >pq;
map<pair<int,int>,int>mp;

vector<int>tree[505];
int adjmat[505][505];
int parent[505],rnk[505];
int cost;
int root;

void initialize()
{
    cost=0;
    for(int i=0; i<502; i++)
    {
        tree[i].clear();
        parent[i]=i;
        rnk[i]=0;
    }
}

int find_parent_without(int x){
    if(parent[x]=x){
        return x;
    }
    else{
```

```
        return find_parent_without(parent[x]);
    }
}

int find_parent_with(int x){
    if(parent[x]=x){
        return x;
    }
    else{
        return parent[x]=find_parent_with(parent[x]);
    }
}

void kruskal_without_rank_compressed_djset(){
    priority_queue<pp, vector<pp>, greater<pp>> pq1 = pq;

    for(int i=0; i<pq1.size();i++){
        pp p1=pq1.top();
        pq1.pop();
        pair<int,int>p2;
        p2=p1.second;

        int x=find_parent_without(p2.first);
        int y=find_parent_without(p2.second);

        if(x!=y){
            parent[x]=y;
            cost+=p1.first;
            tree[p2.first].push_back(p2.second);
            tree[p2.second].push_back(p2.first);
        }
    }
}

void kruskal_with_rank_compressed_djset(){
    priority_queue<pp, vector<pp>, greater<pp>> pq1 = pq;
```

```
    for(int i=0; i<pq1.size();i++){
        pp p1=pq1.top();
        pq1.pop();
        pair<int,int>p2;
        p2=p1.second;

        int x=find_parent_with(p2.first);
        int y=find_parent_with(p2.second);

        if(x!=y){
            if(rnk[x]>rnk[y]){
                swap(x,y);
            }
            parent[x]=y;
            if(rnk[x]==rnk[y]){
                rnk[y]+=1;
            }
            cost+=p1.first;
            tree[p2.first].push_back(p2.second);
            tree[p2.second].push_back(p2.first);
        }
    }
}

int main(){

    memset(adjmat,0,sizeof(adjmat));

    int n,e,a,b,c;

    cout<<"Enter N & E: ";
    cin>>n>>e;
    if(n==0&& e==0)
    {
        return 0;
    }
```

```
//    srand(time(0));
    for(int i=0; i<e; i++)
    {
        a=(rand()%n)+1;
        b=(rand()%n)+1;
        if(mp[ {a,b}]!=0 || mp[ {b,a}]!=0 || a==b)
        {
            i-=1;
            continue;
        }
        mp[ {a,b}]=1;
        mp[ {b,a}]=1;
        c=(rand()%7)+3;
        cout<<"Enter E to E & C: ";
        cout<<a<<" "<<b<<" "<<c<<endl;
        pq.push({c,{a,b}});
        adjmat[a][b]=c;
        adjmat[b][a]=c;
    }

    initialize();
    auto start = high_resolution_clock::now();
    kruskal_with_rank_compressed_djset();
    auto stop = high_resolution_clock::now();

    auto duration = duration_cast<microseconds>(stop - start);
    cout<<"kruskal_with_rank_compressed_djset:
"<<duration.count()<<" Microseconds"<<endl;
    cout<<cost<<endl;

    initialize();
    start = high_resolution_clock::now();
    kruskal_without_rank_compressed_djset();
    stop = high_resolution_clock::now();

    duration = duration_cast<microseconds>(stop - start);
```

```
    cout<<"kruskal_without_rank_compressed_djset:
"<<duration.count()<<" Microseconds"<<endl;
    cout<<cost<<endl;



    return 0;
}
```

❖ **Output:**

V	E	Time(with path/rank based approach)	Time(with path/rank based approach)	Solution (total cost)
100	**120**	**0**	**0**	**253**
500	**500**	**0**	**997 micro sec.**	**1071**
600	**600**	**0**	**999 micro sec.**	**1319**

Longest Common Subsequence

❖ Source Code:

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

int main(){
    string a,b;

    cout<<"Enter First String: ";
    cin>>a;
    cout<<"Enter Second String: ";
    cin>>b;

    int la,lb,i,j,n,m;

    la=a.size();
    lb=b.size();

    int arr[la+3][lb+3];

    for(i=0;i<=la;i++){
        for(j=0;j<=lb;j++){
            if(i==0 || j==0){
                arr[i][j]=0;
            }
            else if(a[i-1]==b[j-1]){
                arr[i][j]=arr[i-1][j-1]+1;
            }
            else{
                arr[i][j]=max(arr[i][j-1],arr[i-1][j]);
            }
        }
    }
```

```
    stack<char>st;

    i=la; j=lb;
    while(i>0 && j>0){
        if(a[i-1]==b[j-1]){
            st.push(a[i-1]);
            i-=1;
            j-=1;
        }
        else if(arr[i][j-1]>arr[i-1][j]){
            j-=1;
        }
        else{
            i-=1;
        }
    }


    cout<<"Size: "<<st.size()<<endl;
    cout<<"\nLCS: ";
    while(!st.empty()){
        cout<<st.top();
        st.pop();
    }
    cout<<endl;


    return 0;
}
```

❖ Output:

```
"F:\4th Semester\CSE\CSE.2202\Lab 9\LCS.exe"
Enter First String: ABCDGH
Enter Second String: AEDFHR

Size: 3

LCS: ADH

Process returned 0 (0x0)   execution time : 16.779 s
Press any key to continue.
```

 # END #