**INDEX**

**1.)WAP TO PERFORM QUICK SORT USING DIVIDE AND CONQUER.**

**2.)WAP TO PERFORM MERGE SORT USING DIVIDE AND CONQUER.**

**3.)WAP TO PERFORM BINARY SEARCH USING DIVIDE AND CONQUER.**

**4.)WAP TO PERFORM SELECTION SORT.**

**5.)WAP TO PERFORM INSERTION SORT.**

**6.)WAP TO IMPLEMENT KNAPSACK ALGORITHM.**

**7.)WAP TO IMPLEMENT KRUSKAL ALGORITHM TO GENERATE MINIMUM COST SPANNING TREE.**

**8.)WAP TO IMPLEMENT PRIM’S ALGO TO GENERATE MINIMUM COST SPANNING TREE.**

**9.)WAP TO SOLVE SHORTEST PATH PROBLEM USING DIJKSTRA’S ALGORITHM.**

**10.WAP TO IMPLEMENT ACTIVITY SELECTION ALGORITHM**

**7.)WAP TO IMPLEMENT KRUSKAL ALGORITHM TO GENERATE MINIMUM COST SPANNING TREE**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

int cost[10][10],i,j,k,n,m,c,visit,visited[10],l,v,count,count1,vst,p;

main()

{

int dup1,dup2;

cout<<"enter no of vertices";

cin >> n;

cout <<"enter no of edges";

cin >>m;

cout <<"EDGE Cost";

for(k=1;k<=m;k++)

{

cin >>i >>j >>c;

cost[i][j]=c;

cost[j][i]=c;

}

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(cost[i][j]==0)

cost[i][j]=31999;

visit=1;

while(visit<n)

{

v=31999;

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(cost[i][j]!=31999 && cost[i][j]<v && cost[i][j]!=-1 )

{

int count =0;

for(p=1;p<=n;p++)

{

if(visited[p]==i || visited[p]==j)

count++;

}

if(count >= 2)

{

for(p=1;p<=n;p++)

if(cost[i][p]!=31999 && p!=j)

dup1=p;

for(p=1;p<=n;p++)

if(cost[j][p]!=31999 && p!=i)

dup2=p;

if(cost[dup1][dup2]==-1)

continue;

}

l=i;

k=j;

v=cost[i][j];

}

cout <<"edge from " <<l <<"-->"<<k;

cost[l][k]=-1;

cost[k][l]=-1;

visit++;

int count=0;

count1=0;

for(i=1;i<=n;i++)

{

if(visited[i]==l)

count++;

if(visited[i]==k)

count1++;

}

if(count==0)

visited[++vst]=l;

if(count1==0)

visited[++vst]=k;

}

}



**8.)WAP TO IMPLEMENT PRIM’S ALGO TO GENERATE MINIMUM COST SPANNING TREE**

#include<iostream>

#include<conio.h>

#include<stdlib.h>

using namespace std;

int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10],u;

main()

{

int m,c;

cout <<"enterno of vertices";

cin >> n;

cout <<"ente no of edges";

cin >> m;

cout <<"**\n**EDGES Cost**\n**";

for(k=1;k<=m;k++)

{

cin >>i>>j>>c;

cost[i][j]=c;

}

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(cost[i][j]==0)

cost[i][j]=31999;

cout <<"ORDER OF VISITED VERTICES";

k=1;

while(k<n)

{

m=31999;

if(k==1)

{

for(i=1;i<=n;i++)

for(j=1;j<=m;j++)

if(cost[i][j]<m)

{

m=cost[i][j];

u=i;

}

}

else

{

for(j=n;j>=1;j--)

if(cost[v][j]<m && visited[j]!=1 && visit[j]!=1)

{

visit[j]=1;

stk[top]=j;

top++;

m=cost[v][j];

u=j;

}

}

cost[v][u]=31999;

**v=u;**

cout<<v << " ";

k++;

visit[v]=0; visited[v]=1;

}

}

OUTPUT

enterno of vertices7  
ente no of edges9

EDGES Cost  
1 6 10  
6 5 25  
5 4 22  
4 3 12  
3 2 16  
2 7 14  
5 7 24  
4 7 18  
1 2 28  
ORDER OF VISITED VERTICES1 6 5 4 3 2

**9.)WAP TO SOLVE SHORTEST PATH PROBLEM USING DIJKSTRA’S ALGORITHM**

#include<iostream>

#include<conio.h>

#include<stdio.h>

using namespace std;

int shortest(int ,int);

int cost[10][10],dist[20],i,j,n,k,m,S[20],v,totcost,path[20],p;

main()

{

int c;

cout <<"enter no of vertices";

cin >> n;

cout <<"enter no of edges";

cin >>m;

cout <<"**\n**enter**\n**EDGE Cost**\n**";

for(k=1;k<=m;k++)

{

cin >> i >> j >>c;

cost[i][j]=c;

}

for(i=1;i<=n;i++)

for(j=1;j<=n;j++)

if(cost[i][j]==0)

cost[i][j]=31999;

cout <<"enter initial vertex";

cin >>v;

cout << v<<"**\n**";

shortest(v,n);

}

int shortest(int v,int n)

{

int min;

for(i=1;i<=n;i++)

{

S[i]=0;

dist[i]=cost[v][i];

}

path[++p]=v;

S[v]=1;

dist[v]=0;

for(i=2;i<=n-1;i++)

{

k=-1;

min=31999;

for(j=1;j<=n;j++)

{

if(dist[j]<min && S[j]!=1)

{

min=dist[j];

k=j;

}

}

if(cost[v][k]<=dist[k])

p=1;

path[++p]=k;

for(j=1;j<=p;j++)

cout<<path[j];

cout <<"**\n**";

//cout <<k;

S[k]=1;

for(j=1;j<=n;j++)

if(cost[k][j]!=31999 && dist[j]>=dist[k]+cost[k][j] && S[j]!=1)

dist[j]=dist[k]+cost[k][j];

}

}

**OUTPUT**

enter no of vertices6  
enter no of edges11

enter  
EDGE Cost  
1 2 50  
1 3 45  
1 4 10  
2 3 10  
2 4 15  
3 5 30  
4 1 10  
4 5 15  
5 2 20  
5 3 35  
6 5 3  
enter initial vertex1  
1  
14  
145  
1452  
13

**10.WAP TO IMPLEMENT ACTIVITY SELECTION ALGORITHM**

#include<iosteam.h>

#include<conio.h>

void activities(int s[], int f[], int n)

{

int i, j;

cout>>"Selected Activities are:\n");

i = 1;

cout>>A , i;

for (j = 1; j < n; j++)

{

if (s[j] >= f[i])

{

Cout>>A , j+1;

i = j;

}

}

}

void main()

{

int s[] = {1, 3, 0, 5, 3, 5, 6, 8, 8, 2, 12};

int f[] = {4, 5, 6, 7, 9, 9, 10, 11, 12, 14, 16};

int n = sizeof(s)/sizeof(s[0]);

clrscr();

activities(s, f, n);

getchar();

getch();

}

**Output**

