***Department Of Computer Science &Engineering***

***Assignment***

***ON***

***Algorithms Lab***

***Course Code-206***

**Submitted By**

**Assaduzzaman**

**Senior Lecturer**

***Department Of Computer Science &Engineering***

**Submitted By:**

**Ibrahim Khalil**(CSE)

**ID:181002134**

***Department Of Computer Science &Engineering***

***Algorithm Lab CSE-206***

***Assignment***

***//Breath First Search…………………………………***

#include<stdio.h>

int gr[20][20],visited[20],q[20],f=0,r=0,n;

void enqueue(int a)

{

q[r++]=a;

}

int dequeue()

{

return q[f++];

}

void BFS(int s)

{

int u,v;

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

{

visited[u]=0;

}

visited[s]=1;

enqueue(s);

while(f!=r)

{

u=dequeue();

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

{

if(gr[u][v]==1)

{

if(visited[v]==0)

{

visited[v]=1;

enqueue(v);

}

}

}

}

}

void main()

{

int i,j,s;

printf("enter number of nodes: ");

scanf("%d",&n);

printf("Enter the matrix:\n");

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

{

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

{

scanf("%d",&gr[i][j]);

}

}

printf("Enter the source node: ");

scanf("%d",&s);

BFS(s);

printf("Visiting order is:");

for(i=0;i<n;i++)

{

printf("%d ",q[i]);

}

}

***//Depath Frist Search………………………***

#include<stdio.h>

int gr[20][20],col[20],f[20],d[20],prev[20],n,time=0;

void dfs\_visit(int u)

{

int v;

col[u]=1;

time++;

d[u]=time;

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

{

if(gr[u][v]==1)

{

if(col[v]==0)

{

prev[v]=u;

dfs\_visit(v);

}

}

}

time++;

f[u]=time;

col[u]=2;

}

void dfs()

{

int u;

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

{

col[u]=0;

prev[u]=-1;

d[u]=-1;

f[u]=-1;

}

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

{

if(col[u]==0)

dfs\_visit(u);

}

}

void main()

{

int i,j,s,de;

printf("enter number of nodes: ");

scanf("%d",&n);

printf("Enter the matrix:\n");

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

{

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

{

scanf("%d",&gr[i][j]);

}

}

dfs();

printf("Starting time is:");

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

{

printf("%d ",d[i]);

}

printf("\nFinishing time is:");

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

{

printf("%d ",f[i]);

}

}

***//Disconnected.........................................***

#include<stdio.h>

int gr[20][20],visited[20],q[20],f=0,r=0,n;

void enqueue(int a)

{

q[r++]=a;

}

int dequeue()

{

return q[f++];

}

void BFS(int s)

{

int u,v;

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

{

visited[u]=0;

}

visited[s]=1;

enqueue(s);

while(f!=r)

{

u=dequeue();

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

{

if(gr[u][v]==1)

{

if(visited[v]==0)

{

visited[v]=1;

enqueue(v);

}

}

}

}

}

void main()

{

int i,j,s;

printf("enter number of nodes: ");

scanf("%d",&n);

printf("Enter the matrix:\n");

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

{

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

{

scanf("%d",&gr[i][j]);

}

}

printf("Enter the source node: ");

scanf("%d",&s);

BFS(s);

printf("Visiting order is:");

for(i=0;i<n;i++)

{

printf("%d ",q[i]);

}

int check=0;

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

{

if(visited[i])

{

check=1;

break;

}

}

if(check==1)

printf("\nDisconnect grap\n");

else

printf("\Connect grap\n");

}

***//Cycle....................................................***

#include<stdio.h>

int gr[20][20],visited[20],pre[20],q[20],f=0,r=0,n,cycle=0;

void enqueue(int a)

{

q[r++]=a;

}

int dequeue()

{

return q[f++];

}

void BFS(int s)

{

int u,v;

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

{

visited[u]=0;

pre[u]=-1;

}

visited[s]=1;

enqueue(s);

while(f!=r)

{

u=dequeue();

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

{

if(gr[u][v]==1)

{

if(visited[v]==0)

{

visited[v]=1;

enqueue(v);

pre[v]=u;

}

else if(pre[u]!=v)

{

cycle=1;

}

}

}

}

}

void main()

{

int i,j,s;

printf("enter number of nodes: ");

scanf("%d",&n);

printf("Enter the matrix:\n");

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

{

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

{

scanf("%d",&gr[i][j]);

}

}

printf("Enter the source node: ");

scanf("%d",&s);

BFS(s);

printf("Visiting order is:");

for(i=0;i<n;i++)

{

printf("%d ",q[i]);

}

if(cycle==1)

printf("\nCycle\n");

else

printf("\nNocycle\n");

}

***//Distance.........................................................***

#include<stdio.h>

int gr[20][20],visited[20],q[20],d[20],f=0,pre[20],r=0,n;

void enqueue(int a)

{

q[r++]=a;

}

int dequeue()

{

return q[f++];

}

void BFS(int s)

{

int u,v;

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

{

visited[u]=0;

pre[u]=-1;

d[u]=-1;

}

visited[s]=1;

d[s]=0;

enqueue(s);

while(f!=r)

{

u=dequeue();

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

{

if(gr[u][v]==1)

{

if(visited[v]==0)

{

visited[v]=1;

enqueue(v);

pre[v]=u;

d[v]=d[u]+1;

}

}

}

}

}

void main()

{

int i,j,s,de;

printf("enter number of nodes: ");

scanf("%d",&n);

printf("Enter the matrix:\n");

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

{

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

{

scanf("%d",&gr[i][j]);

}

}

printf("Enter the source node: ");

scanf("%d",&s);

printf("Enter the Destination node: ");

scanf("%d",&de);

BFS(s);

printf("Distance from %d to %d is %d\n",s,de,d[de]);

printf("Visiting order is:");

for(i=0;i<n;i++)

{

printf("%d ",q[i]);

}

***//Prims minimum Spanning tree……………………………***

void main()

{

int i,j,s,n,u,v,min,index;

int gr[20][20],visited[20],key[20],prev[20];

printf("enter number of nodes: ");

scanf("%d",&n);

printf("Enter the matrix:\n");

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

{

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

{

scanf("%d",&gr[i][j]);

}

}

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

{

visited[i]=0;

key[i]=999999;

prev[i]=-1;

}

s=1;

key[s]=0;

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

{

min=9999999;

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

{

if(key[j]<min && visited[j]==0)

{

min=key[j];

u=j;

}

}

visited[u]=1;

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

{

if(gr[u][v]>0 && visited[v]==0)

{

if(gr[u][v]<key[v])

{

key[v]=gr[u][v];

prev[v]=u;

}

}

}

}

printf("Edges of the minimum spanning tree are:\n");

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

{

printf("%d->%d\n",prev[i],i);

}

}

***//QuickSort.......................................................***

#include<stdio.h>

int A[20];

void Quicksort(int Left,int Right)

{

int q;

if(Left<Right)

{

q=Partition(Left,Right);

Quicksort(Left,q-1);

Quicksort(q+1,Right);

}

}

int Partition(int Left, int Right)

{

int x,i,j,temp;

x = A[Right];

i = Left-1;

for(j=Left;j<=Right-1; j++)

{

if(A[j]<=x)

{

i = i + 1;

temp=A[i];

A[i]=A[j];

A[j]=temp;

}

}

temp=A[i+1];

A[i+1]=A[Right];

A[Right]=temp;

return (i + 1);

}

void main()

{

int n,i;

printf("Enter size of array: ");

scanf("%d",&n);

printf("Enter data array: ");

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

scanf("%d",&A[i]);

Quicksort(1,n);

printf("Sorted Data: \n");

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

printf("%d ",A[i]);

}

***//MargeSort.......................................................***

#include<stdio.h>

int A[20];

void margesort(int Left,int Right)

{

int q;

if(Left<Right)

{

q=(Left+Right)/2;

margesort(Left,q);

margesort(q+1,Right);

marge(Left,q,Right);

}

}

void marge(int Left,int q,int Right)

{

int n1,n2,i,j,k;

n1=q-Left+1;

n2=Right-q;

int L[n1],R[n2];

for(i=1;i<=n1;i++)

{

L[i]=A[Left+i-1];

}

for(j=1;j<=n2;j++)

{

R[j]=A[q+j];

}

L[n1+1]=999;

R[n1+1]=999;

i=1;

j=1;

for( k=Left;k<=Right;k++)

{

if(L[i]<=R[j])

{

A[k]=L[i];

i=i+1;

}

else

{

A[k]=R[j];

j=j+1;

}

}

}

void main()

{

int n,i;

printf("Enter size of array: ");

scanf("%d",&n);

printf("Enter data array: ");

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

{

scanf("%d",&A[i]);

}

margesort(1,n);

printf("Sorted Data: \n");

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

{

printf("%d",A[i]);

}

}

***//0-1 knapse sort…………………………………………***

#include<stdio.h>

void main()

{

int p[20][20],i,j,weight,v[20],w[20],item;

printf("Enter total weight: ");

scanf("%d",&weight);

printf("Enter item number: ");

scanf("%d",&item);

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

{

printf("Enter value of %d: ",i);

scanf("%d",&v[i]);

printf("Enter weight of %d: ",i);

scanf("%d",&w[i]);

}

for(i=0;i<=item;i++)

{

for(j=0;j<=weight;j++)

{

if(i==0 || j==0)

{

p[i][j]=0;

}

else if(j-w[i]<0)

{

p[i][j]=p[i-1][j];

}

else if((v[i]+p[i-1][j-w[i]])>p[i-1][j])

p[i][j]=v[i]+p[i-1][j-w[i]];

else

{

p[i][j]=p[i-1][j];

}

}

}

printf("Output Table:\n");

for(i=0;i<=item;i++)

{

for(j=0;j<=weight;j++)

{

printf("%d ",p[i][j]);

}

printf("\n");

}

}