**Divide and Conquer Technique**

//Name : R.Giridhar Register Number : 15TD0244

//Binary Search

**Source Code:**

#include<stdio.h>

#include<conio.h>

int a[10];

void main()

{

int l,h,k,n,i;

printf("Enter value of n");

scanf("%d",&n);

printf("Enter The numbers");

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

{

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

}

l=1;

h=n;

printf("Enter the search value");

scanf("%d",&k);

BS(l,h,k);

printf("Not found");

getch();

}

BS(int l,int h,int k)

{

int m;

if(l<=h)

{

m=(l+h)/2;

if(a[m]==k)

{

printf("Found");

exit(0);

}

if(a[m]>k)

BS(l,m-1,k);

else

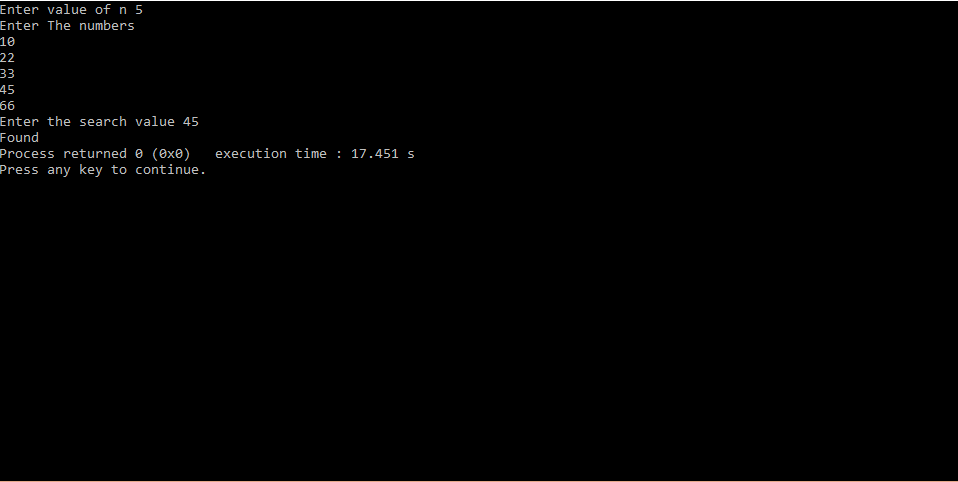
BS(m+1,h,k);

}

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//Merge Sort

**Source Code:**

#include<stdio.h>

#include<conio.h>

int a[200];

int ms(int,int);

int merge(int,int,int);

void main()

{

int n,i;

printf("Enter the value of n:");

scanf("%d",&n);

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

{

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

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

}

ms(1,n);

printf("Elements after Sorting:");

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

{

printf("\na[%d]=%d",i,a[i]);

}

}

int ms(int l,int k)

{

int m;

if(l<k)

{

m=(l+k)/2;

ms(l,m);

ms(m+1,k);

merge(l,m,k);

}

}

int merge(l,m,k)

{

int i,h,j,b[500],x;

i=l;

h=l;

j=m+1;

while((h<=m)&&(j<=k))

{

if(a[h]<=a[j])

{

b[i]=a[h];

h=h+1;

}

else

{ b[i]=a[j];

j=j+1;

}

i=i+1;

}

if(h>m)

{

for(x=j;x<=k;x++)

{ b[i]=a[x];

i=i+1;

}

}

else

{

for(x=h;x<=m;x++)

{

b[i]=a[x];

i=i+1;

}

}

for(x=l;x<=k;x++)

{

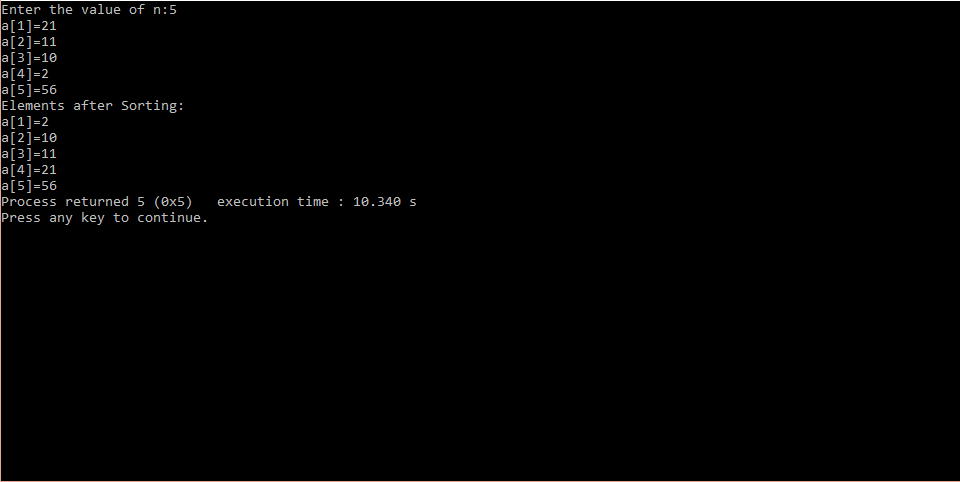
a[x]=b[x];

}

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//Quick Sort

**Source code:**

#include<stdio.h>

#include<conio.h>

int a[200];

int qs(int,int);

int part(int ,int);

void main()

{

int n,i;

printf("Enter the value of n:");

scanf("%d",&n);

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

{ printf("a[%d]=",i);

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

}

a[i]=32676;

qs(1,n);

printf("Elements after sorting");

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

{

printf("\na[%d]=%d",i,a[i]);

}

return 0;

}

int qs(p,q)

{

int j;

if(p<q)

{

j=part(p,q+1);

qs(p,j-1);

qs(j+1,q);

}

}

int part(m,p)

{

int v,i,j,t;

v=a[m];

i=m;

j=p;

do

{

do

{

i=i+1;

}

while(a[i]<=v);

do

{

j=j-1;

}

while(a[j]>v);

if(i<j)

{

t=a[i];

a[i]=a[j];

a[j]=t;

}

}

while(i<j);

a[m]=a[j];

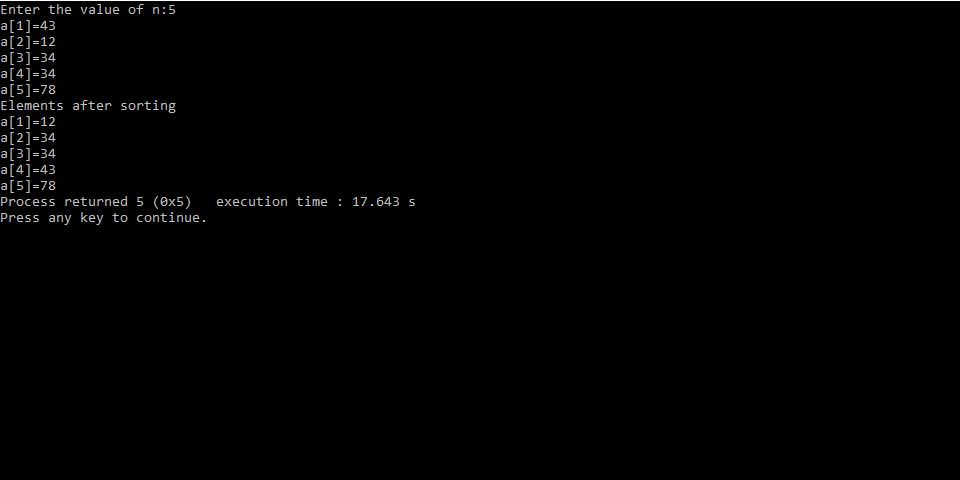
a[j]=v;

return j;

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



**Greedy**

**Technique**

//Name : R.Giridhar Register Number : 15TD0244

//Knapsack

**Source code:**

#include<stdio.h>

void main()

{

int i,j,n,m,u;

float value[20],x[20],w[20],p[20],temp,tot;

printf("Enter n");

scanf("%d",&n);

printf("\nEnter elements for weight");

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

{

printf("\n Enter the weight[%d]",i);

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

printf("\n Enter the profit[%d]",i);

scanf("%f",&p[i]);

value[i]= p[i] / w[i];

}

printf("\n Enter the maximum weight :");

scanf("%d",&m);

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

{

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

{

if(value[i]<value[j])

{

temp=w[i];

w[i]=w[j];

w[j]=temp;

temp=p[i];

p[i]=p[j];

p[j]=temp;

temp=value[i];

value[i]=value[j];

value[j]=temp;

}

}

}

printf(" \tWeight \t Profit \t Value");

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

{

printf("\n \t %f \t %f \t %f ",w[i],p[i],value[i]);

}

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

{

x[i]=0.0;

u=m;

}

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

{

if(w[i]<=u)

{

x[i]=1.0;

u = u - w[i];

}

else

{

break;

}

}

if(i!=n)

{

x[i]= u / w[i];

}

tot=0.0;

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

{

tot = tot +( p[i]\*x[i] );

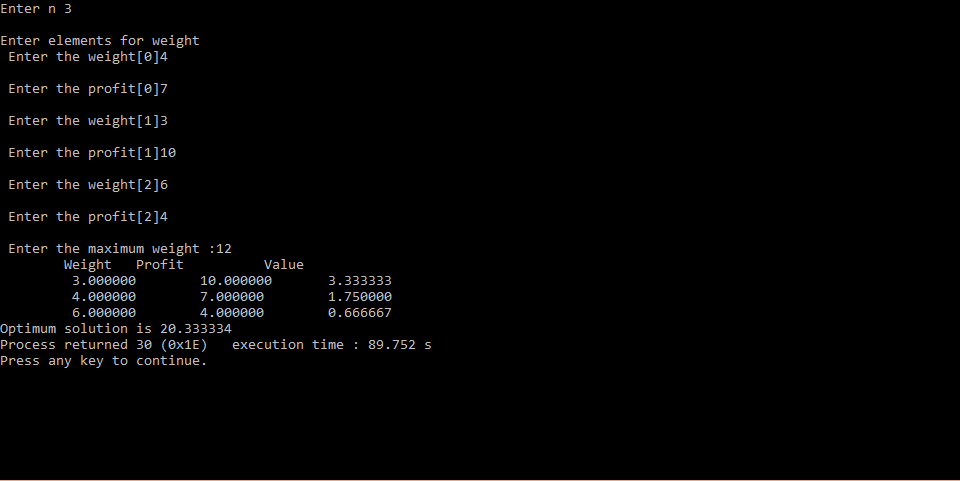
}

printf("\nOptimum solution is %f",tot);

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//Minimum Spanning Tree using Prims Algorithm

**Source code:**

#include<stdio.h>

int n,i,temp,j,k,l,cost[10][10],near[10],t[10][10],mincost=0,min=999;

void main()

{

printf("Enter the number of nodes:");

scanf("%d",&n);

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

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

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

{

printf("\n cost[%d][%d]=",i,j);

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

}

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

{

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

{

if(min>cost[i][j])

{

min=cost[i][j];

k=i;

l=j;

}

}

}

mincost=cost[k][l];

t[1][1]=k; t[1][2]=l;

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

{

if(cost[i][l]<cost[i][k])

{

near[i]=l;

}

else

{

near[i]=k;

}

}

near[k]=near[l]=0;

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

{

min=999;

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

{

if(near[j]!=0)

{

if(min>cost[j][near[j]])

{

min=cost[j][near[j]];

temp=j;

}

}

}

j=temp;

t[i][1]=j;t[i][2]=near[j];

mincost=(mincost+cost[j][near[j]]);

near[j]=0;

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

{

if((near[k]!=0)&&(cost[k][near[k]]>cost[k][j]))

near[k]=j;

}

}

mincost=(mincost+cost[j][near[j]]);

printf("\n mincost=%d",mincost);

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//Single Source Shortest Path

**Source code:**

#include<stdio.h>

int v=1,d[10],s[20],min=999,temp;

void main()

{

int n,cost[20][20],i,j,u,w,num;

printf("Enter the number of vertices");

scanf("%d",&n);

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

{

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

{

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

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

}

}

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

{

s[i]=0;

d[i]=cost[v][i];

}

s[v]=1;

d[v]=0;

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

{

min=999;

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

{

if(s[u]==0)

{

if(min>d[u])

{

min=d[u];

temp=u;

}

}

}

u=temp;

s[u]=1;

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

{

if(s[w]==0)

{

if(d[w]>d[u]+cost[u][w])

{

d[w]=d[u]+cost[u][w];

}

}

}

}

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

{

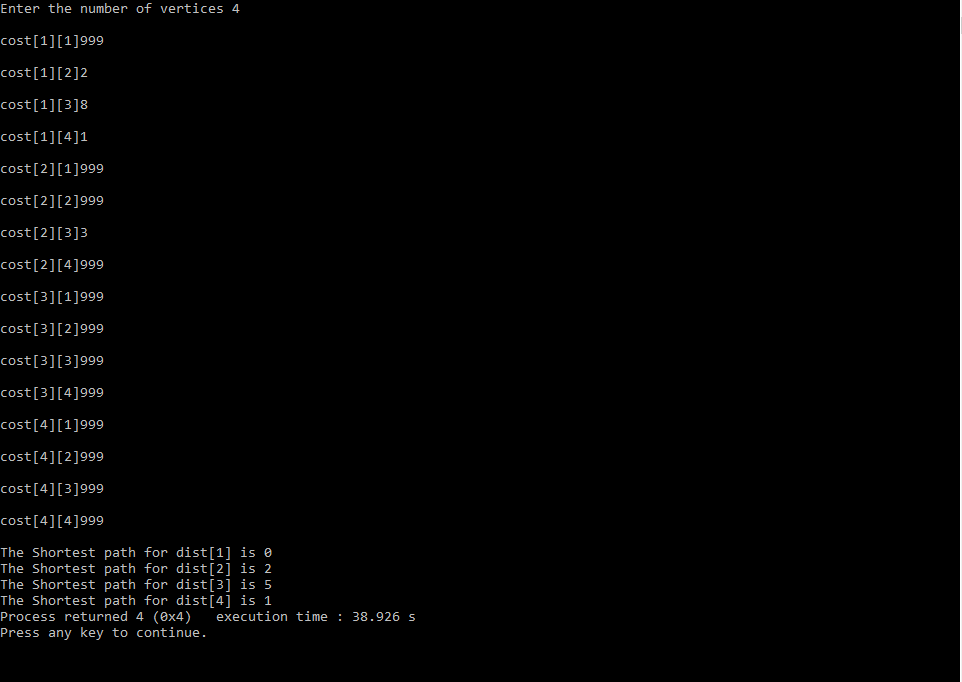
printf("\nThe Shortest path for dist[%d] is %d",i,d[i]);

}

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



**Dynamic**

**Programming & Traversal**

**Techniques**

//Name : R.Giridhar Register Number : 15TD0244

//MultiStage Graph

**Source code:**

#include<stdio.h>

void main()

{

int path[5][5],i,j,min,a[5][5],p,stp,st=1,ed=5,edp,t[5],index;

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

for(i=1;i<=5;i++)

{

for(j=1;j<=5;j++)

{

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

}

}

printf("Enter the number of paths \n");

scanf("%d",&p);

printf("Enter the possible paths:\n");

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

{

for(j=1;j<=5;j++)

{

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

}

}

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

{

t[i]=0;

stp=st;

for(j=1;j<=5;j++)

{

edp=path[i][j+1];

t[i]=t[i]+a[stp][edp];

if(edp==ed)

{

break;

}

else

{

stp=edp;

}

}

min=t[st];

index=st;

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

{

if(min>t[i])

{

min=t[i];

index=i;

}

}

}

printf("Minimum cost %d",min);

printf("\n Minimum cost path");

for(i=1;i<=5;i++)

{

printf("-> %d", path[index][i]);

if(path[index][i]==ed)

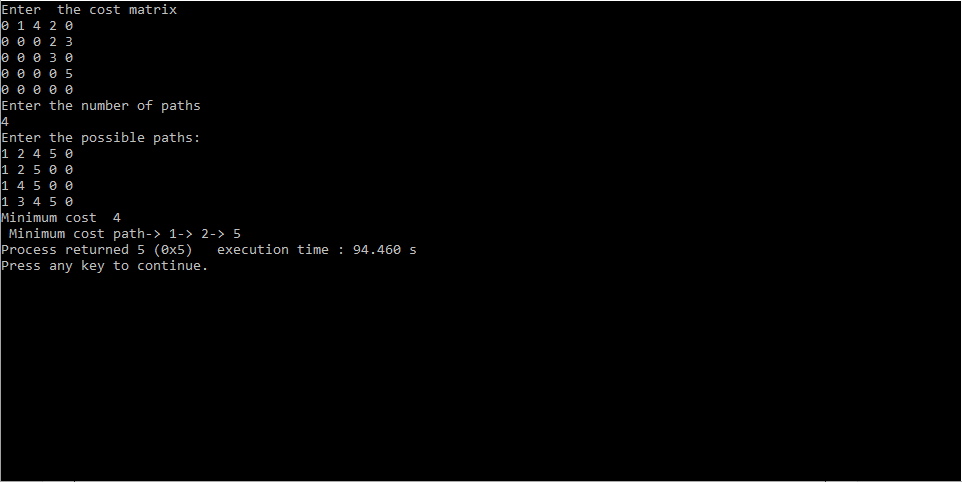
break;

}

}

//Name : R.Giridhar Register Number : 15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//All Pair Shortest Path

**Source code:**

#include<stdio.h>

int cost[20][20],n;

void main()

{

int i,j;

printf("Enter the number of nodes:");

scanf("%d",&n);

printf("Enter the cost matrix");

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

{

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

{

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

}

}

allpair();

}

allpair()

{

int i,j,k,a[20][20];

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

{

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

{

a[i][j]=cost[i][j];

}

}

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

{

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

{

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

{

if(a[i][j]<(a[i][k]+a[k][j]))

{

a[i][j]=a[i][j];

}

else

{

a[i][j]=(a[i][k]+a[k][j]);

}

}

}

printf("Step:%d\n",k);

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

{

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

{

printf("\t%d",a[i][j]);

}

printf("\n");

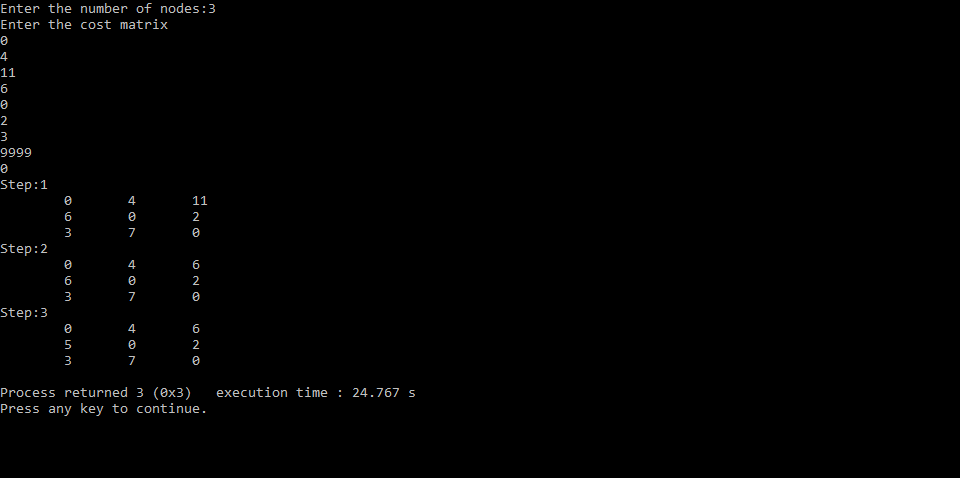
}

}

}

//Name : R.Giridhar Register Number : 15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//DFS Traversal

**Source code:**

#include<stdio.h>

int g[10][10],vis[10],n;

void main()

{

int i,j;

printf("Enter number of nodes");

scanf("%d", &n);

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

{

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

{

printf("\Enter G[%d][%d]:",i,j);

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

}

}

dfs(1);

}

dfs(int v)

{

int w;

vis[v]=1;

printf("\n Visited node is %d",v);

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

{

if((g[v][w]==1)&& (vis[w]==0))

{

dfs(w);

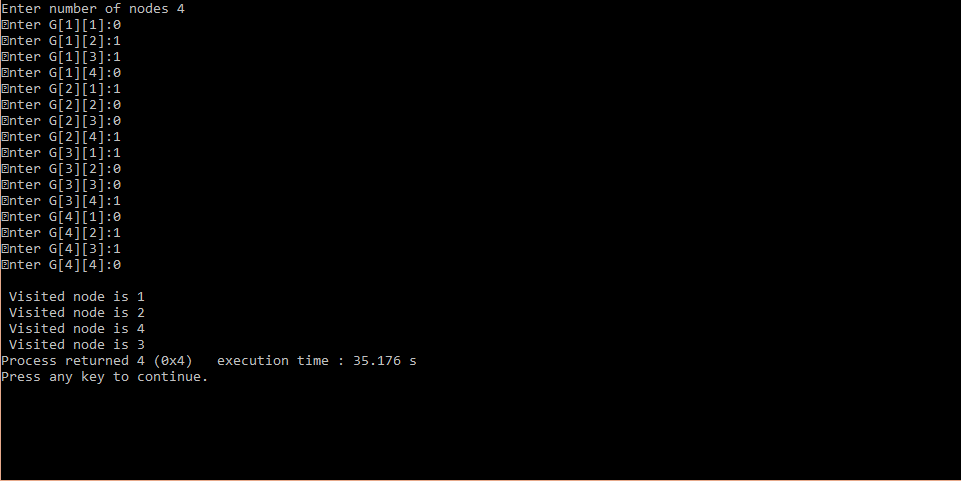
}

}

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//BFS Traversal

**Source code:**

#include<stdio.h>

int g[10][10],vis[10],n,max=10,q[20],rear,front;

void main()

{

int i,j;

printf("Enter number of nodes");

scanf("%d", &n);

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

{

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

{

printf("\n Enter G[%d][%d]:",i,j);

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

}

}

bfs(1);

}

bfs(int v)

{

int y,w;

y=v;

vis[v]=1;

printf("\n Visited node is %d",v);

while(1)

{

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

{

if((g[y][w]==1)&&(vis[w]==0))

{

if(rear!=max)

{

rear=rear+1;

q[rear]=w;

vis[w]=1;

printf("\n Visited node is %d",w);

}

}

}

if(front!=rear)

{

y=q[front];

front++;

}

else

{

exit(0);

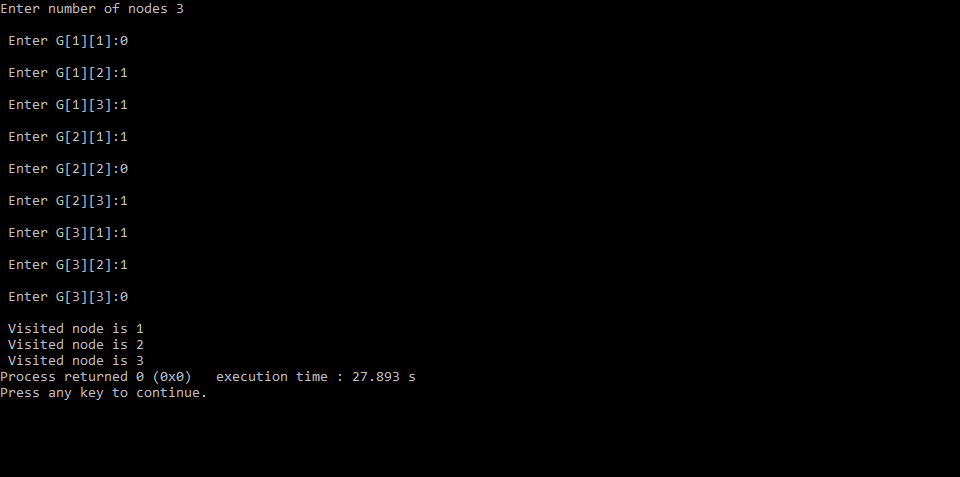
}

}

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



**Backtracking**

//Name : R.Giridhar Register Number : 15TD0244

//nQueen

**Source code:**

#include<stdio.h>

int x[20];

int nqueen(int k,int n)

{

int i,j;

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

{

if(place(k,i))

{

x[k]=i;

if(k==n)

{

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

{

printf("x[%d]=%d \t",j,x[j]);

}

printf("\n");

}

else

nqueen(k+1,n);

}

}

return 0;

}

int place(int k,int i)

{

int j;

for(j=1;j<=k-1;j++)

{

if( (x[j]==i) || (abs(x[j]-i)==abs(j-k)) )

{

return 0;

}

}

return 1;

}

void main()

{

int n ;

printf("Enter the number of Queens");

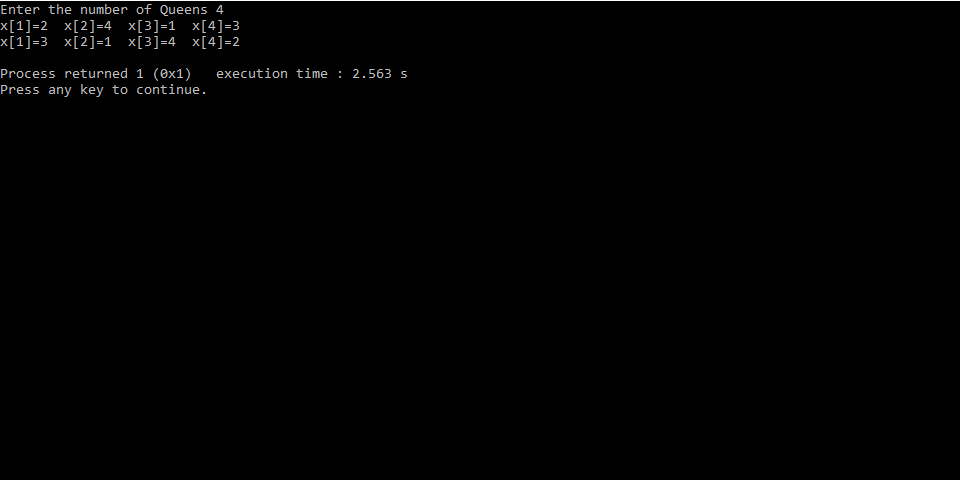
scanf("%d",&n);

nqueen(1,n);

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number : 15TD0244

//Sum of Subsets

**Source code:**

#include<stdio.h>

int w[20],m,x[20];

void main()

{

int n,r=0,i;

printf("Enter the number of elements");

scanf("%d",&n);

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

{

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

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

}

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

{

r=r+w[i];

}

printf("\nEnter the constraint");

scanf("%d",&m);

sos(0,1,r);

}

int sos(int s,int k,int r)

{

int j;

x[k]=1;

if((s+w[k])==m)

{

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

{

printf("x[%d]=%d\n",j,x[j]);

printf("\n");

}

}

else if((s+w[k]+w[k+1])<=m)

{

sos(s+w[k],k+1,r-w[k]);

}

if( (s+r-w[k])>=m && (s+w[k+1])<=m )

{

x[k]=0;

sos(s,k+1,r-w[k]);

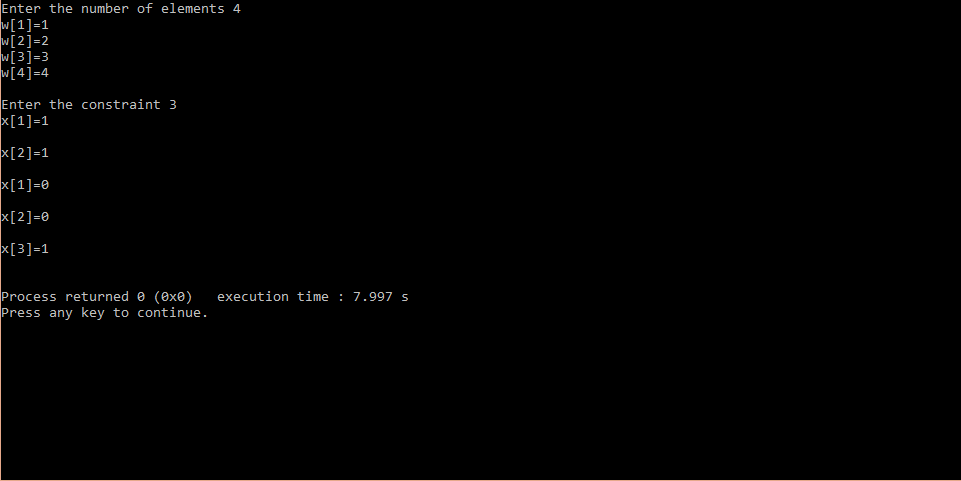
}

return 0;

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number:15TD0244

//Hamilton Cycle

**Source code:**

#include<stdio.h>

void Hamiltonian(int k);

void nextvalue(int k);

int x[10],g[14][12];

int k=2,n,i,j;

void main()

{

x[1]=1;

printf("\nEnter the no of nodes:");

scanf("%d",&n);

printf("enter the value of nodes");

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

{

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

{

printf("\ng[%d][%d]=",i,j);

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

}

}

Hamiltonian(2);

}

void Hamiltonian(int k)

{

do

{

nextvalue(k);

if(x[k]==0)

return;

if(k==n)

{

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

{

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

}

printf("\n");

//break;

//break;

}

else

Hamiltonian(k+1);

}while(1);

}

void nextvalue(k)

{

do

{

x[k]=(x[k]+1)%(n+1);

if(x[k]==0)

return ;

if(g[x[k-1],x[k]]!=0)

{

for(j=1;j<=k-1;j++)

{

if(x[j]==x[k])

{

break;

}

}

}

if(j==k)

{

if((k<n)||((k==n)&&g[x[n],x[1]]!=0))

{

return;

}

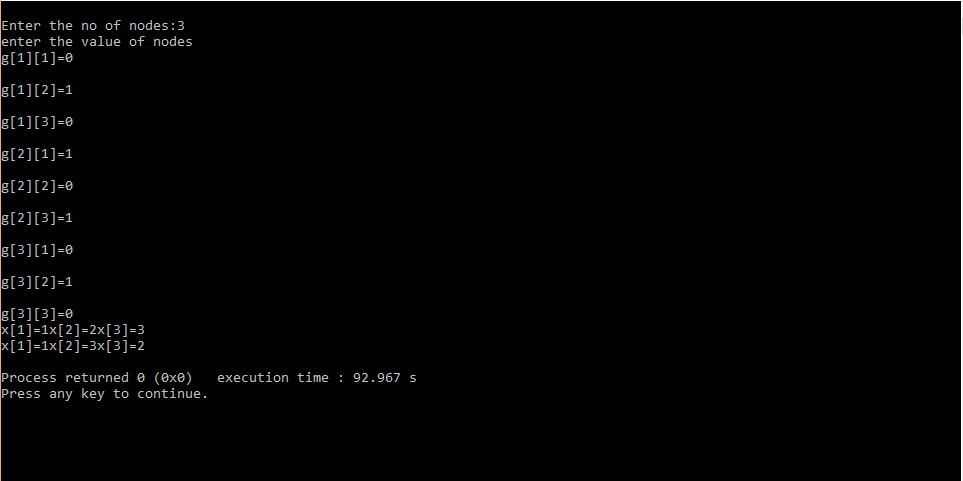
}

}while(1);

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**



//Name : R.Giridhar Register Number:15TD0244

//mcolouring

**Source code:**

#include<stdio.h>

int g[10][10],x[10],n,m=4;

void main()

{

int i,j;

printf("Enter the nodes");

scanf("%d",&n);

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

{

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

{

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

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

}

}

mcolouring(1);

}

int mcolouring(int k)

{

int i;

do

{

nextvalue(k);

if(x[k]==0)

return;

if(k==n)

{

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

{

printf("X[%d]=%d \t",i,x[i]);

}

exit(0);

}

else

mcolouring(k+1);

}while(1);

return 0;

}

int nextvalue(int k)

{

int j;

do

{

x[k]=(x[k]+1)%(m+1);

if(x[k]==0)

return;

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

{

if( (g[k][j]!=0) && (x[k]==x[j]) )

break;

}

if(j==n+1)

return;

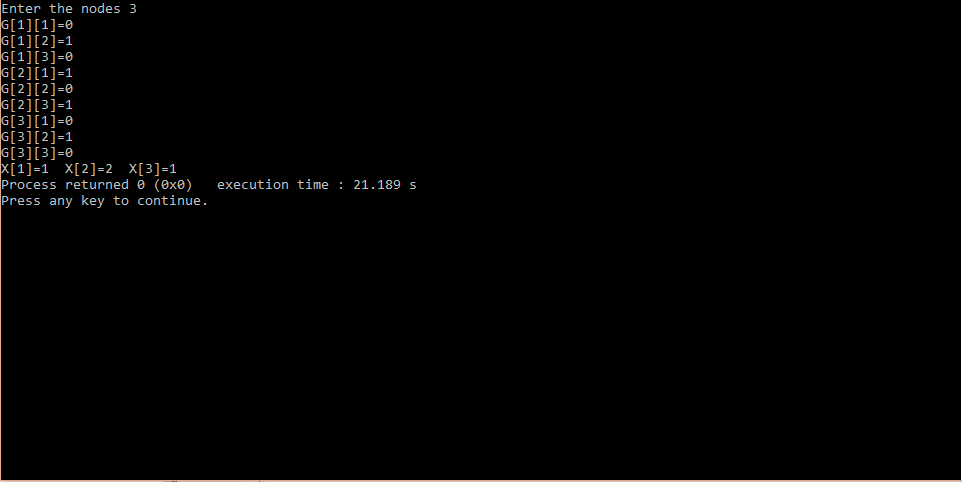
}while(1);

return 0;

}

//Name : R.Giridhar Register Number:15TD0244

**Output:**

****