**CODE: Sorting1**

#include<stdio.h>

#include<string.h>

#define SIZE 30

int a[SIZE],n;

void getArray();

void display(int a[], int n);

void insertion\_sort(int a[], int n);

void merge\_sort(int a[]);

void merge(int l[], int r[], int a[]);

int length(int a[]);

int main()

{

int choice;

do

{

printf("\n\*\*\*\*\*MAIN MENU\*\*\*\*\*\n");

printf("1. Enter array\n");

printf("2. Insertion Sort\n");

printf("3. Merge Sort\n");

printf("4. EXIT\n");

printf("Enter your choice: ");

scanf("%d",&choice);

printf("\n");

switch(choice)

{

case 1 : memset(a,0,SIZE\*sizeof(int));

getArray();

display(a,n);

break;

case 2 : insertion\_sort(a,n);

display(a,n);

break;

case 3 : merge\_sort(a);

printf("Successfully sorted using Merge Sort Algorithm\n");

display(a,n);

break;

case 4 : break;

}

}while(choice!=4);

return 0;

}

void getArray()

{

int i;

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

scanf("%d",&n);

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

{

printf("Enter a value: ");

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

}

}

void display(int a[],int n)

{

int i;

printf("The Array is: ");

for(i=0;i<length(a);i++)

{

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

}

printf("\n");

}

void insertion\_sort(int a[], int n)

{

int key,i,j;

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

{

key=a[j];

i=j-1;

while(a[i]>key && i!=0)

{

a[i+1]=a[i];

i=i-1;

}

a[i+1]=key;

}

printf("Successfully sorted using Insertion Sort Algorithm\n");

}

int length(int a[])

{

int i=0;

while(a[i]!='\0')

{

i++;

}

return i;

}

void merge\_sort(int a[])

{

int i,len=0,l[SIZE]={0},r[SIZE]={0},mid;

len=length(a);

if(len>1)

{

mid=(len-1)/2;

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

{

l[i]=a[i];

}

for(i=mid+1;i<len;i++)

{

r[i-mid-1]=a[i];

}

merge\_sort(l);

merge\_sort(r);

merge(l,r,a);

}

else

return ;

}

void merge(int l[], int r[], int a[])

{

int i=0,j=0,k=0;

while(i<length(l) && j<length(r))

{

if(l[i]<r[j])

{

a[k]=l[i];

i++;

k++;

}

else

{

a[k]=r[j];

k++;

j++;

}

}

while(i<length(l))

{

a[k]=l[i];

i++;

k++;

}

while(j<length(r))

{

a[k]=r[j];

j++;

k++;

}

}

**OUTPUT:**

ubuntu@ubuntu:~$ cd Desktop

ubuntu@ubuntu:~/Desktop$ gcc Sort1.c -o Sort1

ubuntu@ubuntu:~/Desktop$ ./Sort1

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Insertion Sort

3. Merge Sort

4. EXIT

Enter your choice: 1

Enter the number of elements: 7

Enter a value: 30

Enter a value: 50

Enter a value: 40

Enter a value: 20

Enter a value: 70

Enter a value: 10

Enter a value: 60

The Array is: 30 50 40 20 70 10 60

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Insertion Sort

3. Merge Sort

4. EXIT

Enter your choice: 2

Successfully sorted using Insertion Sort Algorithm

The Array is: 10 20 30 40 50 60 70

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Insertion Sort

3. Merge Sort

4. EXIT

Enter your choice: 1

Enter the number of elements: 5

Enter a value: 5

Enter a value: 4

Enter a value: 3

Enter a value: 2

Enter a value: 1

The Array is: 5 4 3 2 1

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Insertion Sort

3. Merge Sort

4. EXIT

Enter your choice: 3

Successfully sorted using Merge Sort Algorithm

The Array is: 1 2 3 4 5

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Insertion Sort

3. Merge Sort

4. EXIT

Enter your choice: 4

**CODE: Sorting2**

#include<stdio.h>

#include<string.h>

#define SIZE 30

int a[SIZE],n;

void getArray();

void display(int a[], int n);

void selection\_sort(int a[], int n);

int partition(int a[], int start, int end);

void quick\_sort(int a[], int start, int end);

void swap(int i, int j);

int length(int a[]);

int main()

{

int choice;

do

{

printf("\n\*\*\*\*\*MAIN MENU\*\*\*\*\*\n");

printf("1. Enter array\n");

printf("2. Selection Sort\n");

printf("3. Quick Sort\n");

printf("4. EXIT\n");

printf("Enter your choice: ");

scanf("%d",&choice);

printf("\n");

switch(choice)

{

case 1 : memset(a,0,SIZE\*sizeof(int));

getArray();

display(a,n);

break;

case 2 : selection\_sort(a,n);

display(a,n);

break;

case 3 : quick\_sort(a,0,n-1);

printf("Successfully sorted using Quick Sort Algorithm\n");

display(a,n);

break;

case 4 : break;

}

}while(choice!=4);

return 0;

}

void getArray()

{

int i;

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

scanf("%d",&n);

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

{

printf("Enter a value: ");

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

}

}

void display(int a[],int n)

{

int i;

printf("The Array is: ");

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

{

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

}

printf("\n");

}

void selection\_sort(int a[], int n)

{

int imin, i, j, temp;

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

{

imin=i;

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

{

if(a[imin]>a[j])

{

imin=j;

}

}

temp=a[i];

a[i]=a[imin];

a[imin]=temp;

}

printf("Successfully sorted using Selection Sort Algorithm\n");

}

void swap(int i, int j)

{

int temp=a[i];

a[i]=a[j];

a[j]=temp;

}

void quick\_sort(int a[], int start, int end)

{

int p;

if(start<end)

{

p=partition(a,start,end);

quick\_sort(a,start,p-1);

quick\_sort(a,p+1,end);

}

}

int partition(int a[], int start, int end)

{

int pivot, pindex, i,temp;

pivot=a[end];

pindex=start;

for(i=start;i<=end-1;i++)

{

if(a[i]<pivot)

{

swap(i,pindex);

pindex++;

}

}

swap(pindex,end);

return pindex;

}

**OUTPUT:**

ubuntu@ubuntu:~$ cd Desktop

ubuntu@ubuntu:~/Desktop$ gcc Sort2.c -o Sort2

ubuntu@ubuntu:~/Desktop$ ./Sort2

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Selection Sort

3. Quick Sort

4. EXIT

Enter your choice: 1

Enter the number of elements: 7

Enter a value: 30

Enter a value: 50

Enter a value: 40

Enter a value: 10

Enter a value: 70

Enter a value: 60

Enter a value: 20

The Array is: 30 50 40 10 70 60 20

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Selection Sort

3. Quick Sort

4. EXIT

Enter your choice: 2

Successfully sorted using Selection Sort Algorithm

The Array is: 10 20 30 40 50 60 70

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Selection Sort

3. Quick Sort

4. EXIT

Enter your choice: 1

Enter the number of elements: 8

Enter a value: 136

Enter a value: 32

Enter a value: 75

Enter a value: 23

Enter a value: 97

Enter a value: 21

Enter a value: 65

Enter a value: 26

The Array is: 136 32 75 23 97 21 65 26

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Selection Sort

3. Quick Sort

4. EXIT

Enter your choice: 3

Successfully sorted using Quick Sort Algorithm

The Array is: 21 23 26 32 65 75 97 136

\*\*\*\*\*MAIN MENU\*\*\*\*\*

1. Enter array

2. Selection Sort

3. Quick Sort

4. EXIT

Enter your choice: 4

**CODE: Tennis**(Non-Recursive Algorithm)

#include<stdio.h>

#include<stdlib.h>

void main()

{

int a[100],n,count,i;

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

scanf("%d",&n);

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

{

a[i]=rand()%100+1;

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

}

printf("\n");

count=n;

while(count>1)

{

count=0;

for(i=0;i<n-1;i=i+2)

{

if(a[i]>a[i+1])

{

a[count++]=a[i];

}

else

{

a[count++]=a[i+1];

}

}

if(n%2==0)

{

n=n/2;

}

else

{

a[count++]=a[n-1];

n=(n/2)+1;

}

}

printf("The winner is: %d\n",a[0]);

}

**OUTPUT:**

ubuntu@ubuntu:~$ cd Desktop

ubuntu@ubuntu:~/Desktop$ gcc Tennis.c -o Tennis

ubuntu@ubuntu:~/Desktop$ ./Tennis

Enter the number of players: 20

84 87 78 16 94 36 87 93 50 22 63 28 91 60 64 27 41 27 73 37

The winner is: 94

**CODE: Tennis**(Recursive Algorithm)

int winner(int c[],int start,int end);

void main()

{

int a[100],n,w,i;

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

scanf("%d",&n);

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

{

a[i]=rand()%100+1;

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

}

printf("\n");

w=winner(a,0,n-1);

printf("The winner is: %d\n",w);

}

int winner(int c[],int start,int end)

{

int w1,w2,mid;

if(start==end)

{

return c[start];

}

else if(start==end-1)

{

if(c[start]>c[end])

return c[start];

else

return c[end];

}

else

{

mid=(start+end)/2;

w1=winner(c,start,mid);

w2=winner(c,mid+1,end);

if(w1>w2)

return w1;

else

return w2;

}

}

**OUTPUT:**

ubuntu@ubuntu:~$ cd Desktop

ubuntu@ubuntu:~/Desktop$ gcc Tennis.c -o Tennis

ubuntu@ubuntu:~/Desktop$ ./Tennis

Enter the number of players: 7

84 87 78 16 94 36 87

The winner is: 94

**CODE: Knapsack(Greedy)**

#include<stdio.h>

int weight,n;

float p\_w[10],x[10],p[10],w[10];

void greedy(int weight, int n)

{

int i;

float rem\_weight=weight, profit=0;

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

x[i]=0;

printf(" x[i] \t w[i] \t p[i]\n");

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

{

if(w[i]>rem\_weight)

break;

x[i]=1;

rem\_weight=rem\_weight-w[i];

profit=profit+p[i];

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

}

if(i<n)

{

x[i]=rem\_weight/w[i];

profit=profit+x[i]\*p[i];

rem\_weight=rem\_weight-x[i]\*w[i];

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

}

printf("The maximum profit is: %f\n",profit);

}

int main()

{

int i,j;

float temp;

printf("Enter the weight of Knapsack and number of objects: ");

scanf("%d %d",&weight,&n);

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

{

printf("Enter the weight and profit of object: ");

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

p\_w[i]=p[i]/w[i];

}

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

{

for(j=0;j<n-1;j++)

{

if(p\_w[j]<p\_w[j+1])

{

temp=p\_w[j];

p\_w[j]=p\_w[j+1];

p\_w[j+1]=temp;

temp=p[j];

p[j]=p[j+1];

p[j+1]=temp;

temp=w[j];

w[j]=w[j+1];

w[j+1]=temp;

}

}

}

greedy(weight,n);

return 0;

}

**OUTPUT:**

ubuntu@ubuntu:~$ cd Desktop

ubuntu@ubuntu:~/Desktop$ gcc Greedy.c -o Greedy

ubuntu@ubuntu:~/Desktop$ ./Greedy

Enter the weight of Knapsack and number of objects: 20 3

Enter the weight and profit of object: 18 25

Enter the weight and profit of object: 15 24

Enter the weight and profit of object: 10 15

x[i] w[i] p[i]

1.000000 15.000000 24.000000

0.500000 10.000000 15.000000

The maximum profit is: 31.500000

**CODE: TSP**

#include<stdio.h>

int a[10][10],visited[10],n,cost=0;

void get()

{

int i,j;

printf("Enter No. of Cities: ");

scanf("%d",&n);

printf("\nEnter Cost Matrix\n");

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

{

printf("\nEnter Elements of Row # : %d\n",i+1);

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

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

visited[i]=0;

}

printf("\n\nThe cost list is:\n\n");

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

{

printf("\n\n");

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

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

}

}

void mincost(int city)

{

int i,ncity;

visited[city]=1;

printf("%d -->",city+1);

ncity=least(city);

if(ncity==999)

{

ncity=0;

printf("%d",ncity+1);

cost+=a[city][ncity];

return;

}

mincost(ncity);

}

int least(int c)

{

int i,nc=999;

int min=999,kmin;

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

{

if((a[c][i]!=0)&&(visited[i]==0))

if(a[c][i] < min)

{

min=a[i][0]+a[c][i];

kmin=a[c][i];

nc=i;

}

}

if(min!=999)

cost+=kmin;

return nc;

}

void put()

{

printf("\n\nMinimum cost:");

printf("%d",cost);

}

void main()

{

get();

printf("\n\nThe Path is:\n\n");

mincost(0);

put();

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc TSP.c -o TSP

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./TSP

Enter No. of Cities: 4

Enter Cost Matrix

Enter Elements of Row # : 1

1

5

4

2

Enter Elements of Row # : 2

2

1

5

4

Enter Elements of Row # : 3

9

6

2

4

Enter Elements of Row # : 4

7

5

3

4

The cost list is:

1 5 4 2

2 1 5 4

9 6 2 4

7 5 3 4

The Path is:

1 –->4 -–>3 -–>2 -–>1

**CODE: 15 Puzzle**

#include<stdio.h>

int m=0,n=4;

int cal(int temp[10][10],int t[10][10])

{

int i,j,m=0;

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

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

{

if(temp[i][j]!=t[i][j])

m++;

}

return m;

}

int check(int a[10][10],int t[10][10])

{

int i,j,f=1;

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

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

if(a[i][j]!=t[i][j])

f=0;

return f;

}

void main()

{

int p,i,j,n=4,a[10][10],t[10][10],temp[10][10],r[10][10];

int m=0,x=0,y=0,d=1000,dmin=0,l=0;

clrscr();

printf("\nEnter the matrix to be solved,space with zero :\n");

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

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

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

printf("\nEnter the target matrix,space with zero :\n");

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

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

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

printf("\nEntered Matrix is :\n");

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

{

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

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

printf("\n");

}

printf("\nTarget Matrix is :\n");

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

{

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

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

printf("\n");

}

while(!(check(a,t)))

{

l++;

d=1000;

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

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

{

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

{

x=i;

y=j;

}

}

//To move upwards

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

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

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

if(x!=0)

{

p=temp[x][y];

temp[x][y]=temp[x-1][y];

temp[x-1][y]=p;

}

m=cal(temp,t);

dmin=l+m;

if(dmin<d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

//To move downwards

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

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

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

if(x!=n-1)

{

p=temp[x][y];

temp[x][y]=temp[x+1][y];

temp[x+1][y]=p;

}

m=cal(temp,t);

dmin=l+m;

if(dmin<d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

//To move right side

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

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

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

if(y!=n-1)

{

p=temp[x][y];

temp[x][y]=temp[x][y+1];

temp[x][y+1]=p;

}

m=cal(temp,t);

dmin=l+m;

if(dmin<d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

//To move left

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

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

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

if(y!=0)

{

p=temp[x][y];

temp[x][y]=temp[x][y-1];

temp[x][y-1]=p;

}

m=cal(temp,t);

dmin=l+m;

if(dmin<d)

{

d=dmin;

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

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

r[i][j]=temp[i][j];

}

printf("\nCalculated Intermediate Matrix Value :\n");

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

{

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

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

printf("\n");

}

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

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

{

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

temp[i][j]=0;

}

printf("Minimum cost : %d\n",d);

}

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc 15Puzzle.c -o 15

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./15

Enter the matrix to be solved,space with zero:

1

2

3

4

5

6

0

8

9

10

7

11

13

14

15

12

Enter the target matrix,space with zero :

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

0

Entered Matrix is :

1 2 3 4

5 6 0 8

9 10 7 11

13 14 15 12

Target Matrix is :

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Calculated Intermediate Matrix Value :

1 2 3 4

5 6 7 8

9 10 0 11

13 14 15 12

Minimum cost : 4

Calculated Intermediate Matrix Value :

1 2 3 4

5 6 7 8

9 10 11 0

13 14 15 12

Minimum cost : 4

Calculated Intermediate Matrix Value :

1 2 3 4

5 6 7 8

9 10 11 12

13 14 15 0

Minimum cost : 3

**CODE: Djsktra**

#include<stdio.h>

#define infinity 999

void dij(int n,int v,int cost[10][10],int dist[])

{

int i,u,count,w,flag[10],min;

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

{

flag[i]=0;

if(i!=v)

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

else

dist[i]=0;

}

count=1;

while(count<n)

{

min=99;

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

if(dist[w]<min && !flag[w])

min=dist[w],u=w;

flag[u]=1;

count++;

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

{

if((dist[u]+cost[u][w]<dist[w]) && !flag[w])

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

}

}

}

void main()

{

int n,v,i,j,cost[10][10],dist[10];

printf("\nEnter the number of nodes:");

scanf("%d",&n);

printf("\nEnter the cost matrix:\n");

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

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

{

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

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

cost[i][j]=infinity;

}

printf("\nEnter the source of the matrix:");

scanf("%d",&v);

dij(n,v,cost,dist);

printf("\nShortest path:\n");

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

printf("%d->%d,cost=%d\n",v,i,dist[i]);

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc Djsktra.c –o Djsktra

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./Djkstra

Enter the number of nodes:9

Enter the cost matrix:

0 4 0 0 0 0 0 8 0

4 0 8 0 0 0 0 11 0

0 8 0 7 0 4 0 0 2

0 0 7 0 9 14 0 0 0

0 0 0 9 0 10 0 0 0

0 0 4 14 10 0 2 0 0

0 0 0 0 0 2 0 1 6

8 11 0 0 0 0 1 0 7

0 0 2 0 0 0 6 7 0

Enter the source of the matrix:1

Shortest path:

1->0,cost=4

1->1,cost=0

1->2,cost=8

1->3,cost=15

1->4,cost=22

1->5,cost=12

1->6,cost=12

1->7,cost=11

1->8,cost=10

**CODE: Prims**

#include<stdio.h>

int a,b,u,v,n,i,ne=1,j;

int visited[10]={0},min,mincost=0,cost[10][10];

void main()

{

printf("\nEnter the number of nodes:");

scanf("%d",&n);

printf("\nEnter the adjacency matrix:\n");

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

{

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

{

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

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

cost[i][j]=999;

}

}

visited[1]=1;

printf("\n");

while(ne < n)

{

for(i=1,min=999;i<=n;i++)

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

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

if(visited[i]!=0)

{

min=cost[i][j];

a=u=i;

b=v=j;

}

if(visited[u]==0 || visited[v]==0)

{

printf("\n Edge %d:(%d %d) cost:%d",ne++,a,b,min);

mincost+=min;

visited[b]=1;

}

cost[a][b]=cost[b][a]=999;

}

printf("\nMinimun cost=%d",mincost);

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc Prim.c -o Prim

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./Prim

Enter the number of nodes:5

Enter the adjacency matrix:

0 3 0 0 1

3 0 9 2 2

0 9 0 3 7

0 2 3 0 0

1 2 7 0 0

Edge 1:(1 5) cost:1

Edge 2:(5 2) cost:2

Edge 3:(2 4) cost:2

Edge 4:(4 3) cost:3

Minimun cost=8

**CODE: Kruskals**

#include<stdio.h>

int min,mincost=0,cost[9][9],parent[9];

int find(int i)

{

while(parent[i])

i=parent[i];

return i;

}

int uni(int i,int j)

{

if(i!=j)

{

parent[j]=i;

return 1;

}

return 0;

}

void main()

{

int i,j,k,a,b,u,v,n,ne=1;

printf("\nEnter the no. of vertices :\n");

scanf("%d",&n);

printf("\nEnter the cost adjacency matrix :\n");

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

{

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

{

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

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

cost[i][j]=9999;

}

}

printf("\nThe edges of Minimum Cost Spanning Tree are :\n\n");

while(ne<n)

{

min=9999;

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

{

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

{

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

{

min=cost[i][j];

a=u=i;

b=v=j;

}

}

}

u=find(u);

v=find(v);

if(uni(u,v))

{

printf("%d. edge (%d,%d) =%d\n",ne++,a,b,min);

mincost +=min;

}

cost[a][b]=cost[b][a]=999;

}

printf("\n\t\*\*\*Minimum cost = %d\*\*\*\n",mincost);

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc Kruskal.c -o Kruskal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./Kruskal

Enter the no. of vertices :

7

Enter the cost adjacency matrix :

0 28 0 0 0 10 0

28 0 16 0 0 0 14

0 16 0 12 0 0 0

0 0 12 0 22 0 18

0 0 0 22 0 0 24

10 0 0 0 25 0 0

0 14 0 18 24 0 0

The edges of Minimum Cost Spanning Tree are :

1. edge (1,6) =10

2. edge (3,4) =12

3. edge (2,7) =14

4. edge (2,3) =16

5. edge (4,5) =22

6. edge (6,5) =25

\*\*\*Minimum cost = 99\*\*\*

**CODE: LCS**

#include<stdio.h>

#include<string.h>

int i,j,m,n,a,c[20][20];

char x[15],y[15],b[20][20];

void print\_lcs(int i,int j);

void lcs\_length(void);

void main()

{

printf("Enter 1st sequence : ");

scanf("%s",x);

printf("Enter 2nd sequence : ");

scanf("%s",y);

lcs\_length();

printf("\n");

}

void print\_lcs(int i,int j)

{

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

return;

if(b[i][j]=='c')

{

print\_lcs(i-1,j-1);

printf(" %c",x[i-1]);

}

else if(b[i][j]=='u')

print\_lcs(i-1,j);

else

print\_lcs(i,j-1);

}

void lcs\_length(void)

{

m=strlen(x);

n=strlen(y);

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

c[i][0]=0;

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

{

printf("0 \t");

c[0][i]=0;

}

printf("\n");

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

{

printf("0 \t");

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

{

if(x[i-1]==y[j-1])

{

c[i][j]=1+c[i-1][j-1];

b[i][j]='c';

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

}

else if(c[i-1][j] >= c[i][j-1])

{

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

b[i][j]='u';

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

}

else

{

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

b[i][j]='l';

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

}

}

printf(" \n");

}

printf("\nLongest common subsequence is :");

print\_lcs(m,n);

printf("\n");

printf("The length of Subsequence is: %d",c[m][n]);

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc LCS.c -o LCS

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./LCS

Enter 1st sequence : 1000101011

Enter 2nd sequence : 00011101

0 0 0 0 0 0 0 0 0

0 0 U 0 U 0 U 1 D 1 D 1 D 1 L 1 D

0 1 D 1 D 1 D 1 U 1 U 1 U 2 D 2 L

0 1 D 2 D 2 D 2 L 2 L 2 L 2 D 2 U

0 1 D 2 D 3 D 3 L 3 L 3 L 3 D 3 L

0 1 U 2 U 3 U 4 D 4 D 4 D 4 L 4 D

0 1 D 2 D 3 D 4 U 4 U 4 U 5 D 5 L

0 1 U 2 U 3 U 4 D 5 D 5 D 5 U 6 D

0 1 D 2 D 3 D 4 U 5 U 5 U 6 D 6 U

0 1 U 2 U 3 U 4 D 5 D 6 D 6 U 7 D

0 1 U 2 U 3 U 4 D 5 D 6 D 6 U 7 D

Longest common subsequence is : 0 0 0 1 1 0 1

The length of Subsequence is: 7

**CODE: Sum of Subset**

#include<stdio.h>

void subset(int n,int w[],int d);

int main()

{

int n;

int w[10];

int d;

int i;

printf("Enter number of elements in array\n");

scanf("%d",&n);

printf("Enter array elements in ascending order\n");

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

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

printf("Enter max subset value\n");

scanf("%d",&d);

subset(n,w,d);

}

void subset(int n,int w[],int d)

{

int i;

int s=0;

int x[10]; // tracking of values in array w

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

x[i]=0;

int k=1;

x[k]=1;

while(1)

{

if(k<=n && x[k]==1)

{

if(s+w[k]==d) // soln found

{

printf("Solution is\n");

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

{

if(x[i]==1)

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

}

printf("\n");

x[k]=0;

}

else if(s+w[k]<d)

{

s+=w[k];

}

else

{

x[k]=0;

}

}

else

{

k--;

while(k>0 && x[k]==0)

{

k--;

}

if(k==0)

break;

x[k]=0;

s=s-w[k];

}

k+=1;

x[k]=1;

}

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc SumofSubset.c -o SoS

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./SoS

Enter number of elements in array

5

Enter array elements in ascending order

1 2 4 5 6

Enter max subset value

6

Solution is

1 5

Solution is

2 4

Solution is

6

**CODE: nQueen**

#include<stdio.h>

#include<math.h>

char a[10][10];

int n;

void printmatrix() {

int i, j;

printf("\n");

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

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

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

printf("\n\n");

}

}

int getmarkedcol(int row) {

int i;

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

if (a[row][i] == 'Q') {

return (i);

break;

}

}

int feasible(int row, int col) {

int i, tcol;

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

tcol = getmarkedcol(i);

if (col == tcol || abs(row - i) == abs(col - tcol))

return 0;

}

return 1;

}

void nqueen(int row) {

int i, j;

if (row < n) {

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

if (feasible(row, i)) {

a[row][i] = 'Q';

nqueen(row + 1);

a[row][i] = '.';

}

}

} else {

printf("\nThe solution is:- ");

printmatrix();

}

}

int main() {

int i, j;

printf("\nEnter the no. of queens:- ");

scanf("%d", &n);

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

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

a[i][j] = '.';

nqueen(0);

return (0);

}

**OUTPUT:**

student@student-HP-ProOne-400-G1-AiO:~$ cd Desktop

student@student-HP-ProOne-400-G1-AiO:~/Desktop$ cd Vatsal

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ gcc 8Queen.c -o 8Queen

student@student-HP-ProOne-400-G1-AiO:~/Desktop/Vatsal$ ./8Queen

Enter the no. of queens:- 8

The solution is:-

. . . . . . . Q

. . . Q . . . .

Q . . . . . . .

. . Q . . . . .

. . . . . Q . .

. Q . . . . . .

. . . . . . Q .

. . . . Q . . .