Assignment No:1

#include<stdio.h>

int a[10],n,max,min;

int main()

{

int i;

printf("\n enter the list size");

scanf("%d",&n);

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

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

max=a[0];

min=a[0];

minmax(0,n-1);

printf("\nmax element=%d",max);

printf("\nmin element=%d",min);

return 0;

}

int minmax(int i,int j)

{

int min1,max1,mid;

if(i==j)

{

max=a[i];

min=a[i];

}

else if(i==j-1)

{

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

{

min=a[i];

max=a[j];

}

else

{

min=a[j];

max=a[i];

}

}

else

{

mid=(i+j)/2;

minmax(i,mid);

max1=max;

min1=min;

minmax(mid+1,j);

if(max<max1)

{

max=max1;

}

if(min>min1)

{

min=min1;

}

}

return 0;

}

----------output--------------

it@it-Veriton-Series:~$ gcc minmax.c

it@it-Veriton-Series:~$ ./a.out

enter the list size 5

22

7

23

8

9

max element=23

min element=7it@it-Veriton-Series:~$

------------end---------------

Assignment NO:2

#include<stdio.h>

void optimal(int L[], int n);

int L[10],n,R[10];

int p=0, sum=0;

double MT;

int main()

{

int i,j,temp;

printf("\n enter the number of Objects\n");

scanf("%d",&n);

printf("enter the length of program of n objects\n");

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

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

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

{

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

{

if(L[j]>L[j+1])

{

temp=L[j];

L[j]=L[j+1];

L[j+1]=temp;

}

}

}

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

{

p=p+L[i];

sum=sum+p;

R[i]=sum;

}

MT=(float)R[i-1]/n;

printf("Mean retrieveal time= %f ",MT);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

it@it-Veriton-Series:~$ gcc optimal.c

it@it-Veriton-Series:~$ ./a.out

enter the number of Objects

5

enter the length of program of n objects

2

4

6

7

9

Mean retrieveal time= 13.400000 it@it-Veriton-Series:~$ gcc optimal.c

it@it-Veriton-Series:~$ ./a.out

enter the number of Objects

3

enter the length of program of n objects

1 4 6 8 9

Mean retrieveal time= 5.666667 it@it-Veriton-Series:~$

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Assignment NO:3

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <limits.h>

struct Edge

{

int source, destination, weight;

};

// a structure to represent a connected, directed and weighted graph

struct Graph

{

int V, E;

struct Edge\* edge;

};

struct Graph\* createGraph(int V, int E)

{

struct Graph\* graph = (struct Graph\*) malloc( sizeof(struct Graph));

graph->V = V;

graph->E = E;

graph->edge = (struct Edge\*) malloc( graph->E \* sizeof( struct Edge ) );

return graph;

}

void FinalSolution(int dist[], int n)

{

printf("\nVertex\tDistance from Source Vertex\n");

int i;

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

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

}

}

void BellmanFord(struct Graph\* graph, int source)

{

int V = graph->V;

int E = graph->E;

int StoreDistance[V];

int i,j;

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

StoreDistance[i] = INT\_MAX;

StoreDistance[source] = 0;

for (i = 1; i <= V-1; i++)

{

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

{

int u = graph->edge[j].source;

int v = graph->edge[j].destination;

int weight = graph->edge[j].weight;

if (StoreDistance[u] + weight < StoreDistance[v])

StoreDistance[v] = StoreDistance[u] + weight;

}

}

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

{

int u = graph->edge[i].source;

int v = graph->edge[i].destination;

int weight = graph->edge[i].weight;

if (StoreDistance[u] + weight < StoreDistance[v])

printf("This graph contains negative edge cycle....so shortest path is not found.\n");

}

FinalSolution(StoreDistance, V);

return;

}

int main()

{

int V,E,S;

printf("Enter number of vertices in graph\n");

scanf("%d",&V);

printf("Enter number of edges in graph\n");

scanf("%d",&E);

printf("Enter your source vertex number\n");

scanf("%d",&S);

struct Graph\* graph = createGraph(V, E);

int i;

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

printf("\nEnter edge %d properties Source, destination, weight respectively\n",i+1);

scanf("%d",&graph->edge[i].source);

scanf("%d",&graph->edge[i].destination);

scanf("%d",&graph->edge[i].weight);

}

BellmanFord(graph, S);

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

it@it-Veriton-Series:~$ gcc bellman.c

it@it-Veriton-Series:~$ ./a.out

Enter number of vertices in graph

5

Enter number of edges in graph

8

Enter your source vertex number

0

Enter edge 1 properties Source, destination, weight respectively

0

1

-1

Enter edge 2 properties Source, destination, weight respectively

0

2

4

Enter edge 3 properties Source, destination, weight respectively

1

3

2

Enter edge 4 properties Source, destination, weight respectively

3

1

1

Enter edge 5 properties Source, destination, weight respectively

3

2

5

Enter edge 6 properties Source, destination, weight respectively

1

2

3

Enter edge 7 properties Source, destination, weight respectively

1

4

2

Enter edge 8 properties Source, destination, weight respectively

4

3

3

Vertex Distance from Source Vertex

0 0

1 -1

2 2

3 1

4 1

it@it-Veriton-Series:~$

\*\*negative edge cycle:

it@it-Veriton-Series:~$ gcc bellman.c

it@it-Veriton-Series:~$ ./a.out

Enter number of vertices in graph

3

Enter number of edges in graph

3

Enter your source vertex number

0

Enter edge 1 properties Source, destination, weight respectively

0

1

-8

Enter edge 2 properties Source, destination, weight respectively

1

2

2

Enter edge 3 properties Source, destination, weight respectively

2

0

4

This graph contains negative edge cycle....so shortest path is not found.

Vertex Distance from Source Vertex

0 -4

1 -10

2 -8

it@it-Veriton-Series:~$

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Assignment No:4

#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);

}

int main()

{

get();

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

mincost(0);

put();

return(0);

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*output\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

it@it-Veriton-Series:~$ gcc dynamic.c

it@it-Veriton-Series:~$ ./a.out

Enter No. of Cities: 3

Enter Cost Matrix

Enter Elements of Row # : 1

2

3

4

Enter Elements of Row # : 2

3

4

5

Enter Elements of Row # : 3

4

5

6

The cost list is:

2 3 4

3 4 5

4 5 6

The Path is:

1 -->3 -->2 -->1

Minimum cost:12it@it-Veriton-Series:~$

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*end\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Assignment NO:5

#include<stdio.h>

#include<math.h>

int a[100];

int solution=1;

void display(int n)

{

int i,j;

printf("\n\nSolution:%d \n",solution);

solution++;

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

{

printf("\t%d",i);

}//for i

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

{

printf("\n\n %d",i);

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

{

if(a[i] == j)

printf("\tQ");

else

printf("\t\*");

}//for j

}//for i

}//display

int place(int row,int col)

{

int i;

for(i=1;i<=row-1;i++)

{

if(a[i] == col)

return 0;

else if(abs(a[i]-col) == abs(i-row))

return 0;

}//for i

return 1;

}//place

void queen(int row,int n)

{

int col;

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

{

if(place(row,col))

{

a[row] = col;

if(row == n)

display(n);

else

queen(row+1,n);

}//if

}//for i

}//queen

int main()

{

int n;

printf("\nEnter the value of sqaure matrix:");

scanf("%d",&n);

queen(1,n);

printf("\n\n");

}//main

OUTPUT:

it@it-Veriton-Series:~$ gedit RR.c

it@it-Veriton-Series:~$ gcc RR.c

it@it-Veriton-Series:~$ ./a.out

Enter the value of sqaure matrix:4

Solution:1

1 2 3 4

1 \* Q \* \*

2 \* \* \* Q

3 Q \* \* \*

4 \* \* Q \*

Solution:2

1 2 3 4

1 \* \* Q \*

2 Q \* \* \*

3 \* \* \* Q

4 \* Q \* \*

it@it-Veriton-Series:~$

Assignment NO:6

TRAVELLING SALESMAN PROBLEM(TSP)

#include<stdio.h>

#include<stdlib.h>

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

void get()

{

int i,j;

printf("\t TRAVELLING SALESMAN PROBLEM SOLUTION IN C\n");

printf("Enter Number of Cities: ");

scanf("%d",&n);

printf("\nEnter Cost Matrix:");

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

{

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

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

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

visited[i]=0;

}

printf("\nThe Cost Matrix is:\n");

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

{

printf("\n");

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

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

}

}//get

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);

}//mincost

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;

}//least

void put()

{

printf("\nMinimum cost:");

printf("%d",cost);

}//put

int main()

{

get();

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

mincost(0);

put();

printf("\n");

return 0;

}//main

OUTPUT:

it@it-Veriton-Series:~$ gedit tsp.c

it@it-Veriton-Series:~$ gcc tsp.c

[it@it-Veriton-Series:~$./a.out](mailto:it@it-Veriton-Series:~$./a.out)

TRAVELING SELSMAN PROBLEM

Enter Number of Cities:4

Enter Cost matrix:

Enter Element of row #:1

999

2

4

7

Enter Element of row#:2

2

999

8

3

Enter Element of row#:3

4

8

999

1

Enter Element of row#:4

7

3

5

999

The cost matrix is

999 2 4 7

2 999 8 3

4 8 999 1

7 3 1 999

The path is:1==> 2==> 4==> 3==>1

minimum cost:10