|  |
| --- |
| **Subject: DAA Class: S.E.(Comp)**    **Practical No.: 3 Date:** |

**AIM: Implement of Dynamic programming by traveling sales person problem.**

**Title:** Write algorithm and program for Dynamic programming by traveling sales person problem.

**Theory:**

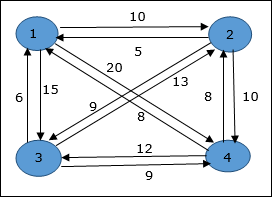
**Dynamic Programming:**

Dynamic Programming is mainly an optimization over plain [recursion](https://www.geeksforgeeks.org/recursion/). Wherever we see a recursive solution that has repeated calls for same inputs, we can optimize it using Dynamic Programming. The idea is to simply store the results of subproblems, so that we do not have to re-comupute them when needed later. This simple optimization reduces time complexities from exponential to polynomial.

**Travelling Salesman Problem:**

Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

In the following example, we will illustrate the steps to solve the travelling salesman problem.



From the above graph, the following table is prepared.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 |
| 1 | 0 | 10 | 15 | 20 |
| 2 | 5 | 0 | 9 | 10 |
| 3 | 6 | 13 | 0 | 12 |
| 4 | 8 | 8 | 9 | 0 |

S = Φ

Cost(2,Φ,1)=d(2,1)=5Cost(2,Φ,1)=d(2,1)=5

Cost(3,Φ,1)=d(3,1)=6Cost(3,Φ,1)=d(3,1)=6

Cost(4,Φ,1)=d(4,1)=8Cost(4,Φ,1)=d(4,1)=8

S = 1

Cost(i,s)=min{Cost(j,s–(j))+d[i,j]}Cost(i,s)=min{Cost(j,s–(j))+d[i,j]}

Cost(2,{3},1)=d[2,3]+Cost(3,Φ,1)=9+6=15Cost(2,{3},1)=d[2,3]+Cost(3,Φ,1)=9+6=15

Cost(2,{4},1)=d[2,4]+Cost(4,Φ,1)=10+8=18Cost(2,{4},1)=d[2,4]+Cost(4,Φ,1)=10+8=18

Cost(3,{2},1)=d[3,2]+Cost(2,Φ,1)=13+5=18Cost(3,{2},1)=d[3,2]+Cost(2,Φ,1)=13+5=18

Cost(3,{4},1)=d[3,4]+Cost(4,Φ,1)=12+8=20Cost(3,{4},1)=d[3,4]+Cost(4,Φ,1)=12+8=20

Cost(4,{3},1)=d[4,3]+Cost(3,Φ,1)=9+6=15Cost(4,{3},1)=d[4,3]+Cost(3,Φ,1)=9+6=15

Cost(4,{2},1)=d[4,2]+Cost(2,Φ,1)=8+5=13Cost(4,{2},1)=d[4,2]+Cost(2,Φ,1)=8+5=13

S = 2

Cost(2,{3,4},1)={d[2,3]+Cost(3,{4},1)=9+20=29d[2,4]+Cost(4,{3},1)=10+15=25=25Cost(2,{3,4},1)={d[2,3]+Cost(3,{4},1)=9+20=29d[2,4]+Cost(4,{3},1)=10+15=25=25

Cost(3,{2,4},1)={d[3,2]+Cost(2,{4},1)=13+18=31d[3,4]+Cost(4,{2},1)=12+13=25=25Cost(3,{2,4},1)={d[3,2]+Cost(2,{4},1)=13+18=31d[3,4]+Cost(4,{2},1)=12+13=25=25

Cost(4,{2,3},1)={d[4,2]+Cost(2,{3},1)=8+15=23d[4,3]+Cost(3,{2},1)=9+18=27=23Cost(4,{2,3},1)={d[4,2]+Cost(2,{3},1)=8+15=23d[4,3]+Cost(3,{2},1)=9+18=27=23

S = 3

Cost(1,{2,3,4},1)=⎧⎩⎨d[1,2]+Cost(2,{3,4},1)=10+25=35d[1,3]+Cost(3,{2,4},1)=15+25=40d[1,4]+Cost(4,{2,3},1)=20+23=43=35Cost(1,{2,3,4},1)={d[1,2]+Cost(2,{3,4},1)=10+25=35d[1,3]+Cost(3,{2,4},1)=15+25=40d[1,4]+Cost(4,{2,3},1)=20+23=43=35

The minimum cost path is 35.

Start from cost **{1, {2, 3, 4}, 1}**, we get the minimum value for **d [1, 2]**. When **s = 3**, select the path from 1 to 2 (cost is 10) then go backwards. When **s = 2**, we get the minimum value for **d [4, 2]**. Select the path from 2 to 4 (cost is 10) then go backwards.

When **s = 1**, we get the minimum value for **d [4, 2]** but 2 and 4 is already selected. Therefore, we select **d [4, 3]** (two possible values are 15 for d [2, 3] and d [4, 3], but our last node of the path is 4). Select path 4 to 3 (cost is 9), then go to **s = Φ** step. We get the minimum value for **d [3, 1]** (cost is 6).

Values

**Time Complexity:**

There are at the most 2n.n2n.n sub-problems and each one takes linear time to solve. Therefore, the total running time is O(2n.n^2)O(2n.n^2).

**Algorithm:**

If size of S is 2, then S must be {1, i},

C(S, i) = dist(1, i)

Else if size of S is greater than 2.

C(S, i) = min { C(S-{i}, j) + dis(j, i)} where j belongs to S, j != i and j != 1.

**Source Code:**

#include<stdio.h>

#include<conio.h>

int ary[10][10],completed[10],n,cost=0;

void takeInput()

{

int i,j;

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

scanf("%d",&n);

printf("\nEnter the 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",&ary[i][j]);

completed[i]=0;

}

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

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

{

printf("\n");

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

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

}

}

void mincost(int city)

{

int i,ncity;

completed[city]=1;

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

ncity=least(city);

if(ncity==999)

{

ncity=0;

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

cost+=ary[city][ncity];

return;

}

mincost(ncity);

}

int least(int c)

{

int i,nc=999;

int min=999,kmin;

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

{

if((ary[c][i]!=0)&&(completed[i]==0))

if(ary[c][i]+ary[i][c] < min)

{

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

kmin=ary[c][i];

nc=i;

}

}

if(min!=999)

cost+=kmin;

return nc;

}

int main()

{

takeInput();

clrscr();

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

mincost(0); //passing 0 because starting vertex

printf("\n\nMinimum cost is %d\n ",cost);

getch();

return 0;

}

**Output:**

