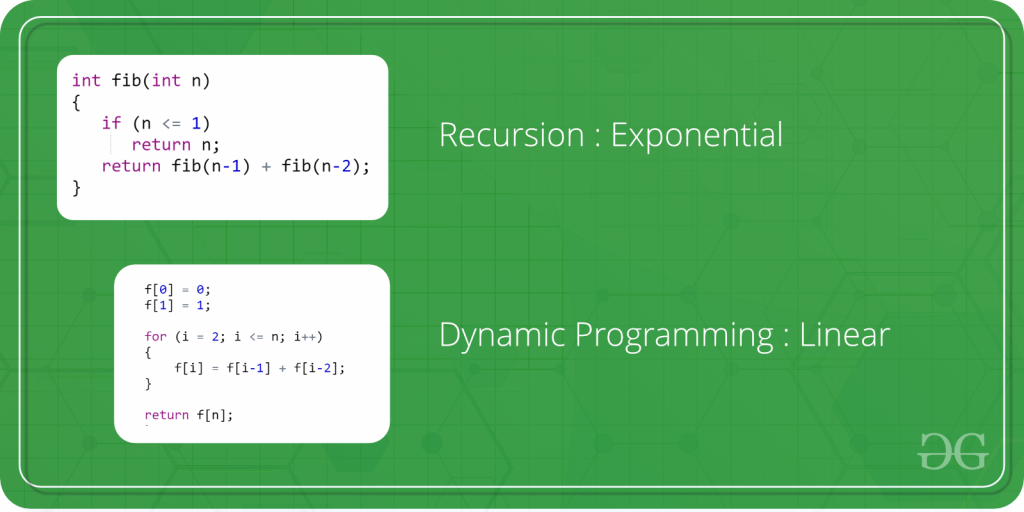
Aim:-Implement Travelling salesman problem using dynamic programming.

# 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. For example, if we write simple recursive solution for [Fibonacci Numbers](https://www.geeksforgeeks.org/program-for-nth-fibonacci-number/), we get exponential time complexity and if we optimize it by storing solutions of subproblems, time complexity reduces to linear.



Program for TSP

#include<stdio.h>

#define V 4

#define START 0

int count;int answer[V];

void copyArray(int oldArray[V])

{

int i=0;

for(;i<V;i++)

answer[i]=oldArray[i];

}

int g(int graph[][V],int source,int done[V],int path[V])

{

int j=0,flag=0;

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

if(done[j]==0)

{

flag=1;

break;

}

if(flag==0)

{

done[source]=0;path[count]=source;

return graph[source][START];

}

int a1=999;int i=0;

for(;i<V;i++)

{

int temp=0;

if(done[i]==0)

{

done[i]=1;path[count++]=i;

temp=graph[source][i]+g(graph,i,done,path);

if(a1>temp)

{

copyArray(path);

a1=temp;

}

done[i]=0;

count--;

}

}

//printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*i'm printing a1 :%d\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n",a1);

return a1;

}

int main()

{

int graph[][V] = { { 0, 10, 15, 20 },

{ 10, 0, 35, 25 },

{ 15, 35, 0, 30 },

{ 20, 25, 30, 0 } };

printf("graph is\n");

int k=0,m=0;

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

{

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

printf("%d\t",graph[k][m]);

printf("\n");

}

int sourec=0;count=0;

int done[V]={0,0,0,0};

done[0]=1;int path[V]={0,0,0,0};int answer[V]={0,0,0,0};path[0]=0;count++;

printf("%d\n path is:\n",g(graph,0,done,path));

int i=0;

for(;i<V;i++)

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

return 0;

}

Output:-

graph is

0       10      15      20

10      0       35      25

15      35      0       30

20      25      30      0

Conclusion:-

80

 path is:

0       1       3       2