Module 26: Multistage Graphs and TSP problem

This module 26 focuses on introducing dynamic programming design strategy and applying it to problems like multistage graphs and traveling salesperson problem (TSP). The objectives of this module are

- **To Explain Dynamic Programming**
- **To explain Multistage Graph problem**
- **To explain Traveling Salesman Problem**

Dynamic Programming

Dynamic programming is useful for solving multistage optimization problems, especially sequential decision problems. Richard Bellman is widely considered as the father of dynamic programming. He was an American mathematician. Richard Bellman is also credited with the coining of the word "Dynamic programming". Here, the word "dynamic" refers to some sort of time reference and "programming" is interpreted as planning or tabulation rather than programming that is encountered in computer programs.

Dynamic programming is used in variety of applications. *Dynamic programming* (DP) is used to solve discrete optimization problems such as scheduling, string-editing, packaging, and inventory management [1,2].

Dynamic programming employs the following steps as shown below:

Step 1: The given problem is divided into a number of subproblems as in "divide and conquer" strategy. But in divide and conquer, the subproblems are independent of each other but in dynamic programming case, there are all overlapping subproblems. A recursive formulation is formed of the given problem.

Step 2: The problem, usually solved in the bottom-up manner. To avoid, repeated computation of multiple overlapping subproblems, a table is created. Whenever a subproblem is solved, then its solution is stored in the table so that in future its solutions can be reused. Then the solutions are combined to solve the overall problem.

There are certain rules that govern dynamic programming. One is called Principle of optimality.

Rule 1:

Bellman's principle of optimality states that at a current state, the optimal policy depends only on the current state and independent of the decisions that are taken in the previous stages. This is called the principle of optimality.

In simple words, the optimal solution to the given problem has optimal solution for all the subproblems that are contained in the given problem.

Rule 2

Dynamic programming problems have overlapping subproblems. So, the idea is to solve smaller instances once and records solutions in a table. This is called memoization, a corrupted word of memorization.

Rule 3

Dynamic programming computes in bottom-up fashion. Thus, the solutions of the subproblems are extracted and combined to give solution to the original problem.

Let us discuss the application of dynamic programming to some of the selected problems now.

Travelling Salesperson Problem (TSP)

Travelling salesman problem (TSP) is an interesting problem. It can be stated as follows: Given a set of n cities and distances between the cities in the form a graph. TSP finds a tour that start and terminate in the source city. The restriction is that, every other cities should be visited exactly once and the focus is to find the tour of shortest length.

It can be said as a function, f(i,s) , shortest sub-tour given that we are at city i and still have to visit the cities in s (and return to home city). In other words, we move from city I to city j and focus is that all cities needs to be visited except city j and should be back to city i.

Let g(i, S) be the length of a shortest path starting at vertex i, going through all vertices in S and terminating at vertex 1.

This can be stated formulated as follows:

$$g(1, V-\{1\}) = \min_{2 \le k \le n} \{c_{1k} + g(k, V-\{1, k\})\}$$

This represents the optimal tour. In general, the recursive function is given as follows:

$$g(i, S) = \min_{j \in S} \{c_{ij} + g(j, S-\{j\})\}$$

This concept is illustrated through this numerical example.

Example 1: Apply dynamic programming for the following graph as shown in Fig. 1. and find the optimal TSP tour.

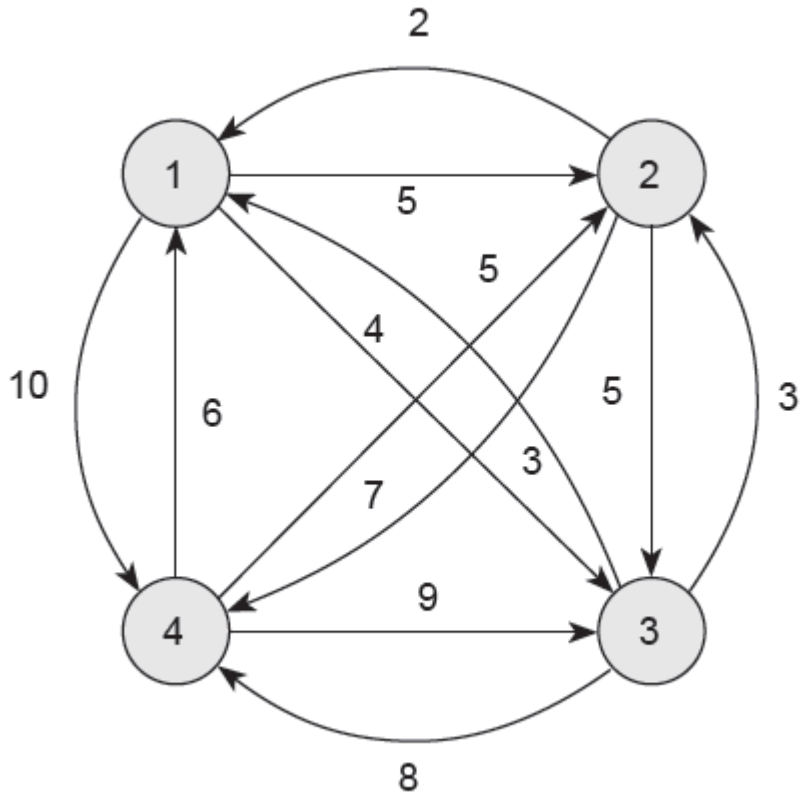

Fig. 1: Sample Graph.

Solution:

Assuming that the starting city 1, let us find the optimal tour.

Let us initialize s = null and find the cost as follows:

$g(2,\varnothing) = C_{21} = 2$

$g(3,\varnothing) = C_{31} = 4$

$g(4,\varnothing) = C_{41} = 6$

Let us consider size =1 , then the possible values are {1}, {2}, {3}.

$$g(2,\{3\}) = C_{23} + g(3,\varnothing)$$
$$= 5 + 4 = 9$$
$$g(2,\{4\}) = C_{24} + g(4,\varnothing)$$
$$= 7 + 6 = 13$$
$$g(3,\{2\}) = C_{32} + g(2,\varnothing)$$
$$= 3 + 2 = 5$$
$$g(3,\{4\}) = C_{34} + g(4,\varnothing)$$
$$= 8 + 6 = 14$$
$$g(4,\{2\}) = C_{42} + g(2,\varnothing)$$
$$= 5 + 2 = 7$$
$$g(4,\{3\}) = C_{43} + g(3,\varnothing)$$
$$= 9 + 4 = 13$$

Let the size s increased by 1, that is, |s| = 2

Now, *g(1,s)* is computed with $|s|=2,\ i \neq 1,\ 1 \notin s,\ i \notin n$, i.e, involves two intermediate nodes.

$$g(2,\{3,4\}) = min\{ C_{23} + g(3,\{4\}) , C_{24} + g(4,\{3\}) \}$$
$$= min\{ 5 + 14 , 7 + 13 \}$$
$$= min\{ 19 , 20 \} = 19.$$
$$g(3,\{2,4\}) = min\{ C_{32} + g(2,\{4\}) , C_{34} + g(4,\{2\}) \}$$
$$= min\{ 3 + 13 , 8 + 7 \}$$
$$= min\{ 16 , 15 \} = 15.$$
$$g(4,\{2,3\}) = min\{ C_{42} + g(2,\{3\}) , C_{43} + g(3,\{2\}) \}$$
$$= min\{ 5 + 9 , 9 + 5 \}$$
$$= min\{ 14 , 14 \} = 14.$$

Let the size is increased by 1, that is, |s| = 3

Finally, the total cost is calculated involving three intermediate nodes. That is $|s|=3$. As $|s|=n-1$, where n is the number of nodes, the process terminates.

$$g(1,\{2,3,4\}) = min\{ C_{12} + g(2,\{3,4\}) , C_{13} + g(3,\{2,4\}) , C_{14} + g(4,\{2,3\}) \}$$
$$= min\{ 5 + 19 , 3 + 15 , 10 + 14 \}$$
$$= min \{24 ,18 , 24 \}$$
$$= 18.$$

Hence, the minimum cost tour is 18. The path can be constructed by noting down the *'k'* that yielded the minimum value. It can be seen that the minimum was possible via route 3.

$\therefore$ *P(1,{2,3,4}) = 3.* Thus tour goes from $1 \rightarrow 3$. It can be seen that *C(3,{2,4})* $\wedge$ minimum.

$\therefore$ *P(3,{2,4}) = 2*

Hence the path goes like $1 \rightarrow 3 \rightarrow 2$ *and the final TSP tour is given as* $1 \rightarrow 3 \rightarrow 2 \rightarrow 1$

Complexity Analysis:

$$n + \sum_{k=2}^{n} (n-1)\binom{n-2}{n-k}(n-k)$$
$$= O(n^2 2^n)$$

Multistage Graphs

The idea for Stagecoach problem is that a salesman is travelling from one town to another town, in the old west. His means of travel is a stagecoach. Each leg of his trip cost a certain amount and he wants to find the minimum cost of his trip, given multiple paths. A sample multistage graph is shown in Fig. 2. And different stage transitions are shown in Fig. 3.

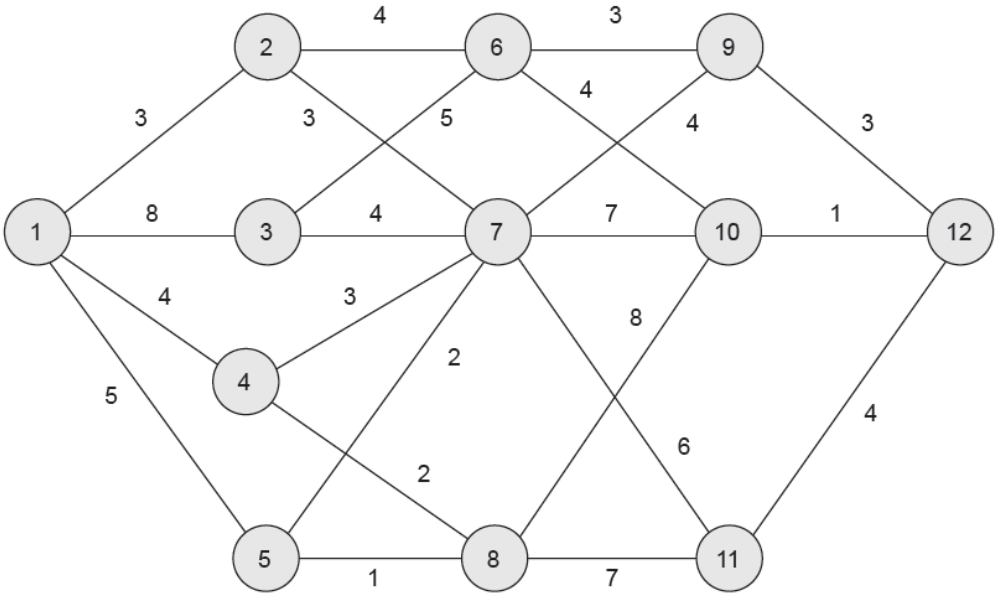

Fig. 2 Sample Multistage Graph

	2	3	4	5
i=1	3	8	4	5

Stage 1-2

	6	7	8
i=2	4	3	2
i=3	5	4	-
i=4	-	3	2
i=5	-	2	1

Stage 2-3

	9	10	11
i=6	3	4	-
i=7	4	7	6
i=8	-	8	7

Stage 3-4

	12
i=9	3
i=10	1
i=11	4

Stage 4-5

Fig. 3: Different Stages of Multistage Graph

Solution:

Stage 4-5: There are three possibilities for going to destination given that one is at points 9, 10 or 11.

cost(4,9) = 3

cost(4,10) = 1

cost(4,11) = 4

$$\cos t(3,6) = \min \begin{cases} 3+\cos t(4,9) = 3+3 = 6 \\ 4+\cos t(4,10) = 4+1 = 5^{*} \end{cases}$$

$$\cos t(3,7) = \min \begin{cases} 4+\cos t(4,9) = 4+3 = 7^{*} \\ 7+\cos t(4,10) = 7+1 = 8 \\ 6+\cos t(4,11) = 6+4 = 10 \end{cases}$$

$$\cos t(3,8) = \min \begin{cases} 8+\cos t(4,10) = 8+1 = 9^{*} \\ 7+\cos t(4,11) = 7+4 = 11 \end{cases}$$

Stage 2-3: The optimal choices at stages 2 can be given as follows:

$$cost(2,2) = \min \begin{cases} 4 + cost(3,6) = 4 + 5 = 9^{*} \\ 3 + cost(3,7) = 3 + 7 = 10 \end{cases}$$

$$cost(2,3) = \min \begin{cases} 5 + cost(3,6) = 5 + 5 = 10^{*} \\ 4 + cost(3,7) = 4 + 7 = 11 \end{cases}$$

$$cost(2,4) = \min \begin{cases} 3 + cost(3,7) = 3 + 7 = 10^{*} \\ 4 + cost(3,8) = 4 + 9 = 13 \end{cases}$$

$$cost(2,5) = \min \begin{cases} 2 + cost(3,7) = 2 + 7 = 9^{*} \\ 2 + cost(3,8) = 2 + 9 = 11 \end{cases}$$

$$cost(1,5) = \min \begin{cases} 3 + cost(2,2) = 3 + 9 = 12^{*} \\ 8 + cost(2,3) = 8 + 10 = 18 \\ 4 + cost(2,4) = 4 + 10 = 14 \\ 5 = cost(2,5) = 5 + 9 = 14 \end{cases}$$

The formal algorithm is given as follows:

Algorithm Fgraph(G)

Begin

cost = 0

n = |V|

stage = n-1

while (j <= stage) do

Choose a vertex k such that C[j,k] + cost k is minimum.

cost[j] = c[j,k]+cost(k)

j = j-1

Add the cost of C(j,r) to record k.

d[j] = k

End while

return cost[j]

end

The path recovery is done as follows:

Algorithm path(G,d,n,k)

Begin

$n = |V|$

stage = n-1

for j = 2 to stage

path[j] = d[path[j-1]]

End for

End.

Complexity Analysis:

Time efficiency: $\Theta(n^3)$

Space efficiency: $\Theta(n^2)$.

Backward Reasoning

A sample graph is shown in Fig. 4. Let us solve this problem using backward reasoning.

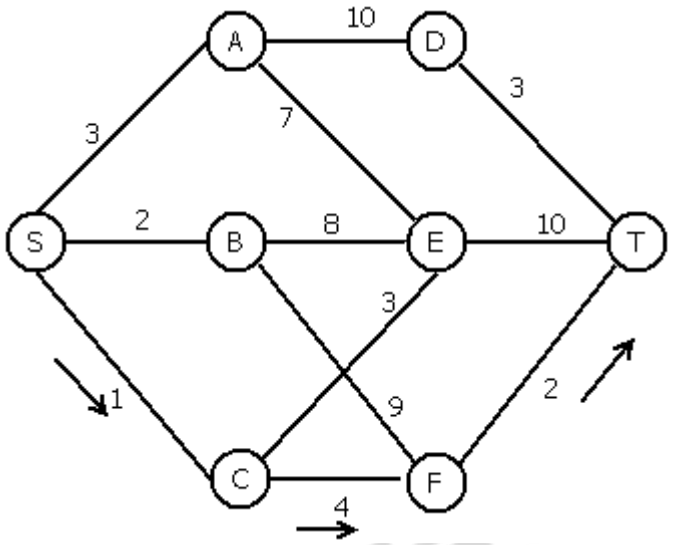

Fig. 4: Sample Graph

Backward approach starts like this

$$C(S, A) = 3$$

$$C(S, B) = 2$$

$$C(S, C) = 1$$

$$C(S, D) = \min\{ 10 + C(S, A) \}$$

$$= \min\{ 10 + 3 \} = 13.$$

$$C(S, E) = \min\{ 7 + C(S, A), 8 + C(S, B), 3 + C(S, C) \}$$

$$= \min\{ 7 + 3, 8 + 2, 3 + 1 \}$$

$$= \min\{ 10, 10, 4 \} = 4$$

$$C(S, F) = \min\{ 9 + Cost(S, B), 4 + Cost(S, C) \}$$

$$= \min\{ 9 + 3, 4 + 1\} = 5$$

$$D(S, T) = \min\{ 3 + C(S, D), 10 + C(S, E), 2 + C(S, F) \}$$

$$= \min\{ 3 + 13, 10 + 4, 2 + 5 \} = 7.$$

The path can be recovered as follows:

$T \rightarrow F \rightarrow C \rightarrow S$

Summary

In short, one can conclude as part of this module 26 that

- Dynamic programming is effective in solving multi-stage graphs
- TSP can be solved effectively using dynamic programming.
- Both forward and backward reasoning can be used find optimal tour.

References:

1. *S.Sridhar , Design and Analysis of Algorithms , Oxford University Press, 2014.*
2. *A.Levitin, Introduction to the Design and Analysis of Algorithms, Pearson Education, New Delhi, 2012.*
3. T.H.Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA 1992.