COST ESTIMATION OF DYNAMIC PROGRAMMING ALGORITHM FOR SOLUTION OF GRAPHICAL AS WELL AS NETWORK PROBLEMS FOR MINIMUM PATH

Anand Kumar Dixit*

Manish Jain**

Adarsh Srivastava***

Ashish Misra****

Abstract:

The cost estimation for any algorithm defines the running time for that algorithm means to say how much time it taken to produce the solution. It defines the performance of algorithm.

The dynamic programming approach is a problem solving technique that solves problems by dividing them into sub problems. Dynamic programming is used when the sub problems are not independent e.g. when the share the same sub problems.

Multistage decision policy with recursive approach will provides a well-organized way while using Dynamic programming. In multistage decision process the problem is divided into several parts called as sub problems and then each sub problem will be solved individually and the final result will be obtained by combining the results of all the sub problems.

With the help of asymptotic notations, calculate the running time complexity of dynamic programming method for solution of graphical as well as network problems for the minimum path between nodes.

Keywords: Algorithm, Asymptotic Notation, Cost Estimation, Complexities for algorithms, Dynamic programming, Optimal Structure, Overlapping Problems, Multistage decision policy.

* Asst. Professor (MCA), Jagran Institute of Management, 620, W Block Saket Nagar, Kanpur, U.P.-208014

** Asst. Professor (MCA), Jagran Institute of Management, 620, W Block Saket Nagar, Kanpur, U.P.-208014

*** Asst. Professor (MCA), Jagran Institute of Management, 620, W Block Saket Nagar, Kanpur, U.P.-208014

**** Asst. Professor (MCA), Jagran Institute of Management, 620, W Block Saket Nagar, Kanpur, U.P.-208014

First we know that what is an algorithm? Algorithm is not a set of instructions that followed, accomplish a work .These are not just simple some steps to solve a problem. If we take an example:

Step 1: Start

Step 2: Read the numbers

Step 3: Stop

From above example we see that we take three numbers and there is no use for these numbers and also no result is produced by these. Hence above steps are aimless. Really "**An algorithm is any well defined computational practice that takes some input, or set of inputs or may be no input, as input and process then produce some value or values as an output.**" It must follow the give below criteria for becoming an algorithm otherwise these are simple steps. The criteria are:

Input: Zero or more quantities must be externally supplied as an input of algorithm for compute the result of that particular problem.

Output: One quantity must be produce as an output.

Finiteness: There should be finite steps in the algorithm. Mean to say the total number of steps may be fixed.

Definiteness: In the algorithm the statements / Instructions must be clear and unambiguous. The unambiguous means steps should clear cut not having confusion for complier that what is done at particular moment? No unambiguousness in steps.

Effectiveness: Each and every statement must be very basic and specifically contribute something in the solution defined by an algorithm.

Example: Algorithm to find the greatest among three numbers

Step 1: Start

Step 2: Read the three number A1, A2, A3

Step 3: Compare A1, A2. If A1 is greater perform step 4 else perform step 5.

Step 4: Compare A1, A3. If A1 is greater, output “A1 is greater” else output “A3 is greater” perform step 6.

Step 5: Compare A2, A3. If A2 is greater, output”A2 is greater” else output “C is greater”.

Step 6: Stop

For any problem we develop the algorithms because there may be different way to solve the problem. A crucial question is “Which one algorithm is better?” The answer for that question is analysis the algorithm. Analysis of algorithm is the quantitative measurement of algorithm performance in terms of times and space requirements. The performance evaluation of an algorithm is gained by totaling the number of occurrences of each operation when the running the algorithm. The performance of algorithm is evaluated as a function of the input size (n) and is to be considered modulo a multiplicative constant.

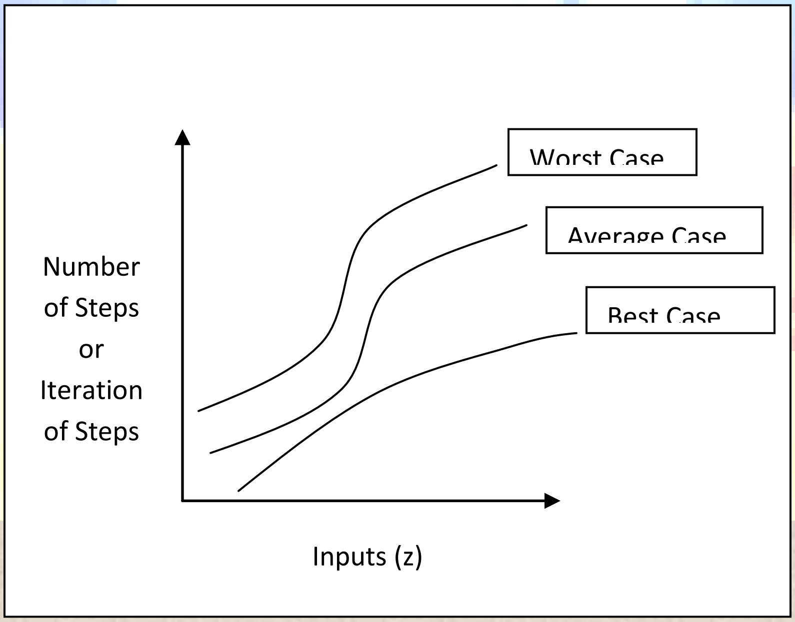


Figure-a

Actually time and space complexity reflect the algorithm`s performance.

Time complexity is defined as running time of the program as a function of size of input.

The space complexity is defined as the amount of computer memory required during the program execution, as a function of input size. The Complexity has three states. These states or cases are: Worst case complexity, Average case complexity, Best case complexity. These are defining as below:

Worst Case Complexity: The worst case complexity of the algorithm is the function defined by the upper limit of steps taken on any instance of size z.

Average Case Complexity: The worst case complexity of the algorithm is the function defined by the average limit of steps taken on any instance of size z.

Best Case Complexity: The worst case complexity of the algorithm is the function defined by the lower limit of steps taken on any instance of size z.

The above graph shows all type of complexity. To represent it in mathematical form we use the concept of asymptotic notations.

Asymptotic Notations notates the asymptotic efficiency. "**The asymptotic efficiency of an algorithm is the order of growth of any algorithm as the input size approaches the limit increases without bound. When the situation arises that the input size is larger enough only the order of growth of the running time is relevant then asymptotic notations are capable to define its complexity**."

There are some notations:

Big – oh – Notations (O),

Omega – Notations (Ω),

Theta – Notations (Θ),

Little- oh- Notations (o),

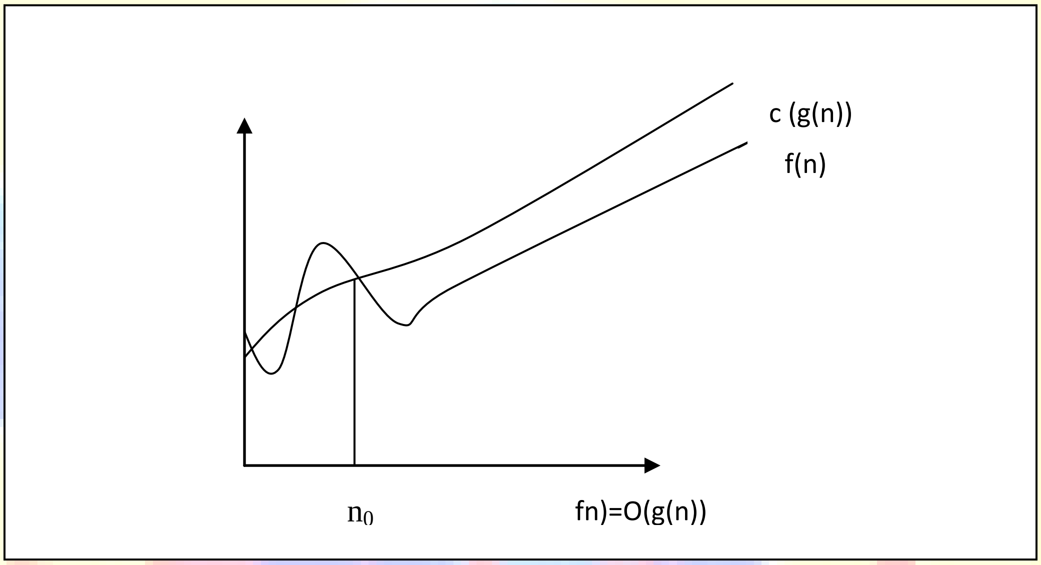
Little –omega – Notations (ω).

Big – oh – Notations (O): When we have only an asymptotic upper bound**,** we use O-notation. For a given function g (n), we denoted by O (g (n)) the set of functions

O(g(n))={ f(n): there exists positive constant c and n_0

such that $0 \le f(n) \le c.g(n)$ for all $n \ge n_0$}

Graphical Illustration:

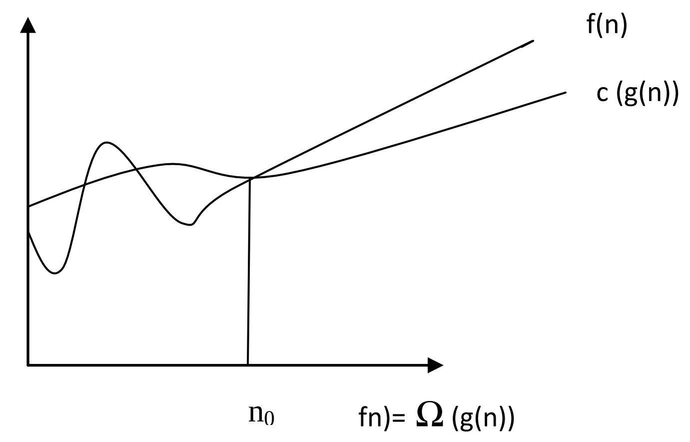


Omega – Notations (Ω): Just as O-notation provides an asymptotic upper bound on a function, Ω -notation provides a asymptotic lower bound. For a given function g (n), we denoted by Ω (g (n)) the set of functions

Ω (g (n)) = {f (n): there exists positive constant c and n_0

such that $0 \le c.g(n) \le f(n)$ for all $n \ge n_0$}

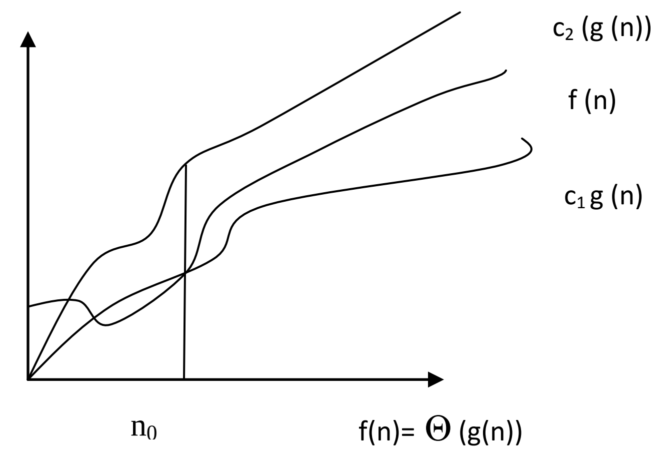
Graphical Illustration:



Theta – Notations (Θ): For a given function g (n), we denoted by Θ (g (n)) the set of functions

Θ (g (n)) = {f (n): there exists positive constant c_1, c_2 and n_0 such that
$0 \leq c_1.g(n) \leq f(n) \leq c_2.g(n)$ for all $n \geq n_0$}

Graphical Illustration:



Little- oh- Notations (o): When we have only an asymptotic upper bound**,** we use O-notation. For a given function g (n), we denoted by O (g (n)) the set of functions

O(g(n))={ f(n): there exists positive constant c >0 and n_0

such that $0 \le f(n) < c.g(n)$ for all $n \ge n_0$}

Below condition is satisfied for little oh notation

Limit f(n) / g(n)= constant

n->∞

Little –omega – Notations (ω): Just as O-notation provides an asymptotic upper bound on a function, Ω -notation provides a asymptotic lower bound. For a given function g (n), we denoted by Ω (g (n)) the set of functions

Ω (g (n)) = {f (n): there exists positive constant c >0 and n_0 such that $0 \le c.g(n) < f(n)$ for all $n \ge n_0$}

Below condition is satisfied for little omega notation

Limit f (n) / g (n) = ∞

n->∞

Now we see the problem for which we want to find the complexity. Let us take an example of a network for which the minimum path between the first to last node would be calculated. But we should also aware with dynamic programming.

Dynamic programming is a useful technique for making a sequence of interrelated decisions. It provides a step wise procedure for finding the optimal combination of decisions. Dynamic programming provides a useful way to find out the minimum distance between the two nodes within the network.

The multistage decision policy with recursive approach will provides an efficient way while using Dynamic programming. In multistage decision process the problem is divided into several parts called as sub problems and then each sub problem will be solved individually and the final result will be obtained by combining the results of all the sub problems.

PROBLEM:

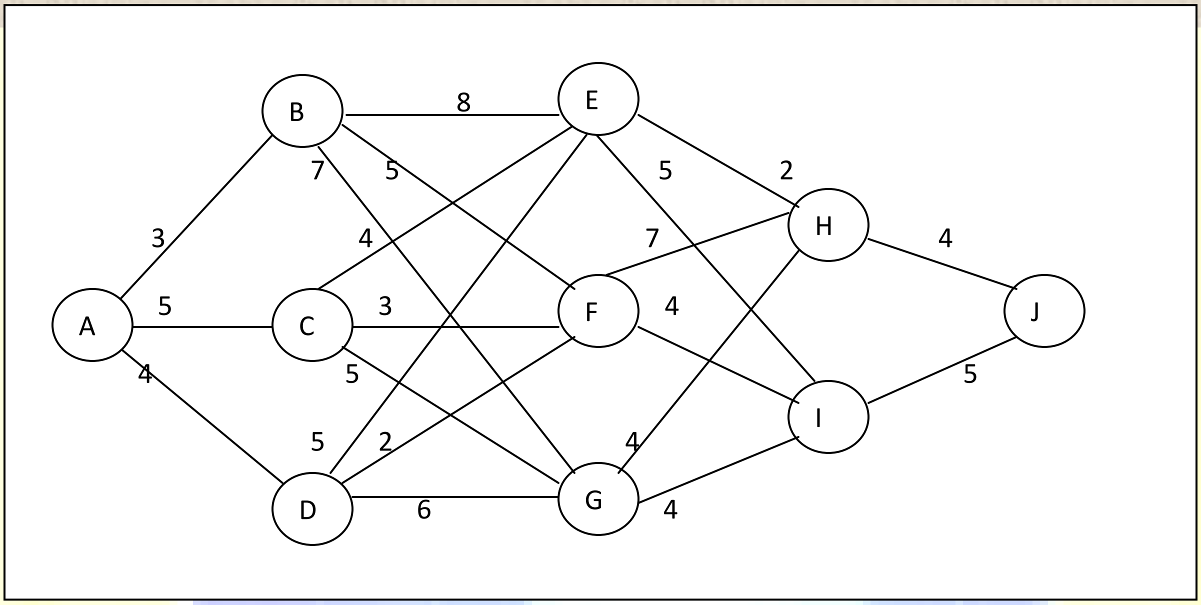


Figure-1 (The network with associated path costs)

The problem in figure-1 shows the road map and the distance between the cities of a transportation problem. If the salesman has to travel from city A to city J, then what should be the best way and the minimum path to reduce the total transportation cost?

While using the Dynamic Programming approach first we have to divide the given network into the multistage problem. Now the problem can be divided into four parts to run from state A to state J.

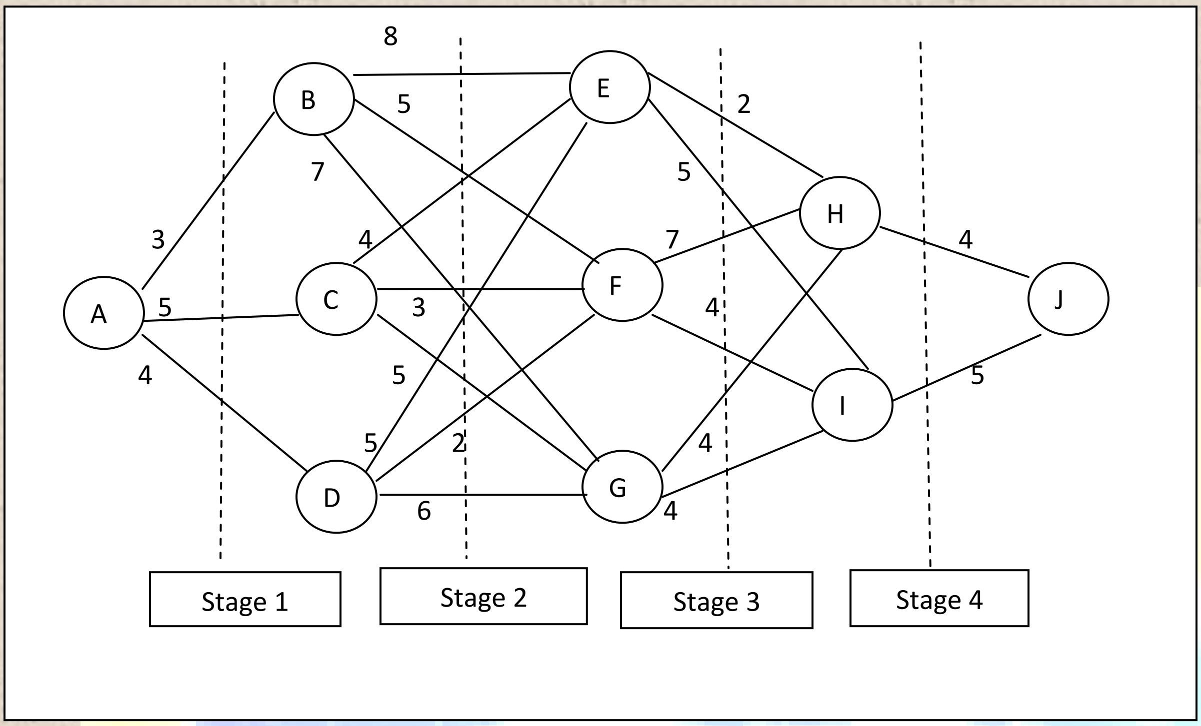


Figure-2 (Multistage Sub division of the Transportation Problem)

The cost associated with each state will be as follows:

	B	C	D
A	3	5	4

	E	F	G
B	8	5	7
C	4	3	5
D	5	2	6

	H	I
E	2	5
F	7	4
G	4	4

	J
H	4
I	5

Let the decision variable x_n(n=1, 2,3,4) be the immediate destination on stage n. Thus the root selected is $A \longrightarrow x_1 \longrightarrow x_2 \longrightarrow x_3 \longrightarrow x_4$, were $x_4 = J$.

Let $f_n(s, x_n)$ be the total cost of the overall transportation for the remaining stages, given that the person is in state s , ready to start stage n, and selects x_n as the immediate destination. Let x_n^* denote any value of x_n that minimizes $f_n(s, x_n)$ and let $f^*_n(s)$ be the corresponding minimum value of $f_n(s, x_n)$.

So $f^*_n(s) = \min f_n(s, x_n) = f_n(s, x^*_n)$.

Where $f_n(s, x_n)$ = Immediate cost(stage n) + Minimum future cost(stage n+1).

Now for n=4 i.e. Fourth stage

N=4

s	$f^*_4(s)$	x^*_4
H	4	J
I	5	J

Now for n=3 i.e. Third stage

$$f_3(s, x_3)= C_{sx3} + f^*_4(x_3)$$

S	H	I	$f^*_3(s)$	x^*_3
E	6	10	6	H
F	11	9	9	I
G	8	9	8	H

Now for n =2 i.e. for Second stage.

$$f_2(s, x_2)= C_{sx2} + f^*_3(x_2)$$

s	E	F	G	$f^*_2(s)$	x^*_2
B	14	14	15	14	E or F
C	10	12	13	10	E
D	11	11	14	11	E or F

Now for n=1 i.e.

$$F_1(s, x_1)= C_{sx1} + f^*_2(x_1)$$

s	B	C	D	$f^*_1(s)$	x^*_1
A	17	15	15	15	C or D

Thus the total minimum cost from A to J is $f^*_1(A) = 15$.

The possible roots are A ⟶ C ⟶ E ⟶ H ⟶ J,

A ⟶ D ⟶ E ⟶ H ⟶ J,

A ⟶ D ⟶ F ⟶ I ⟶ J.

This method will provide a better way to find out all the minimum paths with in a network or any transportation problem. All the routes to reach the destination can be expressed in very précised manner.

To implement this we define the algorithm:

Step 1: First store the graph into computer with the help of cost matrix in stage wise.

Step 2: From n^{th} stage to initial one we find the feasible solution.

Step 3: At last print the possible route with the help of last getting feasible solution that represent also the total minimum cost from starting node to destination node.

Step 4: Stop

Algorithm Analysis:

For first step if we store the data then if layer before stage1 having n_1 nodes and after it before stage2 having node m_1 then the complexity in worst case is

$O(n_{1*} m_1) \approx O(n_{1*} n_1)$ when $m_1= n_1$ or $n_1> m_1$, so $O(n_{1*} n_1) \approx O(p^2)$ so it becomes $O(p^2)$.

Similarly for next stages we get $O(q^2)$, $O(r^2)$, $O(s^2)$, $O(t^2)$ ………… $O(z^2)$.

So total cost for first step $O(n^2)$.

For the second step the cost evaluated when we find the feasible solution: once we add and second when we compare. There are many recursive steps that help to find the solution and generates the complexity but the whole cost can be represented by a quadratic equation then total cost for second step suppose O (m^2).

Similarly for the third steps the cost is also represented by a quadratic equation O (l^2).

Hence the overall cost for above algorithm may be represented as the sum all the inter- mediated cost. So the overall running time is given by

Total Estimated Cost = O (n^2) + O (m^2) + O (l^2)

= O (n^2) where the n is represented to input vertices of graph.

So the Total Estimated Cost of the problem in worst case is O (n^2).

Reference:

- Anand Kumar Dixit, M. Jain, A. Srivastava, S. Ghosh "An initiation of Dynamic Programming to solve the Graphical as well as Network Problems for the minimum path between node." published in "OJCST (International Journal ISSN : 0974-6471) Volume 4 No. 1 225-227 June 2011"
- Andreatta, G. and Romero, L., Stochastic shortest paths with recourse, Networks, 18, 193-204, 1988.
- Daellenbach, H.G. and De Kluyver, C.A., Note on multiple objective dynamic programming, Journal of the Operational Research Society, 31, 591-594, 1980.
- Denardo, E.V. and Fox, B.L., Shortest-route methods: 1. reaching, pruning, and buckets, Operations Research, 27, 161-186, 1979a.
- Denardo, E.V. and Fox, B.L., Shortest-route methods: 2. group knapsacks, expanded networks, and branch- and-bound, Operations Research, 27, 548-566, 1979b.
- Book: Introduction to Algorithms (2009) by Coremen, Leiserson, Rivest, Stein. PHI
- Book: The design and analysis of algorithm by A V Aho.
- Book: Operations Research by Fredric S. Hiller