3. Dynamic Programming

General method:

* Dynamic programming is used to solve hierarchial problems or overlapping problems.
* In Dynamic programming, programming stands for "planning to solve given task".
* It uses the sequence of decisions which are recorded in a "tabular method".
* Summarize or consolidate tabular entries using back tracking method to identify sol^n for a given problem "P".

Applications of Dynamic Programming:

0/1 knapsack:

Procedure:

Step-1: Identify the total no. of items and their corresponding profits and weights.

Step-2: Assume initial profit and weight in the knapsack M as 0 (represent in terms of rows & columns)

Step-3: Maximize the profit in the knapsack by using the formula,

$$P(i,j) = Max(P(i-1, j), P_i + P(i-1, j-w_i))$$

Ste

Step-4: Create a table to record sequence of decisions to identify max. profit (using back tracking method).

Eg: Identify max. profit using 0/1 knap-sack for the given data-

Items : 1 2 3

Profit : 10 20 30 $M = 4$

Weight : 1 2 3

Sol: Step-1: • Create a table with $(M+1)$ no. of columns & $(n+1)$ no. of rows.

- Initially, place zeroes in 0th column & 0th row of the table.
- Arrange items in ascending order with respect to weights & make them reflect on row entries, in the table.
- Calculate every entry in the table using max. profit formula.

	0	1	2	3	4
0	0	0	0	0	0
1	0	10	10	10	10
2	0	10	20	30	30
3	0	10	20	30	40

P_i	w_i
10	1
20	2
30	3

$P(1,1) = \max\left(P(i-1,j), P_i + P(i-1, j-\omega_i)\right)$

$= \max\left(P(0,1), 10+P(0,0)\right)$

$= \max(0, 10+0)$

$= 10$

$P(1,2) = \max(P(0,2), 10+P(0,1))$

$= \max(0, 10+0)$

$= 10$

$P(1,3) = \max(P(0,3), 10+P(0,2))$

$= \max(0, 10+0)$

$= 10$

$P(1,4) = \max(P(0,4), 10+P(0,3))$

$= \max(0, 10+0)$

$= 10$

$P(2,1) = \max(P(1,1), 20+P(1,-1))$

$= \max(10, -)$

$= 10$

$P(2,2) = \max(P(1,2), P_2 + P(1,0))$

$= \max(10, 20+0)$

$= 20$

$P(2,3) = \max(P(1,3), P_2 + P(1,1))$

$= \max(10, 20+10)$

$= 30$

$P(2,4) = MAX(P(1,4), P_2 + P(1,2))$

$= MAX(10, 20+10)$

$= 30$

Similarly, $P(3,1) = 10$

$P(3,2) = 20$

$P(3,3) = 30$

$P(3,4) = 40$

Backtracking:

$P(3,4) = 40 \longrightarrow$ Max. profit

40

$I_3 = 30$ 10 $= I_1$.

$(x_1, x_2, x_3) = (1, 0, 1)$

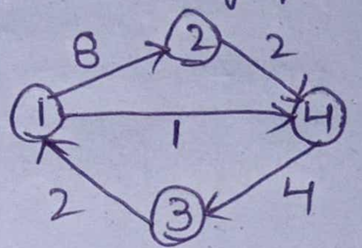
All pair shortest path problem/ Floyd Warshal:

* All pair shortest path problem finds the shortest distance b/w each and every pair in the given graph G. We can use adjacency matrix representation to record all the vertices and their corres- -ponding edge weights or cost.
* We can optimise the cost values of adjacency matrix values by using the formula-

$$A(i,j) = Min(A^{i-1}(i,j),\ A^{i-1}(i,k) + A^{i-1}(k,j))$$

* k is an intermediate vertex which ranges from 1 to n-1 where n is no. of vertices in the given graph G.

Eg: Consider the following graph to identify the shortest distance b/w every pair.



Sol: Step-1: Identifying adjacency matrix for given graph G.

$$\text{Adj Mat } (A^0) = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 8 & \infty & 1 \\ 2 & \infty & 0 & \infty & 2 \\ 3 & 2 & \infty & 0 & \infty \\ 4 & \infty & \infty & 4 & 0 \end{array}$$

All diagonal entries need to be marked zero in adjacency matrix.

$\cancel{A(2,2) = Min(A^{2-1}(2,2), A^{2-1}(}$
$\cancel{k=}$

Step-2: Consider initial vertex = 1

$$k = 2, 3, 4$$

$$A(1,1) = Min(A^0(1,1),\ A^0(1,2) + A^0(2,1))$$

Travelling Salesperson problem:

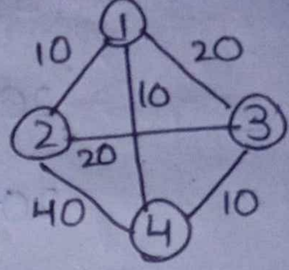
* Travelling salesman problem is an application of dynamic programming approach.
* Travelling salesperson has to start his journey from source or starting city and visit every other city only once and reach back to the source city.
* Identify shortest path to travel all the cities(S) with minimum cost (C).
* Calculate intermediate results of travelling salesperson problem -

  - $g(\underbrace{i}_{\text{source}}, \underbrace{S}_{\text{other vertex}}) = \text{Min}\left[\underbrace{\omega}_{\text{weight}}(i,j) + g(j, \{S-j\})\right];\ j \in S$

  - $g(\underbrace{v}_{\text{vertex}}, \phi) = \omega(v, i)$

* Verify solution of travelling salesperson problem by using travelling salesperson tree (TSP-Tree).

Eg: Calculate shortest path from source to ~~again~~ source by visiting all other vertices only once by considering the following graph.



Sol:

Step-1: Create adjacency matrix.

$$G = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & 20 & 10 \\ 10 & 0 & 20 & 40 \\ 20 & 20 & 0 & 10 \\ 10 & 40 & 10 & 0 \end{bmatrix} \end{array}$$

Step-2:

$g(1, \{2,3,4\})$

$j=2 \longrightarrow w(1,2) + g(2,\{3,4\}) = 10+40 = 50$

$j=3 \longrightarrow w(1,3) + g(3,\{2,4\}) = 20+60 = 80$

$j=4 \longrightarrow w(1,4) + g(4,\{2,3\}) = 10+40 = 50$

Now, $g(2,\{3,4\})$

$j=3 \longrightarrow g(2,3) = w(2,3) + g(3,\{4\}) = 20+20$
$= 40$ ✓

$j=4 \longrightarrow g(2,4) = w(2,4) + g(4,\{3\}) = 40+30$
$= 70$

$g(3,\{4\}) = w(3,4) + g(4,\phi)$

$= 10 + w(4,\underset{\text{source vertex}}{1})$

$= 10+10 = 20.$

$g(3,\{2,4\}) \longrightarrow 60$

$\cancel{j=2 \rightarrow g(3,2)}$

$j=2 \longrightarrow w(3,2)+g(2,\{4\})$

$20+g(2,4)$

$20+w(2,4)+g(2,\phi)$

$20+40+w(2,1)$

$20+40+10$

70

$j=4 \longrightarrow w(3,4)+g(4,\{2\})$

$10+g(4,2)$

$10+w(4,2)+g(4,\phi)$

$10+40+w(4,1)$

$10+40+10$

$60 \checkmark$

$g(4,\{2,3\}) \longrightarrow 40$

$j=2 \longrightarrow w(4,2)+g(2,\{3\})$

$40+g(2,3)$

$40+w(2,3)+g(3,\phi)$

$40+20+w(3,1)$

$40+20+20$

80

$j=3 \longrightarrow w(4,3)+g(3,\{2\})$

$10+g(3,2)$

$10+w(3,2)+g(2,\phi)$

$10+20+\omega(2,1)$

$10+20+10$

40 ✓

$\therefore MIN(50, 80, 50) = 50$

$\therefore$ The path can be -

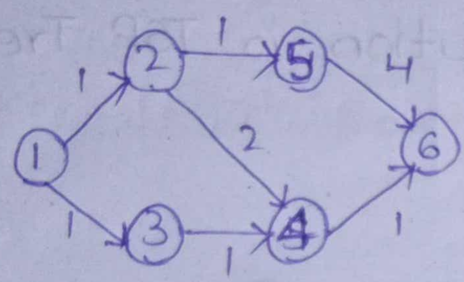
$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$

$1 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$

Multi stage Graph:

* Multi stage Graph is a dynamic programming application which is used to identify the shortest distance with single or multiple paths.

* Multi stage graph overcomes the disadvantage of greedy method's prims algorithm.



1-2-5-6 ⇒ 6 (prims)

1-2-4-6 ⇒ 4
1-3-4-6 ⇒ 4
using multi stage graph

we can find best & optimal sol^n i.e. 2nd or 3rd.

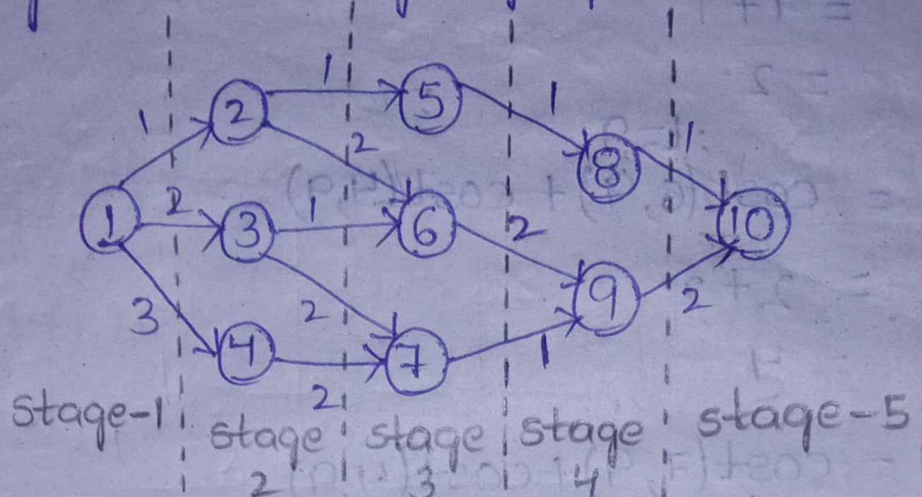
* Multi stage graph cost function can be given as-

$$Cost(i,j) = Min\{cost(j,l) + cost(i+1,l)\}$$

where i = stage, j = vertex, l = set of some vertices

* Record every cost & destination vertex in a tabular format to find optimal cost and it's corresponding path.

Eg: Identify shortest distance or cost using multi stage graph.



Sol:

Step-1: No. of stages = 5

No. of vertices = 10.

Step-2:

~~(4)~~ Stage-5:

vertices = 10.

cost (5, 10) = 0

Step-3:

Stage-4: vertices = 8,9 l = 10

cost(~~8,9~~ 4,8) = cost(8, 10) + cost(5, 10)

= 1+0 ↳ Edge cost

= 1

cost(4,9) = cost(9,10) + cost(5,10)

= 2+0

= 2

Min{cost(4,8), cost(4,9)} = 1

Step-4:

Stage-3: vertices = 5,6,7

$$l=8$$
$$cost(3,5) = cost(5,8) + cost(4,8)$$
$$= 1+1$$
$$= 2$$

$$l=9$$
$$cost(3,6) = cost(6,9) + cost(4,9)$$
$$= 2+2$$
$$= 4$$

$$cost(3,7) = cost(7,9) + cost(4,9)$$
$$= 1+2$$
$$= 3$$

$$Min\{cost(3,5), cost(3,6), cost(3,7)\} = 2.$$

Step-5:

Stage-2: vertices = 2, 3, 4

$$l=5,6$$
$$cost(2,2) = cost(2,5) + cost(3,5)$$
$$= 1+2 = 3$$
$$cost(2,6) + cost(3,6)$$
$$= 2+4 = 5$$
Min = 3

$$l=6,7$$
$$cost(2,3) = cost(3,6) + cost(3,6)$$
$$= 1+4 = 5$$
$$cost(3,7) + cost(3,7)$$
$$= 2+3 = 5$$
Min = 5.

$$l=7$$
$$cost(2,4) = cost(4,7) + cost(3,7)$$
$$= 2+3 = 5$$

$Min\{cost(2,2), cost(2,3), cost(2,4)\} = 3$

So, vertex = 2

Step-6:

Stage-1: vertices = 1

$l = 2,3,4$

$$\left.\begin{aligned} cost(1,1) &= cost(1,2) + cost(2,2) \\ &= 1+3 = 4 \\ &\quad cost(1,3) + cost(2,3) \\ &= 2+5 = 7 \\ &\quad cost(1,4) + cost(2,4) \\ &= 3+5 = 8 \end{aligned}\right\} Min = 4$$

Multi stage graph optimal sol^n table:

Vertex	Cost	Destination
1	4	2
2	3	5
3	5	6
4	5	7
5	2	8
6	4	9
7	3	9
8	1	10
9	2	10
10	0	10

$1 \xrightarrow[1]{} 2 \xrightarrow[1]{} 5 \xrightarrow[1]{} 8 \xrightarrow[1]{} 10$

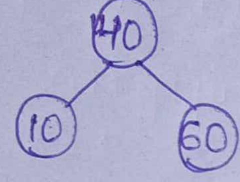
$\therefore$ Total cost = 4

Unit-3

Optimal Binary Search Tree (Using Dynamic Programming):

Consider the following list of elements to construct binary search tree 40, 60, 10. Identify the no. of nodes, successful comparision values & unsuccessful comparision values.

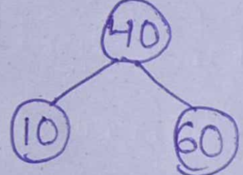
Feasible sol^n-1: 40, 60, 10



$(40 \times 1) + (10 \times 2) + (60 \times 2) = 180$

$p = 3,\ q = 4$

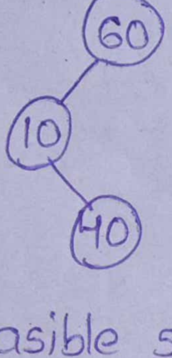
Feasible sol^n-2: 40, 10, 60.



$(40 \times 1) + (10 \times 2) + (60 \times 2) = 180$

$p = 3,\ q = 4$

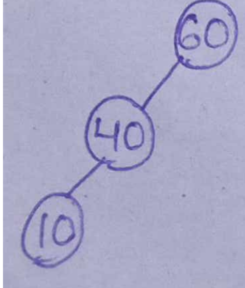
Feasible sol^n-3: 60, 10, 40



$(60 \times 1) + (10 \times 2) + (40 \times 3) = 200$

$p = 3,\ q = 4$

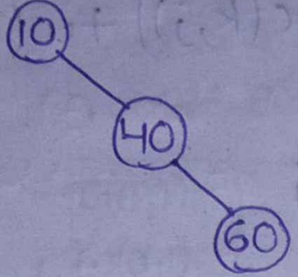
Feasible sol^n-4: 60, 40, 10



$(60 \times 1) + (40 \times 2) + (10 \times 3) = 170$

$p = 3,\ q = 4$

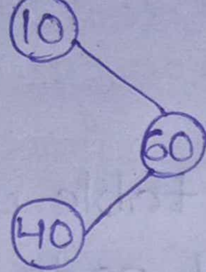
Feasible sol^n - 5: 10, 40, 60



$(10\times1)+(40\times2)+(60\times3)=210$

$p=3;\ q=4$

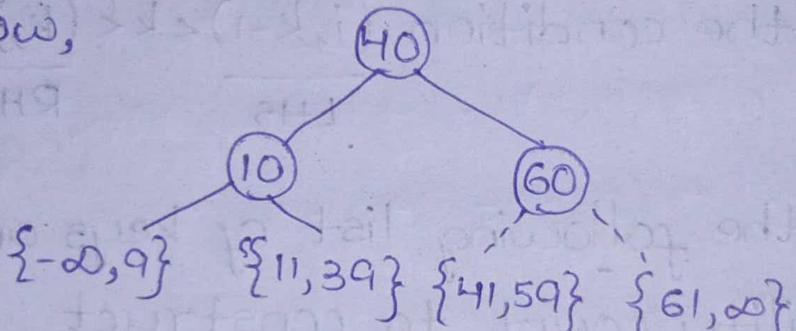
Feasible sol^n - 6: 10, 60, 40



$(10\times1)+(60\times2)+(40\times3)=250$

$p=3,\ q=4.$

Now,



p = successful probability = 3 nodes = 3 (40, 10, 60)

q = unsuccessful probability = 4 nodes = 4

$\therefore \boxed{q>p}$

Procedure:

* Identify given no. of keys, successful probability & unsuccessful probability.
* Construct a table using no. of key count which satisfies the condition $j-i=\{0,1,2,\ldots n\}$
* Fill all the table entries with the help

of cost, weight and rank.

$$c(i,j) = \left(\min_{i<k\leq j} \left(c(i,k-1) + c(k,j) \right) + \omega(i,j) \right)$$

$$\omega(i,j) = \omega(i,j-1) + p_j + q_j$$

$$c(i,i) = 0$$

$$\omega(i,i) = q_i$$

$$r(i,i) = 0$$

$$r(i,j) = k$$

* Identify last entry in the table which is $r(0,n)$ & expand the root node by satisfying the condition $\underbrace{(i,k-1)}_{LHS} < k < \underbrace{(k,j)}_{RHS}$.

Eg: Consider the following list of keys and their probability count to construct optimal binary search tree.

keys = {40, 60, 10}, $p = \{3, 2, 1\}$, $q = \{2, 3, 1, 2\}$

$p_1 \; p_2 \; p_3$ $\quad$ $q_0 \; q_1 \; q_2 \; q_3$

$j-i=0$	$c(0,0)=0$ $w(0,0)=2$ $r(0,0)=0$	$c(1,1)=0$ $w(1,1)=3$ $r(1,1)=0$	$c(2,2)=0$ $w(2,2)=1$ $r(2,2)=0$	$c(3,3)=0$ $w(3,3)=2$ $r(3,3)=0$
$j-i=1$	$c(0,1)=8$ $w(0,1)=8$ $r(0,1)=1$ $k=1$	$c(1,2)=6$ $w(1,2)=6$ $r(1,2)=2$ $k=2$	$c(2,3)=4$ $w(2,3)=4$ $r(2,3)=3$ $k=3$	
$j-i=2$	$c(0,2)=17$ $w(0,2)=11$ $r(0,2)=1$ $k=1,2$	$c(1,3)=13$ $w(1,3)=9$ $r(1,3)=2$ $k=2,3$		
$j-i=3$	$c(0,3)=26$ $w(0,3)=14$ $r(0,3)=2$ $k=1,2,3$			

*$c(0,1)$

$i<k\le j = 0<k\le 1 \Rightarrow k=1$

$c(0,1) = (c(0,0)+c(1,1)) + w(0,1)$

$= 0+0+8$

$= 8$

$r(0,1) = k = 1$

$w(0,1)$

$= w(0,0)+P_1+q_1$

$= 2+3+3 = 8$

---

* $c(1,2)$

$i<k\le j = 1<k\le 2 \Rightarrow k=2$

$c(1,2) = (c(1,1)+c(2,2)) + w(1,2)$

$= (0+0)+6$

$= 6$

$r(1,2) = k = 2.$

$w(1,2)$

$= w(1,1)+P_2+q_2$

$= 3+2+1$

$= 6$

---

* $c(2,3)$

$i < k \le j = 2 < k \le 3 \Rightarrow k = 3$

$c(2,3) = (c(2,2) + c(3,3)) + \omega(2,3)$
$= 0 + 0 + 4$
$= 4$

$\omega(2,3)$
$= \omega(2,2) + P_3 + q_3$
$= 1 + 1 + 2 = 4$

$r(2,3) = k = \cancel{4}3$

---

* $c(0,2)$

$i < k \le j = 0 < k \le 2 \Rightarrow k = 1,2$

~~• k = 1.~~

$\cancel{c(0,2) = (c(0,0) + c(1,2)) + \omega(0,2)}$

$k = 1 \Rightarrow c(0,0) + c(1,2) = 0 + 6 = 6$ ✓ Min.

$k = 2 \Rightarrow c(0,1) + c(2,2) = 8 + 0 = 8$

So, $k = 1$

$c(0,2) = (c(0,0) + c(1,2)) + \omega(0,2)$
$= 0 + 6 + 11$
$= 17$

$\omega(0,2)$
$= \omega(0,1) + P_j + q_j$
$= 8 + 2 + 1 = 11$

$r(0,2) = k = 1$

---

* $c(1,3)$

$i < k \le j = 1 < k \le 3 \Rightarrow k = 2,3$

$k = 2 \Rightarrow c(1,1) + c(2,3) = 0 + 4 = 4$ ✓

$k = 3 \Rightarrow c(1,2) + c(3,3) = 6 + 0 = 6$

So, $k = 2$

$c(1,3) = \left(c(1,1) + c(2,3)\right) + w(1,3)$

$= 0 + 4 + 9$

$= 13$

$w(1,3)$

$= w(1,2) + p_j + q_j$

$= 6 + 2 + 1 = 9$

$r(1,3) = k = 2$

---

* $c(0,3)$

$i < k \leq j = 0 < k \leq 3 \Rightarrow k = 1,2,3$

$k=1 \Rightarrow c(0,0) + c(1,3) = 0 + 13 = 13$

$k=2 \Rightarrow c(0,1) + c(2,3) = 8 + 4 = 12$ ✓

$k=3 \Rightarrow c(0,2) + c(3,3) = 17 + 0 = 17$

So, $k = 2$

$c(0,3) = \left(c(0,1) + c(2,3)\right) + w(0,3)$

$= 8 + 4 + 14$

$= 26$

$w(0,3)$

$= w(0,2) + p_j + q_j$

$= 11 + 2 + 1 = 14$

$r(0,3) = k = 2$