3. Dynamic Programming

(*) General Method: Dynamic Programming is an algorithm design method that can be used when the solution to a problem can be viewed as the result of the sequence of decisions.

Dynamic Programming is used to find optimal solution to the given problem. In this method, solution to a problem is obtained by making sequence of decisions In dynamic programming an optimal sequence of decisions is obtained by using principal of optimality.

Principle of Optimality: the principle of optimality states that " in an optimal sequence of decisions or choices each subsequence must also be optimal"

(*) Differences between Divide and Conquer and Dynamic Programming:

Divide and Conquer:

1. the problem is divided into smaller subproblems. these subproblems are solved independantly. Finally, all the solutions of subproblems are collected together to get the solution to the given problem.
2. In this method duplications in subsolutions are ignored.
3. this method is less efficient.
4. this method uses top down approach of problem solving.
5. this method splits its input at specific points usually in the middle.

Dynamic Programming:

① In this method, many decision sequences are generated and all the overlapping subinstances are considered.

② In this method, duplications are not allowed.

③ This method is efficient than Divide and Conquer

④ In this method, bottom up approach is used.

⑤ Dynamic Programming splits its input at every possible split point. then it determines which split point is optimal.

(*) Differences between Greedy method and Dynamic Programming

Greedy Method:

① Greedy method is used for obtaining optimum solution.

② In Greedy method, from a set of feasible solutions, a solution which satisfies the constraints is optimal solution.

③ In Greedy method, Optimal solution is obtained without revising previously generated solutions.

④ In Greedy method, there is no guarantee of getting optimal sol.

Dynamic Programming:

① Dynamic programming is used for obtaining optimal solution.

② there is no special set of feasible solutions in this method.

③ Dynamic Programming considers all possible sequences in order to obtain the optimal solution.

④ It is guaranteed that the dynamic programming will generate optimal solution using principle of optimality.

⊛ Single Source Shortest Paths: General Weights:

Bellman-Ford algorithm is used to find the shortest path from one source vertex to all other vertices in a graph. Unlike, Dijkstra's algorithm Bellman-ford algorithm is capable of handling negative edges in the graph.

This algorithm initializes the distance to the source to 0 and all other nodes to ∞. then for all edges, if the distance to the destination can be shortened by taking the edge, the distance is updated to the new lower value. At each iteration i that the edges are scanned, the algorithm finds all shortest paths of at most length i edges. Since the longest path without a cycle can be V-1 edges, the edges must be scanned V-1 times to ensure the shortest path has been found for all nodes.

Let $dist^{l}[u]$ be the length of a shortest path from source vertex v to u under the constraint that the shortest path contains at most l edges. Then $dist^{1}[u] = cost[v,u]$, $1 \le u \le n$. Hence $dist^{n-1}[u]$ is the length of an unrestricted shortest path from v to u. Our aim is to find $dist^{n-1}[u]$ for all u. This can be done using the following notations.

(i) If the shortest path from v to u with at most k, k>1, edges has no more than k-1 edges, then $dist^k[u] = dist^{k-1}[u]$.

(ii) If the shortest path from v to u with at most k, k>1, edges has exactly k edges, then it is made up of a shortest path from v to some vertex j following by the edge <j,u>. the path from v to j has k-1 edges, and its length is $dist^{k-1}[j]$. then $dist^{k-1}[j]$ is $dist^{k-1}[i] + cost[i,u]$

$$\therefore dist^k[u] = \min\{dist^{k-1}[u], \min_i\{dist^{k-1}[i] + cost[i,u]\}\}.$$

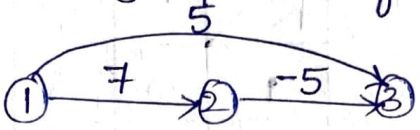
Algorithm: Algorithm BellmanFord (v, cost, dist, n)

```
{
    for i := 1 to n do
        dist [i] := cost [v, i];

    for k = 2 to n-1 do
        for each u such that u ≠ v and u has
            at least one incoming edge do
            for each <i, u> in the graph do
                if dist [u] > dist[i] + cost[i, u] then
                    dist [u] := dist[i] + cost[i, u];

}
```

Example ①: Find the shortest path from node 1 to every other node in the graph using Bellman-Ford algorithm.



Sol: Initially $dist^k[1] = 0 \ \forall \ k \geq 1$

i.e $dist^1[1] = dist^2[1] = 0$

K \ M	1	2	3
1	0	7	5
2	0	7	2

and $dist^1[2] = cost[1,2] = 7$

$dist^1[3] = cost[1,3] = 5$

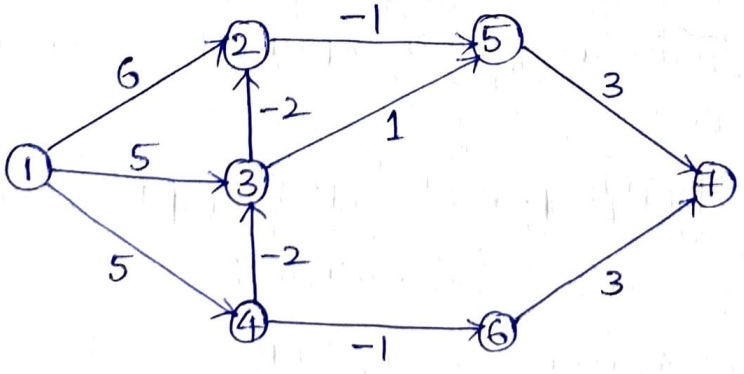
$$dist^2[2] = \min\left\{dist^1[2], \min_{i \neq 3}\{dist^1[i] + cost[i,2]\}\right\}$$

$$= \min\left\{7, \min\{\cancel{\infty+7}, 5+\infty\}\right\} = 7 //$$

$$dist^2[3] = \min\left\{dist^1[3], \min_{i \neq 2}\{dist^1[i] + cost[i,3]\}\right\}$$

$$= \min\left\{5, \min\{\cancel{\infty+\infty}, 7-5\}\right\} = 2 //$$

Example ②: Find the shortest path from 1 to every other node in the graph using Bellman-Ford algorithm



Sol: Initially $dist^k[1] = 0 \ \forall$ values of k

i.e $dist^1[1] = dist^2[1] = dist^3[1] = dist^4[1] = dist^5[1] = dist^6[1] = dist^7[1] = 0$.

For (k=1)

$dist^1[2] = cost(1,2) = 6$

$dist^1[3] = cost(1,3) = 5$

$dist^1[4] = cost(1,4) = 5$

$dist^1[5] = cost(1,5) = \infty$

$dist^1[6] = cost(1,6) = \infty$

$dist^1[7] = cost(1,7) = \infty$

K \ u	1	2	3	4	5	6	7
1	0	6	5	5	∞	∞	∞
2	0	3	3	5	5	4	∞
3	0	1	3	5	2	4	7
4	0	1	3	5	0	4	5
5	0	1	3	5	0	4	3
6	0	1	3	5	0	4	3

For (k=2)

$$dist^2[2] = \min\{dist^1[2], \min_{i=3,\ldots}\{dist^1[i] + cost[i,2]\}\}$$

$$= \min\{6, \min\{\ldots, 5-2, \ldots\}\}$$

$$= 3$$

$$dist^2[3] = \min\{dist^1[3], \min_{i=\ldots,4,\ldots}\{dist^1[i] + cost[i,3]\}\}$$

$$= \min\{5, \min\{\ldots 5-2, \ldots\}\}$$

$$= 3.$$

$$dist^2[4] = \min\{dist^1[4], \min_{i=\ldots}\{dist^1[i] + cost[i,4]\}\}$$

$$= \min\{5, \min\{\ldots\}\}$$

$$= 5$$

$$dist^2[5] = \min\{dist^1[5], \min_{i=2,3,\ldots}\{dist^1[i] + cost[i,5]\}\}$$

$$= \min\{\infty, \min\{6-1, 5+1, \ldots\}\}$$

$$= 5$$

$dist^2[6] = \min\{dist^1[6], \min\{dist^1[i] + cost[i,6]\}\}$

$i = 4, [redacted]$

$= \min\{\infty, \min\{[redacted], 5-1, [redacted]\}\}$

$= 4.$

$dist^2[7] = \min\{dist^1[7], \min\{dist^1[i] + cost[i,7]\}\}$

$i = [redacted] 5,6$

$= \min\{\infty, \min\{[redacted], \infty+3, \infty+3\}\}$

$= \infty.$

For (k=3)

$dist^3[2] = \min\{dist^2[2], \min\{dist^2[i] + cost[i,2]\}\}$

$i = 3, [redacted]$

$= \min\{3, \min\{3+(-2), [redacted]\}\}$

$= 1$

$dist^3[3] = \min\{dist^2[3], \min\{dist^2[i] + cost[i,3]\}\}$

$i = 2, 4[redacted]$

$= \min\{3, \min\{[redacted], 5+(-2), [redacted]\}\}$

$= 3.$

$dist^3[4] = \min\{dist^2[4], \min\{dist^2[i] + cost[i,4]\}\}$

$i = [redacted]$

$= \min\{5, \min\{[redacted]\}\}$

$= 5$

$dist^3[5] = \min\{dist^2[5], \min\{dist^2[i] + cost[i,5]\}\}$

$i = 2, 3, [redacted]$

$= \min\{5, \min\{3+(-1), 3+1, [redacted]\}\}$

$= 2$

$dist^3[6] = \min\{dist^2[6], \min\{dist^2[i] + cost[i,6]\}\}$

$i = [redacted], 4, [redacted]$

$= \min\{4, \min\{[redacted], 5+(-1), [redacted]\}\}$

$= 4$

$dist^3[7] = \min\{dist^2[7], \min\{dist^2[i] + cost[i,7]\}\}$

$i = [redacted], 5, 6$

$= \min\{\infty, \min\{[redacted] 5+3, 4+3\}\}$

$= 7$

For $k=4$

$dist^4[2] = \min\{dist^3[2], \min\{dist^3[i] + cost[i,2]\}\}$

$i = 3, [redacted]$

$= \min\{1, \min\{3+(-2), [redacted]\}\}$

$= 1$

$dist^4[3] = \min\{dist^3[3], \min\{dist^3[i] + cost[i,3]\}\}$

$i = [redacted] 4, [redacted]$

$= \min\{3, \min\{[redacted], 5+(-2), [redacted]\}\}$

$= 3$

$dist^4[4] = \min\{dist^3[4], \min\{dist^3[i] + cost[i,4]\}\}$

$i = [redacted]$

$= \min\{5, \min\{[redacted]\}\}$

$= 5$

$dist^4[5] = \min\{dist^3[5], \min\{dist^3[i] + cost[i,5]\}\}$

$i = 2, 3, [redacted]$

$= \min\{2, \min\{1+(-1), 3+1, [redacted]\}\}$

$= 0$

$dist^4[6] = \min\{dist^3[6], \min\{dist^3[i] + cost[i,6]\}\}$

$i = [redacted], 4, [redacted]$

$= \min\{4, \min\{[redacted], 5+(-1), [redacted]\}\}$

$= 4$

$dist^4[7] = \min\{dist^3[7], \min\{dist^3[i] + cost[i,7]\}\}$

$i = [redacted], 5, 6$

$= \min\{7, \min\{[redacted], \underline{2+3}, 4+3\}\}$

$= 5$

for (K=5)

$dist^5[2] = .\min\{dist^4[2], \min\{dist^4[i] + cost[i,2]\}\}$

$i = 3, [redacted]$

$= \min\{1, \min\{3+(-2), [redacted]\}\}$

$= 1$

$dist^5[3] = \min\{dist^4[3], \min\{dist^4[i] + cost[i,3]\}\}$

$i = [redacted]4[redacted]$

$= \min\{3, \min\{[redacted], 5+(-2), [redacted]\}\} = 3$

$dist^5[4] = \min\{dist^4[4], \min\{dist^4[i] + cost[i,4]\}\}$

$i = [redacted]$

$= \min\{5, \min\{[redacted]\}\} = 5$

$dist^5[5] = \min\{dist^4[5], \min\{dist^4[i] + cost[i,5]\}\}$

$i = 2, 3, [redacted]$

$= \min\{0, \min\{1+(-1), 3+1, [redacted]\}\} = 0$

$dist^5[6] = \min\{dist^4[6], \min\{dist^4[i] + cost[i,6]\}\}$

$i = [redacted], 4[redacted]$

$= \min\{4, \min\{[redacted], 5+(-1), [redacted]\}\} = 4$

$dist^5[7] = \min\{dist^4[7], \min\{dist^4[i] + cost[i,7]\}\}$

$i = [redacted], 5, 6$

$= \min\{5, \min\{[redacted], 0+3, 4+3\}\} = 3$

For (k=6)

$dist^6[2] = \min\{dist^5[2], \min\{dist^5[i] + cost[i,2]\}\}$

$i = 3, [redacted]$

$= \min\{1, \min\{3+(-2), [redacted]\}\} = 1$

$dist^6[3] = \min\{dist^5[3], \min\{dist^5[i] + cost[i,3]\}\}$

$i = [redacted], 4, [redacted]$

$= \min\{3, \min\{[redacted], 5+(-2), [redacted]\}\} = 3$

$dist^6[4] = \min\{dist^5[4], \min\{dist^5[i] + cost[i,4]\}\}$

$i = [redacted]$

$= \min\{5, \min\{[redacted]\}\} = 5$

$dist^6[5] = \min\{dist^5[5], \min\{dist^5[i] + cost[i,5]\}\}$

$i = 2, 3, [redacted]$

$= \min\{0, \min\{1+(-1), 3+1, [redacted]\}\} = 0$

$dist^6[6] = \min\{dist^5[6], \min\{dist^5[i] + cost[i,6]\}\}$

$i = [redacted], 4, [redacted]$

$= \min\{4, \min\{[redacted], 5+(-1), [redacted]\}\} = 4$

$dist^6[7] = \min\{dist^5[7], \min\{dist^5[i] + cost[i,7]\}\}$

$i = [redacted], 5, 6$

$= \min\{3, \min\{[redacted], 0+3, 4+3\}\} = 3$ //

⊛ String Editing:

Consider two strings $X = x_1, x_2 \cdots x_n$ and $Y = y_1, y_2 \cdots y_n$ where x_i, $1 \le i \le n$, and y_j, $1 \le j \le m$, are members of a finite set of symbols known as alphabet. Our aim is to transform X into Y using a sequence of edit operations on X. The permissable edit operations are insert, delete and change, and there is a cost associated with performing each. The cost of sequence of operations is the sum of the costs of the individual operations in the sequence. Here, we have to identify a minimum cost sequence of edit operations that will transform X into Y.

Let $D(x_i)$ be the cost of deleting the symbol x_i from X, $I(y_j)$ be the cost of inserting the symbol y_j into X, and $C(x_i, y_j)$ be the cost of changing the symbol x_i of X into y_j.

A Dynamic Programming solution for this problem can be obtained as follows. Define cost(i,j) to be the minimum cost of any edit sequence for transforming X into y. Compute cost(i,j) for each i and j. Then cost(n,m) is the cost of an optimal edit sequence.

For $i = j = 0$, cost(i,j) = 0, since the two sequences are identical. If $j = 0$, $i > 0$, then transform X into y by a sequence of deletes. Thus,

$$cost(i, 0) = cost(i-1, 0) + D(x_i).$$

Similarly, if $i=0, j>0$ then

$$cost(0,j) = cost(0,j-1) + I(y_j)$$

If $i \neq 0, j \neq 0$, then X can be transformed into Y in one of three ways.

① Transform $x_1, x_2 \cdots x_{i-1}$ into $y_1, y_2 \cdots y_j$ using a minimum cost edit sequence and then delete x_i. The corresponding cost is $cost(i-1, j) + D(x_i)$.

② Transform $x_1, x_2 \cdots x_{i-1}$ into $y_1, y_2 \cdots y_{j-1}$ using a minimum cost edit sequence and then change the symbol x_i to y_j. The associated cost is $cost(i-1, j-1) + C(x_i, y_j)$.

③ Transform $x_1, x_2 \cdots x_i$ into $y_1, y_2 \cdots y_{j-1}$ using a minimum cost edit sequence and then insert y_j. This corresponds to $cost(i, j-1) + I(y_j)$.

The minimum cost of any edit sequence that transforms X into Y is the minimum of the above three costs. Therefore

$$cost(i,j) = \begin{cases} 0 & \text{if } i=j=0 \\ cost(i-1,0) + D(x_i) & \text{if } i>0, j=0 \\ cost(0,j-1) + I(y_j) & \text{if } i=0, j>0 \\ cost'(i,j) & \text{if } i>0, j>0 \end{cases}$$

where $$cost'(i,j) = \min \begin{cases} cost(i-1,j) + D(x_i) \\ cost(i-1,j-1) + C(x_i, y_j) \\ cost(i,j-1) + I(y_j) \end{cases}$$

The value cost(n,m) is the final answer we have to calculate. and a minimum edit sequence can be obtained by a simple backward trace from cost(n,m).

Example①: Let X = a, a, b, a, b and Y = b, a, b, b. Find a minimum cost edit sequence that transforms X into Y.

Sol: First construct a cost table as shown below.

		∅	a	a	b	a	b
j \ i		0	1	2	3	4	5
∅	0	0	1	2	3	4	5
b	1	1	1 → 2	2 → 3	4		
a	2	2	1	1 → 2	2 → 3		
b	3	3	2	2	1 → 2	2	
b	4	4	3	3	2	2	2

↓ insert

delete →

↘ replace

Initially cost(0,0) = 0, cost(0,1) = 1, cost(0,2) = 2, cost(0,3) = 3, cost(0,4) = 4, cost(0,5) = 5.

Similarly cost(1,0) = 1, cost(2,0) = 2, cost(3,0) = 3, cost(4,0) = 4

use the following formula to compute cost(i,j)

$$cost(i,j) = \min\{cost(i-1,j) + D(x_i),\ cost(i-1,j-1) + c(x_i, y_j),\ cost(i,j-1) + I(y_j)\}$$

$$cost(1,1) = \min\{cost(0,1) + D,\ cost(0,0) + c,\ cost(1,0) + I\}$$

$$= \min\{1+1,\ 0+1,\ 1+1\} = 1$$

$cost(2,1) = \min\{cost(1,1)+D, cost(1,0)+I, cost(2,0)+I\}$

$= \min\{1+1, \underline{1+0}, 2+1\} = 2$

$cost(3,1) = \min\{cost(2,1)+D, cost(2,0)+C, cost(3,0)+I\}$

$= \min\{2+1, \underline{2+0}, 3+1\} = 2$

$cost(4,1) = \min\{cost(3,1)+D, cost(3,0)+C, cost(4,0)+I\} = \min\{\underline{2+1}, 3+1, 4+1\}$

$= 3$

$cost(5,1) = \min\{cost(4,1)+D, cost(4,0)+C, cost(5,0)+I\}$

$= \min\{3+1, \underline{4+0}, 5+1\} = 4$

$cost(1,2) = \min\{cost(0,2)+D, cost(0,1)+C, cost(1,1)+I\}$

$= \min\{2+1, \underline{1+0}, 1+1\} = 1$

$cost(2,2) = \min\{cost(1,2)+D, cost(1,1)+C, cost(2,1)+I\}$

$= \min\{1+1, \underline{1+0}, 2+1\} = 1$

$cost(3,2) = \min\{cost(2,2)+D, cost(2,1)+C, cost(3,1)+I\}$

$= \min\{\underline{1+1}, 2+1, 2+1\} = 2$

$cost(4,2) = \min\{cost(3,2)+D, cost(3,1)+C, cost(4,1)+I\}$

$= \min\{2+1, \underline{2+0}, 3+1\} = 2$

$cost(5,2) = \min\{cost(4,2)+D, cost(4,1)+C, cost(5,1)+I\}$

$= \min\{\underline{2+1}, 3+1, 4+1\} = 3$

$cost(1,3) = \min\{cost(0,3)+D, cost(0,2)+C, cost(1,2)+I\}$

$= \min\{3+1, 2+1, \underline{1+1}\} = 2$

$cost(2,3) = \min\{cost(1,3)+D, cost(1,2)+C, cost(2,2)+I\}$

$= \min\{2+1, \underline{1+1}, \underline{1+1}\} = 2$

$cost(3,3) = \min\{cost(2,3)+D, cost(2,2)+C, cost(3,2)+I\}$

$= \min\{2+1, \underline{1+0}, 2+1\} = 1$

$cost(4,3) = \min\{cost(3,3)+D, cost(3,2)+C, cost(4,2)+I\}$

$= \min\{\underline{1+1}, 2+1, 2+1\} = 2$

$cost(5,3) = \min\{cost(4,3)+D, cost(4,2)+C, cost(5,2)+I\}$

$= \min\{2+1, \underline{2+0}, 3+1\} = 2$

$cost(1,4) = \min\{cost(0,4)+D, cost(0,3)+C, cost(1,3)+I\}$

$= \min\{4+1, 3+1, \underline{2+1}\} = 3$

$cost(2,4) = \min\{cost(1,4)+D, cost(1,3)+C, cost(2,3)+I\}$

$= \min\{3+1, \underline{2+1}, \underline{2+1}\} = 3$

$cost(3,4) = \min\{cost(2,4)+D, cost(3,3)+C, cost(3,3)+I\}$

$= \min\{3+1, \underline{2+0}, 1+1\} = 2$

$cost(4,4) = \min\{cost(3,4)+D, cost(3,3)+C, cost(4,3)+I\}$

$= \min\{2+1, \underline{1+1}, 2+1\} = 2$

$cost(5,4) = \min\{cost(4,4)+D, cost(4,3)+C, cost(5,3)+I\}$

$= \min\{2+1, \underline{2+0}, 2+1\} = 2$

Finally $cost(5,4) = 2$. One possible minimum cost edit sequence is replace x_1 with y_1 and delete x_4.

③ 0/1 Knapsack Problem: Consider n no: of objects for $i=1,2\ldots n$ having their corresponding profits P_i and weights w_i. Consider a knapsack or bag with a capacity m. If a fraction x_i, $0 \le x_i \le 1$, of object i is placed into the knapsack, then a profit of $P_i x_i$ is earned. The objective is to obtain a filling of the knapsack that maximizes the total profit earned. Since the knapsack capacity is m, total weight of all chosen objects to be at most m. formally, the problem can be states as

$$\text{maximize} \sum_{1 \le i \le n} P_i x_i \quad \text{subject to} \sum_{1 \le i \le n} w_i x_i \le m$$

$$\text{and} \quad 0 \le x_i \le 1, \quad 1 \le i \le n.$$

A solution to the knapsack can be obtained by making a sequence of decisions on the variables $x_1, x_2 \ldots x_n$. To solve this problem using dynamic programming use the following notations.

① Compute $S^1, S^2 \ldots S^i$. Initially $S^0 = \{(0,0)\}$

② Use the following formulae to compute S^i

$$S^i = S^{i-1} \cup S_1^{i-1}$$

Where $S_1^{i-1} = S^{i-1} + (P_i, w_i)$

③ After computing every S^i value apply purging rule (or) Dominance rule: If S^i contains (P_j, w_j) and (P_k, w_k) such that $P_j \le P_k$ and $w_j \ge w_k$ then eliminate the tuple (P_j, w_j) from S^i.

(*) Ex: Solve the following knapsack instance $m=6, n=3$, $(P_1, P_2, P_3) = (1, 2, 5)$ and $(w_1, w_2, w_3) = (2, 3, 4)$ using dynamic programming.

Sol: Given that $n=3, m=6$, $(P_1, w_1) = (1, 2)$, $(P_2, w_2) = (2, 3)$

$(P_3, w_3) = (5, 4)$.

Compute S^1, S^2, S^3 $\boxed{\because n=3}$

Initially $S^0 = \{(0,0)\}$

$S^i = S^{i-1} \cup S_1^{i-1}$ where $S_1^{i-1} = S^{i-1} + (P_i, w_i)$.

for $i=1$ $S^1 = S^0 \cup S_1^0$

$\therefore S_1^0 = S^0 + (P_1, w_1) = \{(0,0)\} + (1,2) = \{(1,2)\}$

$\Rightarrow S^1 = \{(0,0)\} \cup \{(1,2)\} = \{(0,0), (1,2)\}$

After applying purging rule S^1 is remains unchanged

for $i=2$ $S^2 = S^1 \cup S_1^1$

$\therefore S_1^1 = S^1 + (P_2, w_2) = \{(0,0), (1,2)\} + (2,3)$

$= \{(2,3), (3,5)\}$

$\Rightarrow S^2 = \{(0,0), (1,2)\} \cup \{(2,3), (3,5)\}$

$S^2 = \{(0,0), (1,2), (2,3), (3,5)\}$

After applying purging rule S^2 is uncharged.

for $i=3$ $\quad S^3 = S^2 \cup S_1^2$

$$\therefore S_1^2 = S^2 + (P_3, W_3)$$

$$= \{(0,0), (1,2), (2,3), (3,5)\} + (5,4)$$

$$= \{(5,4), (6,6), (7,7), (8,9)\}$$

$$\Rightarrow S^3 = \{(0,0), (1,2), (2,3), (3,5)\} \cup \{(5,4), (6,6), (7,7), (8,9)\}$$

$$= \{(0,0), (1,2), (2,3), \underbrace{(3,5)}_{P_j, W_j}, (5,4), \underbrace{(6,6)}_{P_k, W_j}, (7,7), (8,9)\}$$

By applying purging rule $\because P_j \leq P_k$ and $W_j \geq W_k$. the tuple $(3,5)$ is eliminated and since capacity of the knapsack is 6 eliminate the tuples having weight greater than 6. Hence

$$S^3 = \{(0,0), (1,2), (2,3), (5,4), (6,6)\}$$

Since $m=6$, search for the tuple whose weight is 6 or closer. Now the selected tuple (P_i, W_i) is $(6,6)$.

if $(P_i, W_i) \in S^i$ and $(P_i, W_i) \notin S^{i-1}$ then $x_i = 1$ else $x_i = 0$.

$(6,6) \in S^3$ and $(6,6) \notin S^2$ then $x_3 = 1$.

Now search for $(P_i - P_3, W_i - W_3) = (6-5, 6-4) = (1,2)$.

$(1,2) \in S^2$ and $(1,2) \notin S^1$ is false then $x_2 = 0$

$(1,2) \in S^1$ and $(1,2) \notin S^0$ is true then $x_1 = 1$

Now search for $(P_i - P_1, W_i - W_1) = (1-1, 2-2) = (0,0)$. Hence optimal solution is $(x_1, x_2, x_3) = (1, 0, 1)$ //

Algorithm for Dynamic 0/1 knapsack:

```
Algorithm DKnap(p, w, x, n, m)
{
  b[1] := 1; pair[1].p = pair[1].w = 0.0;
  t := 1; h := 1;
  b[1] := next := 2;
  for i := 1 to n-1 do
  {
    k := t;
    u := Larges(pair, w, t, h, i, m);
    for j := t to u do
    {
      pp := pair[j].p + p[i]; ww = pair[j].w + w[i];
      while ((k ≤ h) and (pair[k].w ≤ ww)) do
      {
        pair[next].p = pair[k].p;
        pair[next].w = pair[k].w;
        next := next+1; k := k+1;
      }
      if ((k ≤ h) and (pair[k].w = ww)) then
      {
        if pp < pair[k].p then pp := pair[k].p;
        k := k+1;
      }
      if pp > pair[next-1].p then
      {
        pair[next].p := pp; pair[next].w = ww;
        next := next+1;
      }
      while ((k ≤ h) and (pair[k].p ≤ pair[next-1].p)) do
        k := k+1;
    }
```

```
while (k<=h) do
{
  pair[next].p := pair[k].p;
  pair[next].w := pair[k].w;
  next := next+1;
  k := k+1;
}
t := h+1; h := next-1; b(s+1) := next;
}
TraceBack(p, w, pair, x, m, n);
}//
```

* Worst case time complexity is $O(2^{n/2})$.

④ All Pairs Shortest Path Problem: Let $G(V, E)$ be a directed graph with n vertices. Let cost be a cost adjacency matrix for G such that $cost(i,i) = 0$, $1 \le i \le n$. Then $cost(i,j)$ is the length of edge $\langle i,j \rangle$ and $cost(i,j) = \infty$, if there is no edge between i and j. Our aim is to find shortest available paths from every vertex to every other vertex.

The following notations are used to solve this problem using dynamic programming.

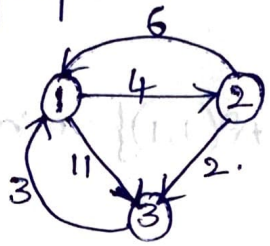
①. Let $A^k(i,j)$ be the length of shortest path from node i to node j. We have to compute A^k for $k = 1, 2 \cdots n$

②. the following formulae is used to compute $A^k(i,j)$.

Initially $A^0(i,j) = cost(i,j)$.

$$A^k(i,j) = \min_{1 \le k \le n} \{ A^{k-1}(i,j), A^{k-1}(i,k) + A^{k-1}(k,j) \} //$$

(*) Ex: Compute All Pairs shortest path problem for the following graph.



Sol: for the given graph cost adjacency matrix is

$$A^0(i,j) = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & \infty & 0 \end{bmatrix}$$

∵ no: of nodes $n=3$ compute A^1, A^2, A^3.

for K=1

$$A^1(i,j) = \begin{bmatrix} A^1(1,1) & A^1(1,2) & A^1(1,3) \\ A^1(2,1) & A^1(2,2) & A^1(2,3) \\ A^1(3,1) & A^1(3,2) & A^1(3,3) \end{bmatrix}$$

∴ for $i=1, j=1$

$$A^1(1,1) = \min\{A^0(1,1), A^0(1,1)+A^0(1,1)\} = \min\{0,0\} = 0.$$

$i=1, j=2$

$$A^1(1,2) = \min\{A^0(1,2), A^0(1,1)+A^0(1,2)\} = \min\{4, 0+4\} = 4$$

$i=1, j=3$

$$A^1(1,3) = \min\{A^0(1,3), A^0(1,1)+A^0(1,3)\} = \min\{11, 0+11\} = 11$$

$i=2, j=1$

$$A^1(2,1) = \min\{A^0(2,1), A^0(2,1)+A^0(1,1)\} = \min\{6, 6+0\} = 6$$

$i=2, j=2$

$$A^1(2,2) = \min\{A^0(2,2), A^0(2,1)+A^0(1,2)\} = \min\{0, 6+4\} = 0$$

$i=2, j=3$

$A^1(2,3) = \min\{A^0(2,3), A^0(2,1)+A^0(1,3)\} = \min\{2, 6+11\} = 2.$

$i=3, j=1$

$A^1(3,1) = \min\{A^0(3,1), A^0(3,1)+A^0(1,1)\} = \min\{3, 3+0\} = 3$

$i=3, j=2.$

$A^1(3,2) = \min\{A^0(3,2), A^0(3,1)+A^0(1,2)\} = \min\{\infty, 3+4\} = 7.$

$i=3, j=3$

$A^1(3,3) = \min\{A^0(3,3), A^0(3,1)+A^0(1,3)\} = \min\{0, 3+11\} = 0.$

$$\therefore A^1(i,j) = \begin{bmatrix} 0 & 4 & 11 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

Now for $\boxed{K=2}$ Compute $A^2 = \begin{bmatrix} A^2(1,1) & A^2(1,2) & A^2(1,3) \\ A^2(2,1) & A^2(2,2) & A^2(2,3) \\ A^2(3,1) & A^2(3,2) & A^2(3,3) \end{bmatrix}$

$i=1, j=1$

$A^2(1,1) = \min\{A^1(1,1), A^1(1,2)+A^1(2,1)\} = \min\{0, 4+6\} = 0.$

$A^2(1,2) = \min\{A^1(1,2), A^1(1,2)+A^1(2,2)\} = \min\{4, 4+0\} = 4$

$A^2(1,3) = \min\{A^1(1,3), A^1(1,2)+A^1(2,3)\} = \min\{11, 4+2\} = 6$

$A^2(2,1) = \min\{A^1(2,1), A^1(2,2)+A^1(2,1)\} = \min\{6, 0+6\} = 6$

$A^2(2,2) = \min\{A^1(2,2), A^1(2,2)+A^1(2,2)\} = \min\{0, 0+0\} = 0$

$A^2(2,3) = \min\{A^1(2,3), A^1(2,2)+A^1(2,3)\} = \min\{2, 0+2\} = 2$

$A^2(3,1) = \min\{A^1(3,1), A^1(3,2)+A^1(2,1)\} = \min\{3, 7+6\} = 3$

$A^2(3,2) = \min\{A^1(3,2), A^1(3,2)+A^1(2,2)\} = \min\{7, 7+0\} = 7$

$A^2(3,3) = \min\{A^1(3,3), A^1(3,2)+A^1(2,3)\} = \min\{0, 7+2\} = 0$ //.

$$\therefore A^2(i,j) = \begin{bmatrix} 0 & 4 & 6 \\ 6 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix}$$

for $\boxed{k=3}$ compute $A^3(i,j) = \begin{bmatrix} A^3(1,1) & A^3(1,2) & A^3(1,3) \\ A^3(2,1) & A^3(2,2) & A^3(2,3) \\ A^3(3,1) & A^3(3,2) & A^3(3,3) \end{bmatrix}$

$i=1, j=1$

$A^3(1,1) = \min\{A^2(1,1), A^2(1,3)+A^2(3,1)\} = \min\{0, 6+3\} = 0.$

$A^3(1,2) = \min\{A^2(1,2), A^2(1,3)+A^2(3,2)\} = \min\{4, 6+7\} = 4.$

$A^3(1,3) = \min\{A^2(1,3), A^2(1,3)+A^2(3,3)\} = \min\{6, 6+0\} = 6$

$A^3(2,1) = \min\{A^2(2,1), A^2(2,3)+A^2(3,1)\} = \min\{6, 2+3\} = 5$

$A^3(2,2) = \min\{A^2(2,2), A^2(2,3)+A^2(3,2)\} = \min\{0, 2+7\} = 0$

$A^3(2,3) = \min\{A^2(2,3), A^2(2,3)+A^2(3,3)\} = \min\{2, 2+0\} = 2.$

$A^3(3,1) = \min\{A^2(3,1), A^2(3,3)+A^2(3,1)\} = \min\{3, 0+3\} = 3.$

$A^3(3,2) = \min\{A^2(3,2), A^2(3,3)+A^2(3,2)\} = \min\{7, 0+7\} = 7$

$A^3(3,3) = \min\{A^2(3,3), A^2(3,3)+A^2(3,3)\} = \min\{0, 0+0\} = 0.$

$$\therefore A^3(i,j) = \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 2 \\ 3 & 7 & 0 \end{bmatrix} //.$$

⊛ Algorithm for All Pairs Shortest Path Problem:

```
Algorithm AllPaths(cost, A, n)
// cost[1:n, 1:n] is the cost adjacency matrix of a graph
// with n vertices; A[i,j] is the cost of a shortest path
// from vertex i to vertex j. cost[i,i] = 0.0 for 1 ≤ i ≤ n.
```

```
{
    for i:=1 to n do
        for j:=1 to n do
            A[i,j]:= cost[i,j];

    for k:=1 to n do
        for i:=1 to n do
            for j:=1 to n do
                A[i,j]:=min (A[i,j], A[i,k]+A[k,j]);
}//
```

⑤ The Travelling Salesperson Problem: Let G(V,E) be a directed graph with V vertices and E edges. The edges are given alongth with their cost C_{ij}. the cost $C_{ij}>0$ for all i and j. and $C_{ij}=\infty$ if there is no edge between i and j.

A Salesperson starts his tour at any vertex and should ends his tour at same vertex. During the tour he should visit all the vertices exactly once. Main objective of travelling salesperson problem is to find the tour of optimum cost. the following notations are used to solve the problem using dynamic programming.

① Let the function $g(1, V-\{1\})$ be total length of the tour terminating at vertex 1.

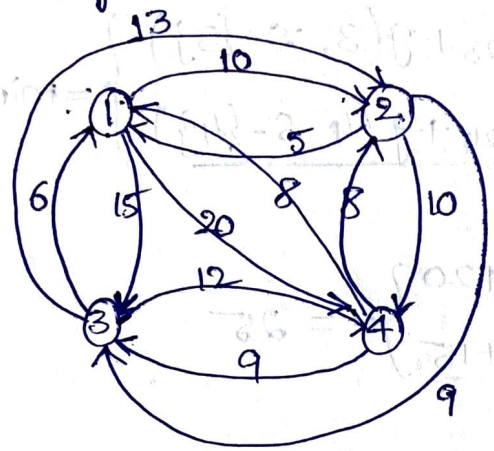
② Let $V_1, V_2 \cdots V_n$ be the sequence of vertices followed in optimal tour.

③ The following formula can be used to obtain tour of optimum cost.

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

$$g(i,\phi) = C_{i1} : \text{Cost of the edge b/w } i \text{ and } 1$$

Ex: Find the shortest tour of a travelling sales person for the following instance using dynamic programming



Sol: Initially construct cost adjacency matrix. i.e

	1	2	3	4
1	∞	10	15	20
2	5	∞	9	10
3	6	13	∞	12
4	8	8	9	∞

Let us assume vertex ① as source vertex. i.e find $g(1, V-\{1\})$ i.e $g(1, \{2,3,4\})$.

$$g(1,\{2,3,4\}) = \min_{\substack{i, S \\ j=2,3,4}} \begin{Bmatrix} C_{12} + g(2, S-\{2\}) \\ C_{13} + g(3, S-\{3\}) \\ C_{14} + g(4, S-\{4\}) \end{Bmatrix}$$

$$= \min \begin{cases} c_{12} + g(2, \{3,4\}) \\ c_{13} + g(3, \{2,4\}) \\ c_{14} + g(4, \{2,3\}) \end{cases}$$

$$= \min \begin{cases} 10 + 25 \\ 15 + 25 \\ 20 + 23 \end{cases} = 35$$

$$g(2, \{3,4\}) = \min \begin{cases} c_{23} + g(3, S-\{3\}) \\ c_{24} + g(4, S-\{4\}) \end{cases} = \min \begin{cases} c_{23} + g(3, \{4\}) \\ c_{24} + g(4, \{3\}) \end{cases}$$

i, S
$j = 3,4$

$$= \min \begin{cases} 9+20 \\ 10+15 \end{cases} = 25$$

$$g(3, \{4\}) = \min \{ c_{34} + g(4, S-\{4\}) \} = \min \{ c_{34} + g(4, \phi) \}$$

$i, S \quad \therefore j = 4$

$$= 12 + 8 = 20.$$

~~$= 9 + 6 = 15$~~

$$g(4, \phi) = c_{i1} = c_{41} = 8.$$

i, S

$$g(4, \{3\}) = \min \{ c_{43} + g(3, S-\{3\}) \} = \min \{ c_{43} + g(3, \phi) \}$$

$i, S \quad \therefore j = 3$

$$= 9 + 6 = 15$$

$$g(3, \phi) = c_{i1} = c_{31} = 6$$

i

$$g(3,\{2,4\}) = \min \begin{Bmatrix} C_{32} + g(2, S-\{2\}) \\ C_{34} + g(4, S-\{4\}) \end{Bmatrix} = \min \begin{Bmatrix} C_{32} + g(2,\{4\}) \\ C_{34} + g(4,\{2\}) \end{Bmatrix}$$

i, S

$\therefore j = 2, 4$

$$= \min \begin{Bmatrix} 13+18 \\ 12+13 \end{Bmatrix} = 25$$

$$g(2,\{4\}) = \min \{C_{24} + g(4, S-\{4\})\} = \min \{C_{24} + g(4,\phi)\}$$

$i, S \therefore j=4$

$$= 10+8 = 18$$

$$g(4,\phi) = C_{i1} = C_{41} = 8.$$

i

$$g(4,\{2\}) = \min \{C_{42} + g(2, S-\{2\})\} = \min \{C_{42} + g(2,\phi)\}$$

$i, S \therefore j=2$

$$= 8+5 = 13.$$

$$g(2,\phi) = C_{i1} = C_{21} = 5.$$

$$g(4,\{2,3\}) = \min \begin{Bmatrix} C_{42} + g(2, S-\{2\}) \\ C_{43} + g(3, S-\{3\}) \end{Bmatrix} = \min \begin{Bmatrix} C_{42} + g(2,\{3\}) \\ C_{43} + g(3,\{2\}) \end{Bmatrix}$$

$i, S \therefore j=2,3$

$$= \min \begin{Bmatrix} 8+15 \\ 9+18 \end{Bmatrix} = 23.$$

$$g(2,\{3\}) = \min \{C_{23} + g(3, S-\{3\})\} = \min \{C_{23} + g(3,\phi)\}$$

$i, S \therefore j=3$

$$= 9+6 = 15$$

$$g(3,\phi) = C_{i1} = C_{31} = 6$$

$$g(3,\{2\}) = \min \left(C_{32} + g(2, S-\{2\})\right) = \min \{C_{32} + g(2,\phi)\}$$

$$= 13+5 = 18.$$

$g(2, \phi) = C_{21} = 5$.

Hence length of the shortest tour is 35. and path is

$1 \to 2 \to 4 \to 3 \to 1$.

$C_{12} + g(2, \{3,4\})$. 1–2

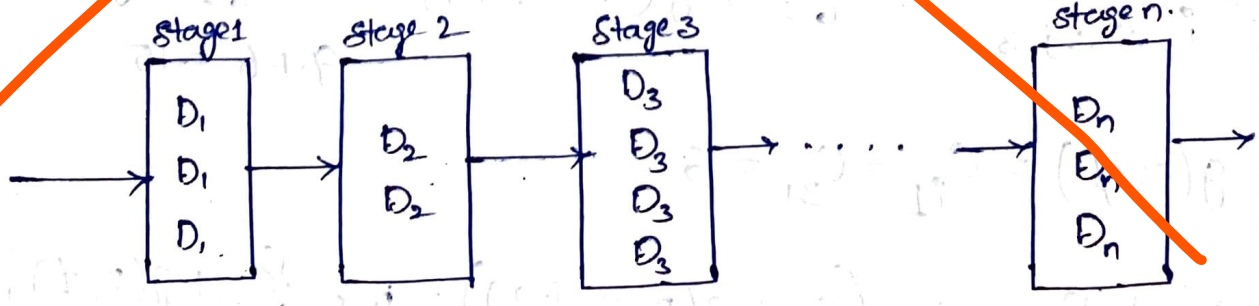
$C_{24} + g(4, \{3\})$ 2–4

$C_{43} + g(3, \phi)$ 4–3

C_{31}. 3–1.

(6) Reliability Design: The problem is to design a system that is composed of several devices connected in series.

Let γ_i be the reliability of device D_i i.e, γ_i is the probability that device i will function properly. Then, the reliability of the entire system is $\Pi \gamma_i$. Even if the individual devices are very reliable, the reliability of the system may not be very good. Hence, it is desirable to duplicate devices. Multiple copies of the same device are connected in parallel at each stage.

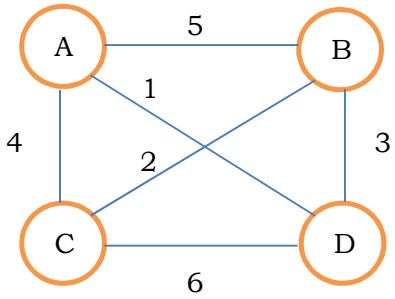


FREQUENTLY ASKED QUESTIONS

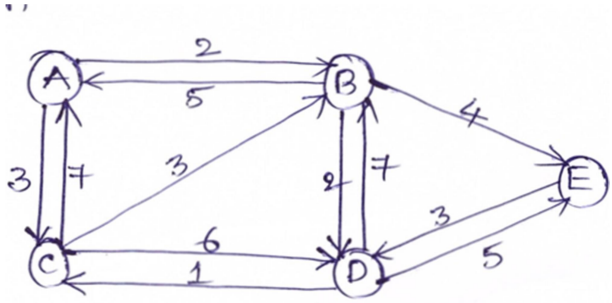
1. Differentiate Divide & Conquer and Dynamic Programming?
2. Differentiate Greedy and Dynamic Programming?
3. Explain the general concept of Dynamic Programming?
4. Explain how to solve 0/1 Knapsack problem using dynamic programming?
5. Find the solution for the knapsack problem when n=3, m=20, (p1,p2, p3)=(25, 24, 15), (w1, w2, w3)=(18, 15, 10) using dynamic programming.
6. Solve the following 0/1 knapsack problem using dynamic programming: n=3, m=8, (p1,p2, p3)=(3, 6, 6), (w1, w2, w3)=(2, 3, 4)
7. What is 0/1 knapsack problem? Define merging and purging rule of 0/1 knapsack problem.
8. Solve the following 0/1 knapsack problem using dynamic programming: n=4, m=40, (p1,p2, p3, p4)=(11, 21, 31, 33), (w1, w2, w3, w4)=(2, 11, 22, 15).
9. Solve the following 0/1 knapsack problem using dynamic programming: n=3, m=6, (p1,p2, p3)=(1, 2, 4), (w1, w2, w3)=(2, 3, 3)
10. Solve the following 0/1 knapsack problem using dynamic programming: n=6, m=165, (p1:p6) = (w1:w6) = (100, 50, 20, 10, 7, 3).
11. Solve the following 0/1 knapsack problem using dynamic programming: n=4, m=30, (p1,p2, p3, p4)=(2, 5, 8, 1), (w1, w2, w3, w4)=(10, 15, 6, 9).

    (OR)

    Generate the sets $0 \le i \le 6$, when n=4, m=30, (p1,p2, p3, p4)=(2, 5, 8, 1), (w1, w2, w3, w4)=(10, 15, 6, 9).
12. Given n=3, weights and profits as (2, 3, 4) and (1, 2, 5) respectively and knapsack capacity m=6. Compute the set S^i containing the pair (P_i, W_i).
13. Explain the process for solving All pairs shortest path problem using dynamic programming.
14. Find the shortest path between all pair of nodes in the following graph.



15. Find the shortest path between all pair of nodes in the following graph.



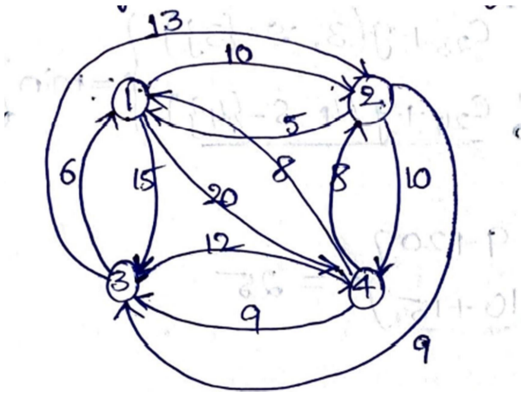
16. Write a function to compute lengths of shortest paths between all pairs of nodes for the given adjacency matrix.

$$\begin{pmatrix} 0 & 6 & 13 \\ 8 & 0 & 4 \\ 5 & \infty & 0 \end{pmatrix}$$

17. Find the all pairs shortest path solution for the graph represented by below adjacency matrix:

$$\begin{bmatrix} \infty & 6 & 5 & 4 \\ 3 & \infty & 2 & 6 \\ 18 & 6 & \infty & 7 \\ 8 & 12 & 10 & \infty \end{bmatrix}$$

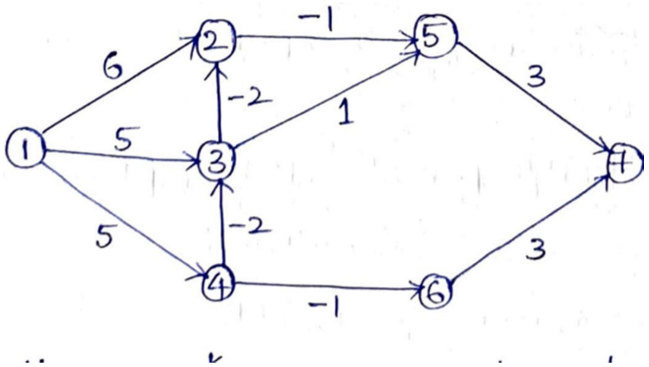
18. What is travelling sales person problem? How it can be solved using dynamic programming approach?
19. Give an example of travelling sales person problem?
20. What is All – Pair Shortest Path problem (APSP)? Discuss the Floyd's APSP algorithm and discuss the analysis of this algorithm.
21. What is principle's of optimality? Explain how travelling sales person problem uses the dynamic programming technique with example?
22. Find the shortest tour of a travelling sales person for the following instance using dynamic programming.



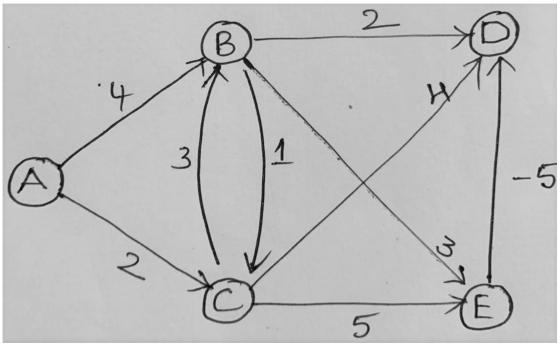
23. Construct an optimal travelling sales person tour using Dynamic Programming for the given data:

$$\begin{bmatrix} 0 & 10 & 9 & 3 \\ 5 & 0 & 6 & 2 \\ 9 & 6 & 0 & 7 \\ 7 & 3 & 5 & 0 \end{bmatrix}$$

24. What is Single source shortest path problem? How it can be solved using dynamic programming approach?
25. Write Bellman-Ford algorithm?
26. Find shortest path from node 1 to every other node in the graph using Bellman-Ford algorithm.



27. Find shortest path from node A to every other node in the graph using Bellman-Ford algorithm.



28. Let X = a,a,b,a,a,b,a,b,a,a and Y = b,a,b,a,a,b,a,b. Find a minimum-cost edit sequence that transforms X and Y.
29. Let X = "INTENTION" and Y = "EXECUTION". Find a minimum-cost edit sequence that transforms X and Y.