122.

Job sequencing with deadline

It is a type of optimization problem that uses greedy strategy. We have n jobs each with a deadline $d_i \geq 0$ and profit $P_i \geq 0$. Our objective is to earn maximum profit. We will earn profit only when a job is completed on or before the deadline. The optimal soln is the feasible soln with max. profit.

Optimization criteria, ~~max $\sum_{i \in J} P_i$~~ $\max \sum_{i \in J_r} P_i$

Consider $n=5$, P_1, P_2, P_3, P_4, P_5 are 20, 15, 10, 5, 1 and deadline 2, 2, 1, 3, 3.

Step 1: Sort the job according to the profit in descending order.

P_1	P_2	P_3	P_4	P_5
20	15	10	5	1
2	2	1	3	3

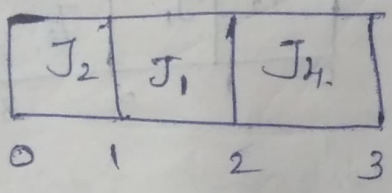
Assume the machine should complete each job in one unit of time.

Deadline: A job should wait until it's completion time.

ex: If the job deadline is ... for 2 has.

Job.	assigned slot	job Considered	action	Profit
$\{\}$	none	J_1	assign $\{1,2\}$	0.
$\{J_1\}$	$\{1,2\}$.	J_2	assign $\{0,1\}$.	20.
$\{J_1, J_2\}$	$\{1,2\}$ $\{0,1\}$.	J_3.	No, reject.	20+15=35
$\{J_1, J_2\}$	$\{1,2\}$ $\{0,1\}$	J_4	assign $\{2,3\}$	35.
$\{J_1, J_2, J_4\}$	$\{1,2\}$ $\{0,1\}$ $\{2,3\}$	J_5	No, reject	35+5=40.

Slot.



Total profit = 40

(Q) Consider the following 5 jobs and their associated deadline and profit

	P_1	P_2	P_3	P_4	P_5
	60	100	20	40	20
deadline	2	1	3	2	1

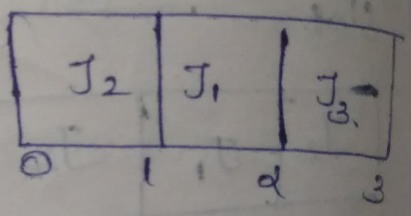
(A) Sort according to profit

P_2	P_1	P_4	P_3	P_5
100	60	40	20	20
1	2	2	3	1

Job	assigned slot			
{ }	none	J_2	assign {0,1}.	0.
$\{J_2\}$	{0,1}.	J_1	assign {1,2}.	100.
$\{J_2, J_1\}$	{0,1} {1,2}.	J_4	NO, reject	100+60=
$\{J_2, J_1\}$	{0,1} {1,2}.	J_3.	assign {2,3}.	160
$\{J_2, J_1, J_3\}$	{0,1} {1,2} {2,3}	J_5	NO, reject	160+20=

Total profit = 180

Job assigned = J_2, J_1, J_3



Procedure for Greedy Job.

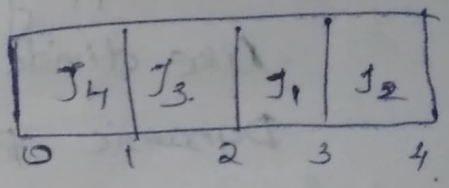
Greedy Job {d, j, n} → deadline, job, no. of Job

```
{ i = {1}.
  for (i=2 to n) do
  { if (all job is j ∪ {i}
      can be completed by deadline then,
      j = j ∪ {i}.
  }.
}.
```

(a) Consider

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7
Profit	35	30	25	20	15	12	5
deadline	3	4	4	2	3	1	2

Job	assigned slot	Job scheduled	action	Profit
{ }	none	J_1	assign {2,3}	0.
$\{J_1\}$	{2,3}	J_2	assign {3,4}	35
$\{J_1, J_2\}$	{2,3} {3,4}	J_3	No, reject	35+30=75.
$\{J_1, J_2\}$	{2,3} {3,4}	J_4	assign {1,2}	75
$\{J_1, J_2, J_4\}$	{2,3} {3,4} {1,2}	J_5		



Job	assigned slot	Job scheduled	action	Profit
{ }	none	J_1	assign {2,3}	0.
$\{J_1\}$	{2,3}	J_2	assign {3,4}	35
$\{J_1, J_2\}$	{2,3} {3,4}	J_3	assign {1,2}	65
$\{J_1, J_2, J_3\}$	{2,3} {3,4} {1,2}	J_4	assign {0,1}	90.
$\{J_1, J_2, J_3, J_4\}$	{0,1} {2,3} {3,4} {1,2}	J_5	no, reject	110
$\{J_1, J_2, J_3, J_4\}$	{0,1} {2,3} {3,4} {1,2}	J_6	no, reject	110.
$\{J_1, J_2, J_3, J_4\}$	{0,1} {2,3} {3,4} {1,2}	J_7	no, reject	110.

Total profit = 110.

Job selected, J_1, J_2, J_3, J_4.

4/1/22

Dynamic Programming

Solving a complex problem by breaking it down into a collection of simpler subproblems. Each solved subproblem are stored in array. Next time the same problem occurs, instead of recomputing it's solution, looks up previous solved computed solution.

Two properties of dynamic programming

(1) Overlapping sub Problem

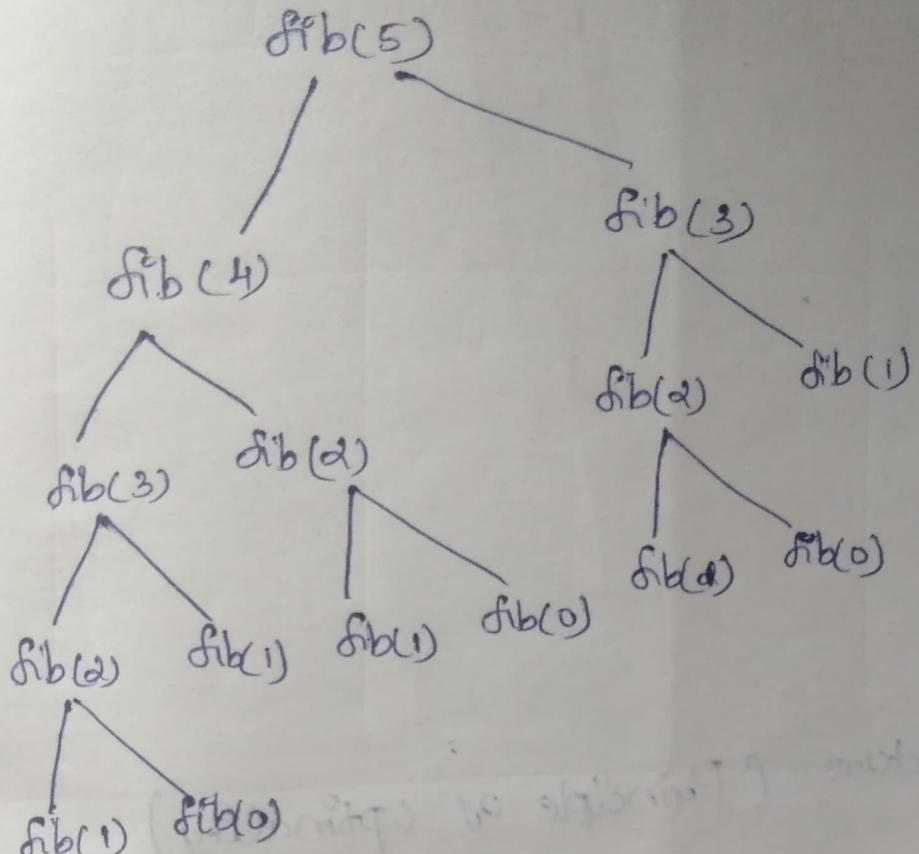
Like divide and conquer

Dynamic programming contains subproblems, solution of same subproblem are needed repeatedly. So the computed solutions are stored in table and reuse the solution.

```
fib(int n)
if (n<=0)
    return 0;
else {
```

```
fib(int n)
if (n=0)
  {return 0}
if (n=1) {
      return 1;}
else:
  {return f(n-2)+fib(n-1);
  }
```

fib(5) → recursion tree



Inorder to find fib(5), we need to divide the problem into 2 recursive fns f(4) & f(3). This subdivision continues until we reach fib(0) and fib(1) ie the leaf node. This causes duplication and this is called overlapping subproblem

2 ways to store data in ~~an array~~ DP.

(1) memonization

(2) Tabulation

-1	-1	-1	-1	-1

Problem behind DP:

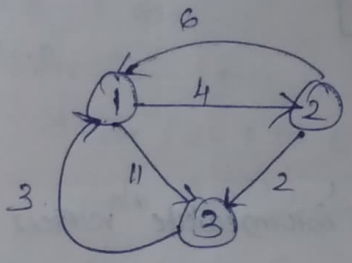
All pair Shortest path problem

Bellman Ford algorithm

Travelling Salesman problem

All pair shortest path problem

example:-



$A_0 =$

	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

via

$A_1 = (0,1)$

	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

$A_2 = (0, 1, 2)$ via

	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

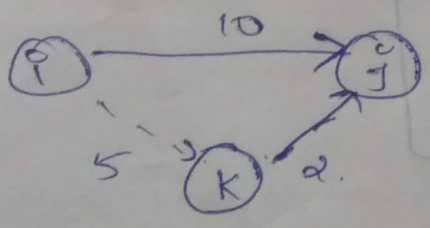
$A_3 = (0, 1, 2, 3)$ via

	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

FLOYD - WARSHALL (W)

This algorithm consider the 'intermediate' vertices of shortest path. This problem is to find the shortest path between all vertices in a given graph. We start this problem by writing the cost or distance using cost adjacency matrix. Then we refine all the vertex by replacing the given cost by a minimum cost.

Intermediate vertex of shortest path $P = \{v_1, v_2, \ldots, v_n\}$. where intermediate vertex is any vertex other than v_1, v_n $\ldots, v_n$.



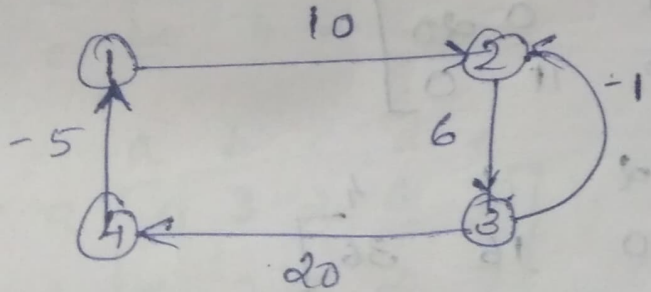
Recursive Equation

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0 \\ \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right) & \text{if } k \geq 1 \end{cases}$$

Algorithm

1. $n \leftarrow$ rows [w]
2. $D^{(0)} \leftarrow W$
3. For $k \leftarrow 1$ to n
4. do for $i \leftarrow 1$ to n
5. do for $j \leftarrow 1$ to n
6. do $d_{ij}^{(k)} \leftarrow \min\left(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right)$
7. return $D^{(n)}$.

(Q) Find the shortest path for the given graph using Floyd-Warshall algorithm. Find the shortest path from 1 to 4.



$$D_0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 10 & \infty & \infty \\ \infty & 0 & 6 & \infty \\ \infty & -1 & 0 & 20 \\ -5 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

$A_1^{via\ (0,1)}$ =

	1	2	3	4
1	0	10	∞	∞
2	∞	0	6	∞
3	∞	-1	0	20
4	-5	5	∞	0

$A_2^{via\ (0,1,2)}$ =

	1	2	3	4
1	0	10	16	∞
2	∞	0	6	∞
3	∞	-1	0	20
4	-5	5	11	0

$A_3^{via\ (0,1,2,3)}$ =

	1	2	3	4
1	0	10	16	36
2	∞	0	6	26
3	∞	-1	0	20
4	-5	5	11	0

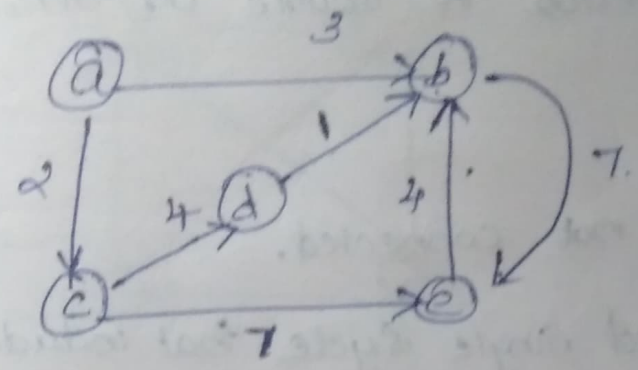
$A_4\ (0,1,2,3,4)$ =

	1	2	3	4
1	0	10	16	36
2	21	0	6	26
3	15	-1	0	20
4	-5	5	11	0

Shortest path from 1 to 4

and the optimal path is $1 \to 2 \to 3 \to 4$

(a) Find the shortest path cost of AC



A_0 =

	a	b	c	d	e
a	0	3	2	∞	∞
b	∞	0	∞	∞	7
c	∞	∞	0	4	7
d	∞	1	∞	0	∞
e	∞	4	∞	∞	0

A_1 =

	a	b	c	d	e
a	0	3	2	∞	∞
b	∞	0	∞	∞	7
c	∞	∞	0	4	7
d	∞	1	∞	0	∞
e	∞	4	∞	∞	0

A_2 (a,b) =

	a	b	c	d	e
a	0	3	2	∞	10
b	∞	0	∞	∞	7
c	∞	∞	0	4	7
d	∞	1	∞	0	8
e	∞	4	∞	∞	0

A_3 (a,b,c) =

	a	b	c	d	e
a	0	3	2	6	9
b	∞	0	∞	∞	7
c	∞	∞	0	4	7
d	∞	1	∞	0	8
e	∞	4	∞	∞	0

A_4 (a,b,c,d) =

	a	b	c	d	e
a	0	3	2	6	9
b	∞	0	∞	∞	7
c	∞	5	0	4	7
d	∞	1	∞	0	8
e	∞	4	∞	∞	0

A_5 =

	a	b	c	d	e
a	0	3	2	6	9
b	∞	0	∞	∞	7
c	∞	5	0	4	7
d	∞	1	∞	0	8
e	∞	4	∞	∞	0

Shortest path cost A→E = 9.

Optimal path ⇒ a → c → e

6/11/24

TSP (Travelling salesman problem)

Start from your city and move through various intermediate cities atleast once, and reach the source or return back to the starting address.

$c_{ij} = \infty$, if a nodes is not connected.

A tour of G is directed single cycle that include every vertex in V.

cost = $\sum$ weight of edges on a tour.

To avoid the complexity of naive method we use TSP through dynamic programming. Each tour consists of an edge $\{i\ k\}$ for some k, element of $V - \{1\}$ and also a path from vertex $k \to 1$

IF the tour is optimal then a path from $k \to 1$ through all vertices, in $V - \{i, k\}$ should be shortest.

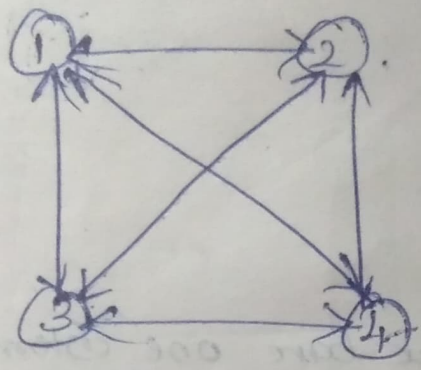
Principle of optimality of TSP

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

$g(i,s)$ → it is the shortest path starting from i and going through all the vertex in S and terminating at vertex 1. [S = intermediate vertices]

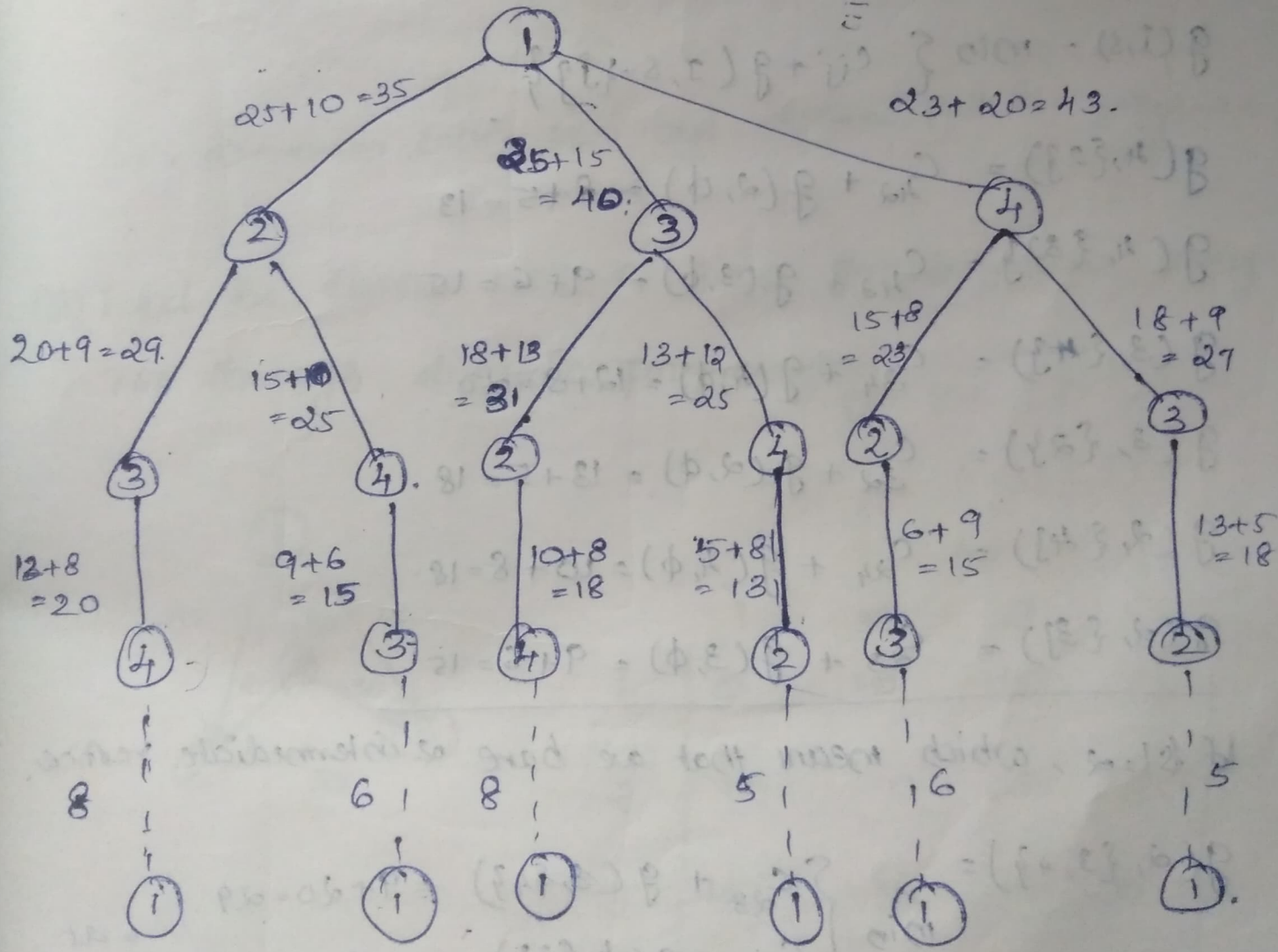
Naive Method

Consider



	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

(A) State space tree



min {35, 40, 43} = 35 and the optimal path is

1 → 2 → 4 → 3

If $|S|=0$ which means that there is no intermediate vertex

so $S=0$, then $g(i,S) = c_{ij}$

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

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

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

If $|S|=1$, which means that we have one intermediate vertex.

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

$g(1,\{2\}) = c_{12} + g(2,\phi) = 8+5 = 13$

$g(1,\{3\}) = c_{13} + g(3,\phi) = 9+6 = 15$

$g(3,\{4\}) = c_{34} + g(4,\phi) = 12+8 = 20$

$g(3,\{2\}) = c_{32} + g(2,\phi) = 13+5 = 18$

$g(2,\{4\}) = c_{24} + g(4,\phi) = 10+8 = 18$

$g(2,\{3\}) = c_{23} + g(3,\phi) = 9+6 = 15$

If $|S|=2$, which means that we have 2 intermediate vertex

$g(2,\{3,4\}) = \min \begin{cases} c_{23} + g(3,\{4\}) = 9+20 = 29 \\ c_{24} + g(4,\{3\}) = 10+15 = 25 \end{cases}$

$g(3,\{2,4\}) = \min \{ c_{32} + g(2,\{4\}) = 13+18 = 31$

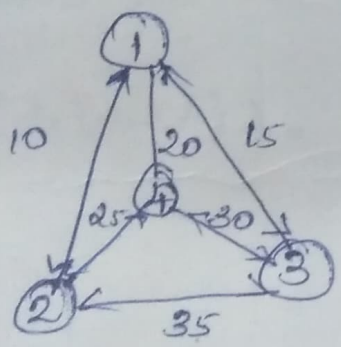
$$g(4,\{2,3\}) = \min \begin{cases} c_{42} + g(2,\{3\}) = 8+15 = 23 \\ c_{43} + g(3,\{2\}) = 9+18 = 27 \end{cases} = 23$$

If |S|=3, which means that we have 3 intermediate vertices.

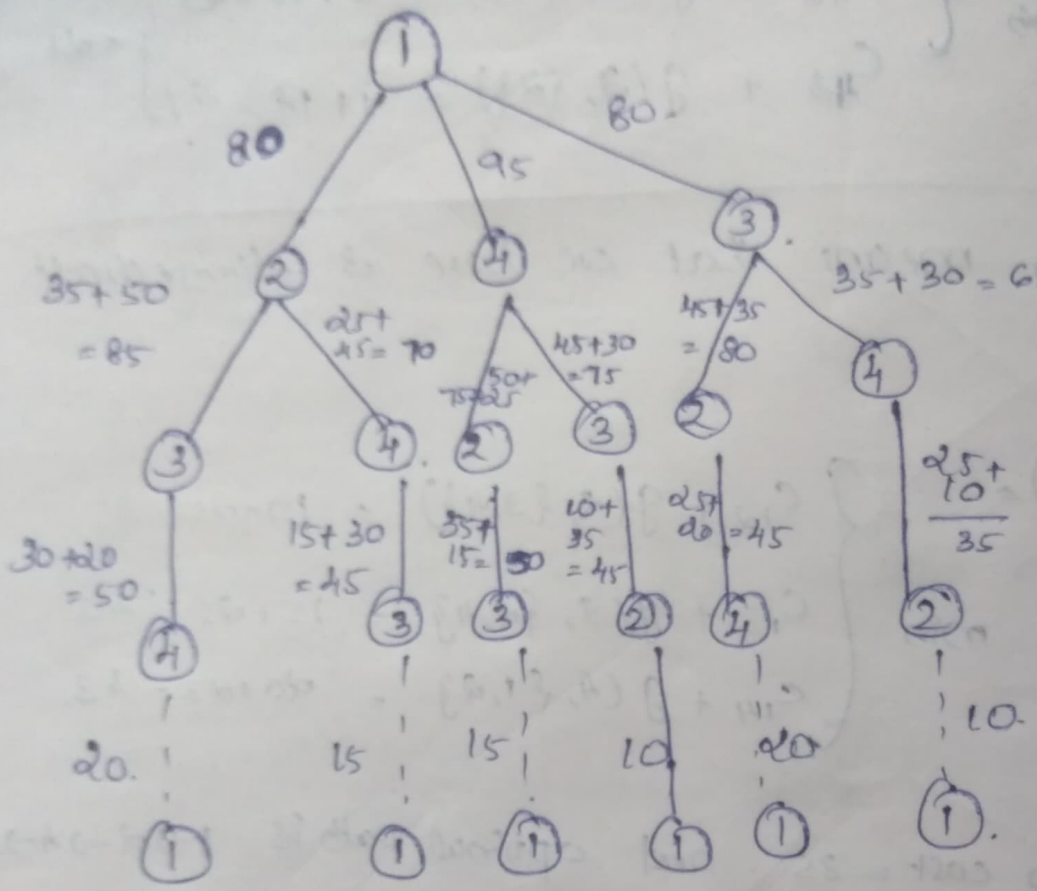
$$g(1,\{2,3,4\}) = \min \begin{cases} c_{12} + g(2,\{3,4\}) = 10+25 = 35 \\ c_{13} + g(3,\{2,4\}) = 15+25 = 40. \\ c_{14} + g(4,\{1,2\}) = 20+23 = 43. \end{cases}$$

∴ minimum cost = 35 and optimal path is $1 \to 2 \to 4 \to 3$

(Q). Find the shortest path of given graph below, using TSP through dynamic programming.



	1	2	3	4
1	0	10	15	20
2	10	0	35	25
3	15	35	0	30
4	20	25	30	0



If $|S| = 0$, then $g(i,j) = c_{ij}$

$g(4, \phi) = 20$

$g(3, \phi) = 15$

$g(2, \phi) = 10$.

If $|S| = 1$ then

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

$g(3,\{4\}) = c_{34} + g(4,\phi) = 30 + 20 = 50$

$g(3,\{2\}) = c_{32} + g(2,\phi) = 35 + 10 = 45$

$g(4,\{3\}) = c_{43} + g(3,\phi) = 30 + 15 = 45$

$g(4,\{2\}) =$

$g(2,\{3\}) = c_{23} + g(3,\phi) = 35 + 15 = 50$

$g(2,\{4\}) = c_{24} + g(4,\phi) = 25 + 20 = 45.$

If $|S| = 2$, then

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

$$g(2,\{3,4\}) = \min \begin{cases} c_{23} + g(3,\{4\}) = 35+50 = 85 \\ c_{24} + g(4,\{3\}) = 25+45 = 70. \end{cases} = 70$$

$$g(4,\{2,3\}) = \min \begin{cases} c_{42} + g(2,\{3\}) = 25+50 = 75 \\ c_{43} + g(3,\{2\}) = 30+45 = 75 \end{cases} = 75$$

$$g(3,\{2,4\}) = \min \begin{cases} c_{32} + g(2,\{4\}) = 35+45 = 80 \\ c_{34} + g(4,\{2\}) = 30+35 = 65 \end{cases}$$

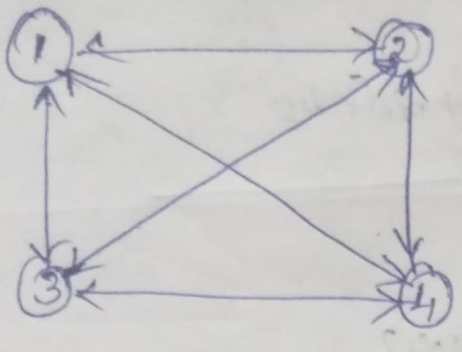
If $|S| = 3$ then

$$g(1,\{2,3,4\}) = \min \begin{cases} c_{12} + g(2,\{3,4\}) = 10+70 = 80 \\ c_{13} + g(3,\{2,4\}) = 15+65 = 80, \\ c_{14} + g(4,\{2,3\}) = 20+75 = 95 \end{cases} = 80$$

Minimum cost = 80.

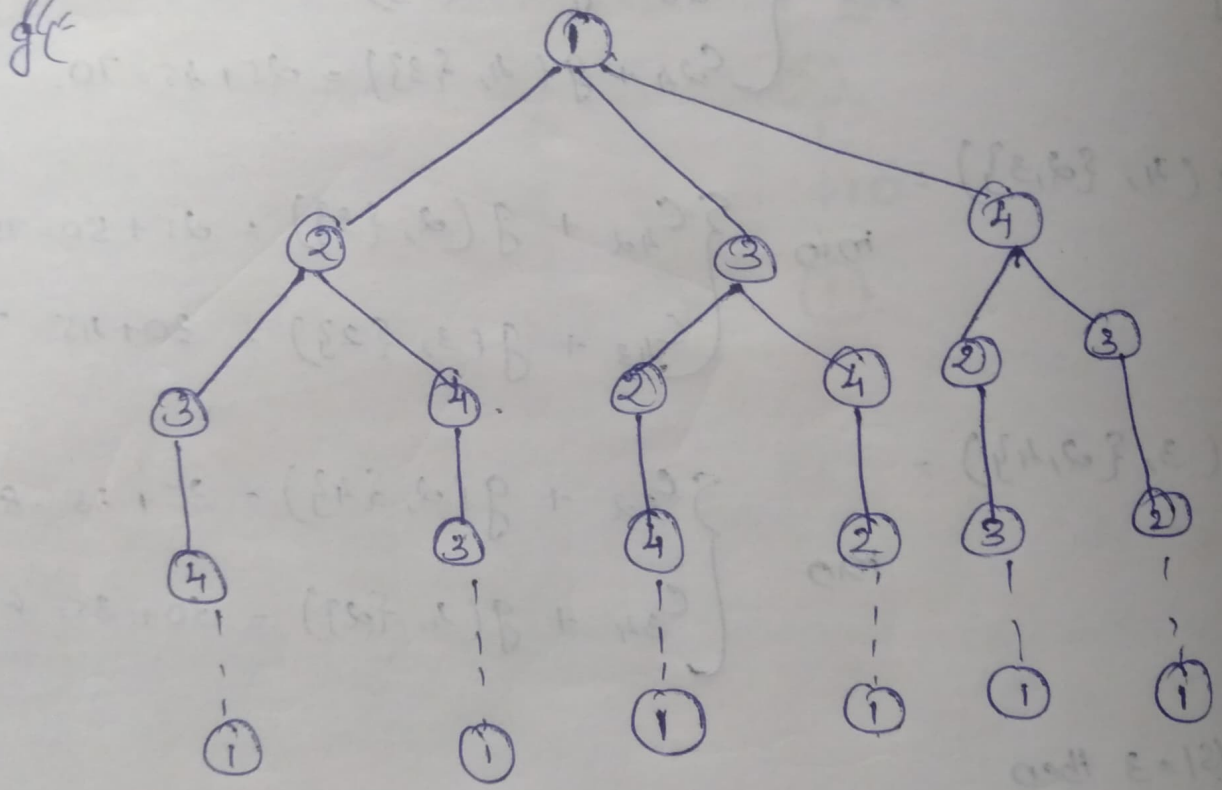
Optimal path $\Rightarrow$ $1 \to 3 \to 4 \to 2$.

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



	1	2	3	4
1	0	5	25	12
2	5	0	10	20
3	25	10	0	9
4	12	20	9	0

If $S \neq \phi$.



If $|S| = 0$.

$g(4, \phi) = 12$

$g(3, \phi) = 25$

$g(2, \phi) = 5$

If $|S| = 1$

$g(3, \{4\}) = c_{34} + g(4, \phi) = 9 + 12 = 21$

$g(2, \{4\}) = c_{24} + g(4, \phi) = 20 + 12 = 32$

$g(2, \{3\}) = c_{23} + g(3, \phi) = 10 + 25 = 35$

$g(4, \{3\}) = c_{43} + g(3, \phi) = 9 + 25 = 34$

$g(4, \{2\}) = c_{42} + g(2, \phi) = 20 + 5 = 25$

If $|S| = 2$.

$$g(2, \{3,4\}) = \min \begin{cases} c_{23} + g(3, \{4\}) = 10 + 21 = 31 \\ c_{24} + g(4, \{3\}) = 20 + 34 = 54 \end{cases} = 31$$

$$g(3, \{2,4\}) = \min \begin{cases} c_{32} + g(2, \{4\}) = 10 + 32 = 42 \\ c_{34} + g(4, \{2\}) = 9 + 25 = 34 \end{cases} = 34$$

$$g(4, \{2,3\}) = \min \begin{cases} c_{42} + g(2, \{3\}) = 20 + 35 = 55 \\ c_{43} + g(3, \{2\}) = 9 + 15 = 24 \end{cases} = 24$$

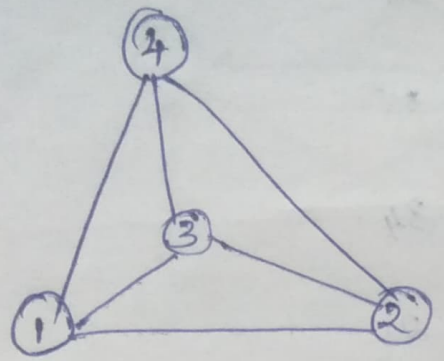
If $|S| = 3$

$$g(1, \{2,3,4\}) = \min \begin{cases} c_{12} + g(2, \{3,4\}) = 5 + 31 = 36 \\ c_{13} + g(3, \{2,4\}) = 25 + 34 = 59 \\ c_{14} + g(4, \{2,3\}) = 12 + 24 = 36. \end{cases}$$

minimum cost = 36

optimal path $\Rightarrow$ $1 \to 2 \to 3 \to 4$

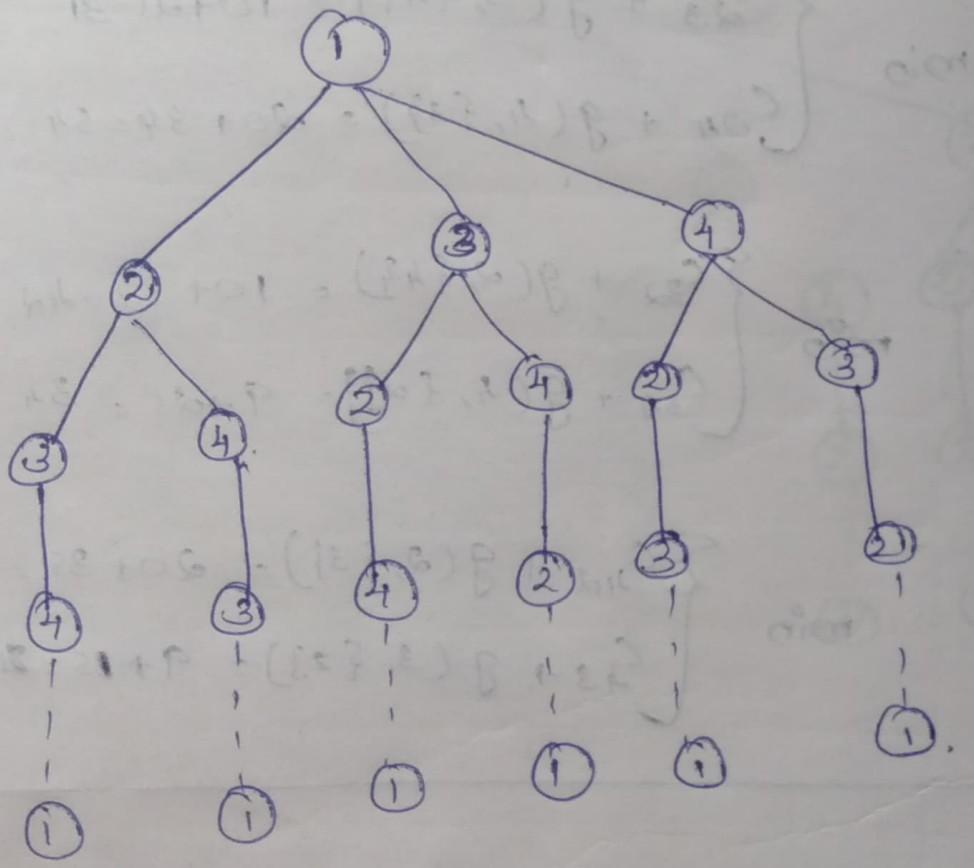
(Q)



	1	2	3	4
1	0	8	5	3
2	4	0	3	5
3	2	1	0	3
4	5	7	2	0.

Find the shortest path of given graph using TSP.

~~If |S|=0.~~



If $|S|=0$

$g(i,j) = c_{ij}$

$g(4,\phi) = 5$

$g(3,\phi) = 2$

$g(2,\phi) = 4.$

If $|S| = 1$

$g(i,j) = \min(C_{ij} + g[J, S - \{j\}])$

$g(3,\{4\}) = C_{34} + g(4,\phi) = 3 + 5 = 8.$

$g(3,\{2\}) = C_{32} + g(2,\phi) = 1 + 4 = 5$

$g(4,\{3\}) = C_{43} + g(3,\phi) = 2 + 2 = 4$

$g(4,\{2\}) = C_{42} + g(2,\phi) = 7 + 4 = 11$

$g(2,\{4\}) = C_{24} + g(4,\phi) = 5 + 5 = 10.$

$g(2,\{3\}) = C_{23} + g(3,\phi) = 3 + 2 = 5.$

If $|S| = 2.$

$$g(2,\{3,4\}) = \min \begin{cases} C_{23} + g(3,\{4\}) = 3+8 = 11 \\ C_{24} + g(4,\{3\}) = 5+4 = 8 \end{cases} = 8.$$

$$g(3,\{2,4\}) = \min \begin{cases} C_{32} + g(2,\{4\}) = 1+10 = 11 \\ C_{34} + g(4,\{2\}) = 3+11 = 14 \end{cases} = 11$$

$$g(4,\{2,3\}) = \min \begin{cases} C_{42} + g(2,\{3\}) = 7+5 = 12 \\ C_{43} + g(3,\{2\}) = 2+5 = 7 \end{cases} = 12.$$

If $|S| = 3.$

$$g(1,\{2,3,4\}) = \min \begin{cases} C_{12} + g(2,\{3,4\}) = 8+8 = 16 \\ C_{13} + g(3,\{2,4\}) = 5+11 = 16 \\ C_{14} + g(4,\{2,3\}) = 3+7 = 10. \end{cases} = 10$$