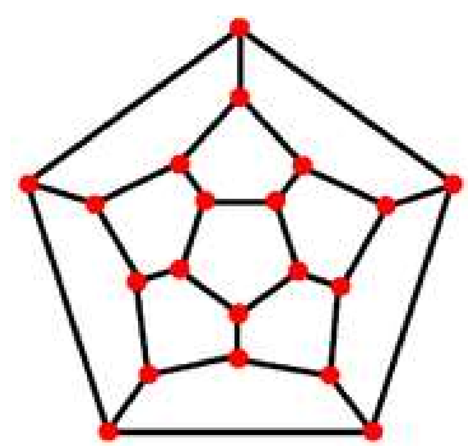
The traveling-salesman problem

Hamiltonian cycles

- Formally, a **Hamiltonian cycle of an undirected graph** G = (V,E) is a simple cycle that contains each vertex in V.
- A graph that contains a Hamiltonian cycle is said to be **Hamiltonian; otherwise, it is nonhamiltonian.**
- A **simple cycle** may be defined either as a closed walk with no repetitions of vertices and edges allowed, other than the repetition of the starting and ending vertex.

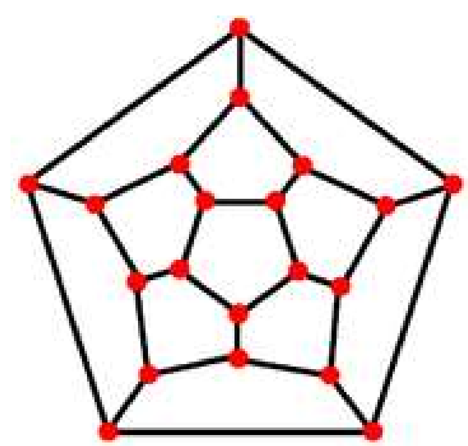
Hamiltonian cycles

- Formally, a **Hamiltonian cycle of an undirected graph** G = (V,E) is a simple cycle that contains each vertex in V.
- A graph that contains a Hamiltonian cycle is said to be **Hamiltonian; otherwise, it is nonhamiltonian.**



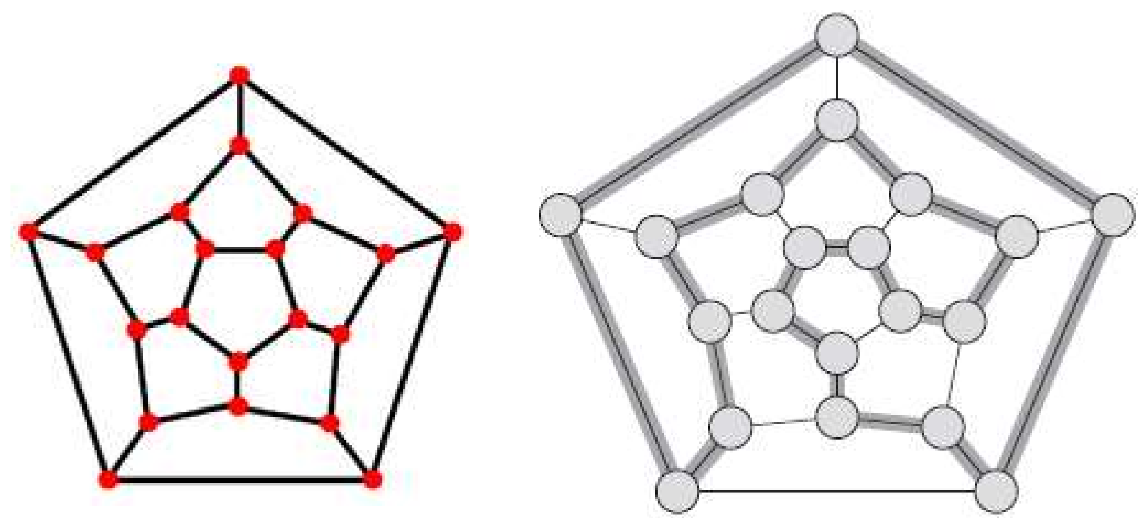
Hamiltonian cycles

- Formally, a **Hamiltonian cycle of an undirected graph** G = (V,E) is a simple cycle that contains each vertex in V.
- A graph that contains a Hamiltonian cycle is said to be **Hamiltonian; otherwise, it is nonhamiltonian.**



Hamiltonian cycles

- Formally, a **Hamiltonian cycle of an undirected graph** G = (V,E) is a simple cycle that contains each vertex in V.
- A graph that contains a Hamiltonian cycle is said to be **Hamiltonian; otherwise, it is nonhamiltonian.**

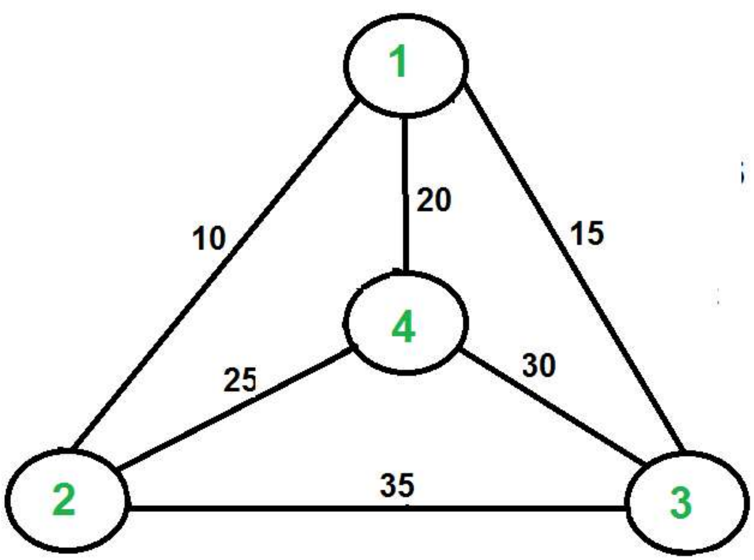


The traveling-salesman problem

- In the **traveling-salesman problem, which is closely related to the Hamiltonian cycle** problem, a salesman must visit ***n*** cities.
- Modeling the problem as a complete graph with ***n*** vertices, we can say that the salesman wishes to make a ***tour, or*** **Hamiltonian cycle**, visiting each city exactly once and finishing at the city he starts from.

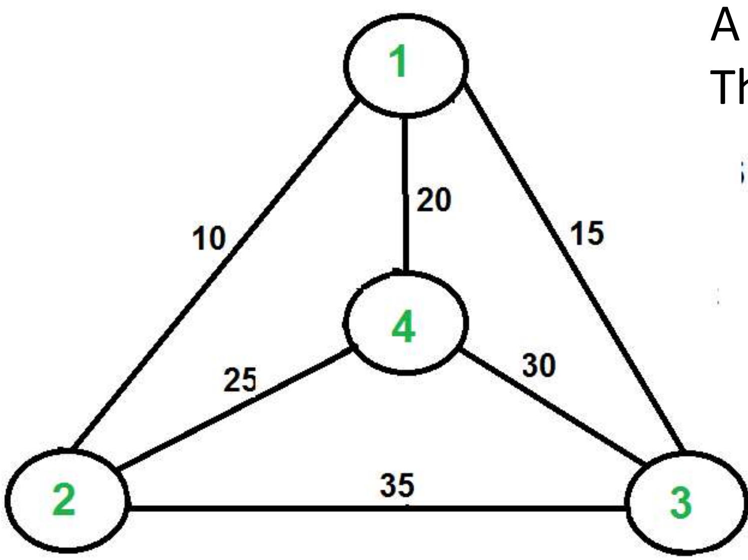
The traveling-salesman problem

- The salesman incurs a nonnegative integer **cost c(i, j)** to travel from **city i to city j**, and the salesman wishes to make the tour whose **total cost is minimum**, where the total cost is the sum of the individual costs along the edges of the tour.



The traveling-salesman problem

- The salesman incurs a nonnegative integer **cost c(i, j)** to travel from **city i to city j**, and the salesman wishes to make the tour whose **total cost is minimum**, where the total cost is the sum of the individual costs along the edges of the tour.



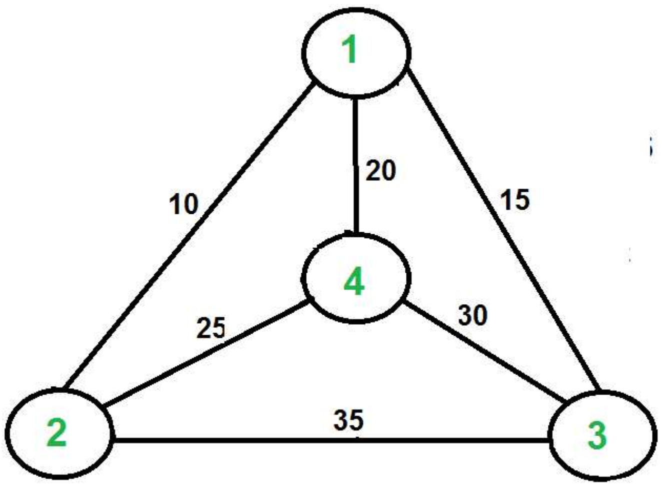
A TSP tour in the graph is **1-2-4-3-1**.
The cost of the tour is **10+25+30+15=80**.

Naive Solution

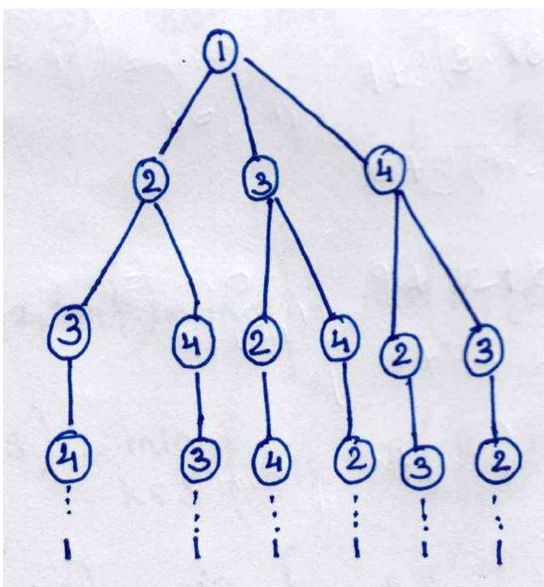
1) Consider city 1 as the starting and ending point.
2) Generate all (n-1)! Permutations of cities.
3) Calculate cost of every permutation and keep track of minimum cost permutation.
4) Return the permutation with minimum cost.

Time Complexity: $\Theta(n!)$

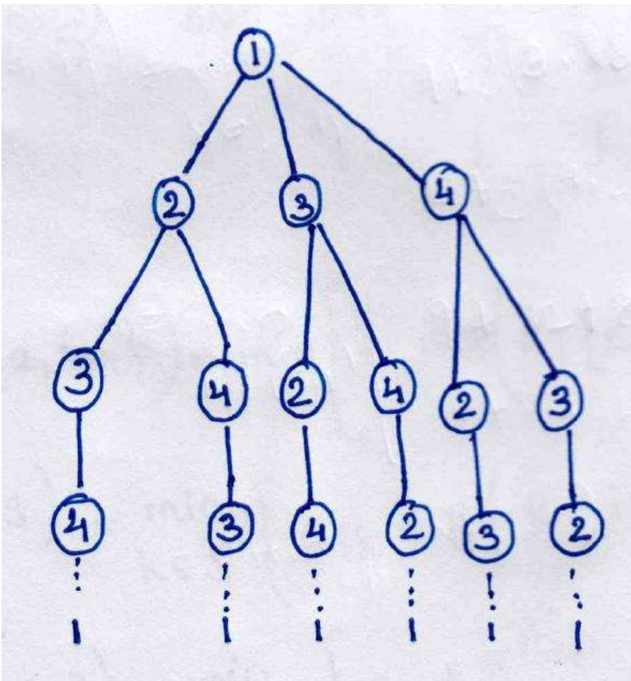
Dynamic Programming

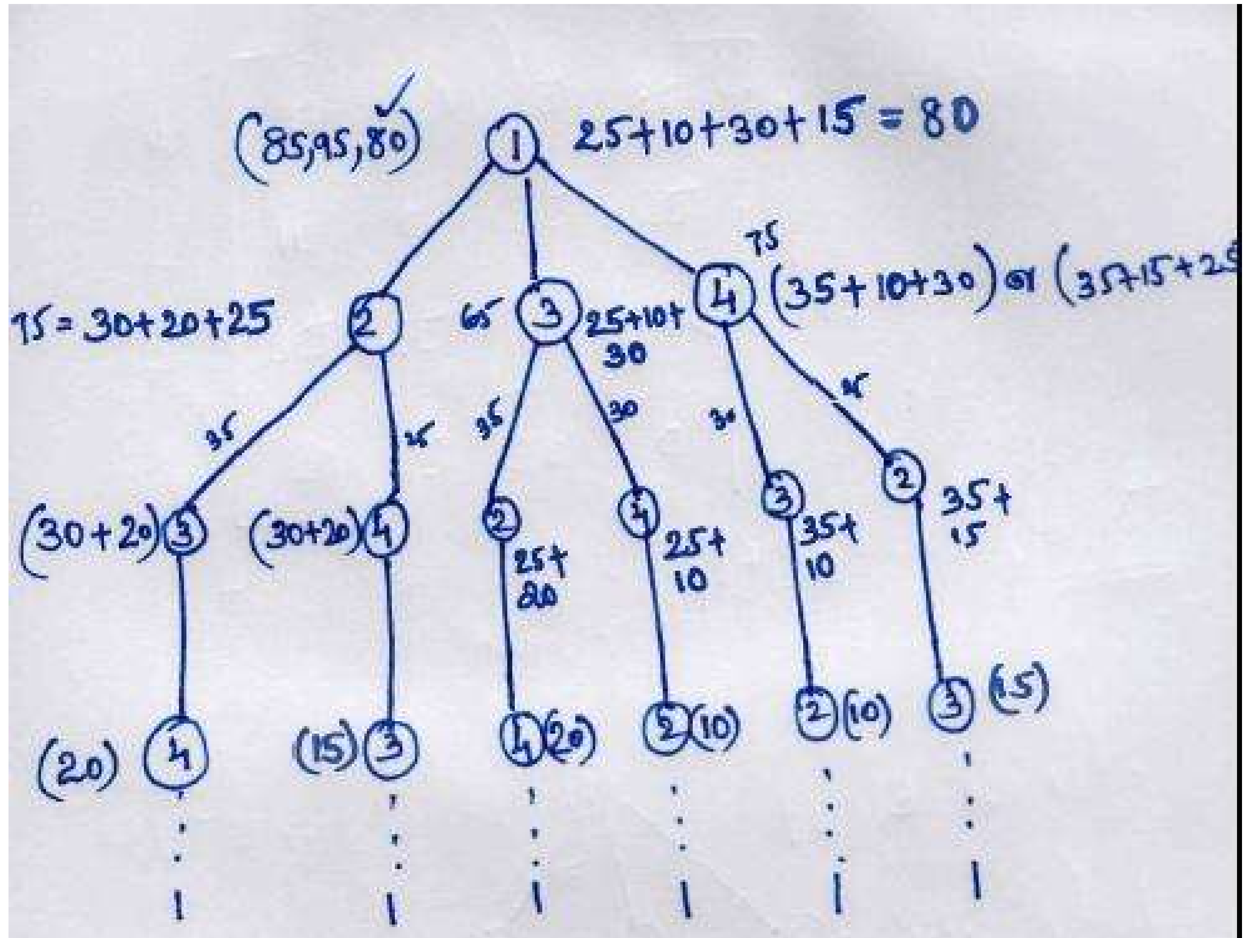


$$
\begin{array}{c}
\\
1 \\ 2 \\ 3 \\ 4
\end{array}
\begin{array}{c}
\begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\
\begin{bmatrix}
0 & 10 & 15 & 20 \\
10 & 0 & 35 & 25 \\
15 & 35 & 0 & 30 \\
20 & 25 & 30 & 0
\end{bmatrix}
\end{array}
$$



Dynamic Programming





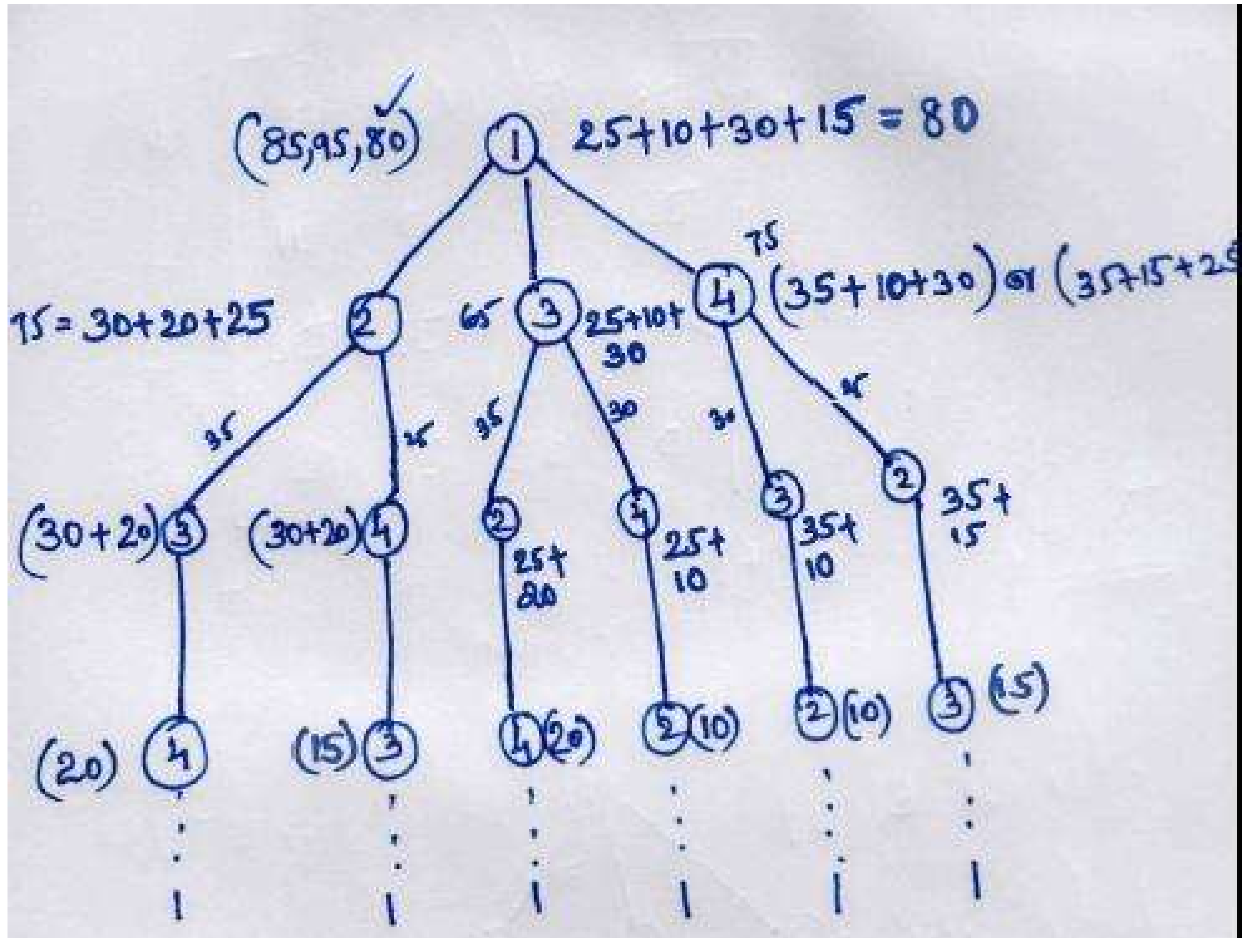
Dynamic Programming

$$g(1,\{2,3,4\}) = \min\left\{c_{1k} + g(k,\{2,3,4\}-k)\right\}$$

$$g(i,S) = \min_{k \in S}\left\{c_{ik} + g(k, S-\{k\})\right\}$$

$$g(2,\{3\}) = \min_{k \in \{3\}}\left\{c_{23} + g(\{3\},\phi)\right\}$$

$$g(2,\{3,4\}) = \min_{k \in \{3,4\}}\left[c_{23} + g(3,4),\ c_{24} + g(4,3)\right]$$



Dynamic Programming

1. If size of **S is 2**, then S must be {1, i},

   C(S, i) = dist(1, i)

2. Else if size of **S is greater than 2**.

   C(S, i) = min { C(S-{i}, j) + dis(j, i)}

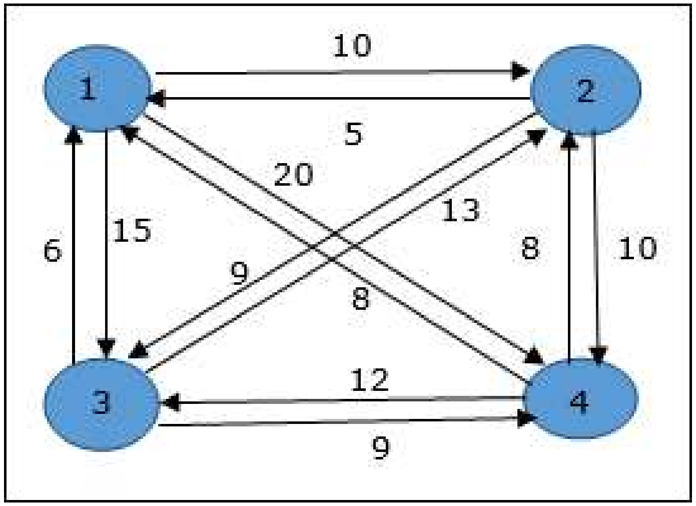
   where j belongs to S, $j \neq i$ and $j \neq 1$.

There are n possible start vertices and 2^n possible sub-graphs.

So this function will be called on at most $n2^n$ distinct arguments (the target never changes).

Each call performs at most O(n) work (there are at most n neighbors). Hence the total work you're doing is $O(n^2 2^n)$

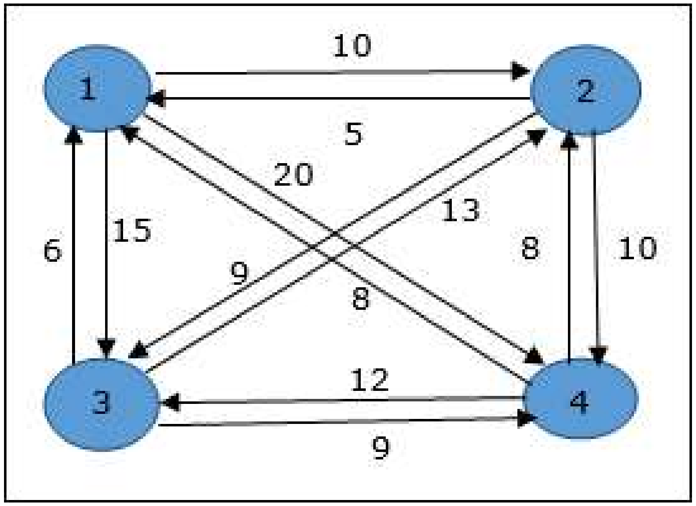
Dynamic Programming



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

S = Φ
Cost(2,Φ,1)=d(2,1)=5
Cost(3,Φ,1)=d(3,1)=6
Cost(4,Φ,1)=d(4,1)=8

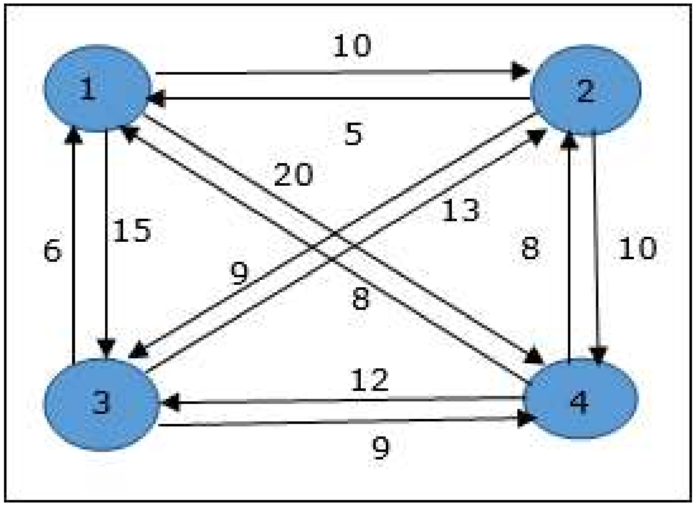
Dynamic Programming



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

S = 1
Cost(i,s)=min{Cost(j,s–(j))+d[i,j]}Cost(i,s)=min{Cost(j,s–(j))+d[i,j]}
Cost(2,{3},1)=d[2,3]+Cost(3,Φ,1)=9+6=15
Cost(2,{4},1)=d[2,4]+Cost(4,Φ,1)=10+8=18
Cost(3,{2},1)=d[3,2]+Cost(2,Φ,1)=13+5=18
Cost(3,{4},1)=d[3,4]+Cost(4,Φ,1)=12+8=20
Cost(4,{3},1)=d[4,3]+Cost(3,Φ,1)=9+6=15
Cost(4,{2},1)=d[4,2]+Cost(2,Φ,1)=8+5=13

Dynamic Programming



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

S = 2

Cost(2,{3,4},1)=min {d[2,3]+Cost(3,{4},1)=9+20=29, d[2,4]+Cost(4,{3},1)=10+15=25}=25

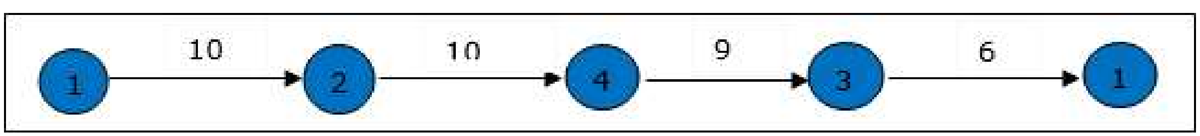
Cost(3,{2,4},1)=min{d[3,2]+Cost(2,{4},1)=13+18=31, d[3,4]+Cost(4,{2},1)=12+13=25}=25

Cost(4,{2,3},1)=min{d[4,2]+Cost(2,{3},1)=8+15=23, d[4,3]+Cost(3,{2},1)=9+18=27}=23

S = 3

Cost(1,{2,3,4},1)=min{d[1,2]+Cost(2,{3,4},1)=10+25=35,
d[1,3]+Cost(3,{2,4},1)=15+25=40,
d[1,4]+Cost(4,{2,3},1)=20+23=43} =35

The minimum cost path is **35**.



Thank you