Design and Analysis of Algorithm (DAA)

Dynamic Programming

(Multi-Stage Graphs)

Dr. Dayal Kumar Behera

School of Computer Engineering
KIIT Deemed to be University, Bhubaneswar, India

What is Multi-Stage Graph Problem?

A multi-stage graph is a directed, weighted graph that can be divided into stages, where each stage contains multiple nodes and edges connecting nodes between consecutive stages.

Multi-stage graph is a technique used to solve optimization problems involving a multi-stage decision process.

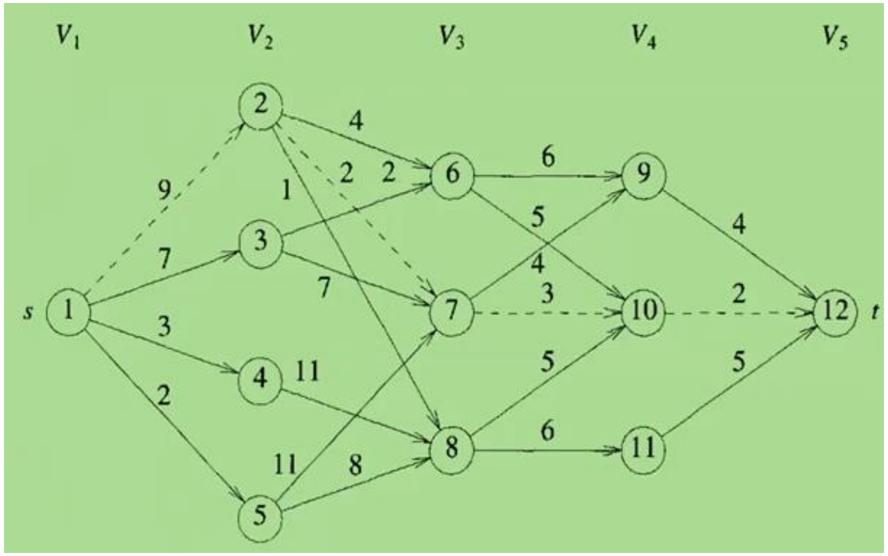
Multi-stage Graph

A multi-stage graph G(V, E) is a directed graph in which the vertices are partitioned into $k \geq 2$ disjoint sets $V_i \quad 1 \leq i < k \quad$ (k : no of stages)

If (u, v) is an edge in E, then

$$u \in V_i \text{ and } v \in V_{i+1}$$

(Vertices on the same stage are not connected by edges)

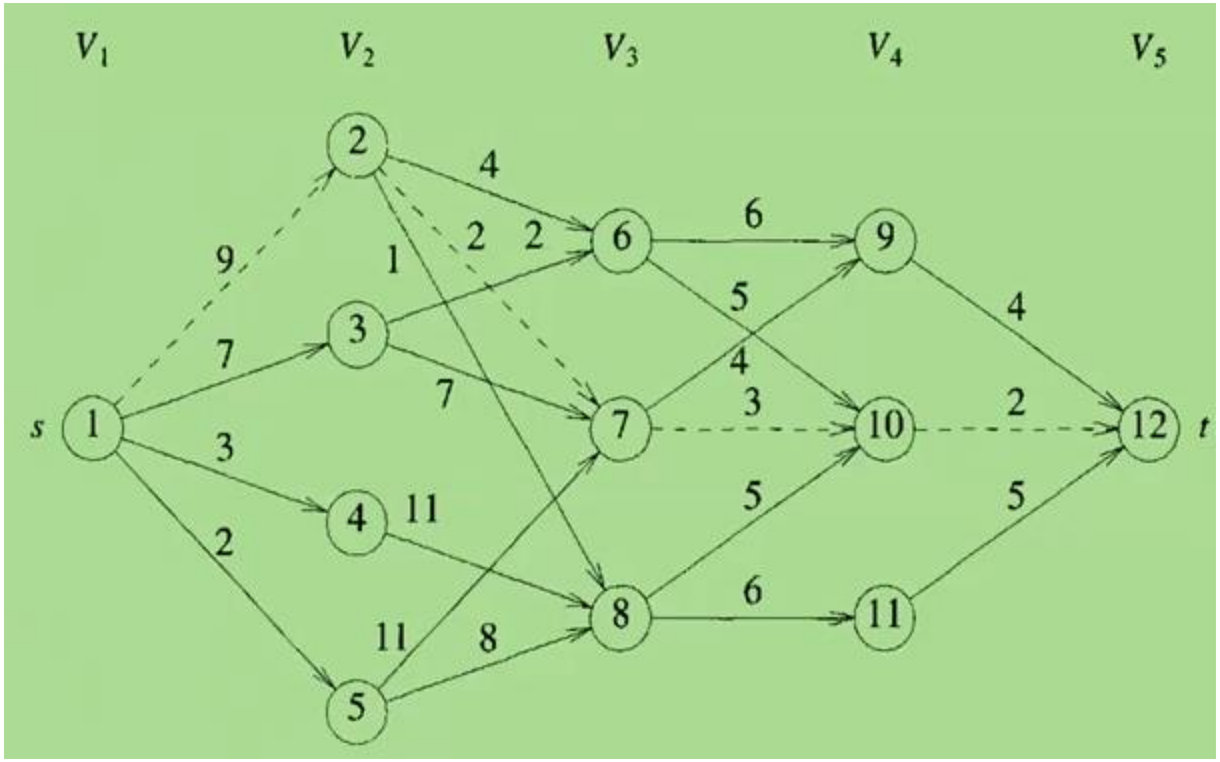


Multi-Stage Graph Problem

Given: A multi-stage graph with stages and weighted edges between consecutive stages.

Goal: find the minimum-cost path from the starting node to the destination node.

Five-stage Graph



Vertex s is the source, and t is the destination/sink.

A minimum-cost path from s to t is indicated by the broken edges.

Approaches

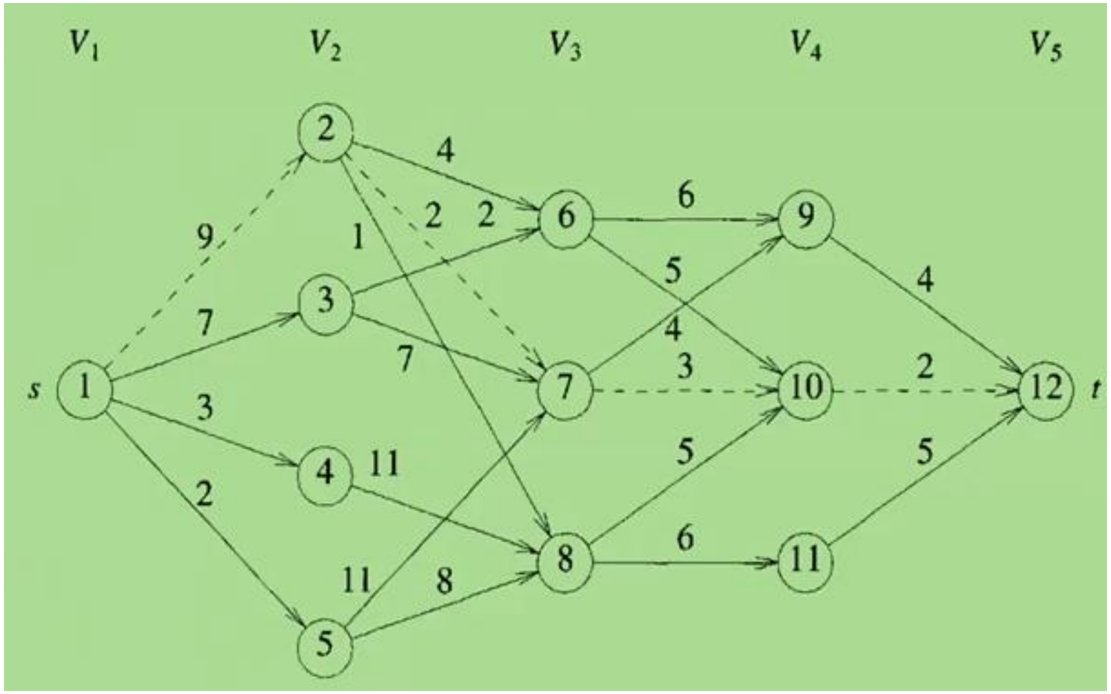
Forward Approach (Backward Reasoning)

Backward Approach (Forward Reasoning)

Forward Approach (Backward Reasoning)

Note# Every s to t path is the result of sequence of k-2 decisions.

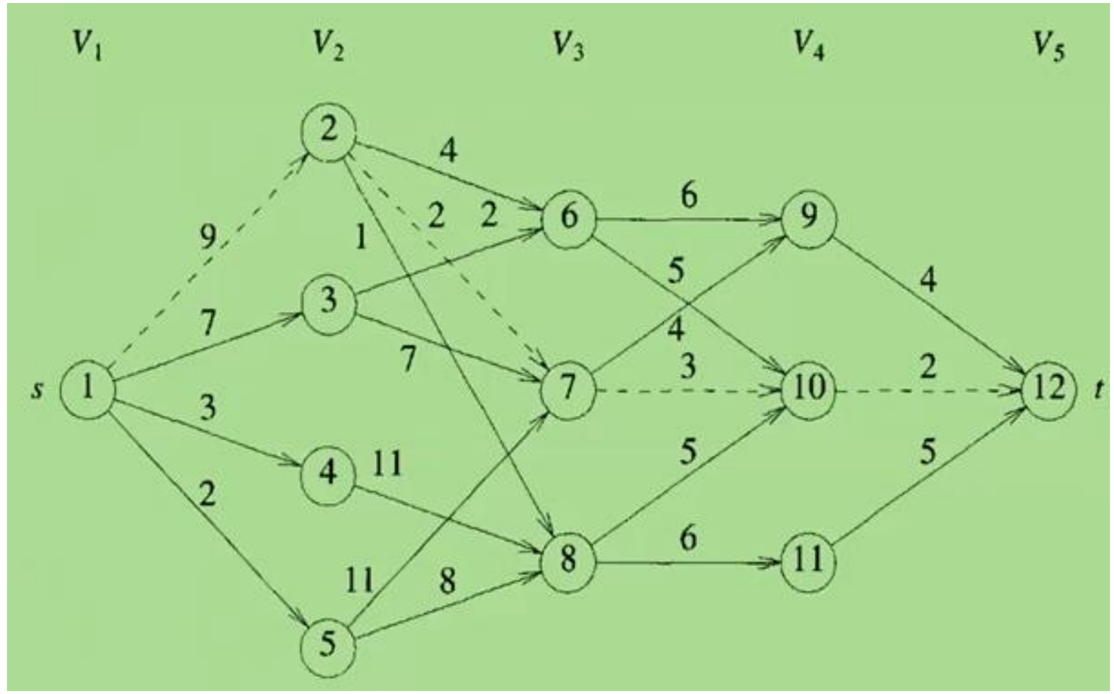
The ith decision determines which vertex in V_{i+1} to be on the path.



Forward Approach (Backward Reasoning)

Let cost(i,j) be the cost of minimum-cost path from vertex j to destination t.

cost(i^{th} stage, j^{th} vertex)



Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost												
d												

Forward Approach (Backward Reasoning)

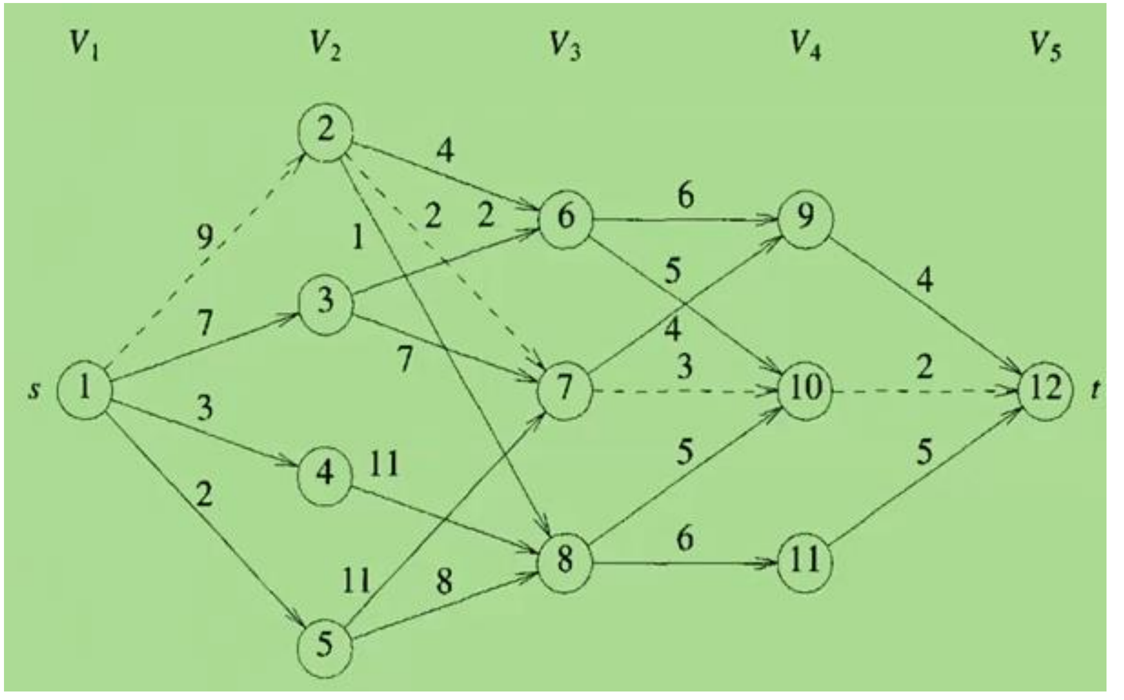
cost(i^{th} stage, j^{th} vertex)

cost(5,12) = 0

cost(4,11) = 5

cost(4,10) = 2

cost(4,9) = 4

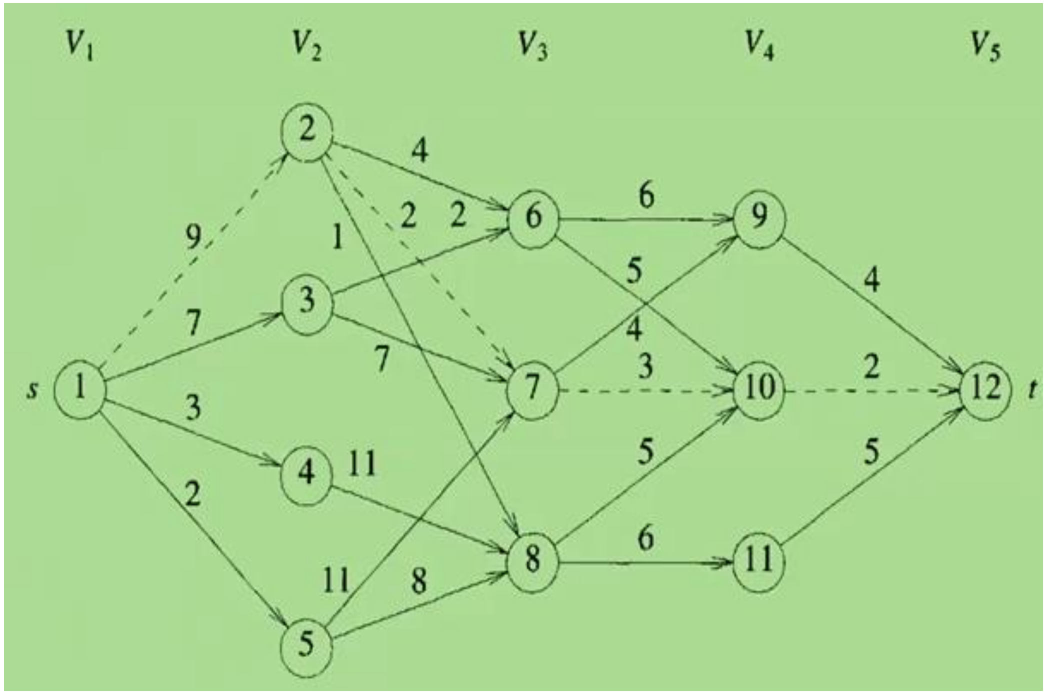


Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost									4	2	5	0
d									12	12	12	12

Forward Approach (Backward Reasoning)

cost(i^{th} stage, j^{th} vertex)

cost(3,8)
= min(6 + 5, 5 + 2)
= min(6 + cost(4,11), 5 + cost(4,10))
= min(11, 7)
= 7



Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost								7	4	2	5	0
d								10	12	12	12	12

Forward Approach (Backward Reasoning)

cost(i^{th} stage, j^{th} vertex)

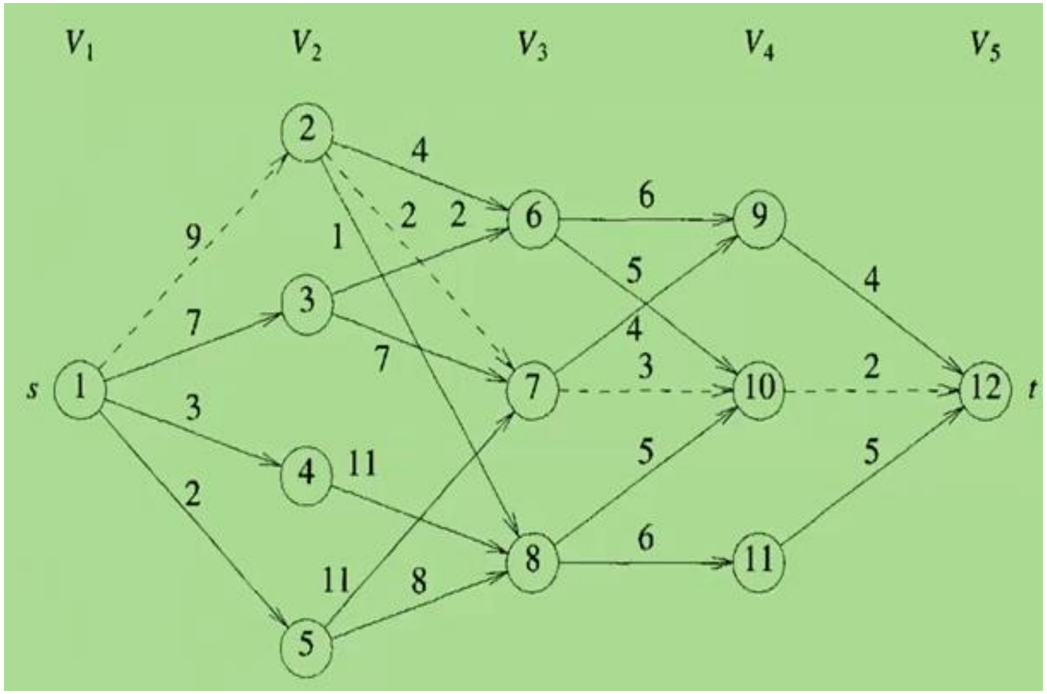
cost(3,7)

= min(3 + cost(4,10), 4 + cost(4,9))

= min(3 + 2, 4 + 4)

= min(5, 8)

= 5



Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost							5	7	4	2	5	0
d							10	10	12	12	12	12

Forward Approach (Backward Reasoning)

cost(ith stage, jth vertex)

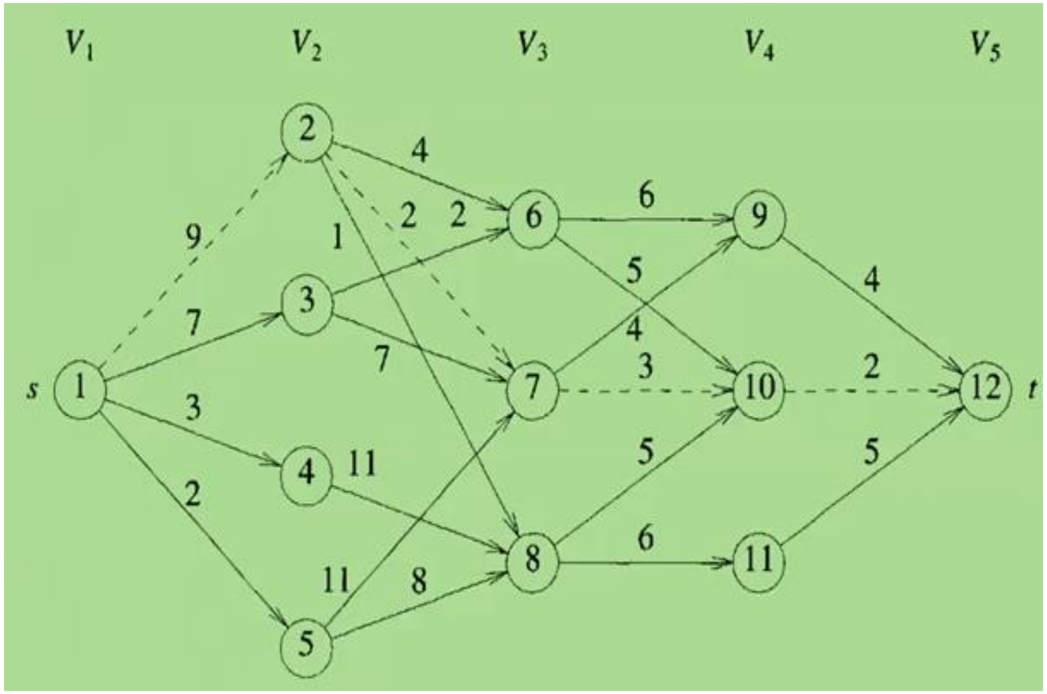
cost(3,6)

= min(6 + cost(4,9), 5 + cost(4,10))

= min(6 + 4, 5 + 2)

= min(10, 7)

= 7

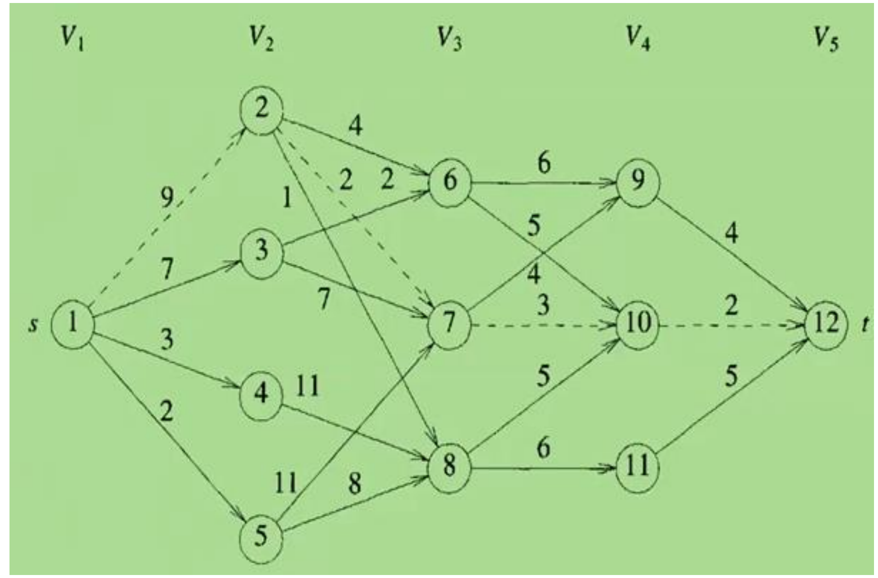


Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost						7	5	7	4	2	5	0
d						10	10	10	12	12	12	12

DP Formula: Forward Approach

Vertex Cost:
cost(i^{th} stage, j^{th} vertex)

Edge Cost:
c(j, r): cost of edge (j,r)



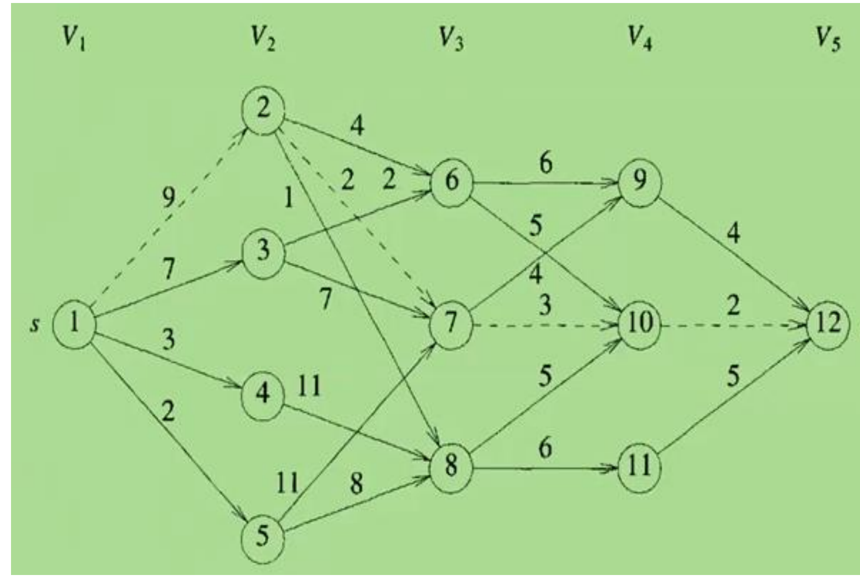
cost(3,6)
= min{6 + cost(4,9), 5 + cost(4,10)}
= **min {c(6, 9) + cost (4, 9), c(6, 10) + cost(4, 10)}**
= min(6 + 4, 5 + 2)
= min(10, 7)
= 7

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost						7	5	7	4	2	5	0
d						10	10	10	12	12	12	12

DP Formula: Forward Approach

Vertex Cost:

cost(ith stage, jth vertex)



Edge Cost:

c(j, r): cost of edge (j,r)

i j

cost(3,6)

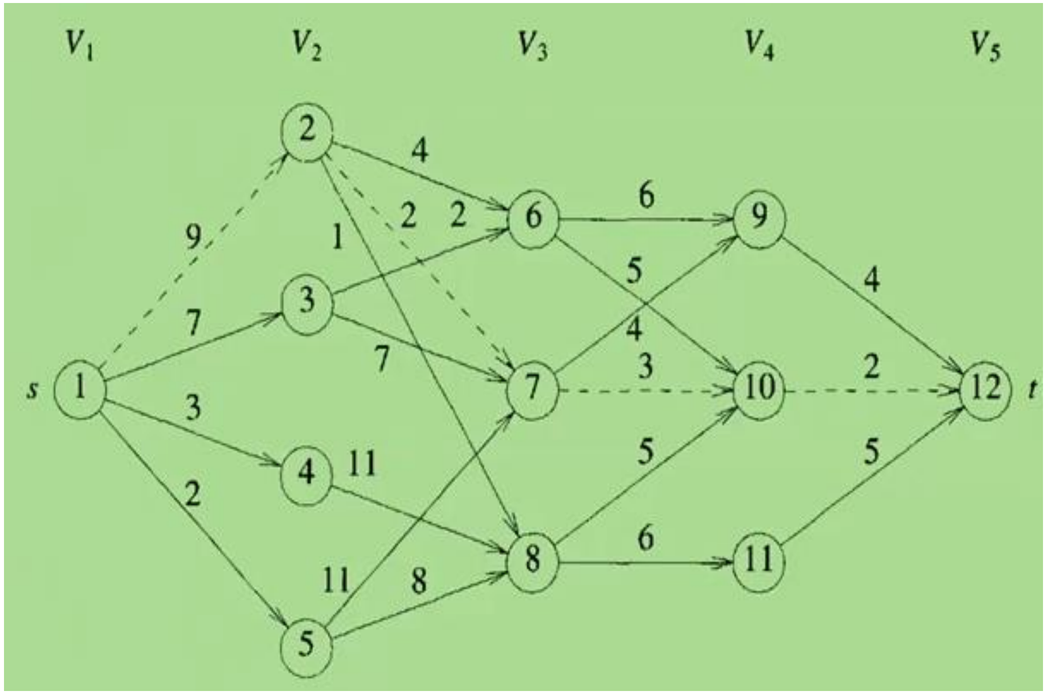
j r i+1 r j r i+1 r

= min {c(6, 9) + cost (4, 9), c(6, 10) + cost(4, 10)}

= min {c(j, r)+ cost(i+1, r)}

$$\boldsymbol{cost(i,j)} = \min_{\substack{\boldsymbol{r} \in \boldsymbol{V_{i+1}} \\ <\boldsymbol{j},\boldsymbol{r}> \in \boldsymbol{E}}} \{\boldsymbol{c(j,r)} + \boldsymbol{cost}\ (\boldsymbol{i}+\boldsymbol{1},\boldsymbol{r})\}$$

Forward Approach (Backward Reasoning)



cost(i^{th} stage, j^{th} vertex)

cost(2,2)

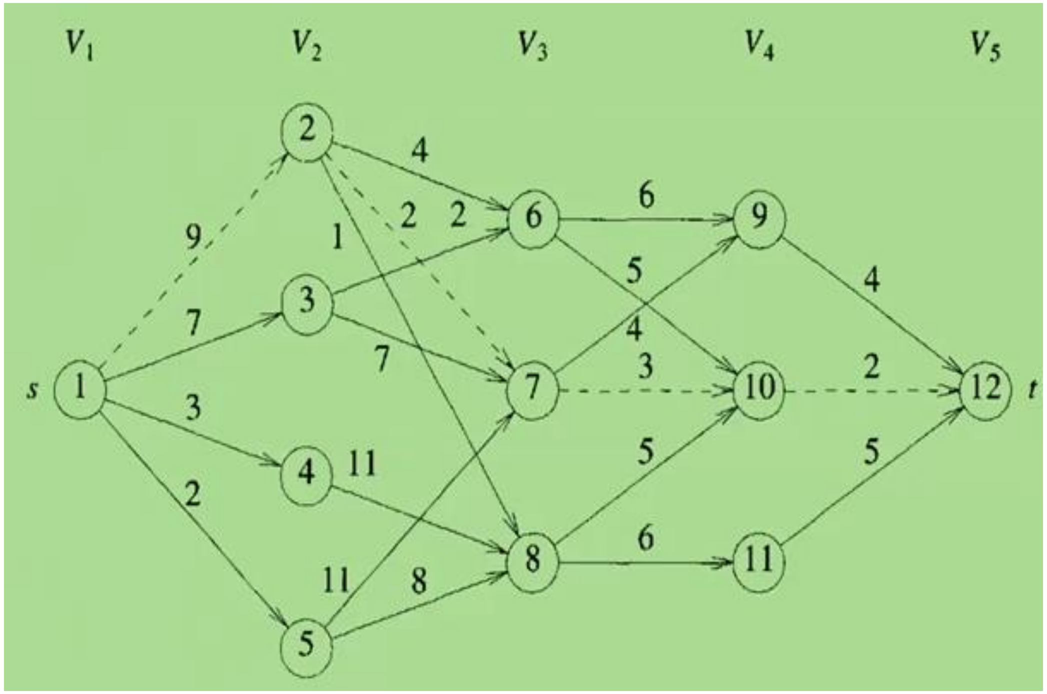
= min(4 + cost(3,6), 2 + cost(3,7), 1+ cost(3,8))

= min(6 + 7, 2 + 5, 1+7)

= 7

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost		7				7	5	7	4	2	5	0
d		7				10	10	10	12	12	12	12

Forward Approach (Backward Reasoning)

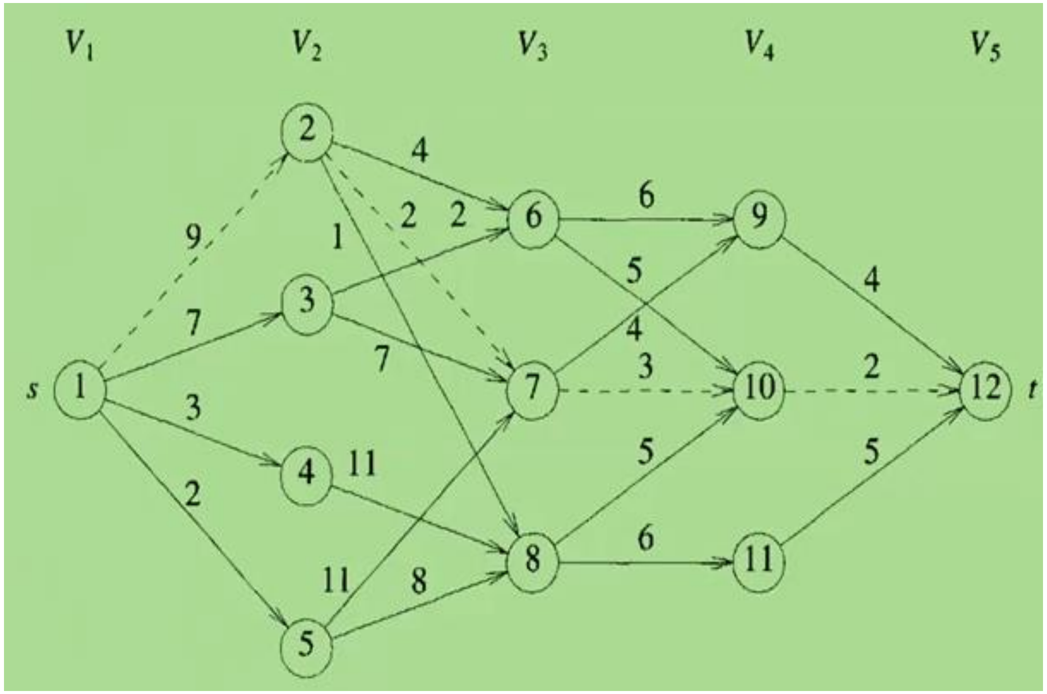


cost(ith stage, jth vertex)

cost(2,3)
= 9

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost		7	9			7	5	7	4	2	5	0
d		7	6			10	10	10	12	12	12	12

Forward Approach (Backward Reasoning)

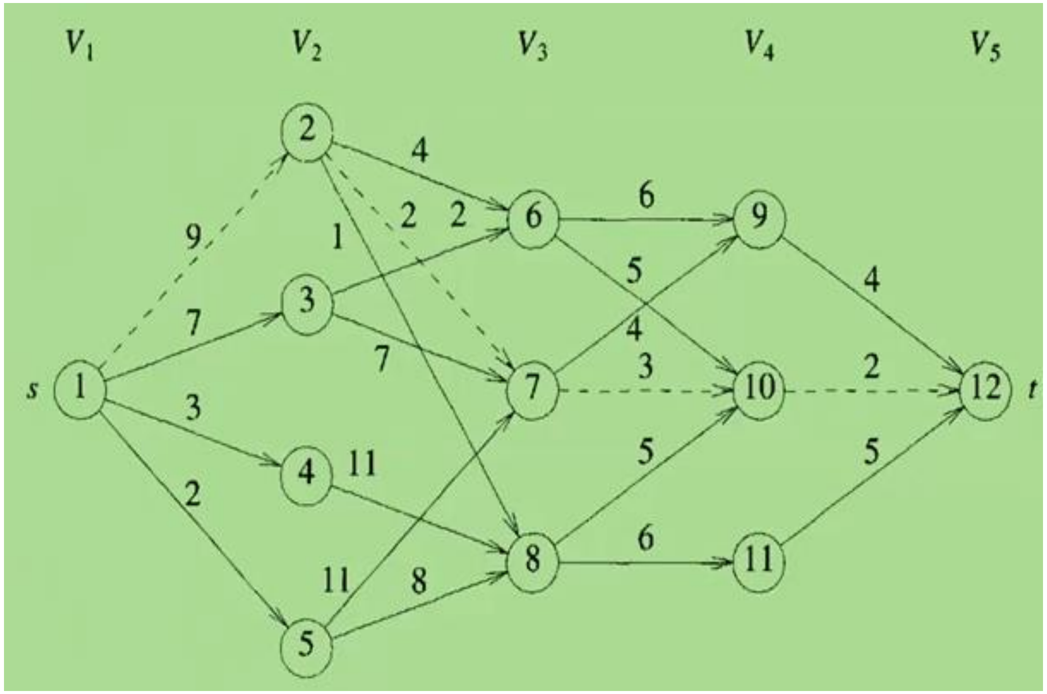


cost(i^{th} stage, j^{th} vertex)

cost(2,4)
= 18

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost		7	9	18		7	5	7	4	2	5	0
d		7	6	8		10	10	10	12	12	12	12

Forward Approach (Backward Reasoning)

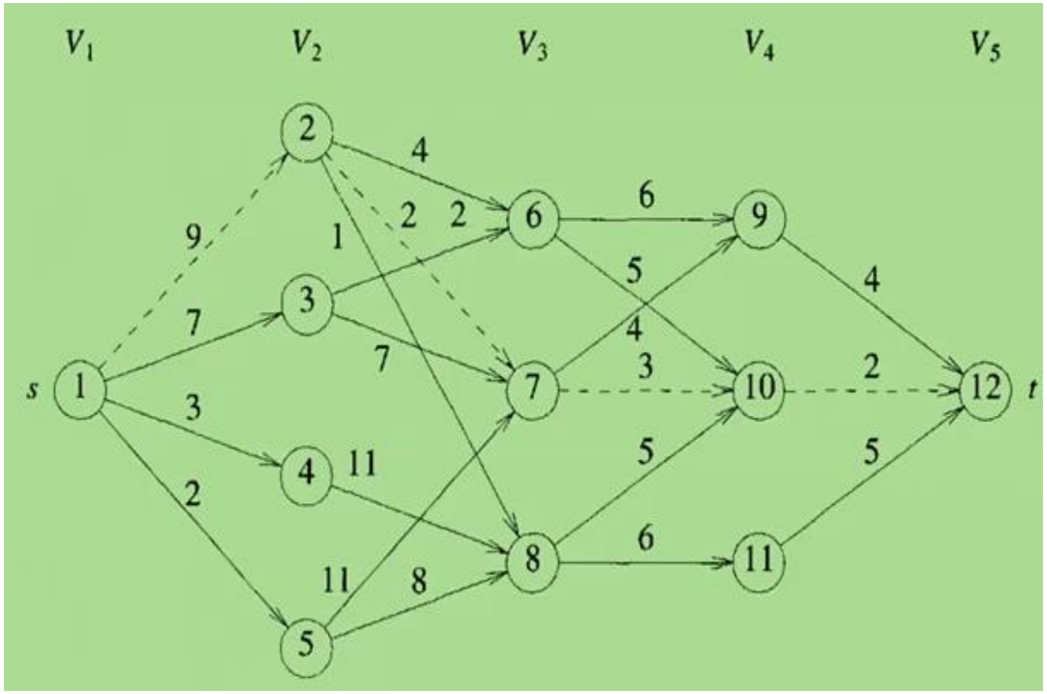


cost(i^{th} stage, j^{th} vertex)

cost(2,5)
= 15

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost		7	9	18	15	7	5	7	4	2	5	0
d		7	6	8	8	10	10	10	12	12	12	12

Forward Approach (Backward Reasoning)



cost(i^{th} stage, j^{th} vertex)

cost(1,1)
= min{9 + cost(2,2), 7 + cost(2,3), 3+ cost(2,4), 2+ cost(2,5)}
= min{9+7, 7+9, 3+18, 2+15}
= 16

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

Forward Approach (Backward Reasoning)

```
Algorithm FGraph(G, k, n, p)
// The input is a k-stage graph G = (V, E) with n vertices
// indexed in order of stages. E is a set of edges and c[i, j]
// is the cost of ⟨i, j⟩. p[1 : k] is a minimum-cost path.
{
    cost[n] := 0.0;
    for j := n − 1 to 1 step −1 do
    { // Compute cost[j].
        Let r be a vertex such that ⟨j, r⟩ is an edge
        of G and c[j, r] + cost[r] is minimum;
        cost[j] := c[j, r] + cost[r];
        d[j] := r;
    }
    // Find a minimum-cost path.
    p[1] := 1; p[k] := n;
    for j := 2 to k − 1 do p[j] := d[p[j − 1]];
}
```

Stage	1	2	3	4	5
P	1				12

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

Forward Approach (Backward Reasoning)

```
Algorithm FGraph(G, k, n, p)
// The input is a k-stage graph G = (V, E) with n vertices
// indexed in order of stages. E is a set of edges and c[i, j]
// is the cost of ⟨i, j⟩. p[1 : k] is a minimum-cost path.
{
    cost[n] := 0.0;
    for j := n − 1 to 1 step −1 do
    { // Compute cost[j].
        Let r be a vertex such that ⟨j, r⟩ is an edge
        of G and c[j, r] + cost[r] is minimum;
        cost[j] := c[j, r] + cost[r];
        d[j] := r;
    }
    // Find a minimum-cost path.
    p[1] := 1; p[k] := n;
    for j := 2 to k − 1 do p[j] := d[p[j − 1]];
}
```

$\Theta\,(\,|V| + \,|E|)$

$\Theta\,(k)$

Time complexity proportional to the degree of vertex j

Overall:

$\Theta\,(\,|V| + \,|E|)$

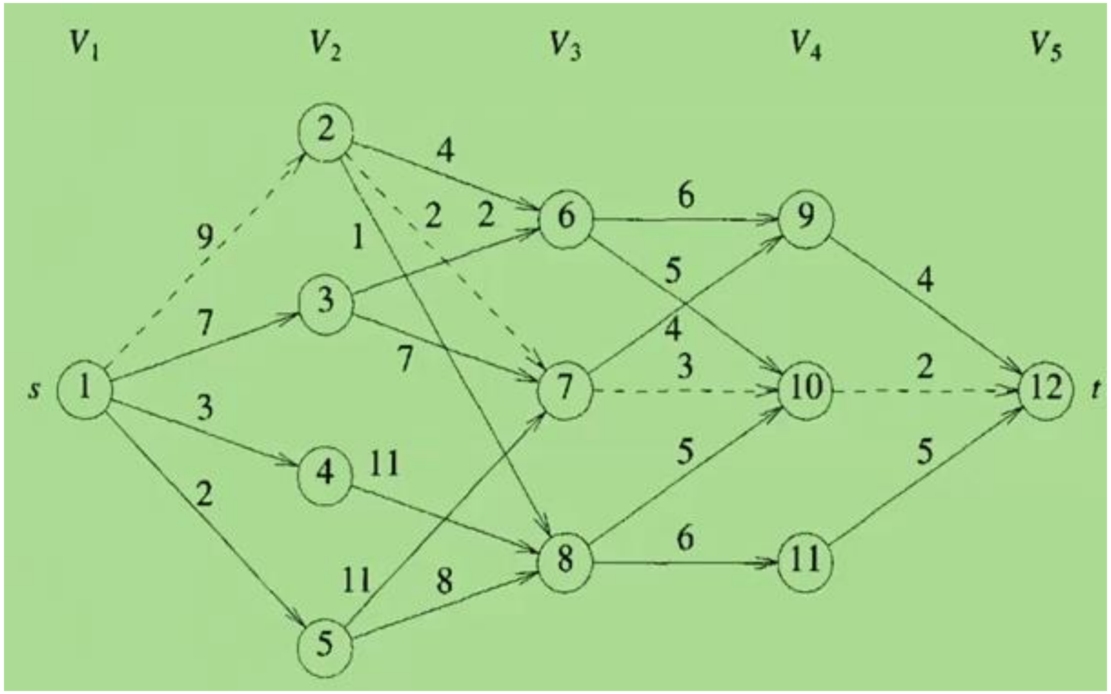
Stage	1	2	3	4	5
P	1	2	7	10	12

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	16	7	9	18	15	7	5	7	4	2	5	0
d	2/3	7	6	8	8	10	10	10	12	12	12	12

Backward Approach (Forward Reasoning)

In forward approach, cost(i,j) was the cost of minimum-cost path from vertex j to destination t.

Here in backward approach bcost(i,j) be the cost of minimum-cost path from source s to vertex j.



bcost(i^{th} stage, j^{th} vertex)

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost												
d												

Backward Approach (Forward Reasoning)

bcost(ith stage, jth vertex)

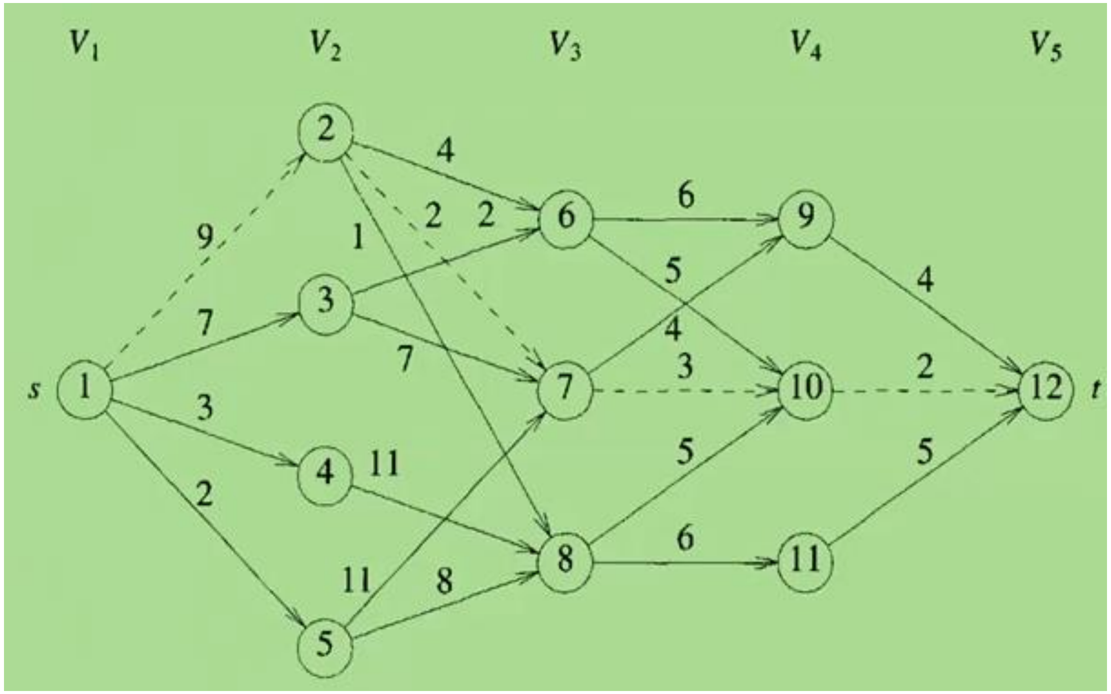
cost(1,1) = 0

cost(2,2) = 9

cost(2,3) = 7

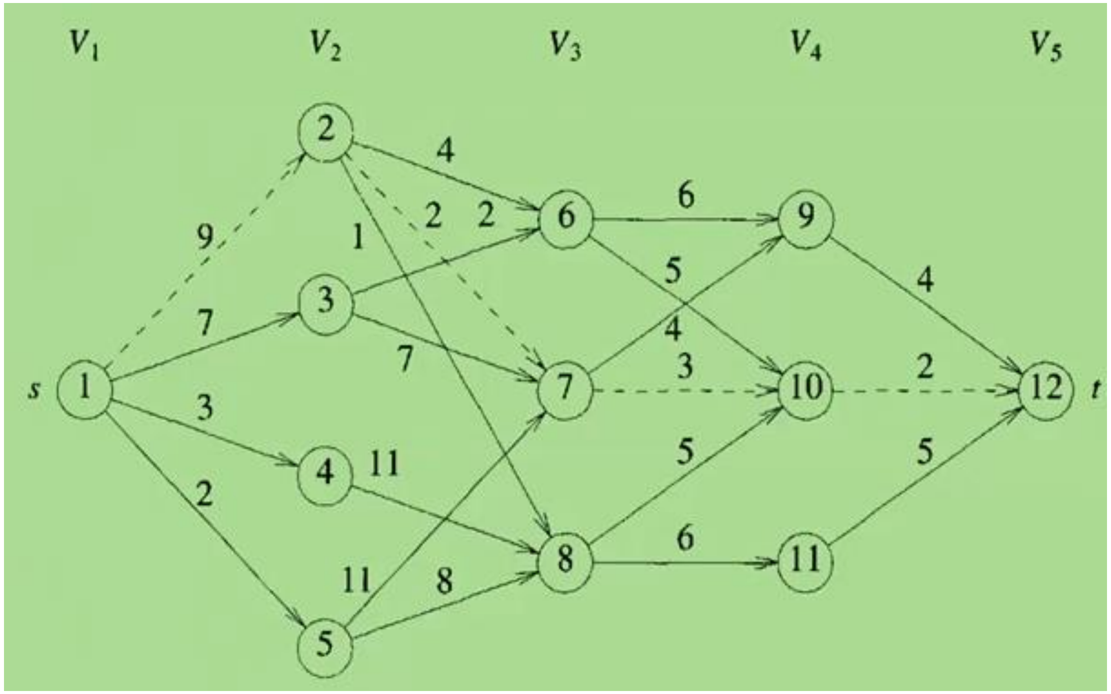
cost(2,4) = 3

cost(2,5) = 2



Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	0	9	7	3	2							
d	1	1	1	1	1							

Backward Approach (Forward Reasoning)



bcost(i^{th} stage, j^{th} vertex)

cost(3,6)
= min(bcost(2,2)+ 4, bcost(2,3)+2)
= min(9+4, 7+2)
= 9

Vertex	1	2	3	4	5	6	7	8	9	10	11	12
cost	0	9	7	3	2	9						
d	1	1	1	1	1	3						

DP Formula: Backward Approach

Vertex Cost:

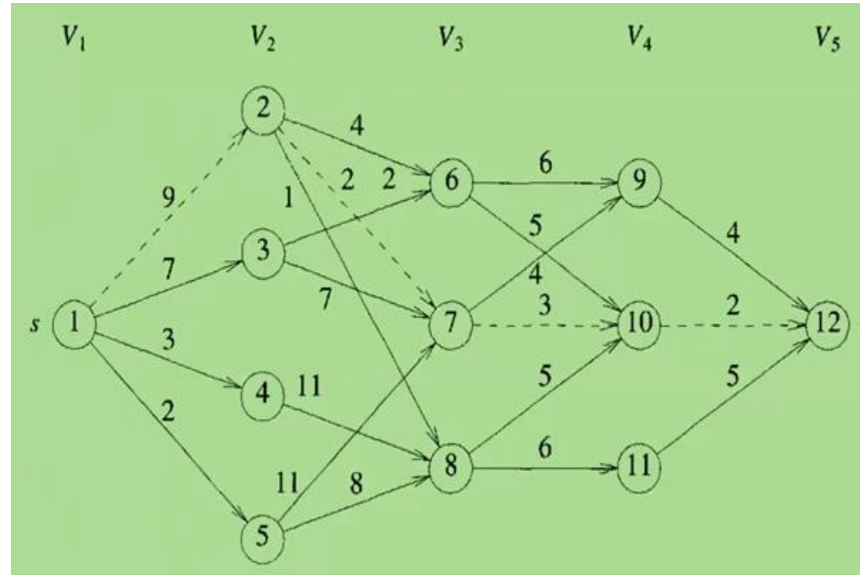
bcost(ith stage, jth vertex)

Edge Cost:

c(r, j): cost of edge (r,j)

i j

cost(3,6)



i-1 r r j i-1 r r j

= min(bcost(2,2)+ c(2,6), bcost(2,3)+c(3,6))

= min {bcost(i-1, r)+ c(r, j)}

$$\boldsymbol{cost(i,j) = \min_{\substack{r \in V_{i-1} \\ <r,j> \in E}} \{bcost(i-1,r) + c\,(r,j)\}}$$

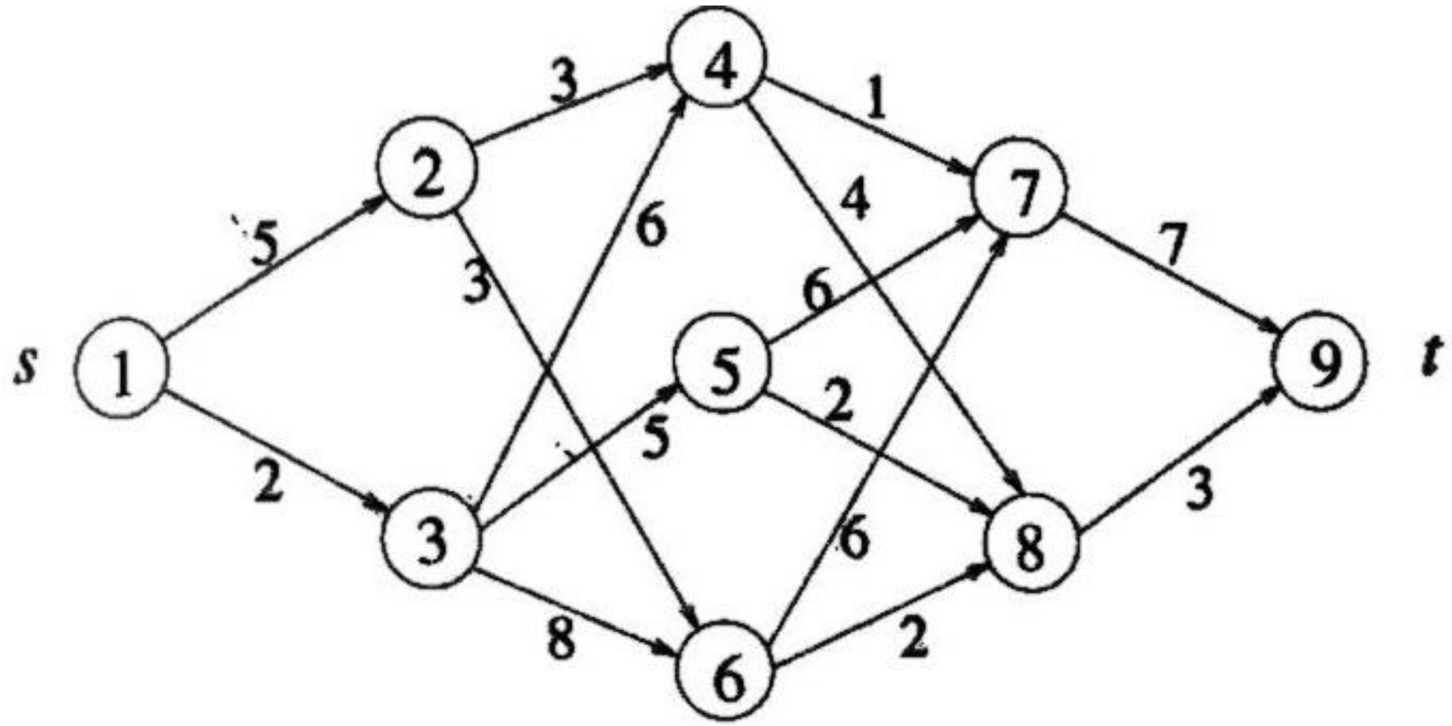
DP Formula: Backward Approach

$$
\begin{aligned}
bcost(3,7) &= 11 \\
bcost(3,8) &= 10 \\
bcost(4,9) &= 15 \\
bcost(4,10) &= 14 \\
bcost(4,11) &= 16 \\
bcost(5,12) &= 16
\end{aligned}
$$

Backward Approach (Forward Reasoning)

```
Algorithm BGraph(G, k, n, p)
// Same function as FGraph
{
    bcost[1] := 0.0;
    for j := 2 to n do
    { // Compute bcost[j].
        Let r be such that ⟨r, j⟩ is an edge of
        G and bcost[r] + c[r, j] is minimum;
        bcost[j] := bcost[r] + c[r, j];
        d[j] := r;
    }
    // Find a minimum-cost path.
    p[1] := 1; p[k] := n;
    for j := k - 1 to 2 do p[j] := d[p[j + 1]];
}
```

Example 2




Each of your
actions will
have an
impact on your
future.


Once you know
who is walking
with you on your path.
you will never
be afraid.

Thank you