Multistage graph

Dr. Bibhudatta Sahoo

Communication & Computing Group

Department of CSE, NIT Rourkela

Email: bdsahu@nitrkl.ac.in, 9937324437, 2462358

Dynamic Programming

- Forward approach and backward approach:
  - Note that if the recurrence relations are formulated using the forward approach then the relations are solved backwards, i.e. beginning with the last decision
  - On the other hand if the relations are formulated using the backward approach, they are solved forwards.

To solve a problem by using dynamic programming:

- Find out the recurrence relations.
- Represent the problem by a multistage graph.

Multistage graph problem

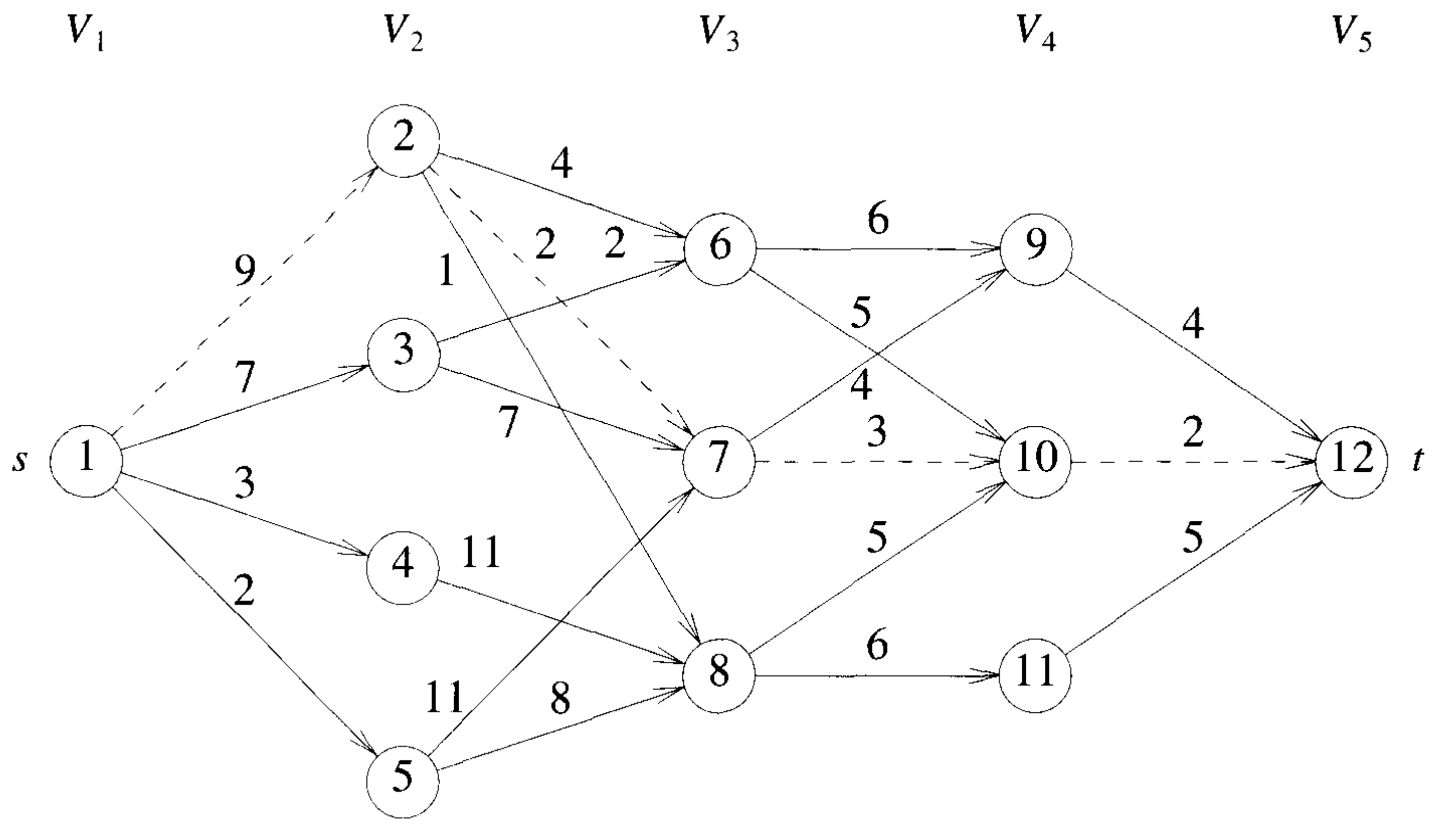
- A **multistage graph** is a directed **graph** in which the nodes can be divided into a set of stages such that all edges are from a stage to next stage only (In other words there is no edge between vertices of same stage and from a vertex of current stage to previous stage).
- A **multistage graph** is a directed graph having a number of multiple stages, where stages element should be connected consecutively. In this multiple stage graph, there is a vertex whose **in degree** is 0 that is known as the **source**.
- The vertex with only one **out degree** is **0** is known as the destination vertex or sink node.
- The **multistage graph problem** is to find a minimum **cost** from a **source** to a **sink**.

Multistage graph problem

- The multistage graph problem is to find a minimum cost from a source to a sink.
- A multistage graph is a directed graph having a number of multiple stages, where stages element should be connected consecutively.
- The one end of the multiple stage graphs is at **i** thus the other reaching end is on **i+1** stage.
- If we denote a graph **G = (V, E)** in which the vertices are partitioned into **k >= 2** disjoints sets, V_i, **1 <= i <=k**. So that, if there is an edge **< u, v >** from **u** to **v** in E, the **u** $\in V_i$ and **v** $\in V_{(i+1)}$, for some **I**, **1 <= i <= K**. And sets V_1 and V_k are such that $|V_1| = |V_k| = 1$.

The "multistage graph problem"

- The "multistage graph problem" is to find the minimum cost path from **s** to **t**.

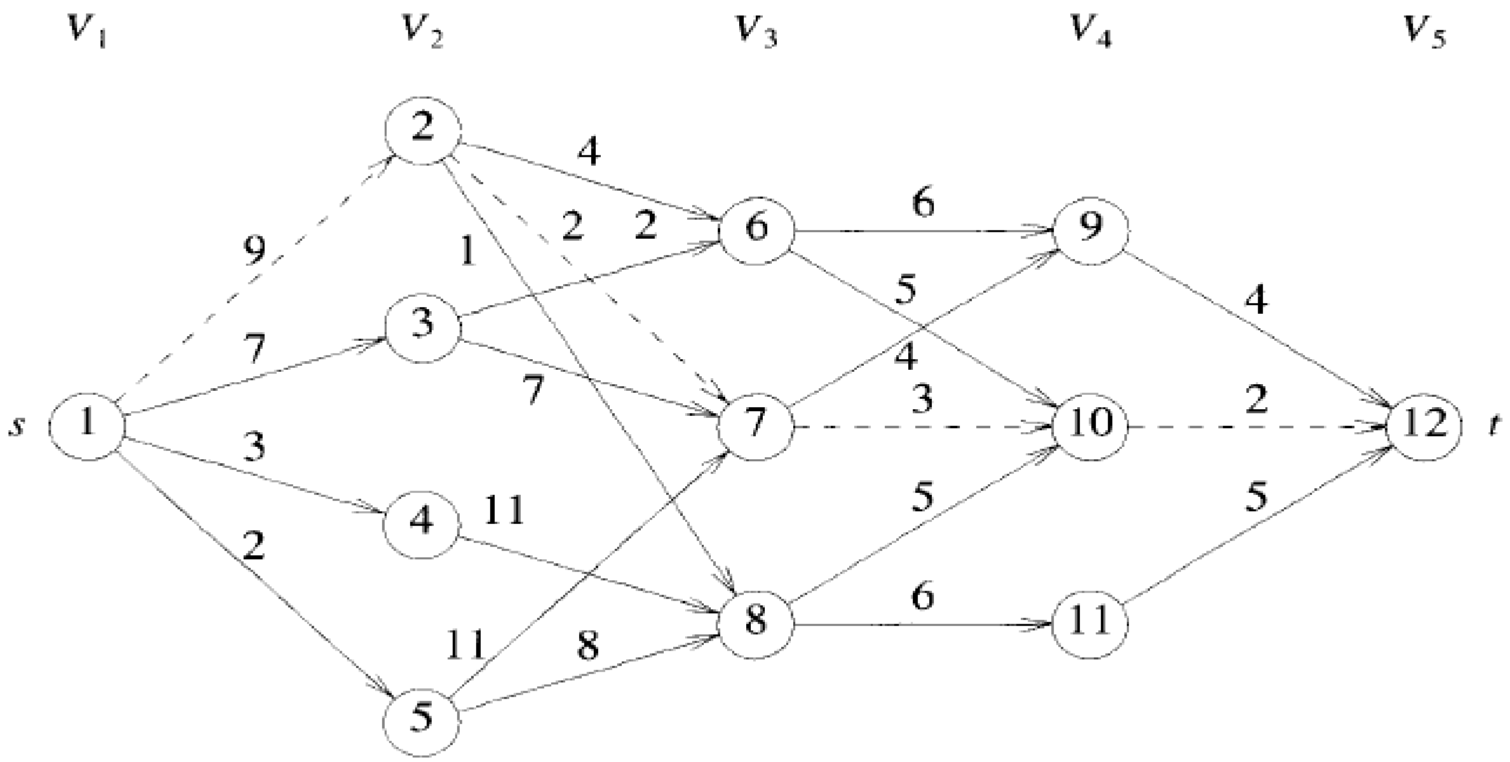


Multi stage graph problem

- A multi stage graph $G = (V, E)$ is a directed graph in which the vertices are partitioned in to $k > 2$ disjoint sets V_i, $1 \leq i \leq k$.
- If$<u,v>$ is an edge in E, then $u \in V_i$ and $v \in Vi+1$, for some $i, 1 \leq i < k$.
- The sets V_1 and V_k are such that $|V_1| = |V_k| = 1$.
- Let s and t, respectively, be the vertices in V_1 and V_k
- The vertex s is the source, and t the sink.
- Let c(i, j)be the cost of **edge(i,j)**.
- The cost of a path from s to t is the sum of the costs of the Edges on the path.

Multi stage graph problem

- The multi stage graph problem is to find a minimum-cost constraints on E, every path from s to t starts in stage1,goes to stage 2, then to stage 3,then to stage , and so on, and eventually terminates in stage k.



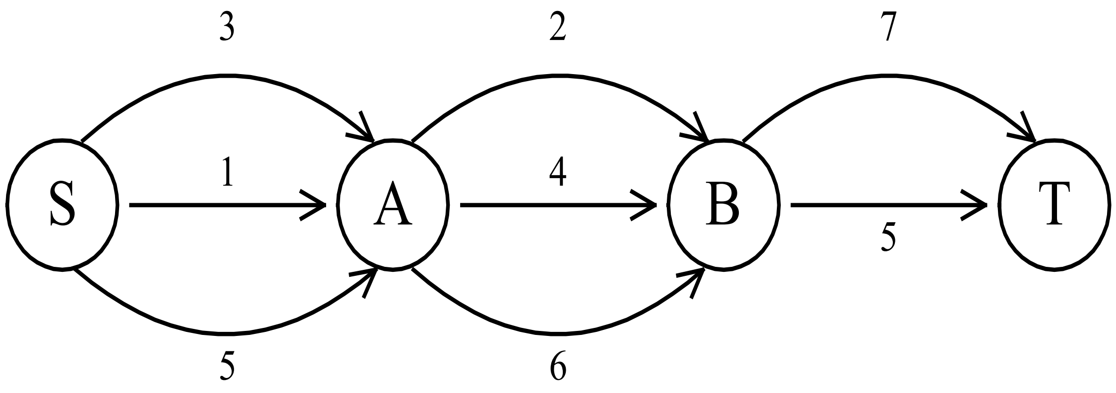
Dynamic Programming

- Forward approach and backward approach:
  - Note that if the recurrence relations are formulated using the forward approach then the relations are solved backwards, i.e. beginning with the last decision
  - On the other hand if the relations are formulated using the backward approach, they are solved forwards.
- To solve a problem by using dynamic programming:
  - Find out the recurrence relations.
  - Represent the problem by a multistage graph.

The shortest path [Transformation to Multistage Graph]

The shortest path

- To find a shortest path in a multi-stage graph



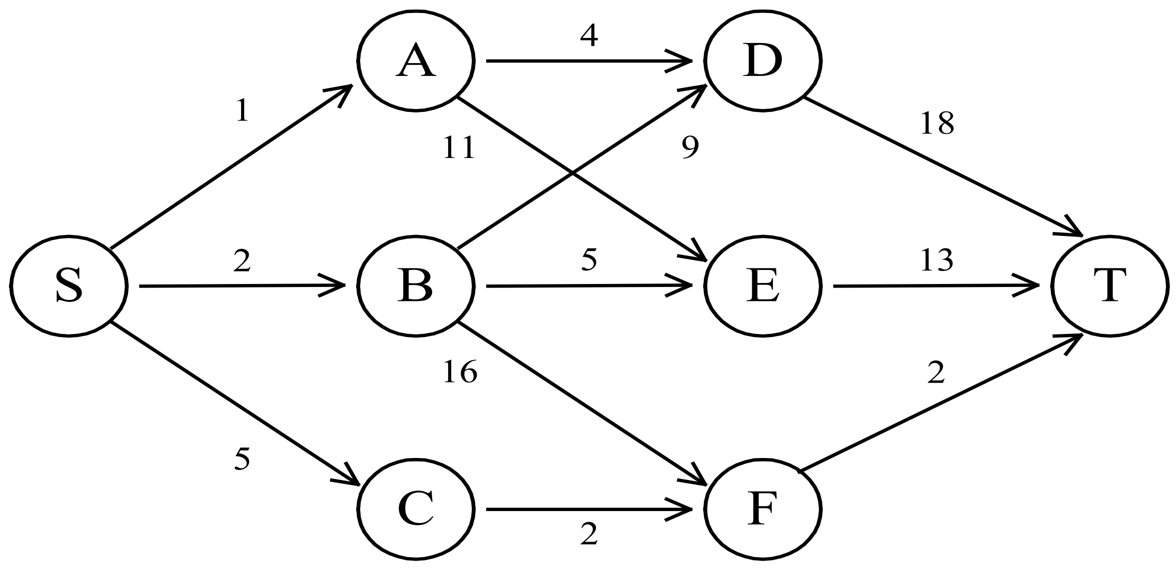
- Apply the **greedy** method :

  the shortest path from S to T :

  $1 + 2 + 5 = 8$

Shortest path in multistage graphs

- e.g.

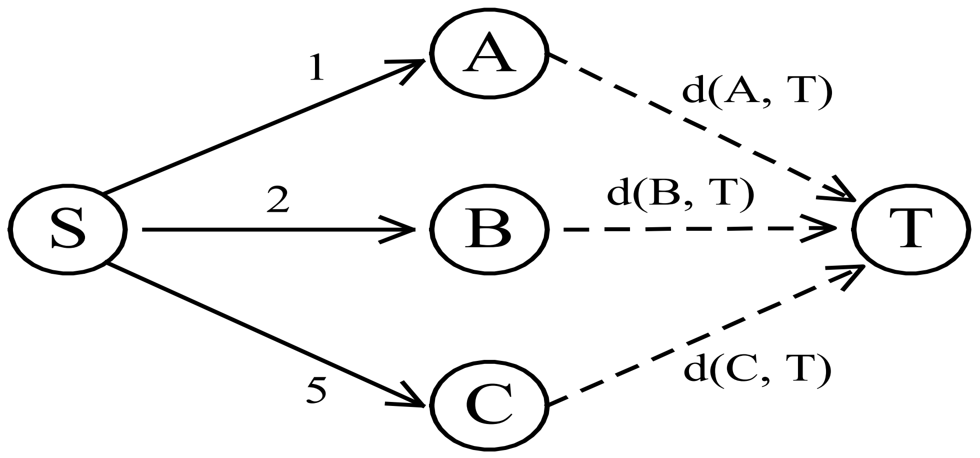


- Greedy method cannot lead to an optimal answer to this case: (S, A, D, T) 1+4+18 = 23.
- Optimal shortest path is:

  (S, C, F, T) 5+2+2 = 9.

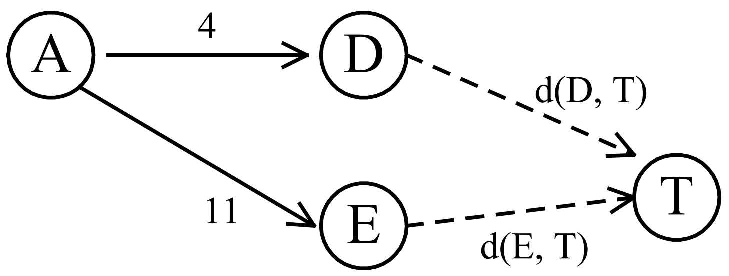
Dynamic Programming Approach

- Dynamic programming approach (forward approach):



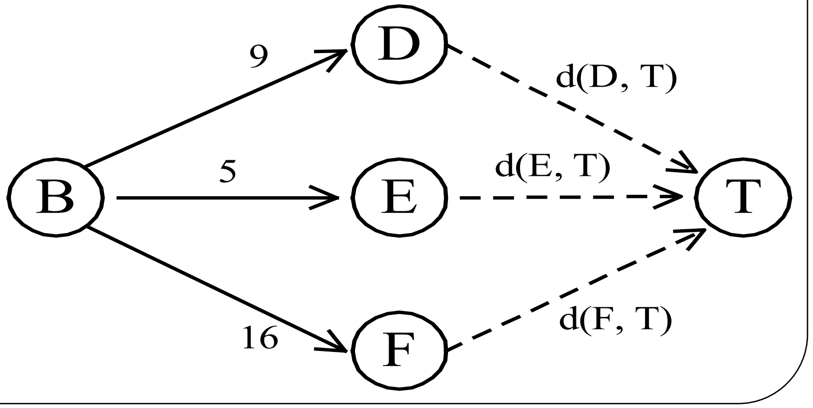
- $d(S, T) = \min\{1+d(A, T), 2+d(B, T), 5+d(C, T)\}$

- $d(A,T) = \min\{4+d(D,T), 11+d(E,T)\}$
  $= \min\{4+18, 11+13\} = 22.$



Dynamic Programming

- $d(B, T) = \min\{9+d(D, T), 5+d(E, T), 16+d(F, T)\}$
  $= \min\{9+18, 5+13, 16+2\} = 18.$
- $d(C, T) = \min\{ 2+d(F, T) \} = 2+2 = 4$
- $d(S, T) = \min\{1+d(A, T), 2+d(B, T), 5+d(C, T)\}$
  $= \min\{1+22, 2+18, 5+4\} = 9.$
- The above way of reasoning is called backward reasoning.



Backward approach (forward reasoning)

- $d(S, A) = 1, d(S, B) = 2, d(S, C) = 5$
- $d(S,D)=\min\{d(S, A)+d(A, D),d(S, B)+d(B, D)\}$

  $= \min\{ 1+4, 2+9 \} = 5$

  $d(S,E)=\min\{d(S, A)+d(A, E),d(S, B)+d(B, E)\}$

  $= \min\{ 1+11, 2+5 \} = 7$

  $d(S,F)=\min\{d(S, A)+d(A, F),d(S, B)+d(B, F)\}$

  $= \min\{ 2+16, 5+2 \} = 7$
- $d(S,T) = \min\{d(S, D)+\mathrm{d}(D, T),d(S,E)+$

  $d(E,T), d(S, F)+d(F, T)\}$

  $= \min\{ 5+18, 7+13, 7+2 \}$

  $= 9$

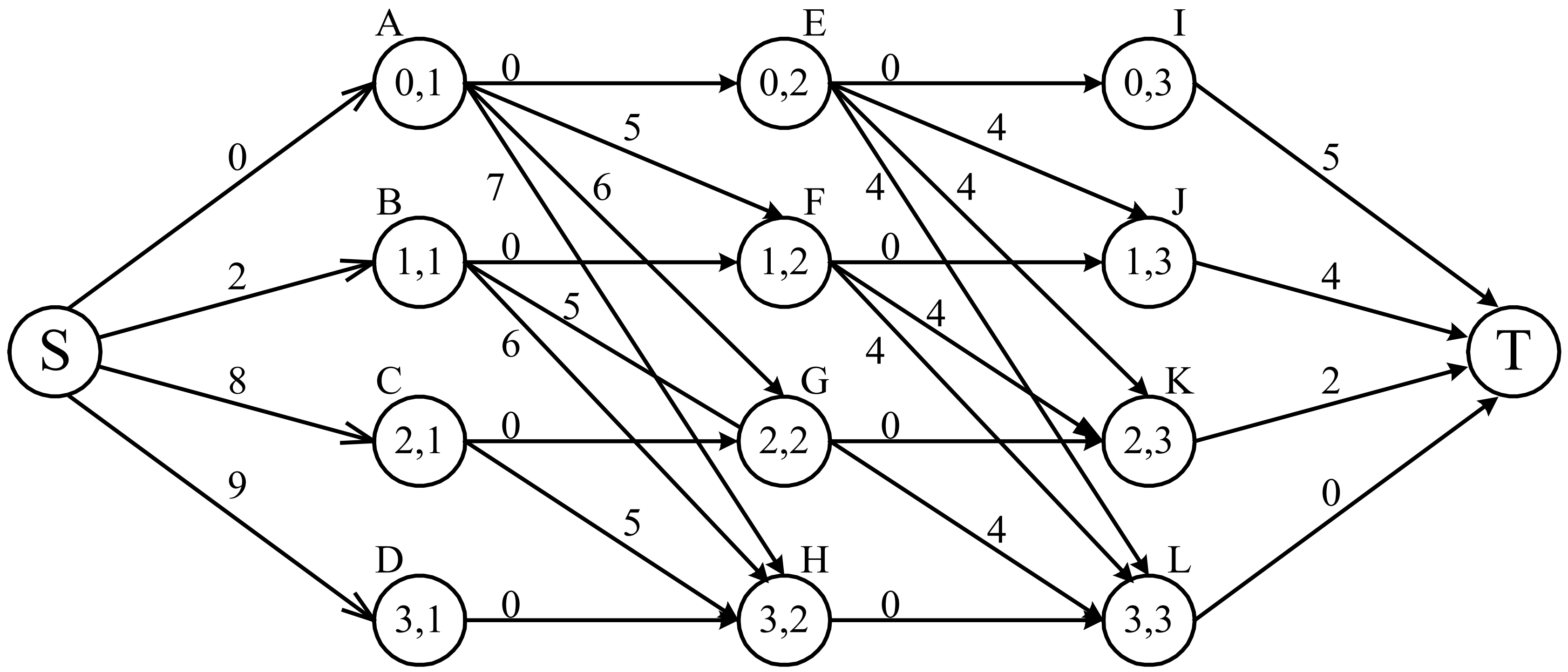
Resource allocation problem

Resource Allocation Problem

- There are *m* resources available to *n* projects
- profit *p*(*i*, *j*) denotes the profits attained through allocating *j* resources to project *i*.
- Goal: Find an allocation that maximizes the total profit.

Resource / Project	1	2	3
1	2	8	9
2	5	6	7
3	4	4	4
4	2	4	5

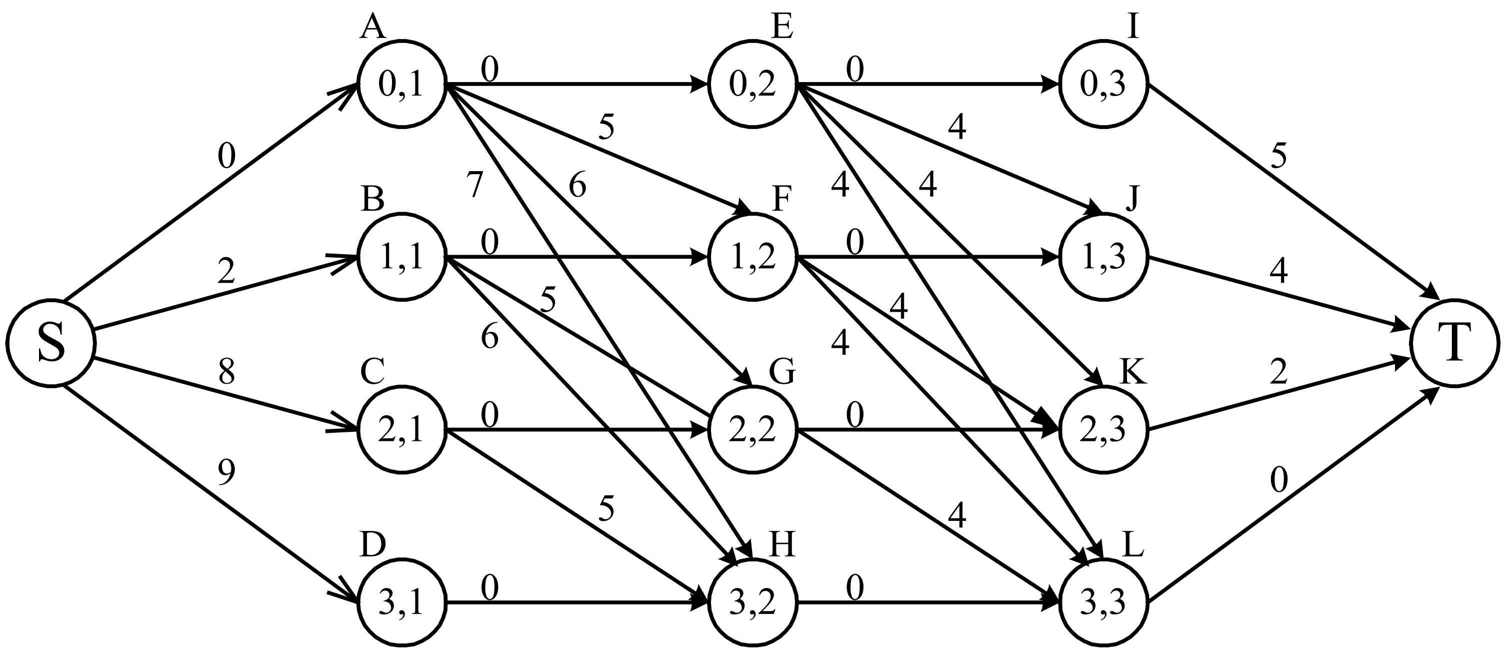
Multistage Graph



- Resource allocation problem can be described as a multistage graph.
- (i, j) : i resources allocated to projects 1, 2, …, j

e.g. node $H = (3, 2)$: 3 resources allocated to projects 1, 2.

Multistage Graph



Find the longest path from *S* to *T* :

(*S*, *C*, *H*, *L*, *T*), 8 + 5 + 0 + 0 = 13

2 resources allocated to project 1.

1 resource allocated to project 2.

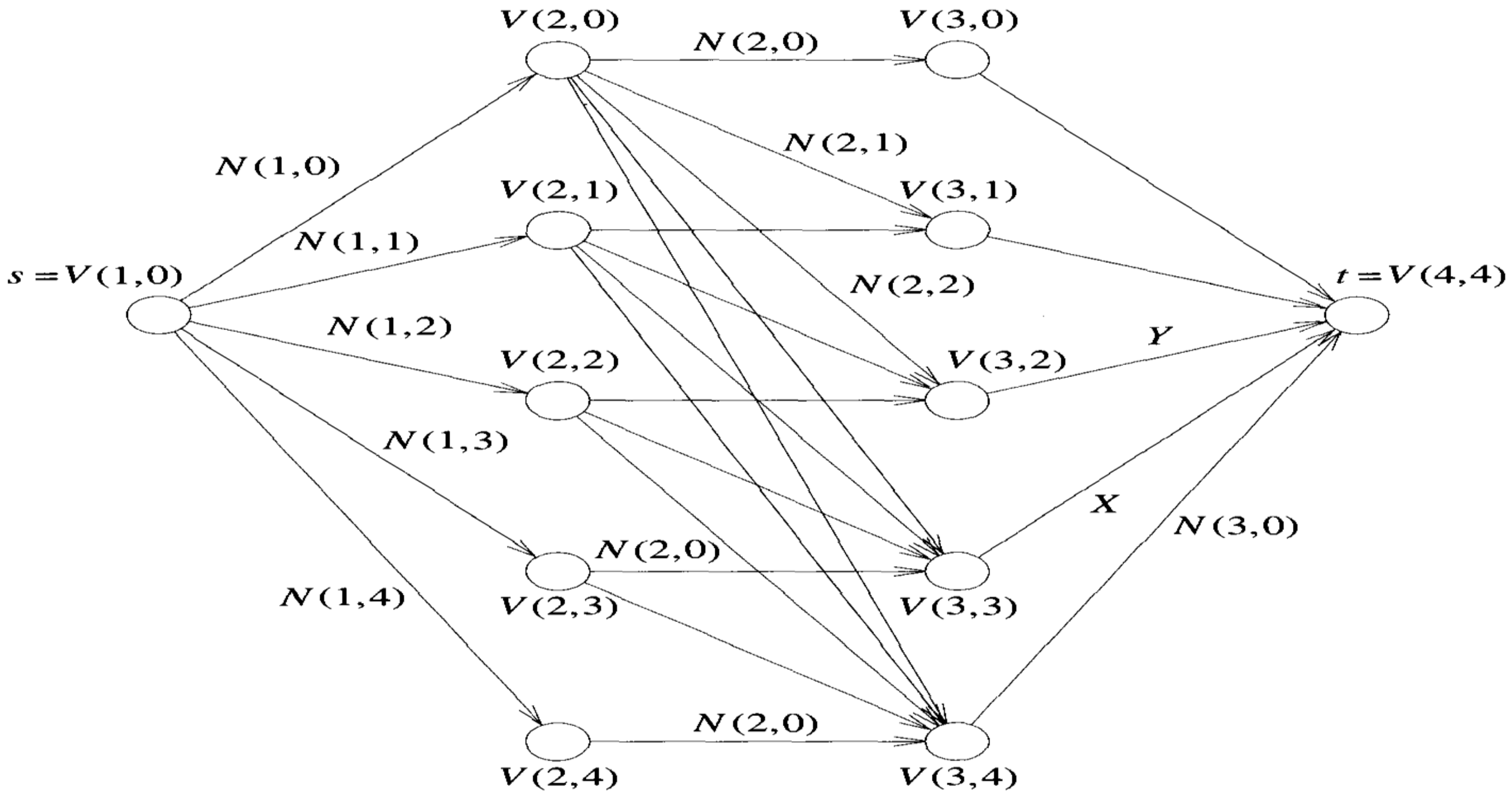
0 resource allocated to projects 3, 4.

Resource allocation problem

- Consider a resource allocation problem in which **n** units of resources are to be allocated to **r** projects.
- If j, $\mathbf{0 \le j < n}$, units of the resource are allocated to project i, then the resulting net profit is **N(i, j)**.
- The problem is to allocate the resource to the r projects in such a way as to **maximize** total net profit.
- This problem can be formulated as an (r+1) stage graph problem as follows.
- Stage i, $1 \le i \le r$, represents project i.
- There are n +1vertices V(i,j), $0 \le j \le n$, associated with stage i, $2 \le i \le r$.

Resource allocation problem

- Stages1 and r + 1each have one vertex, V(l,0) = s and V(r+ 1,n) = t, respectively



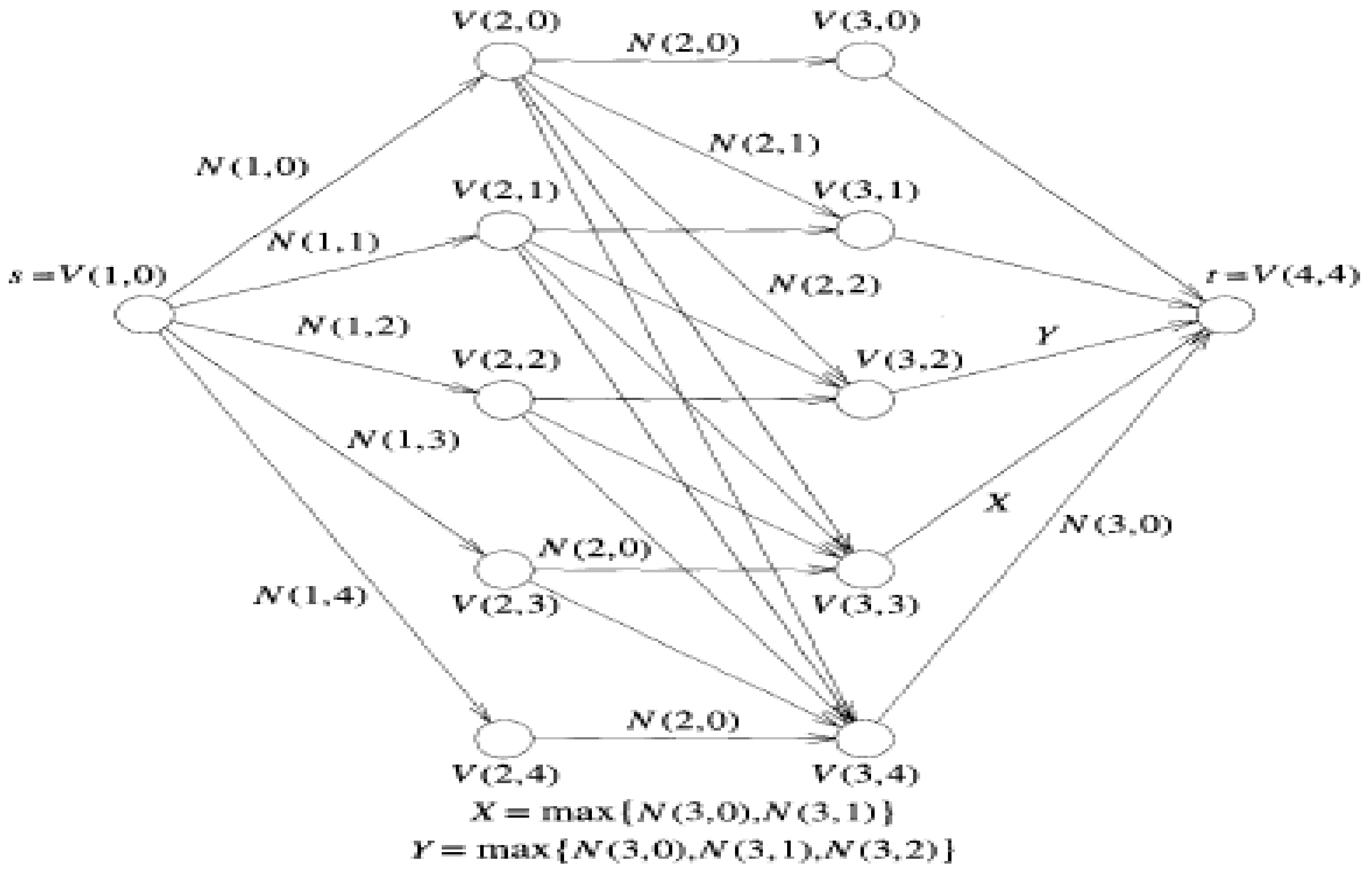
$X = \max\{N(3,0), N(3,1)\}$

$Y = \max\{N(3,0), N(3,1), N(3,2)\}$

Resource allocation problem

- Vertex V(i, j), $2 \leq i < r$, represents the state in which a total of j units of resource have been allocated to project 1, 2,…,i-1.
- The edges in G are of the form <V(i,j), V(i + 1,l)> for all $j \leq l$ and $1 \leq i < r$.
- The edge <V(i,j), V(i + 1,l)> , $j \leq l$ is assigned a weight or cost of N(i, l-j) and corresponds to allocating l-j units of resource to project i, $1 \leq i < r$.
- G has edges of the type <V(r, j)V, (r + l, n)>
- Each edge is assigned a weight $\max_{0 \leq p \leq n-j} \{N(r,p)\}$

Figure5.3 Four-stage graph corresponding to a three-project

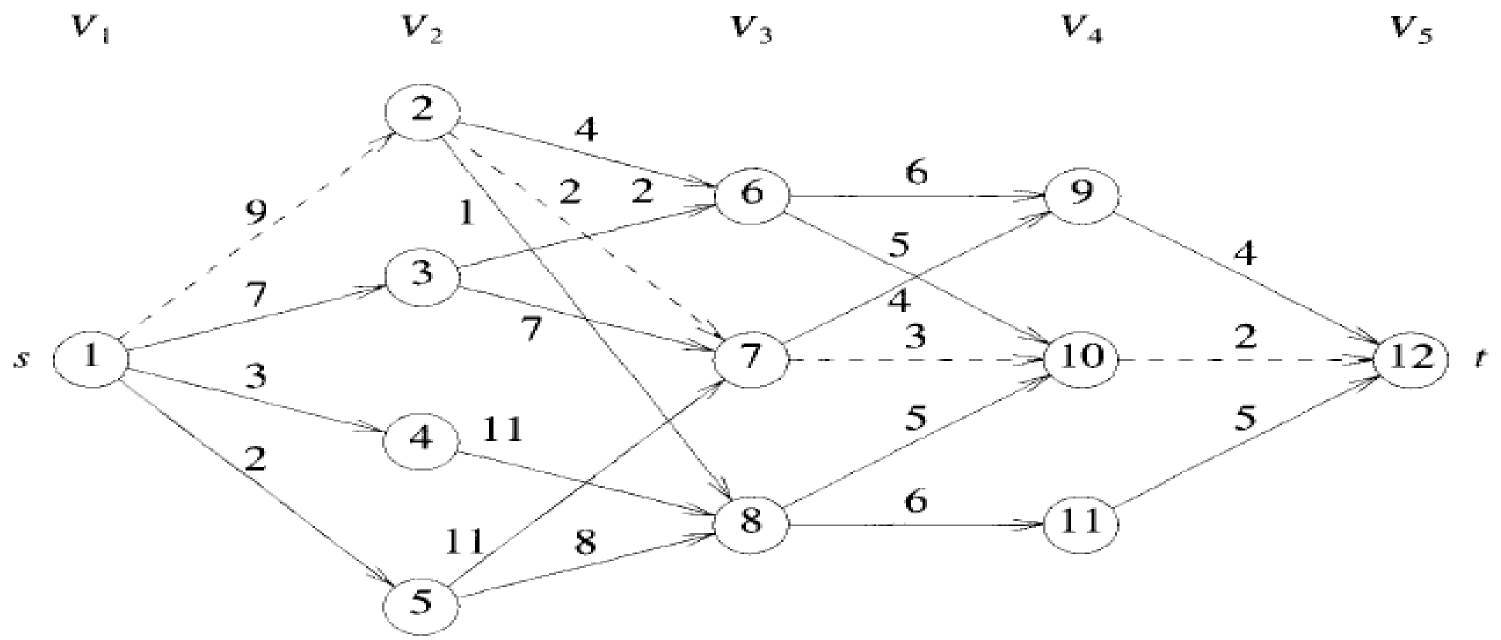


$X = \max\{N(3,0), N(3,1)\}$

$Y = \max\{N(3,0), N(3,1), N(3,2)\}$

An optimal allocation of resources is defined by a maximum cost s to t path

Multistage Graph



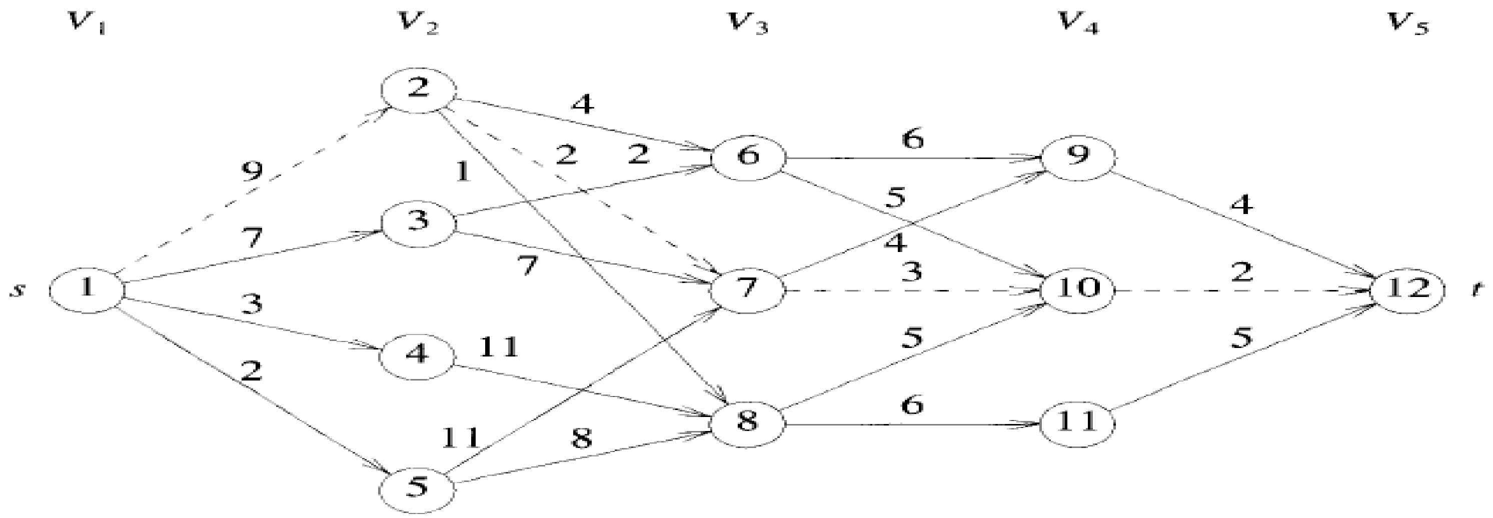
Multistage Graph

- A multistage is a directed graph in which the vertices are partitioned into $k \geq 2$ disjoint sets.
- Multistage graph problem is to determine shortest path from source to destination.
- **Exhaustive search** can guarantee to find an optimal solution.
- However, dynamic programming finds optimal solutions for all scales of sub-problems and finally find an **optimal solution**. That is to say, the global optimum comes from the optimums of all sub-problems.
- Multistage graph problem can be solved by using either **forward** or **backward** approach.
- In **forward approach** we will find the path from **destination to source**, in **backward approach** we will find the path from **source to destination**.

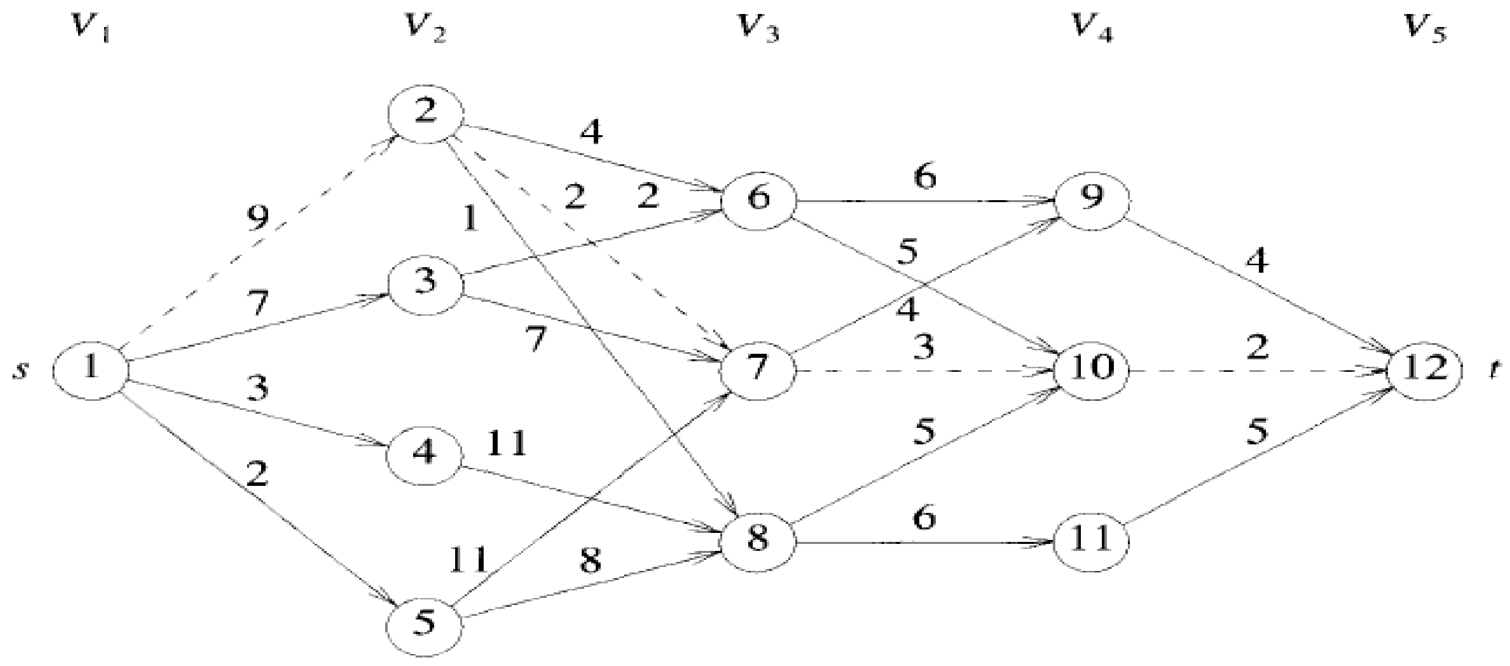
Multistage Graph

A dynamic programming formulation for a k-stage graph problem is obtained by first noticing that every s to t path is the result of a sequence of $k-2$ decisions. The ith decision involves determining which vertex in V_{i+1}, $1 \le i \le k-2$, is to be on the path. It is easy to see that the principle of optimality holds. Let $p(i,j)$ be a minimum-cost path from vertex j in V_i to vertex t. Let $cost(i,j)$ be the cost of this path. Then, using the forward approach, we obtain

$$cost(i,j) = \min_{\substack{l \in V_{i+1} \\ \langle j,l \rangle \in E}} \{c(j,l) + cost(i+1,l)\} \tag{5.5}$$



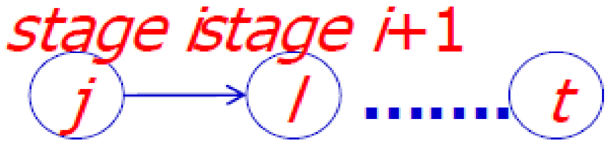
Multistage Graph

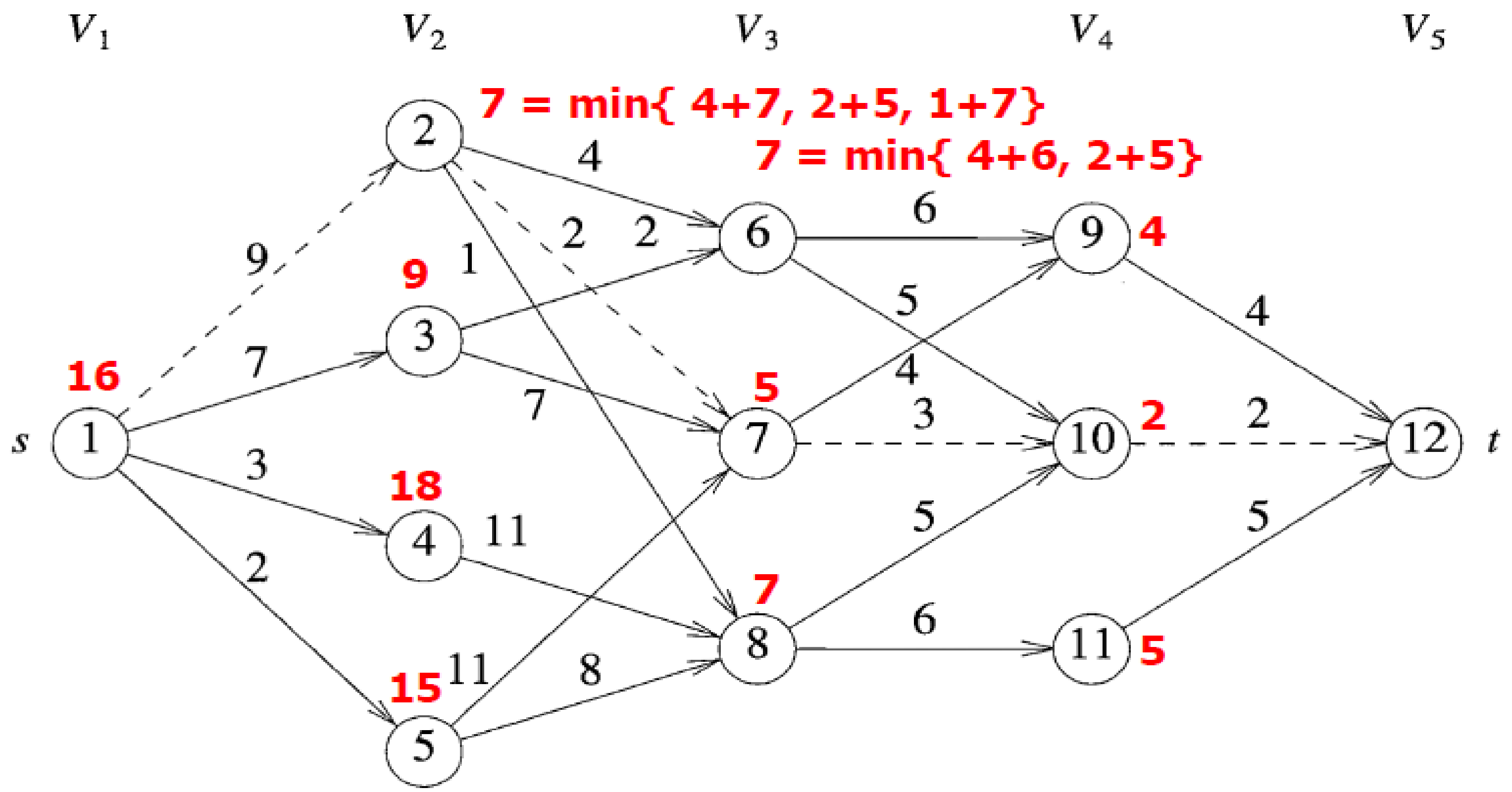


Since, $cost(k-1, j) = c(j, t)$ if $\langle j, t\rangle \in E$ and $cost(k-1, j) = \infty$ if $\langle j, t\rangle \notin E$, (5.5) may be solved for $cost(1, s)$ by first computing $cost(k-2, j)$ for all $j \in V_{k-2}$, then $cost(k-3, j)$ for all $j \in V_{k-3}$, and so on, and finally $cost(1, s)$.

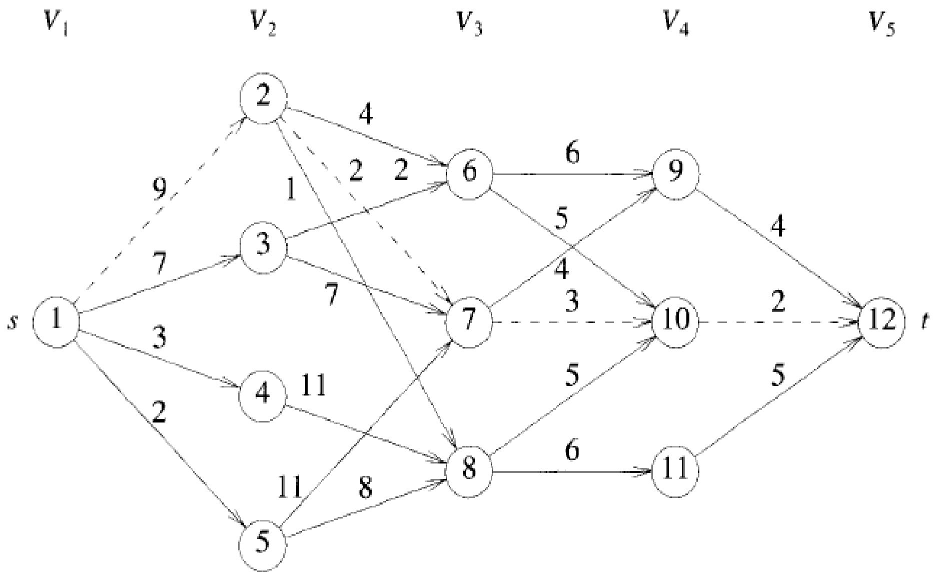
Forward approach

$$cost(i,j) = \min_{\substack{l \in V_{i+1} \\ \langle j,l\rangle \in E}} \{c(j,l) + cost(i+1,l)\}$$





Forward approach



- Cost(5,12)=0;
- Cost(4,9)=4+ Cost(5,12)=4;
- Cost(4,10)=2+ Cost(5,12)=2;
- Cost(4,11)=5+ Cost(5,12)=5;
- Cost(3,6)=min{6+cost(4,9), 5+cost(4,10)}=min{10,7}=7
- Cost(3,7)=min{4+cost(4,9), 3+cost(4,10)}=min{8,5}=5
- Cost(3,8)=min{5+cost(4,10),6+cost(4,11)}=min{7,11}=7
- Cost(2,2)=min{4+cost(3,6),2+cost(3,7),1+cost(3,8)}=min{11,7,8}=7
- Cost(2,3)=min{2+cost(3,6),7+cost(3,7)}=min{9,12}=9
- Cost(2,4)=11+cost(3,8)=18
- Cost(2,5)=min{11+cost(3,7), 8+cost(3,8)}=min{16,15}=15
- Cost(1,1)=min{9+cost(2,2),7+cost(2,3),3+cost(2,4),2+cost(2,5)}
  =min{16,16,21,17}=16

Shortest path 1-2-7-10-12

Cost(i, j)=min { c(j, l) +cost (i+1,l)} $l \in V_{i+1}$, $\langle j,l\rangle \in E$

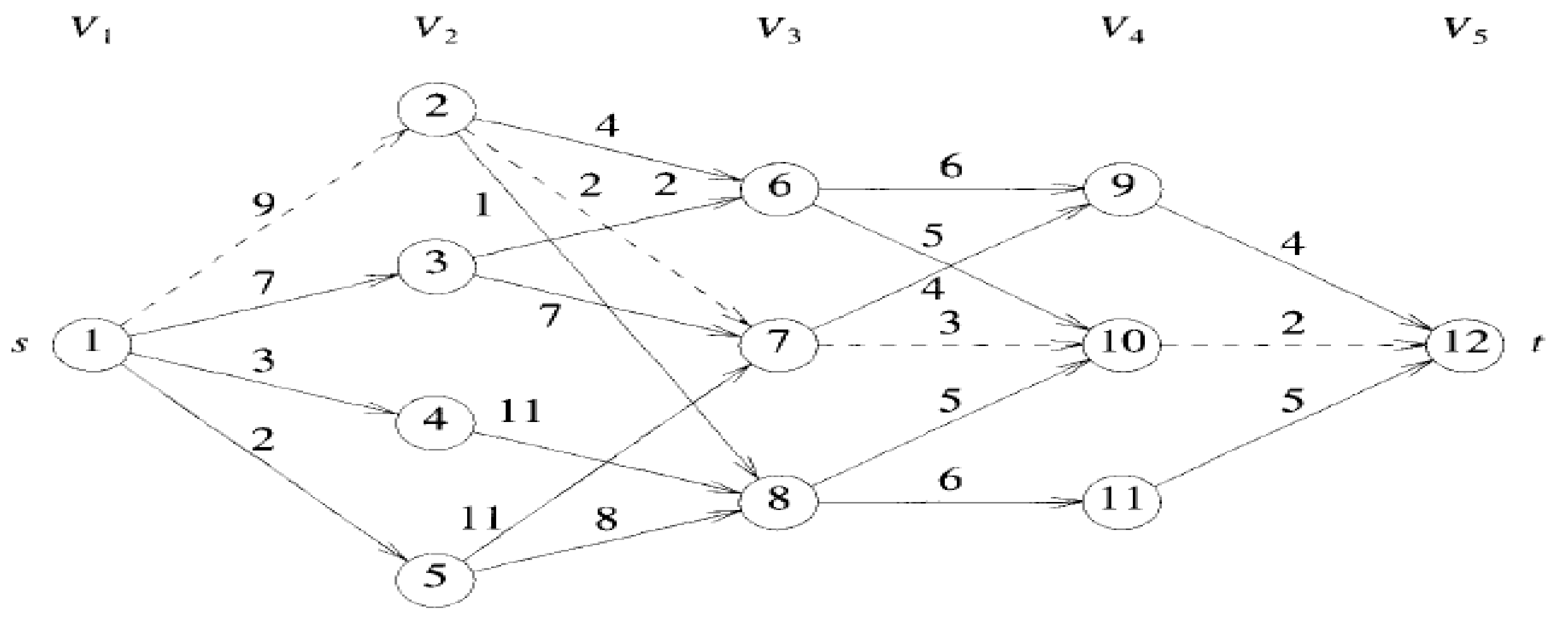
Algorithm 5.1: Forward approach

```
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 $\langle i, j \rangle$. $p[1 : k]$ 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 $\langle j, r \rangle$ 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 $p[j] := d[p[j - 1]]$;
}
```

Forward approach

- The time for the for loop of line 7 is $\Theta(|V| + |E|)$, and the time for the for loop of line 16 is $\Theta(k)$, Hence, the total time is $\Theta(|V| + |E|)$.
- The algorithm also works for the edges crossing more than 1 stage.

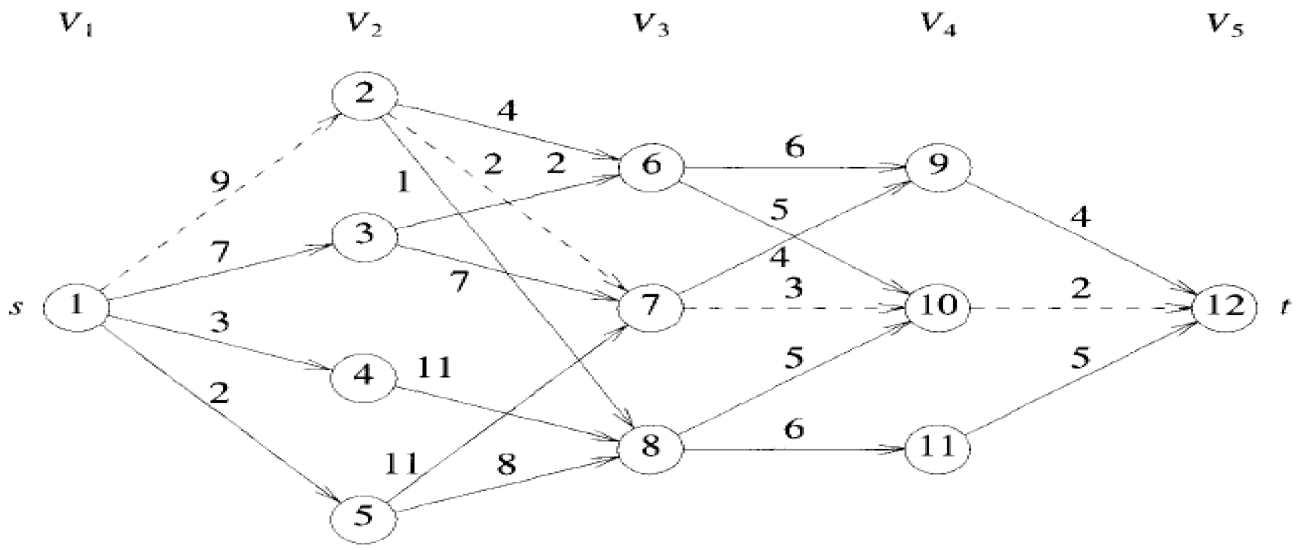
Backward approach



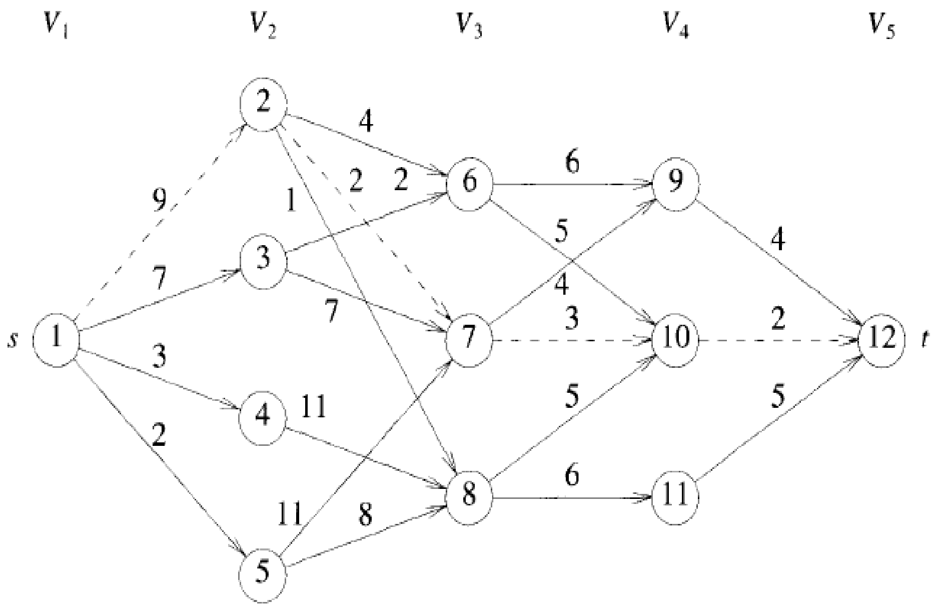
- The multistage graph problem can be solved using backward approach.
- Let bp(i,j) be a minimum-cost path from vertex **s** to vertex **j** in Vi.
- Let bcost(i,j) be cost of bp(i,j).

Backward approach

- Let bcost(i,j) be cost of bp(i,j).
- $bcost(i,j) = \min_{\substack{l \in V_{i-1} \\ <l,j> \in E}} \{bcost(i-1,l) + c(l,j)\}$
- Since bcost(2,j) = c(1,j)if edge <1,j> ∈E and bcost(2,j) = ∞ if edge <1,j> ∉ E, the bcost(I,j) can be computed using the above formulation by first computing bcost for i=3, then i=4 and so on.



Backward approach



- Bcost(1,1)=0;
- Bcost(2,2)=9+bcost(1,1)=9;
- Bcost(2,3)=7++bcost(1,1)=7;
- Bcost(2,4)=3+bcost(1,1)=3;
- Bcost(2,5)=2+bcost(1,1)=2;
- Bcost(3,6)=min{4+bcost(2,2), 2+bcost(2,3)}=min{13,9}=9;
- Bcost(3,7)=11
- Bcost(3,8)=10
- Bcost(4,9)=15
- Bcost(4,10)=14
- Bcost(4,11)=16
- Bcost(5,12)=16

Algorithm 5.2 :Backward approach

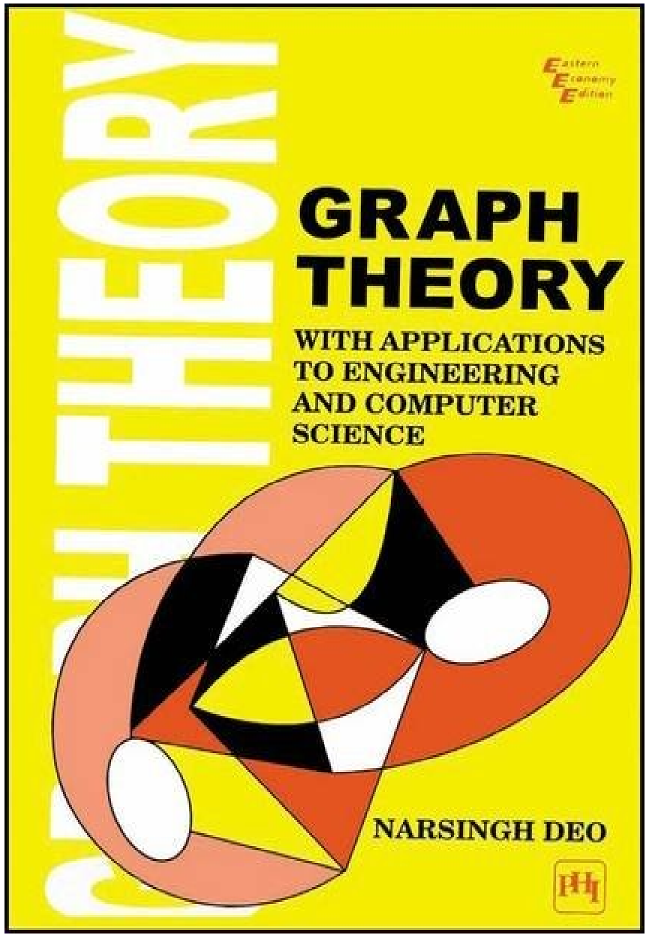
```
Algorithm BGraph($G, k, n, p$)
// Same function as FGraph
{
    $bcost[1] := 0.0$;
    for $j := 2$ to $n$ do
    { // Compute $bcost[j]$.
        Let $r$ be such that $\langle r, j \rangle$ is an edge of
        $G$ and $bcost[r] + c[r, j]$ is minimum;
        $bcost[j] := bcost[r] + c[r, j]$;
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

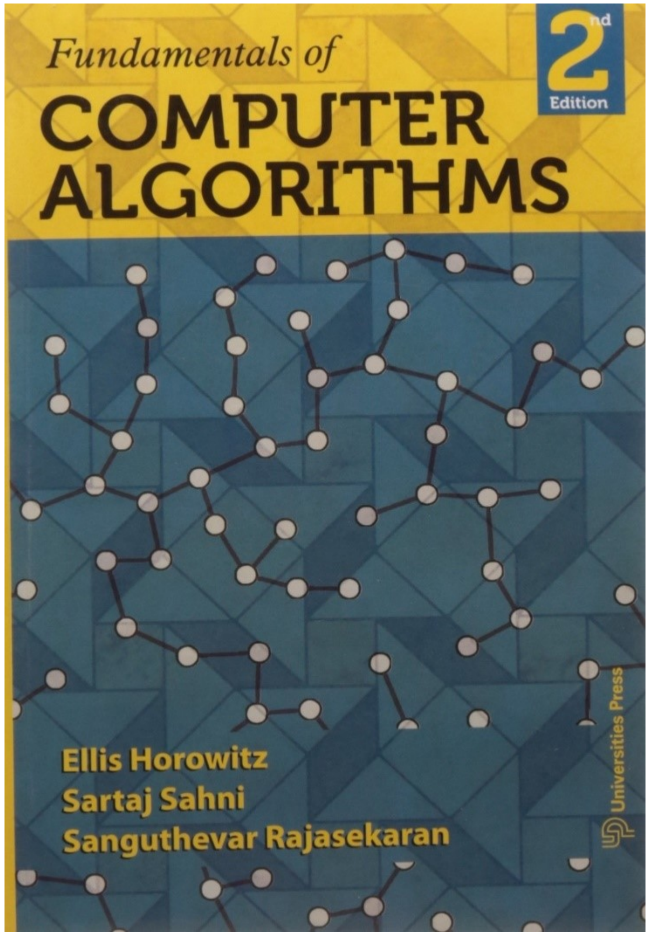
Backward approach

- The time for the for loop of line 5 is $\Theta(|V| + |E|)$, and the time for the for loop of line 14 is $\Theta(k)$, Hence, the total time is $\Theta(|V| + |E|)$.
- The algorithm also works for the edges crossing more than 1 stage.

Thanks for Your Attention!

Suggested reading

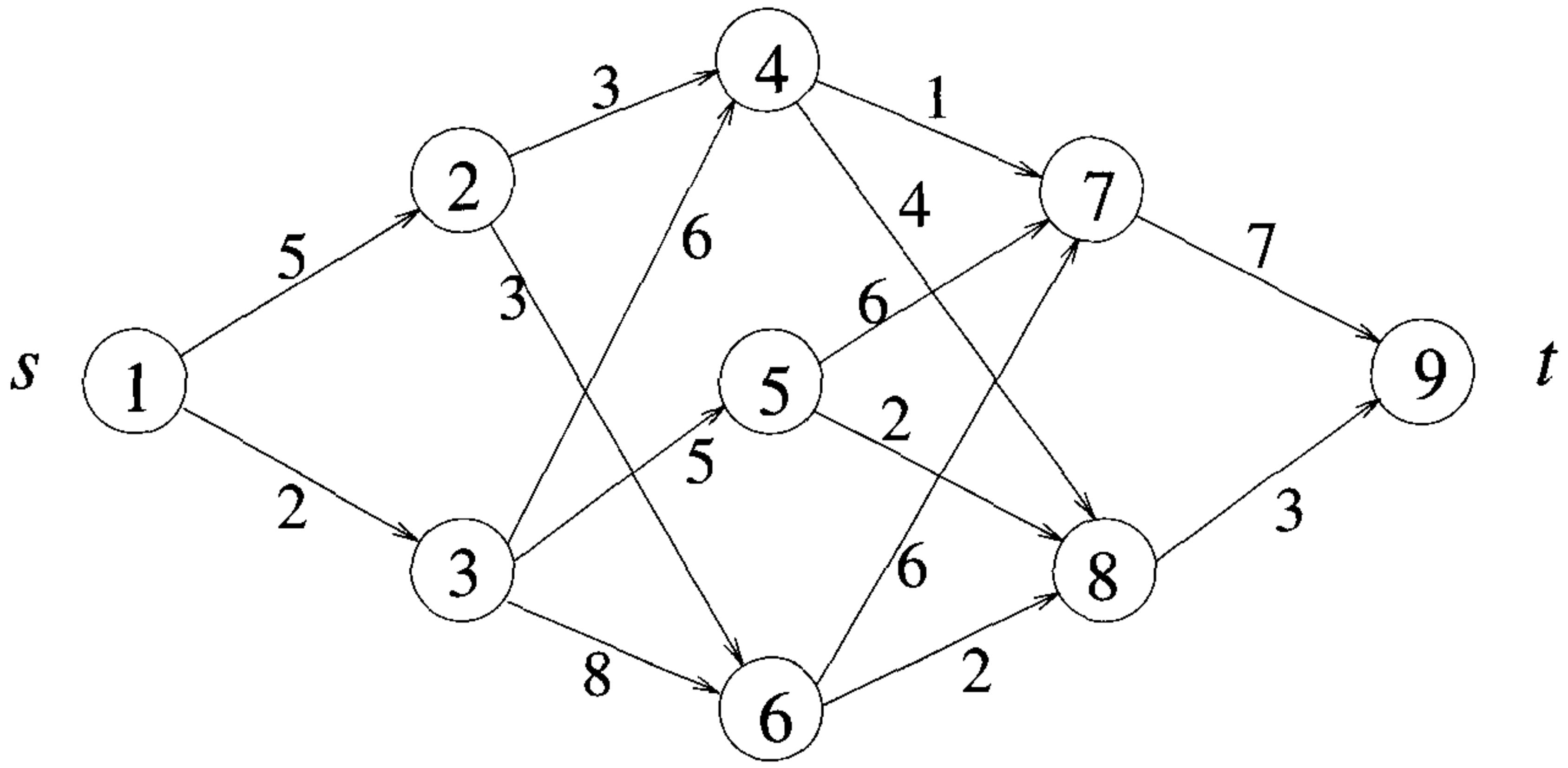




Exercises

Exercise

- Find the minimum cost path from s to t in the following multistage graph using both forward and backward approach.



Exercise

1. Find a minimum-cost path from s to t in the multistage graph of Figure 5.4. Do this first using the forward approach and then using the backward approach.

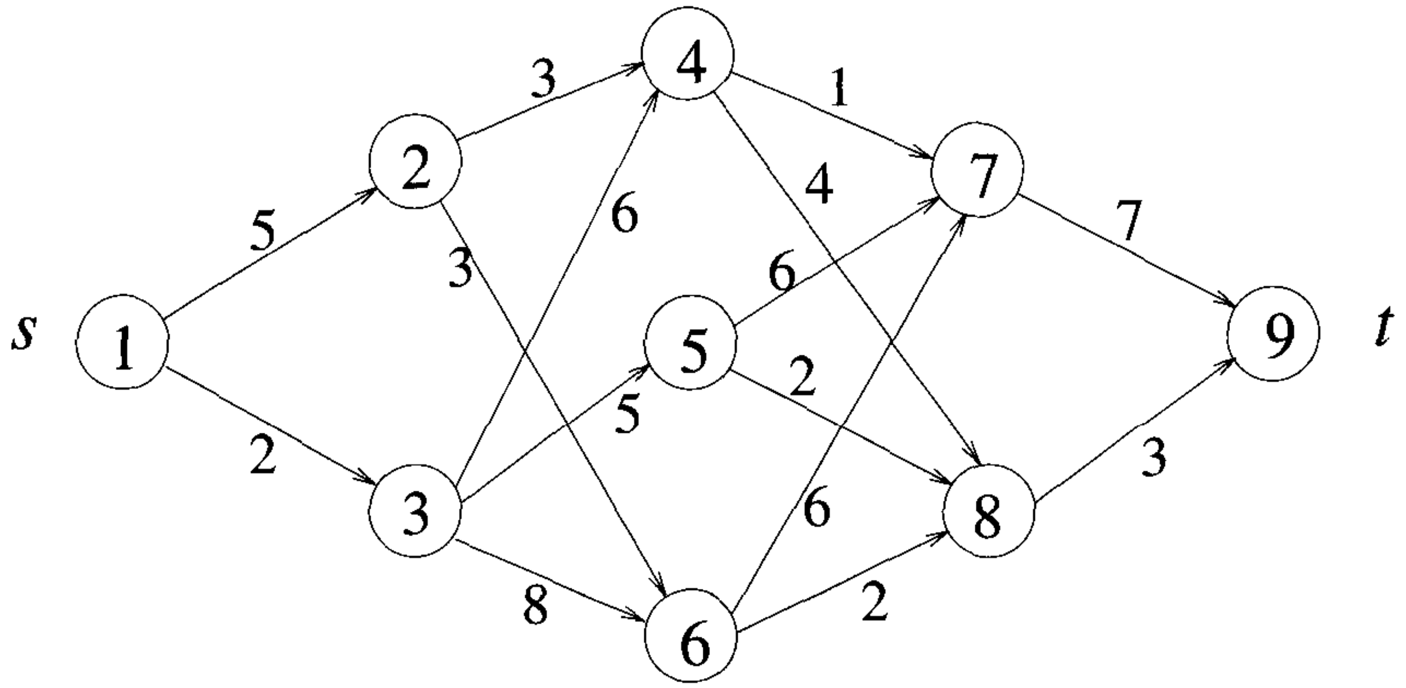
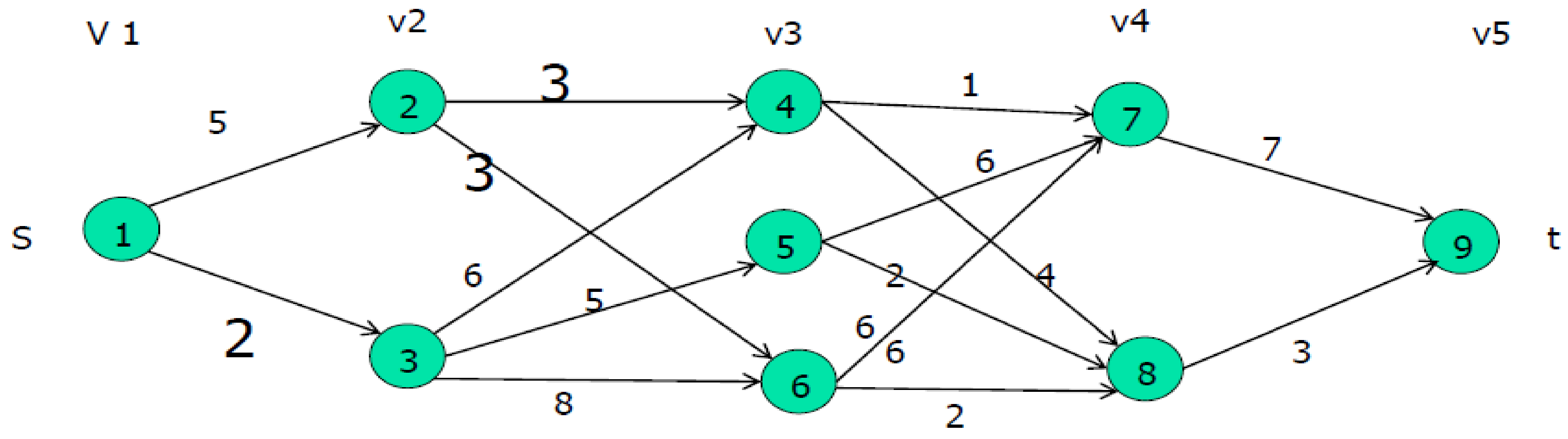
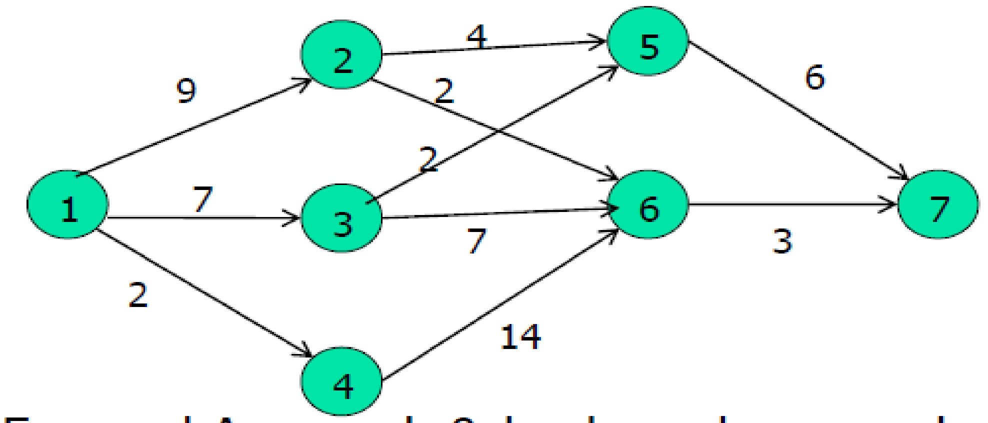


Figure 5.4 Multistage graph for Exercise 1

Exercise



Find Forward Approach & backward approach: answer is 12



Find Forward Approach & backward approach: answer is 14

Programming assignment

2. Refine Algorithm 5.1 into a program. Assume that G is represented by its adjacency lists. Test the correctness of your code using suitable graphs.

3. Program Algorithm 5.1. Assume that G is an array $G[1:e, 1:3]$. Each edge $\langle i, j \rangle$, $i < j$, of G is stored in $G[q]$, for some q and $G[q,1] = i$, $G[q,2] = j$, and $G[q,3] =$ cost of edge $\langle i, j \rangle$. Assume that $G[q,1] \leq G[q+1,1]$ for $1 \leq q < e$, where e is the number of edges in the multistage graph. Test the correctness of your function using suitable multistage graphs. What is the time complexity of your function?

4. Program Algorithm 5.2 for the multistage graph problem using the backward approach. Assume that the graph is represented using inverse adjacency lists. Test its correctness. What is its complexity?

Exercise

5. Do Exercise 4 using the graph representation of Exercise 3. This time, however, assume that $G[q, 2] \leq G[q + 1, 2]$ for $1 \leq q < e$.

6. Extend the discussion of this section to directed acyclic graphs (dags). Suppose the vertices of a dag are numbered so that all edges have the form $\langle i, j \rangle$, $i < j$. What changes, if any, need to be made to Algorithm 5.1 to find the length of the longest path from vertex 1 to vertex n?

7. [W. Miller] Show that BGraph1 computes shortest paths for directed acyclic graphs represented by adjacency lists (instead of inverse adjacency lists as in BGraph).

```
Algorithm BGraph1($G, n$)
{
    $bcost[1] := 0.0$;
    for $j := 2$ to $n$ do $bcost[j] := \infty$;
    for $j := 1$ to $n - 1$ do
        for each $r$ such that $\langle j, r \rangle$ is an edge of $G$ do
            $bcost[r] := \min(bcost[r], bcost[j] + c[j, r])$;
}
```

Note: There is a possibility of a floating point overflow in this function. In such cases the program should be suitably modified.

Dynamic programming

- Dynamic Programming is an algorithm design method that can be used when the solution to a problem may be viewed as the result of a sequence of decisions
- Dynamic programming solves *optimization problems* by combining solutions to sub-problems
- "Programming" refers to a tabular method with a series of choices, not "coding"

4 Basic steps for Dynamic programming paradigm

1. **Characterize the structure of an optimal solution**
2. **Recursively define the value of an optimal solution**
3. **Compute the value of an optimal solution in a bottom-up fashion**
4. **Construct an optimal solution from computed information**

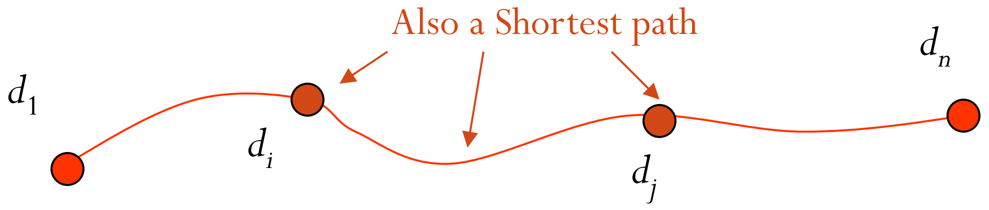
The Principle of Optimality

- In solving optimization problems which require making a sequence of decisions, such as the change making problem, we often apply the following principle in setting up a recursive algorithm:
- Suppose an optimal solution made decisions d_1, d_2, and …, d_n.
- The sub-problem starting after decision point d_i and ending at decision point d_j, also has been solved with an optimal solution made up of the decisions d_i through d_j.
- That is, any subsequence of an optimal solution constitutes an optimal sequence of decisions for the corresponding sub-problem.

The Principle of Optimality:

- This principle of optimality which can be illustrated by the shortest paths in weighted graphs as follows:
- In a shortest path from d_1 to $d_{n,}$ If d_i, d_{i1}, d_{i2}, …,d_j is a shortest path from d_i to d_j, then d_{i1}, d_{i2}, …,d_j must be a shortest path from d_{i1} to d_j

A shortest path from d_1 to d_n



Principle of Optimality

- *Principle of optimality*: Suppose that in solving a problem, we have to make a sequence of decisions D_1, D_2, …, D_n. If this sequence is optimal, then the last k decisions, $1 < k < n$ must be optimal.
- E.g. the shortest path problem

If d_i, d_{i1}, d_{i2}, …,d_j is a shortest path from d_i to d_j, then d_{i1}, d_{i2}, …,d_j must be a shortest path from d_{i1} to d_j

- In summary, if a problem can be described by a **multistage graph**, then it can be solved by dynamic programming.