Dynamic Programming
Multistage Graphs

Dr. K. RAGHAVA RAO

Professor in CSE

KL University

krraocse@gmail.com

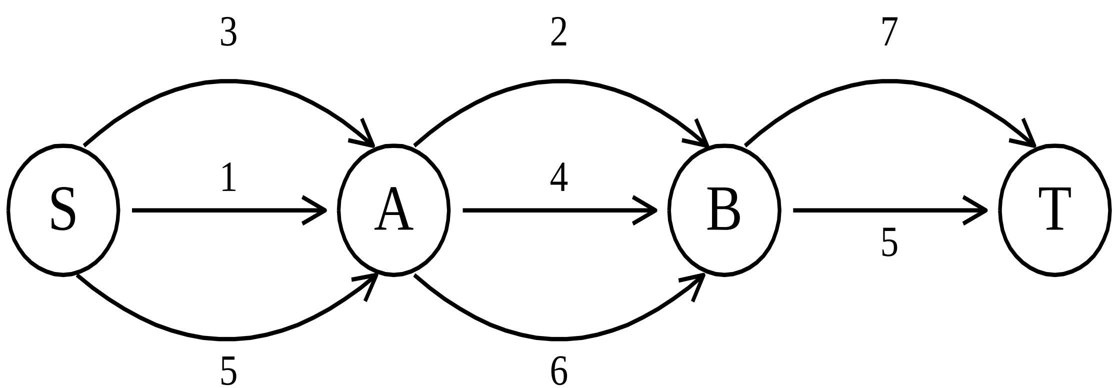
http://mcadaa.blog.com

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

The shortest path

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



- Apply the greedy method :
  the shortest path from S to T :

  $1 + 2 + 5 = 8$

Principle of optimality

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

  If $i, i_1, i_2, \ldots, j$ is a shortest path from i to j, then $i_1, i_2, \ldots, j$ must be a shortest path from i_1 to j
- In summary, if a problem can be described by a multistage graph, then it can be solved by dynamic programming.

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.

Backward chaining vs. forward chaining

- Recursion is sometimes called "backward chaining": start with the goal you want, f(7), choosing your sub goals f(6), f(5), … on an as-needed basis.
  - Reason backwards from goal to facts
    (start with goal and look for support for it)

- Another option is "forward chaining": compute each value as soon as you can, f(0), f(1), f(2), f(3) … in hopes that you'll reach the goal.
  - Reason forward from facts to goal
    (start with what you know and look for things you can prove)

Multistage Graphs

Let G=(v,E) be a directed graph. In this we divide the problem into no. of stages or multiple stages then we try to solve whole problem.

Multistage graph problem is to determine shortest path from source to destination. This 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 Graphs

- A multistage graph G=(V,E) is a directed graph in which the vertices are partitioned into k>=2 disjoint sets Vi, i<=i<=k.
- The vertex s is source and t is 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 costs of the edges on the path.
- The multistage graph problem is to find a minimum-cost path from s to t.

Multistage Graphs

- 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 invloves determining which vertex in Vi+1, 1<=i<=k-2, is on the path. It is easy to see that principal of optimality holds.

- Let p(i,j) be a minimum-cost path from vertex j in Vi to vertex t. Let cost(i,j) be the cost of this path.
- Using forward approach to find cost of the path

- Cost(i,j) = min { c(j, l) + cost(i+1, l)} i-stage number
- $l \in V_{i+1}$ j-vertices available at particular stage
  $< j,l > \in E$ l- vertices away from the vertex

P262, Algorithm 5.1

```
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]];
}
```

Algorithm 5.1 Multistage graph pseudocode corresponding to the forward approach

Multistage Graphs

- The multistage graph problem can be solved using backward approach.
- Let bp(i,j) be a mimimum-cost path from vertex s to vertex j in V_i Let bcost(i,j) be cost of bp(i,j).
- The backward apporach to find min. cost is

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

- Since bcost(2,j) = c(1,j) if $<1,j> \in E$ and bcost(2,j) = ∞ if $<i,j> \notin E$, bcost(i,j) can be computed using above formula.

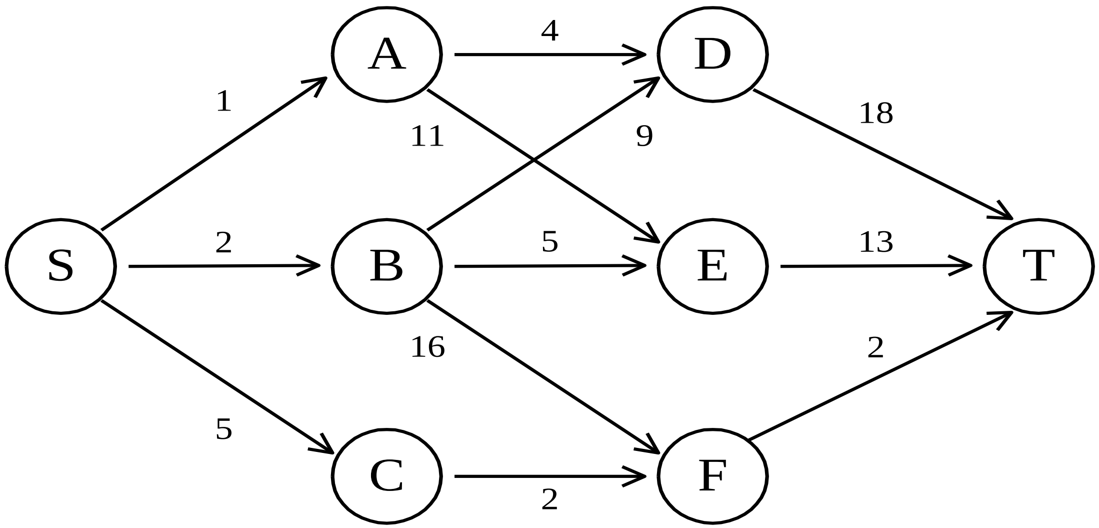
Multistage Graphs : backward approach pseudocode algorithm

- 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
    mimimum;
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]];
}
```

The shortest path in multistage graphs

- e.g.

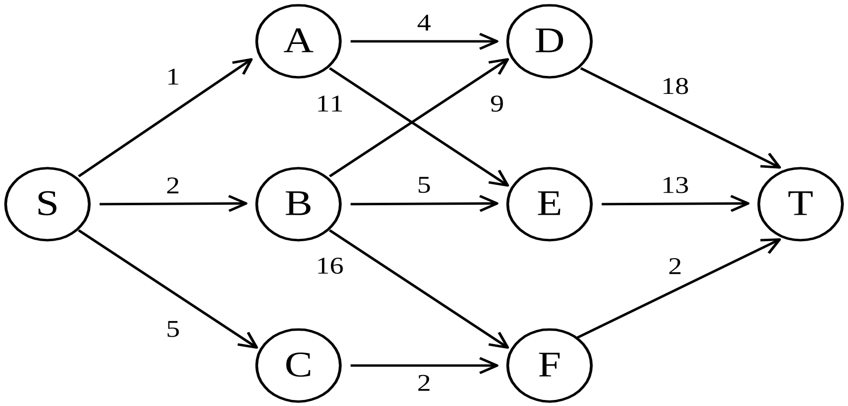


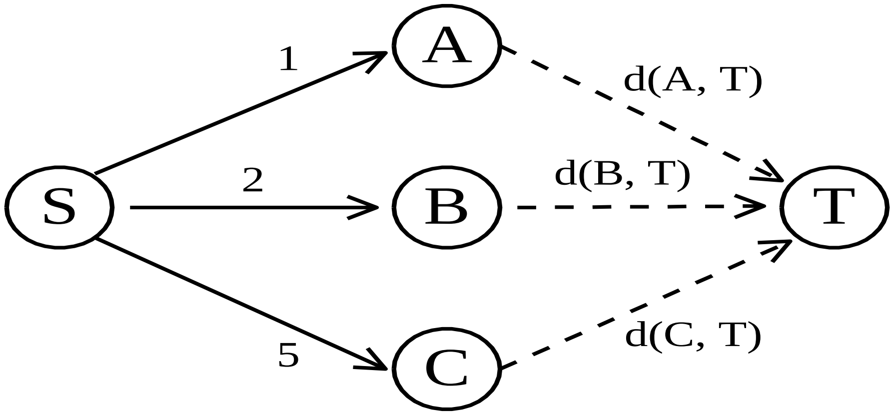
- The greedy method can not be applied to this case: (S, A, D, T) 1+4+18 = 23.
- The real shortest path is:

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

Dynamic programming approach

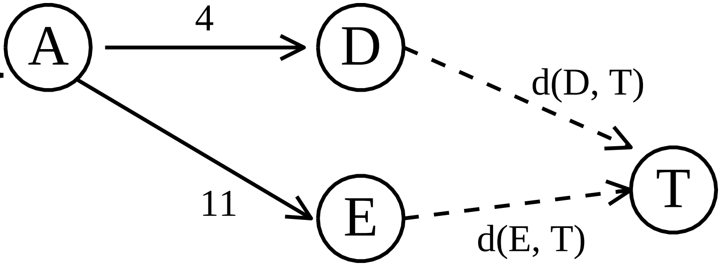
- Dynamic programming approach (forward approach):



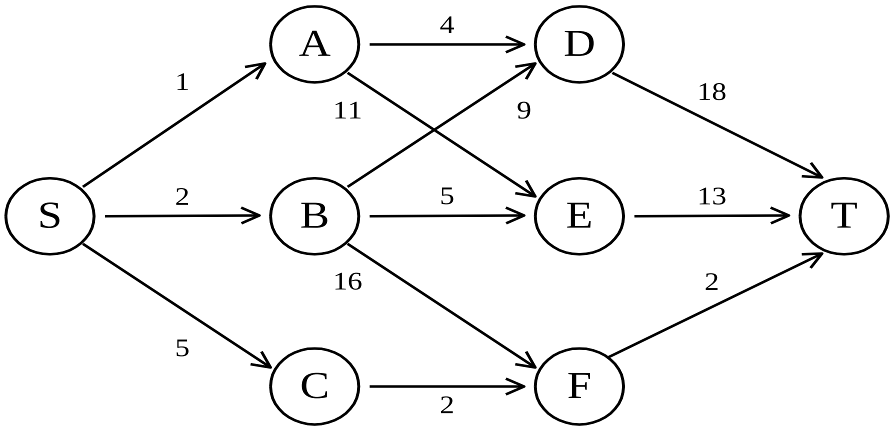


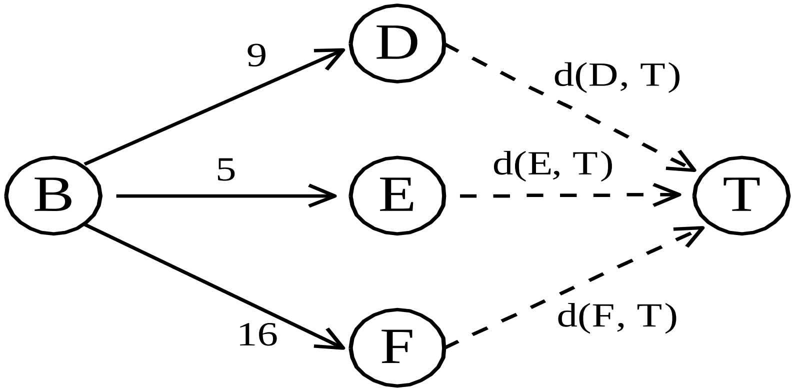
- d(S, T) = min{1+d(A, T), 2+d(B, T), 5+d(C, T)}

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



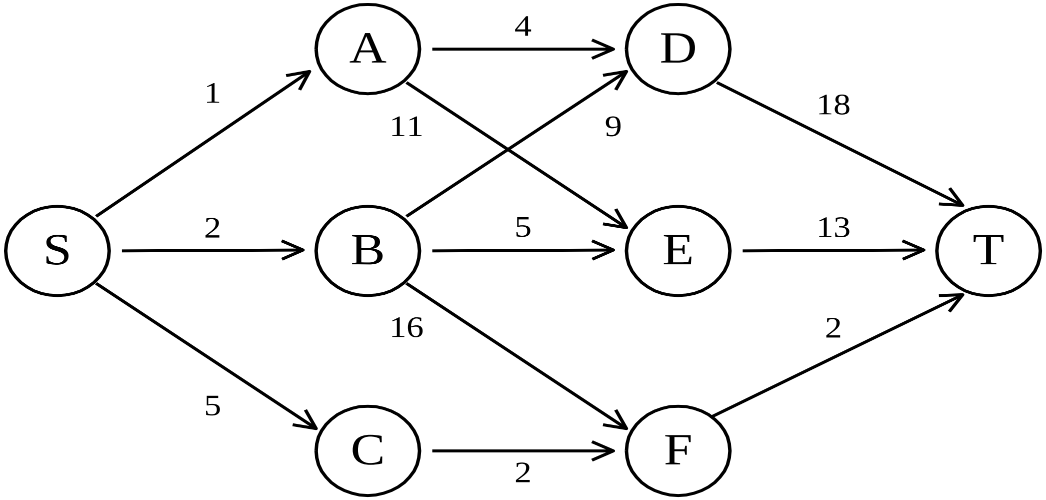
- $d(B, T) = \min\{9+d(D, T), 5+d(E, T), 16+d(F, T)\}$
  $= \min\{9+18, 5+13, 16+2\} = 18.$





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

Backward approach (forward reasoning)

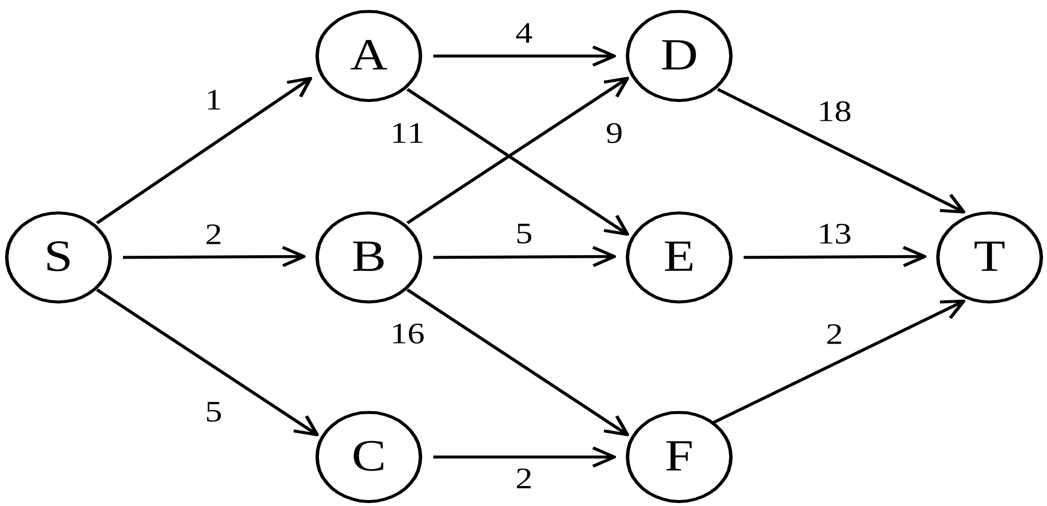


- d(S, A) = 1
  d(S, B) = 2
  d(S, C) = 5
- d(S,D)=min{d(S,A)+d(A,D), d(S,B)+d(B,D)}
  = min{ 1+4, 2+9 } = 5

  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,B)+d(B,F), d(S,C)+d(C,F)}
  = min{ 2+16, 5+2 } = 7

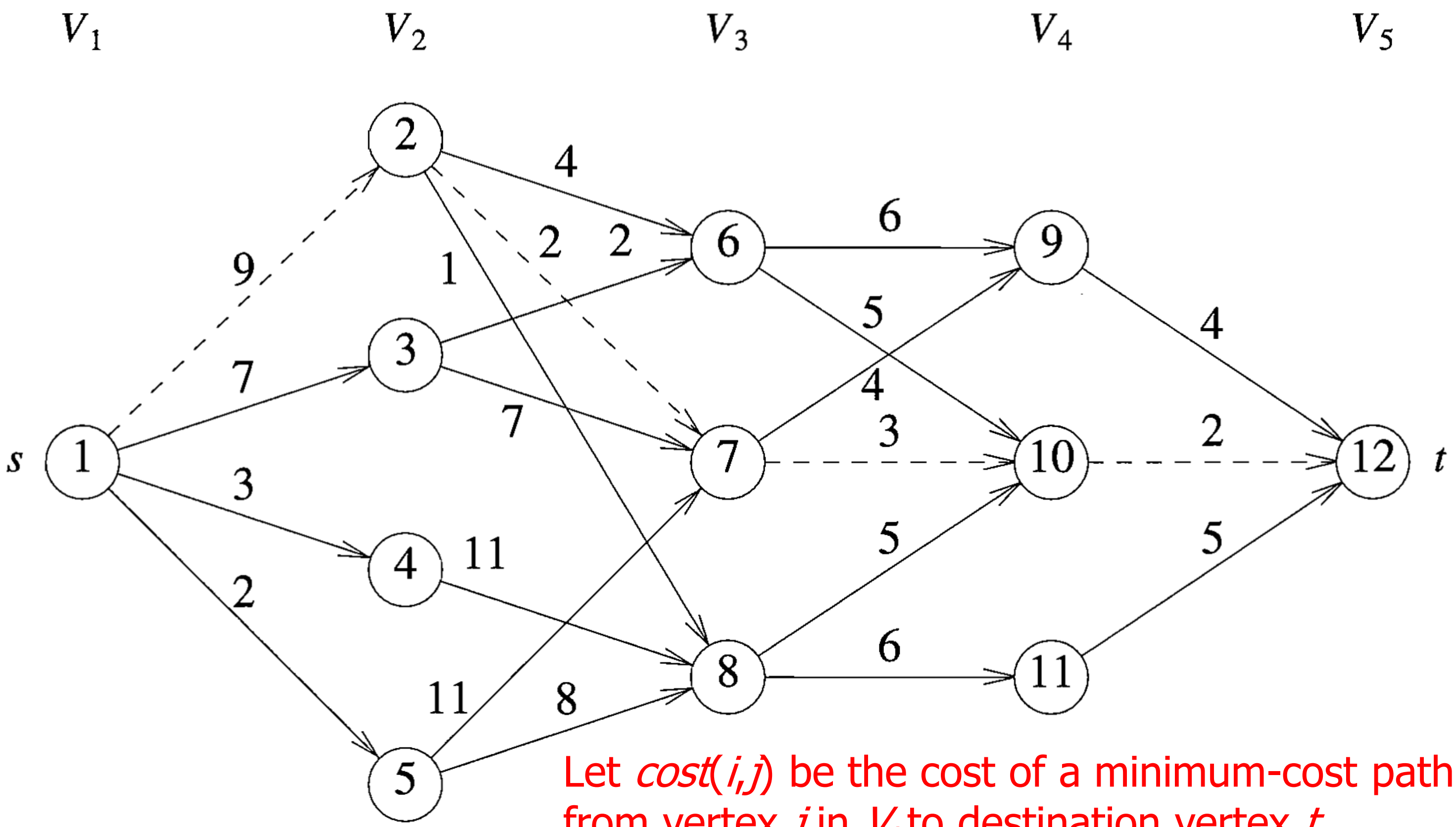
- $d(S,T) = \min\{d(S, D)+d(D, T), d(S,E)+ d(E,T), d(S, F)+d(F, T)\}$
  $= \min\{ 5+18, 7+13, 7+2 \}$
  $= 9$



Example-2
Multistage Graphs

- Principle of optimality (p254)
- 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.
- A multistage is a directed graph in which the vertices are partitioned into $k \geqq 2$ disjoint sets.
- The multistage graph problem is to find a minimum-cost path from *s* to *t*.

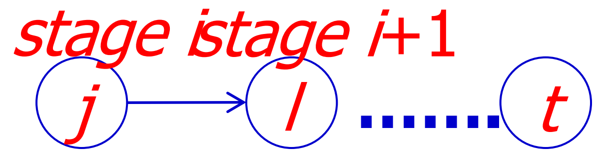
Multistage Graphs: P259, Figure 5.2



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

Figure 5.2 Five-stage graph

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



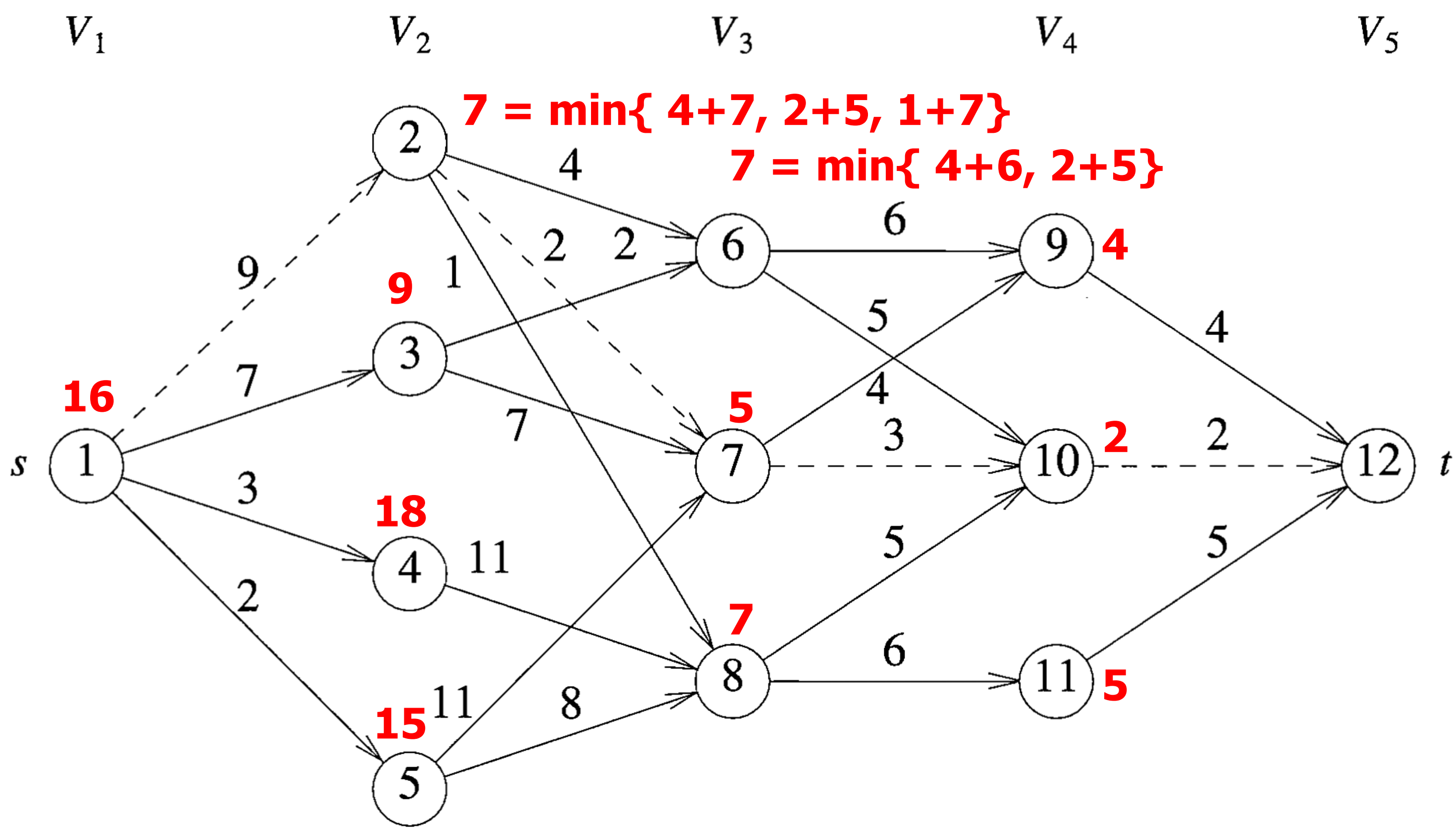
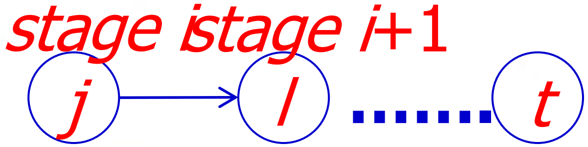


Figure 5.2 Five-stage graph

P259 & P261

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

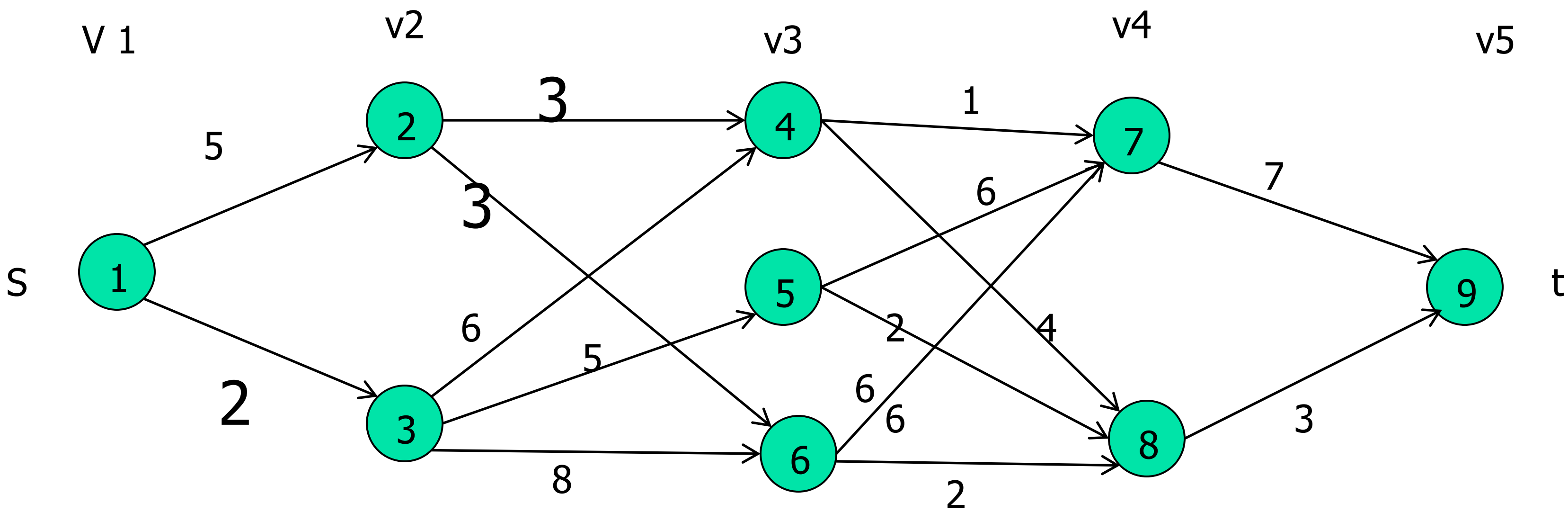


$$\begin{aligned}
cost(3,6) &= \min\ \{6 + cost(4,9), 5 + cost(4,10)\} \\
&= 7 \\
cost(3,7) &= \min\ \{4 + cost(4,9), 3 + cost(4,10)\} \\
&= 5 \\
cost(3,8) &= 7 \\
cost(2,2) &= \min\ \{4 + cost(3,6), 2 + cost(3,7), 1 + cost(3,8)\} \\
&= 7 \\
cost(2,3) &= 9 \\
cost(2,4) &= 18 \\
cost(2,5) &= 15 \\
cost(1,1) &= \min\ \{9 + cost(2,2), 7 + cost(2,3), 3 + cost(2,4), \\
&\qquad\qquad 2 + cost(2,5)\} \\
&= 16
\end{aligned}$$

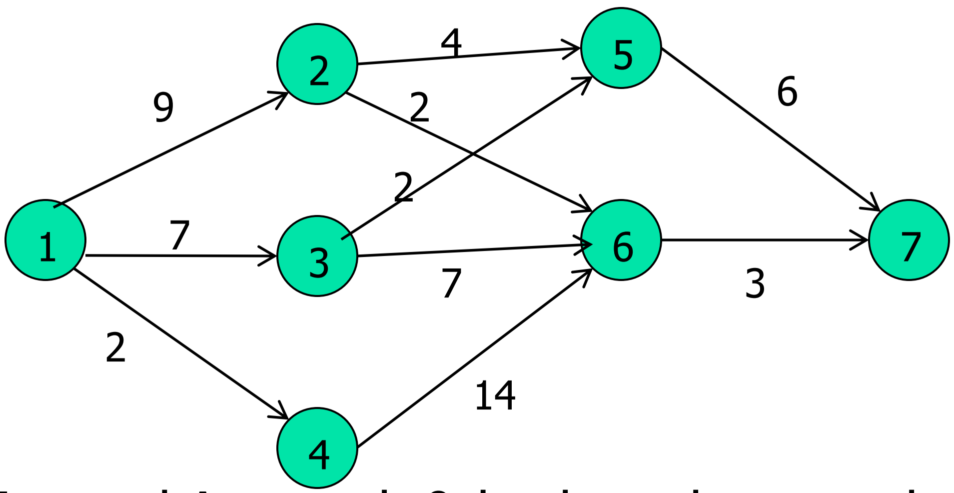
Multistage Graphs

- 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 backward trace from vertex 1 to n also works.
- The algorithm also works for the edges crossing more than 1 stage.

Exercise-1&2



Find Forward Approach & backward approach: answer is 12



Find Forward Approach & backward approach: answer is 14