Design and Analysis of Algorithms

L26: Multi-Stage Graphs

Dynamic Programming

Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

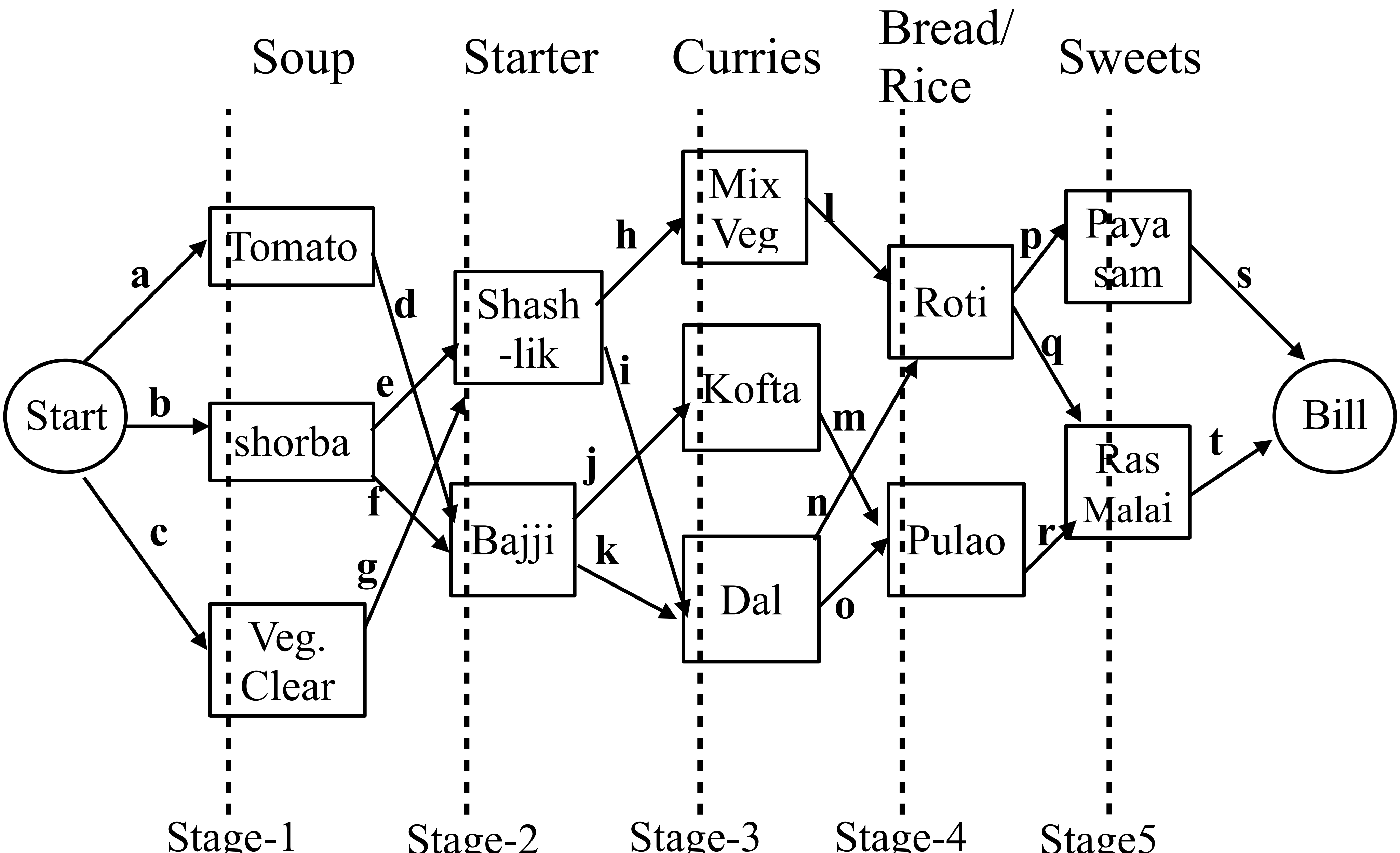
Resources

- Text book 2: Horowitz
  – Sec `5.1,5.2,5.4,5.8,5.9`
- http://www.gdeepak.com/course/adslidesold/26ad.pdf
- https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf
- R1: Introduction to Algorithms
    - Cormen et al.

Consider Restaurant Ordering

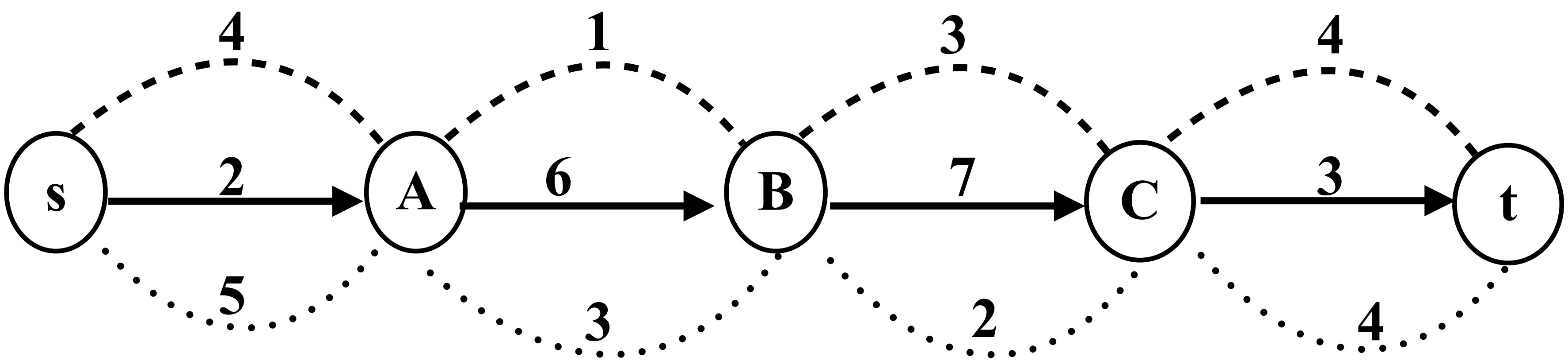
- Food order and serving
  - Soups
  - Starters
  - Main course (curries)
  - Breads/Rice
  - Sweets
  - Mouth freshners
- Each happens in stages
  - Want meal with minimum cost with 1 item in each stage
  - Have multiple choices in each stage.
    - Constraints on what can be chosen in next stage
  - Draw a multi-stage graph

Multi-Stage Graph: Restaurant



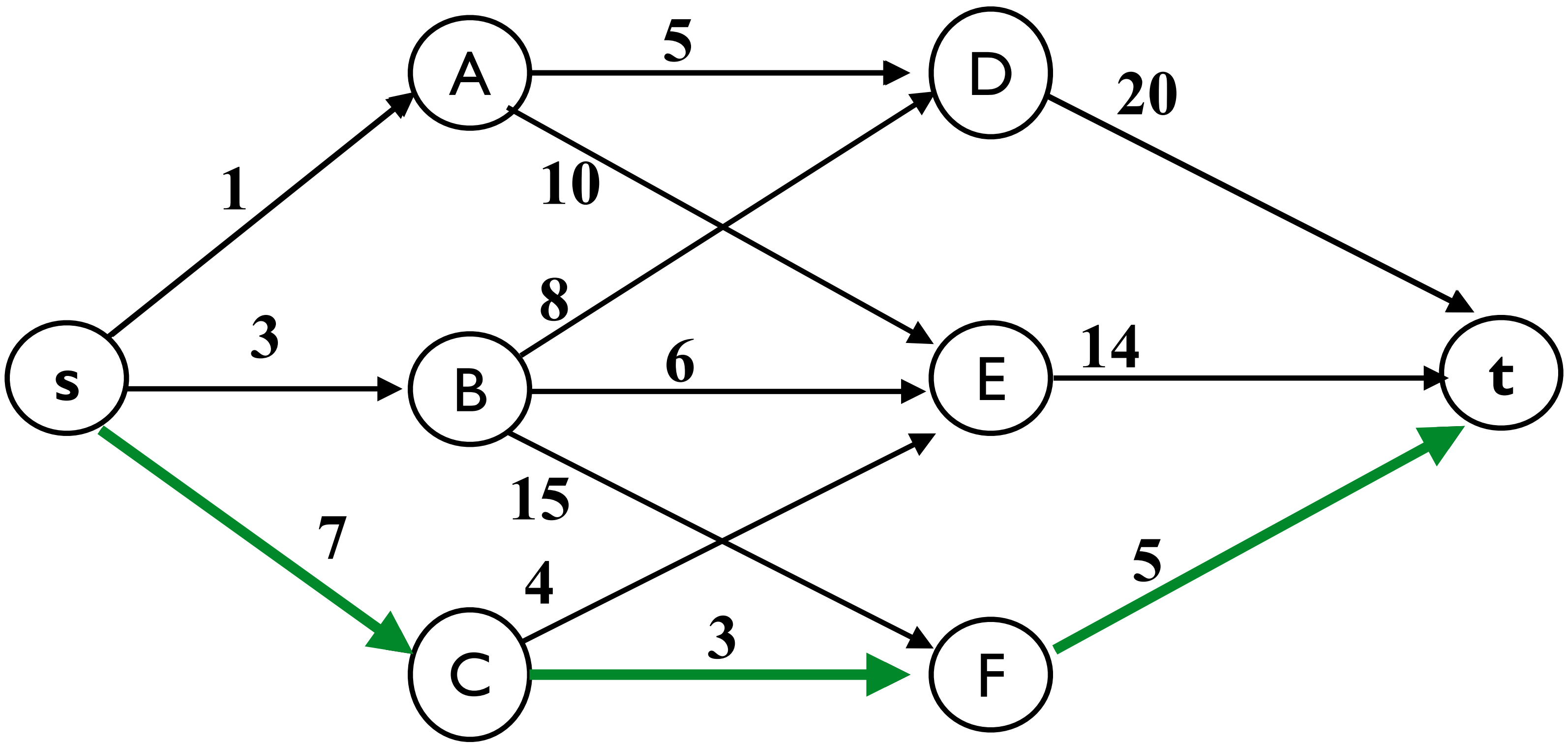
Simple Multi-Stage Graph

- Find shortest path from s to t



– Q: Does Greedy approach work?

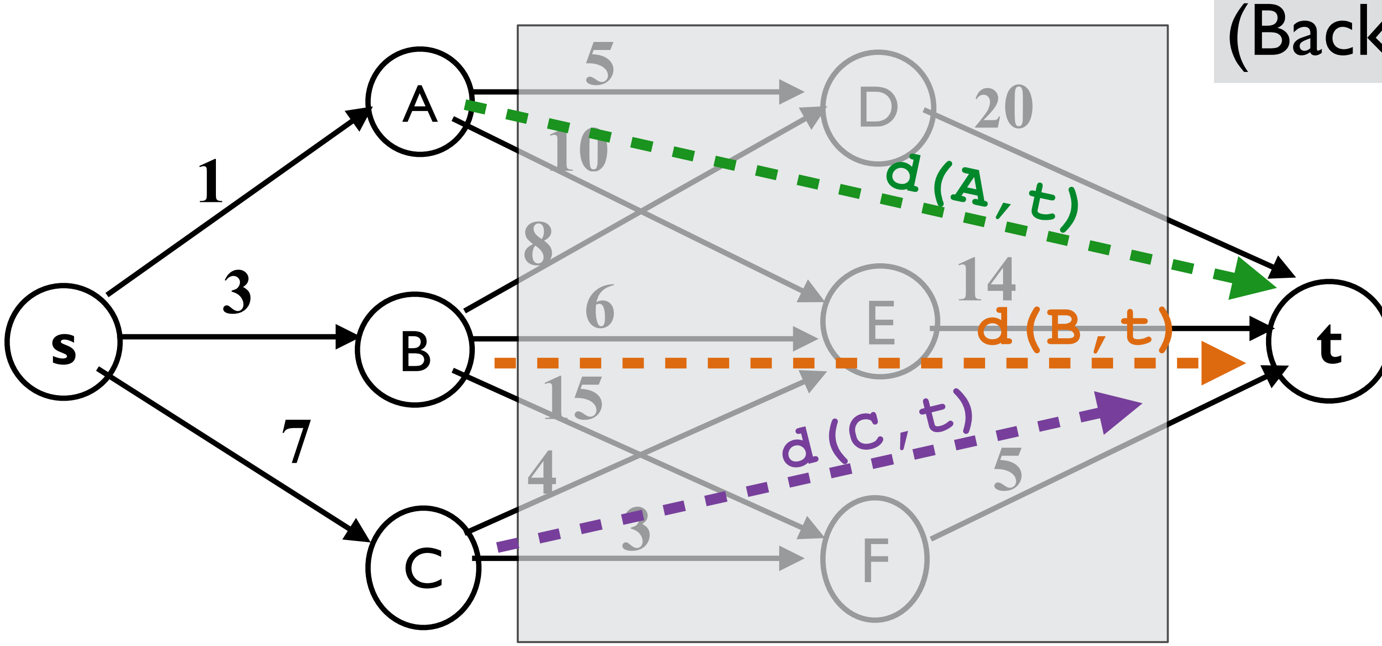
Multistage Graph: Shortest Path



- Find shortest path from s to t
  - Greedy Approach :
    - s→A→D→t = 1+5+20=26
  - Shortest path: s→C→F→t = 7+3+5=15

Dynamic Programming: Forward Approach

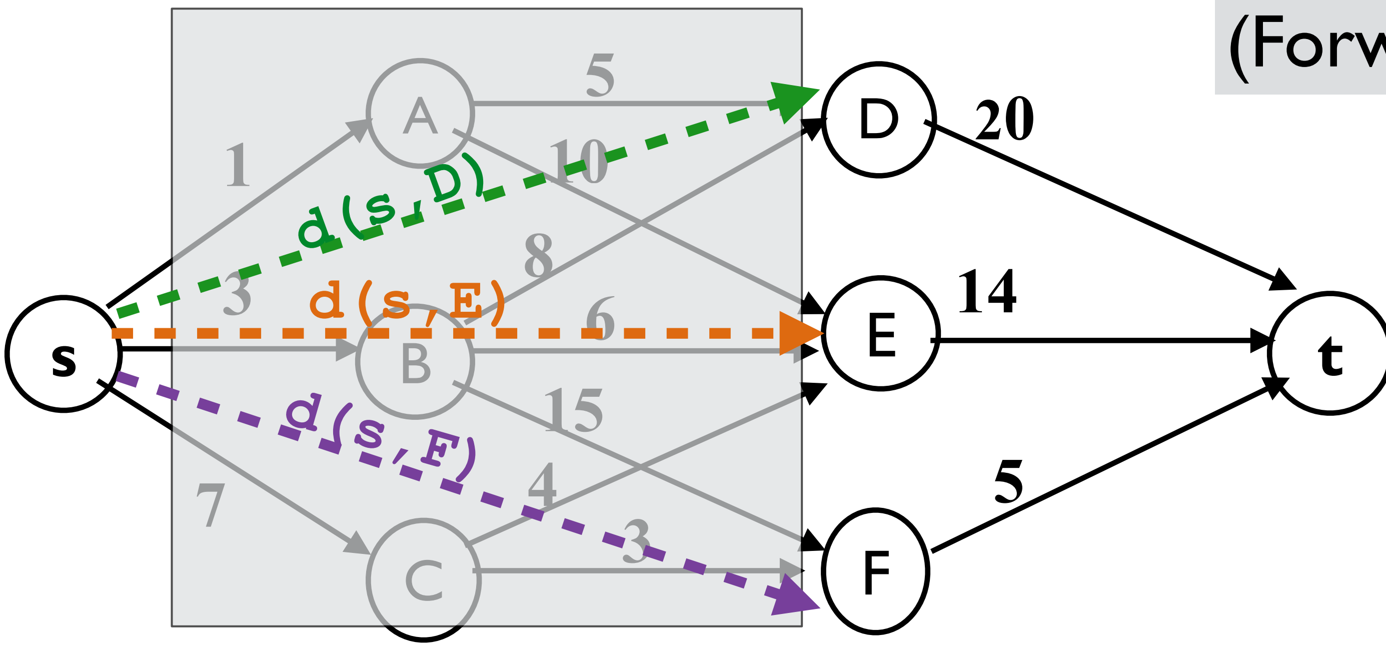
(Backward reasoning)




```
d(s,t)=min{1+d(A,t), 3+d(B,t), 7+d(C,t)}
d(A,t)=min{5+d(D,t),10+d(E,t)}=min(25,24)=24
d(B,t)=min(8+d(D,t),6+d(E,t),15+d(F,t))
       =min{8+20,6+14,15+5)=20
d(C,t)=min{4+d(E,t),3+d(F,t)}=min{18,8)=8
d(s,t)=min{1+24, 3+20, 7+8}=15
```

Dynamic Programming: Backward Approach

(Forward reasoning)



d(s,t)=min{d(s,D)+20, d(s,E)+14, d(s,F)+5}

d(s,D)=min{d(s,A)+5,d(s,B)+8}=min(1+5,3+8)=6

d(s,E)=min(d(s,A)+10,d(s,B)+6,d(s,C)+4)

=min{1+10,3+6,7+4)=9

d(s,F)=min{d(s,B)+15,d(s,C)+3}=min{3+15,7+3)=10

d(s,t)=min{6+20, 9+14, 10+5}=15

Dynamic Programming: Applications

- Resource allocation problem
- Consider the following scenario:
  - A team of 3 students are asked to play 4 games.
    - Table Tennis, Chess, Badminton, Carrom
  - A student can choose to play none, some or all 4.
  - At a time, only one student can play a game.
    - First P_1, then P_2, and then P_3 (in that order)
  - However, no game is to be played by 2 students
  - All the 4 games need to be played.
  - Depending upon games played by a students, different points are awarded as shown next

Resource (Assignment) Allocation

- Award points for games played
  - e.g. P2 plays 3 games, get a total of 8 points
  - Thus, column values are non-decreasing

Student → Games↓	P1	P2	P3
1 game	2	4	5
2 games	5	7	5
3 games	7	8	6
4 games	8	10	6

- Q: How to allocate games among team members so as to get maximum award points

Resource (Assignment) Allocation

- Possible allocations…
- P_1: 0Gs:
  - P_2:4Gs, P_3:0G:
    - Points:0+10+0=10
  - P_2:3Gs, P_3:1G,
    - Points: 0+8+5=13
  - P_2:2Gs, P_3:2Gs,
    - Points: 0+7+5=12
  - P_2:1G, P_3:3Gs,
    - Points:0+4+6=10
  - P_2:0G, P_3:4Gs,
    - Points:0+0+6=6

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

Resource (Assignment) Allocation

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

- Possible allocations…
- P_1: 1G:
  - P_2:3G, P_3:0G,
    - Points: 2+8+0=10
  - P_2:2G, P_3:1Gs,
    - Points: 2+7+5=14
  - P_2:1G, P_3:2Gs,
    - Points:2+4+5=11
  - P_2:0G, P_3:3Gs,
    - Points:2+0+6=8

Resource (Assignment) Allocation

- Possible allocations…
- P_1: 2Gs.
  - P_2:2G, P_3:0G:
    - Points:5+7+0=12
  - P_2:1G, P_3:1G,
    - Points: 5+4+5=14
  - P_2:0G, P_3:2Gs,
    - Points: 5+0+5=10

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

Resource (Assignment) Allocation

- Possible allocations
- P_1: 3Gs
  - P_2:1G, P_3:0G:
    - Points=7+4+0=11
  - P_2:0G, P_3:1G:
    - Points=7+0+5=12
- P_1:4Gs:
  - P_2:0G, P_3:0G
    - Points=8

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

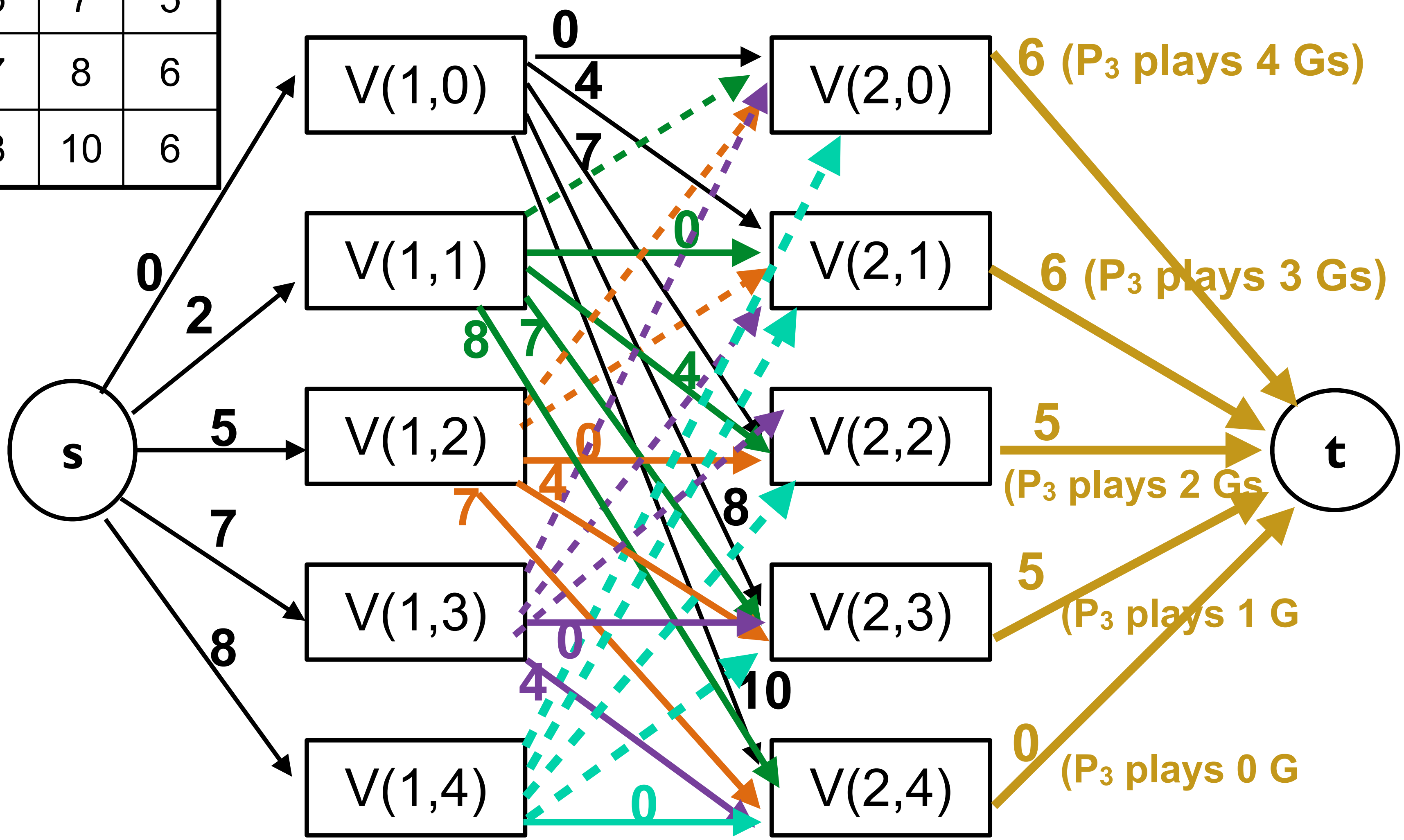
Resource (Assignment) Allocation

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

- Construction of Multistage graph
- The graph has 4 stages
  - Stage 1: Start
  - Stage 2: P_1 plays some games
  - Stage 3: P_2 - plays some of remaining games
  - Stage 4: P_3 - all the remaining games
    - The end stage: all games are played
- From each stage to next stage
  - Draw edge with allowed possibilities
- Each stage (except start, end) has 5 vertices
  - `V(i,j)`: Person P_i, `j`=total num of games played.
    - $1 \le i < 3$; and $0 \le j \le 4$
- *Start*, and *end* stage has one vertex each
  - *start* stage P_1 plays; *end* stage: all 4 games are played
  - Stage 1: P_2 plays; Stage 2: P_3 plays

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

Multistage Graph

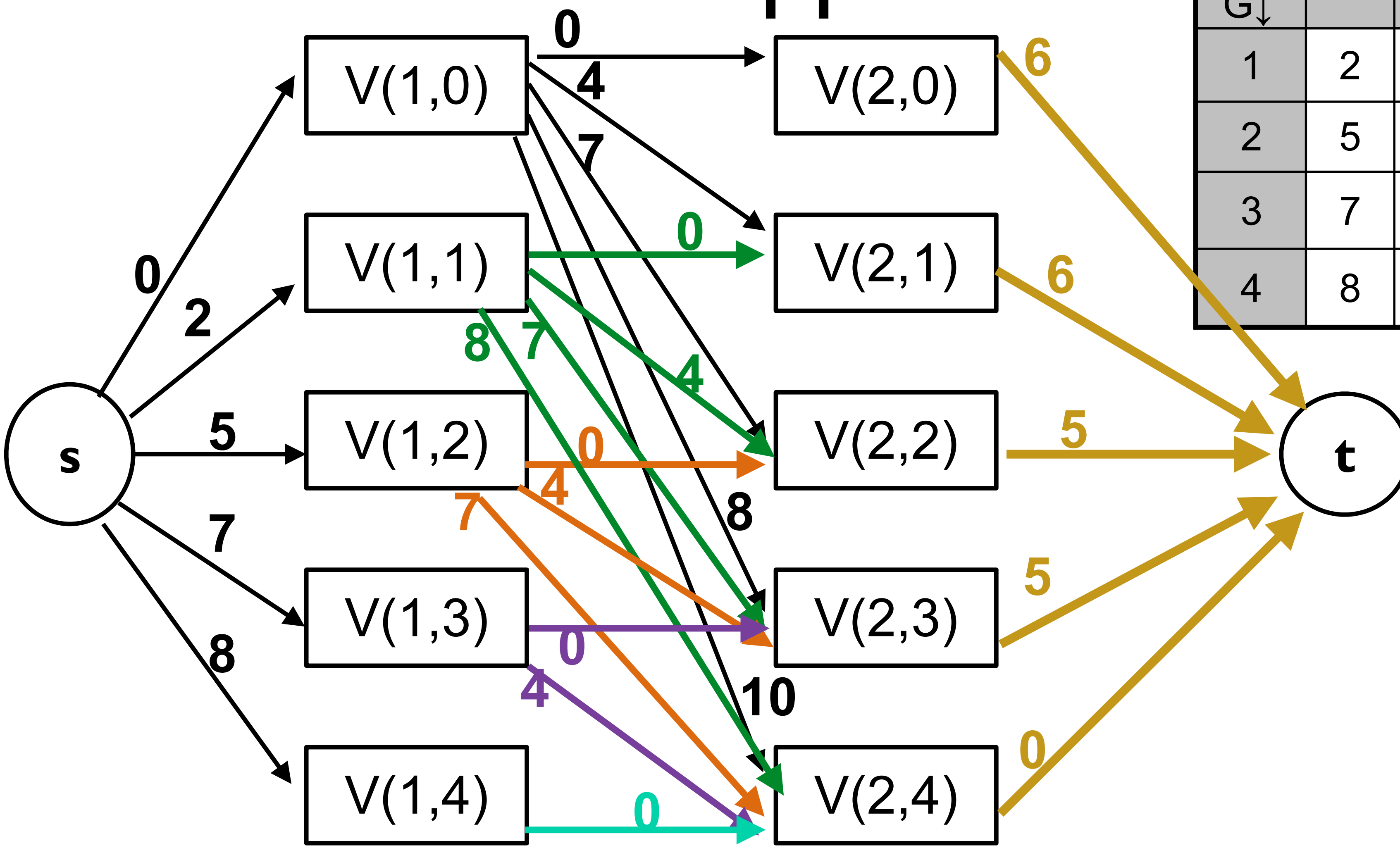


Q: Find max marks using DP Forward approach?

Q: Find max marks using DP Backward approach?

Forward Approach

P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6



Maximum of

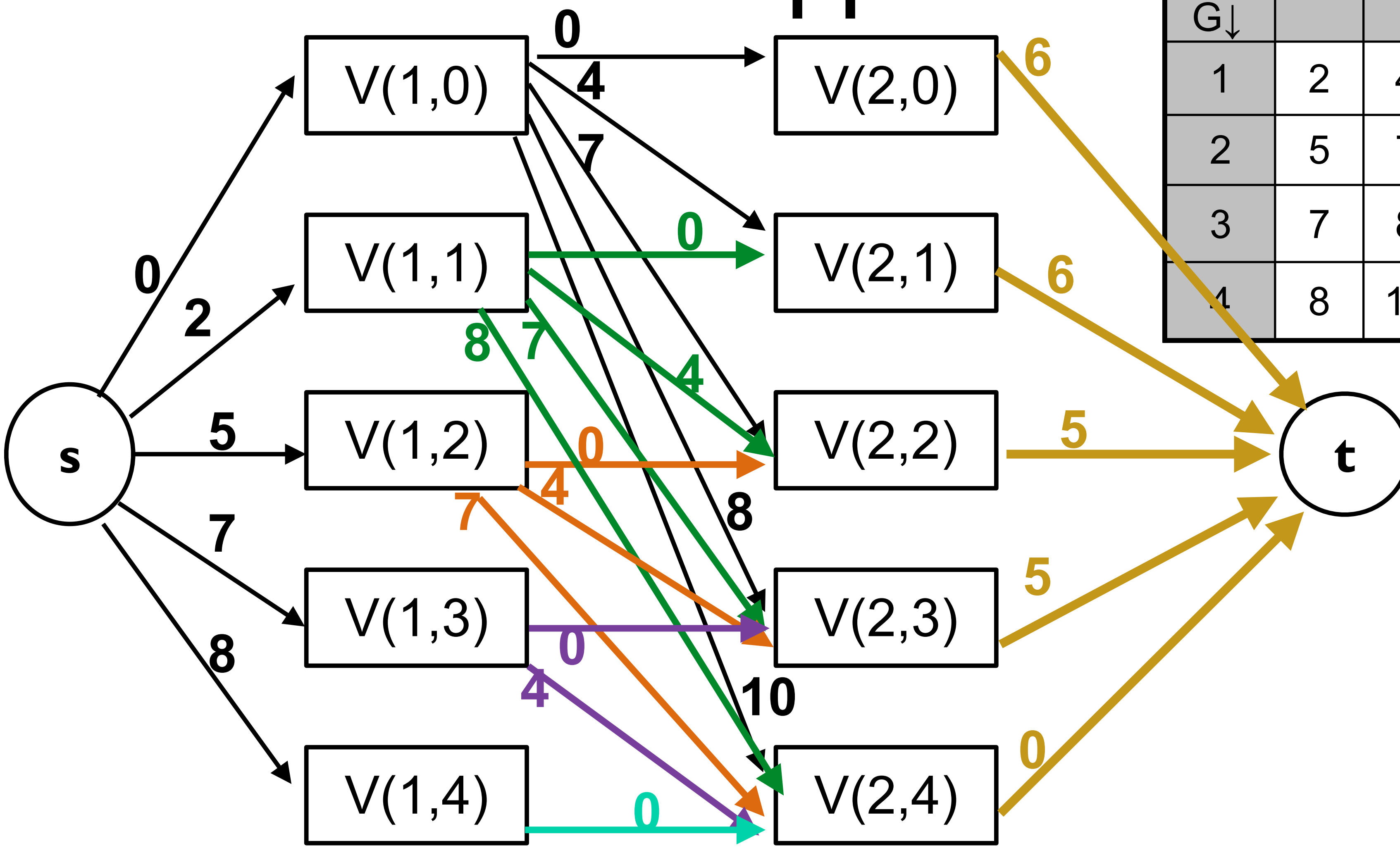
0+d(V(1,0),t), 2+d(V(1,1),t), 5+d(V(1,2),t), 7+d(V(1,3),t), 8+d(V(1,4),t)

=0+13, 2+12, 5+9, 7+5, 8+0=13, 14, 14, 12, 8

= 14

Backward Approach

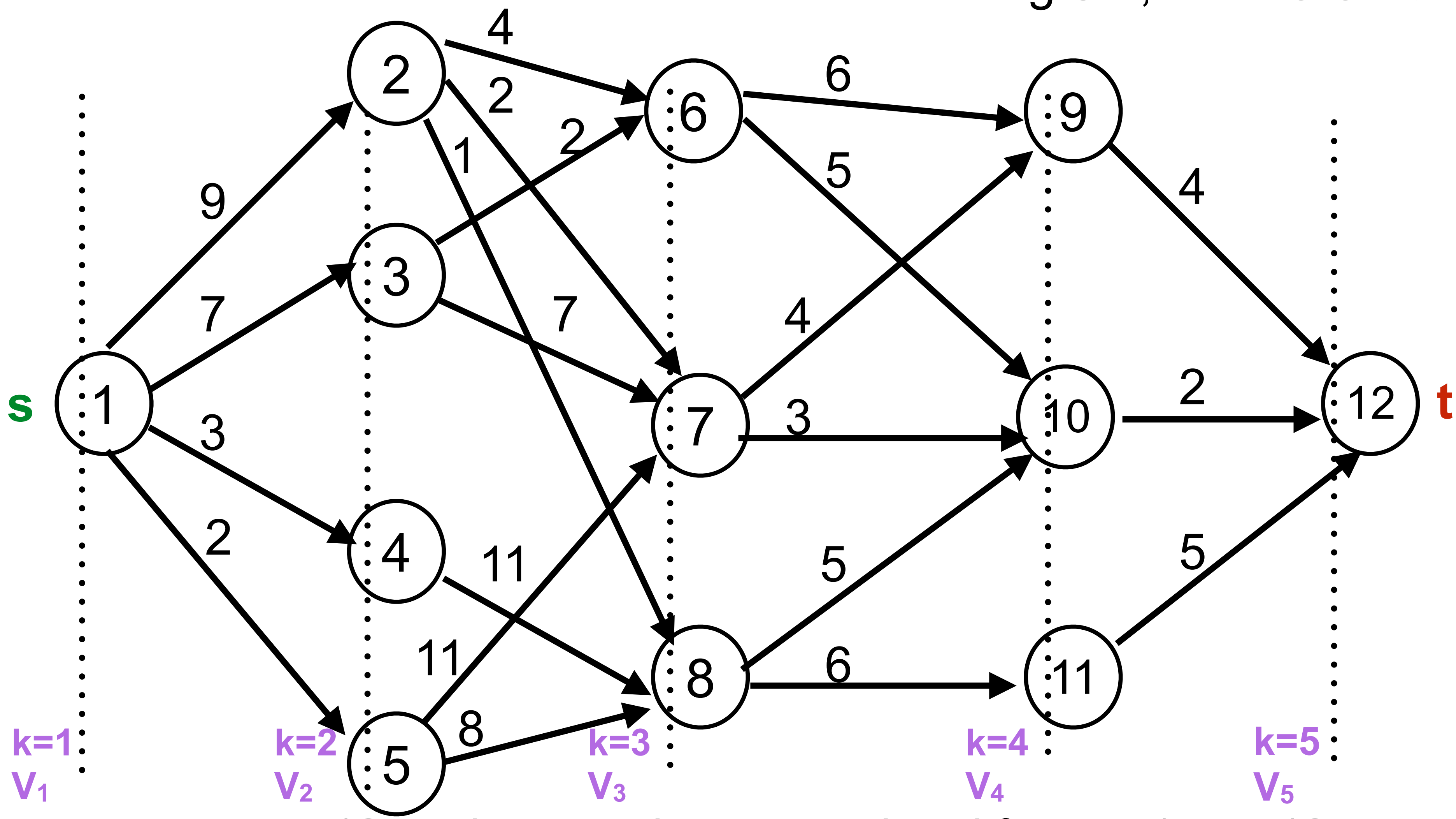
P → G↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6



Maximum of
d(s,V(2,0))+6, d(s,V(2,1))+6, d(s,V(2,2)+5, d(s,V(2,3)+5,d(s,V(2,4))+0
=0+6, 4+6, 7+5, 9+5, 8+0 = 6, 10, 12, 14, 8
= 14

Forward Approach

Fig 5.2, T2: Horowitz…



5 stages, 12 vertices, vertices are ordered from s=1 to t=12

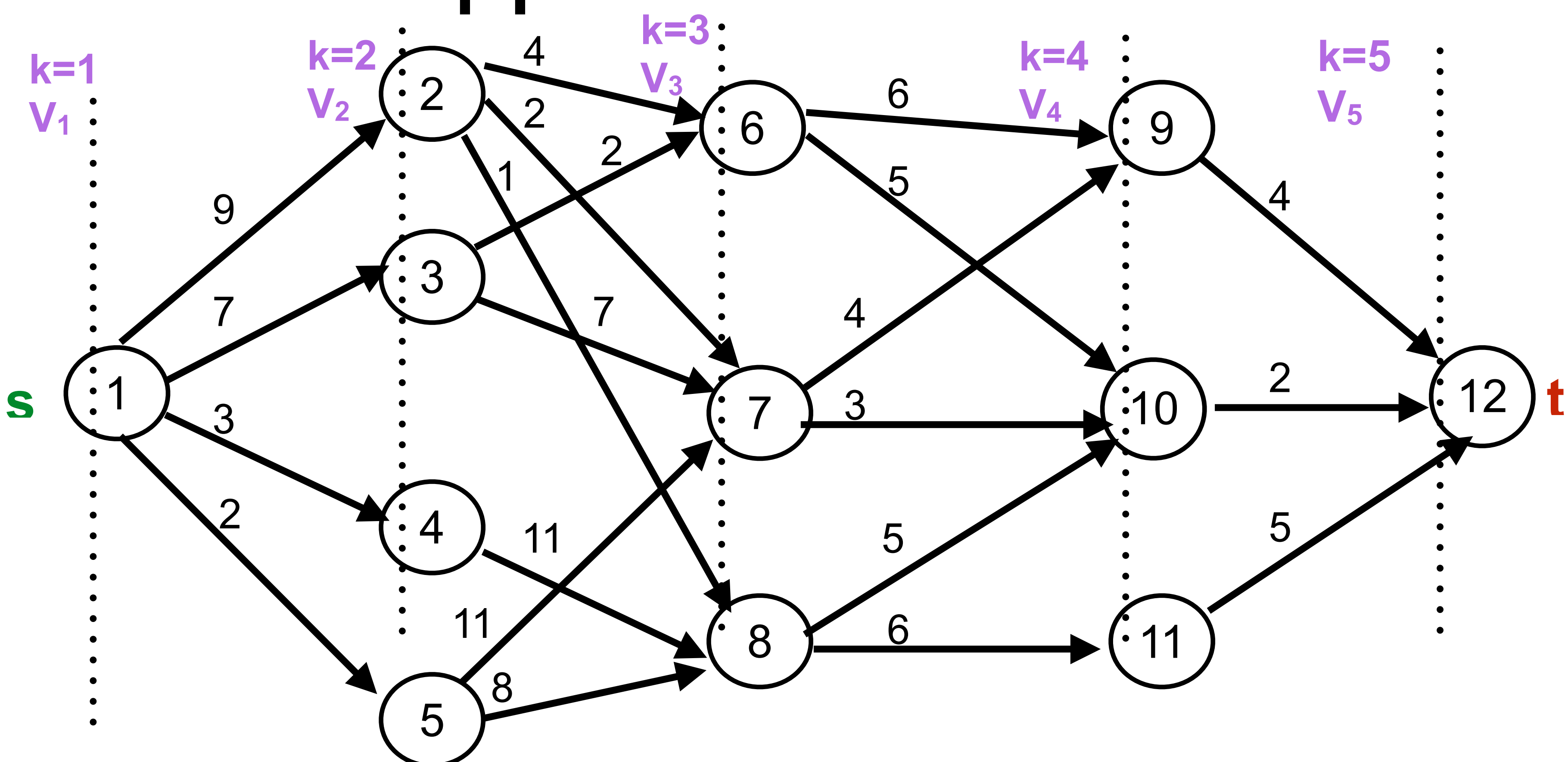
`p(i,j)`: Min cost path from vertex `j` in stage V_i

`cost(i,j)`: Cost of Min cost path `p(i,j)`, or `cost(j)`

`c(j,m)`: Cost of edge `(j,m)` **provided** `(j,m)` $\in E$

Forward Approach

Fig 5.2, T2: Horowitz…

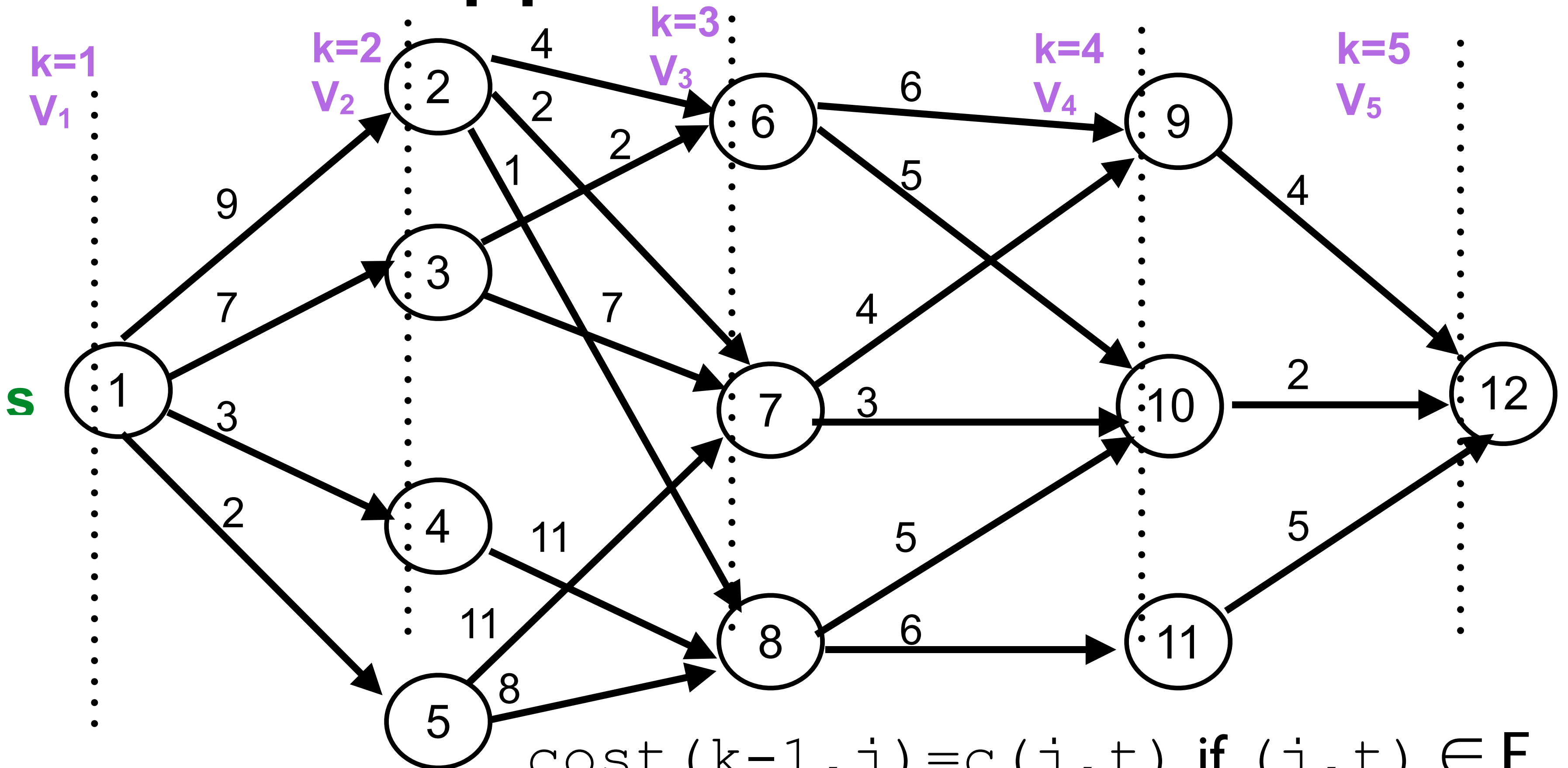


- DP sol^n for k-stage problem is obtained by result of $k-2$ decisions
  - Stage V_2 to V_{k-1}
- i^{th} decision: which vertex in stage V_{i+1} $(1 \le i \le k-2)$ is on the path
- Forward approach gives the solution

$$cost(i,j)=min\{c(j,m)+cost(i+1,m\}, m \in V_{i+1}, (j,m) \in E$$

Forward Approach

Fig 5.2, T2: Horowitz…



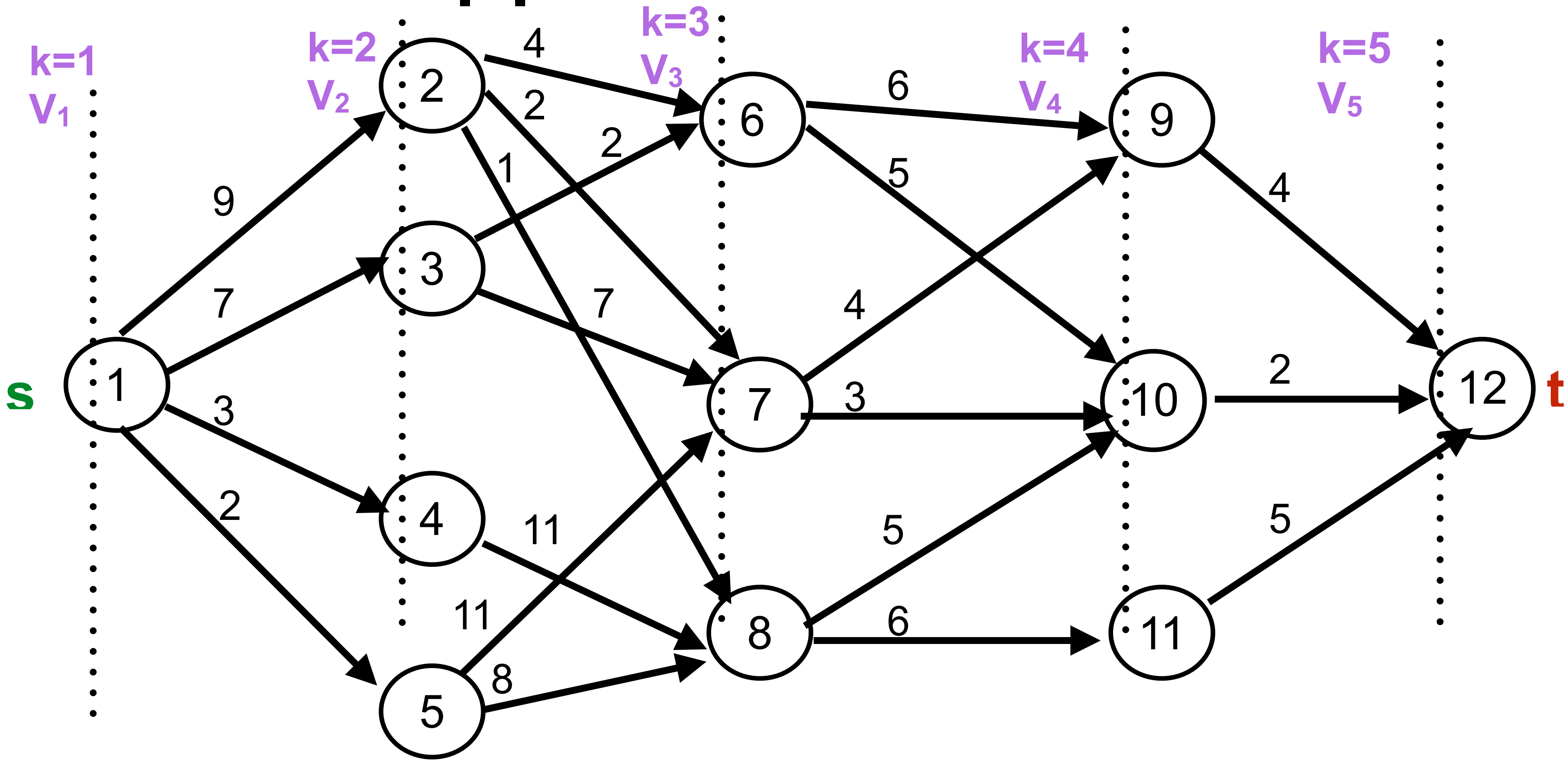
`cost(k-1,j)=c(j,t)` **if** `(j,t)` $\in$ E

`cost(k-1,j)=`∞ **if** `(j,t)` $\notin$ E

Computing `cost(1,s)` requires
  computing first `cost(k-2,j)` $\forall$`j` $\in V_{k-2}$
  then computing `cost(k-3,j)` $\forall$`j` $\in V_{k-3}$
  and so on, and finally `cost(1,s)`

Forward Approach

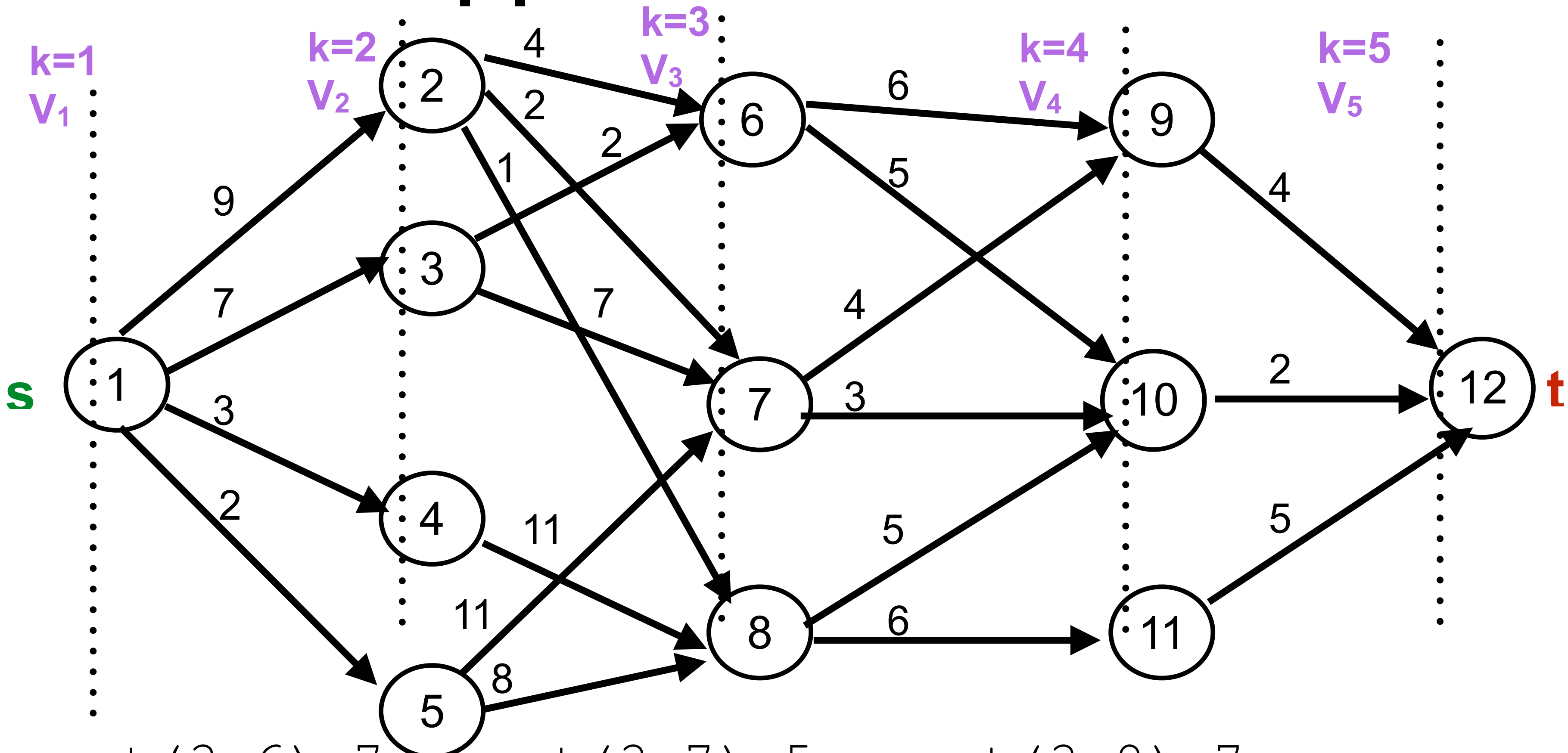
Fig 5.2, T2: Horowitz…



cost(3,6)=min{6+cost(4,9), 5+cost(5,10)}=7
cost(3,7)=min{4+cost(4,9), 3+cost(5,10)}=5
cost(3,8)=min{6+cost(4,10), 6+cost(5,11)}=7

Forward Approach

Fig 5.2, T2: Horowitz…




```
cost(3,6)=7; cost(3,7)=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)=min{2+cost(3,6),7+cost(3,7)}=9
cost(2,4)=min{11+cost(3,8)}=18
cost(2,5)=min{11+cost(3,7), 8+cost(3,8))=15
cost(1,1)=min{9+cost(2,2), 7+cost(2,3),
              3+cost(2,4), 2+cost(2,5)} = 16
```

DP Forward approach: Algo

Algo: `FGraph(Graph G, int k, int p[])`

// i/p `k`-stage graph `n` vertices indexed in order of stages.
// edge `c(i,j)` is cost of edge $v_i \rightarrow v_j$
// `p[i]` is a node on stage `i` in min cost path
// `cost[i]` is minimum from node `i`
// `d[j]` indicates successor of node `j` in min cost path

```
float cost[maxsize]; int d[maxsize], r;
cost[n]=0.0
for j=n-1 to 1 // compute cost[j]
    Let r be a vertex such that vj→vr is an edge, and
    c(j,r)+cost[r] is minimum
    cost[j] = c[j,r) + cost(r)
    d[j]=r
p[1]=1; p[k]=n;
for j=2 to k-1
    p[j]=d[p[j-1]]
```

DP Backward approach: Algo

Algo: `BGraph(Graph G, int k, int p[])`

// i/p `k`-stage graph `n` vertices indexed in order of stages.

// edge `c(i,j)` is cost of edge $v_i \rightarrow v_j$

// `p[1:k]` is a minimum cost path

<u>***float***</u> `bcost[maxsize];` <u>***int***</u> `d[maxsize], r;`

`bcost[n]=0.0`

<u>***for***</u> `j=2` <u>***to***</u> `n` // compute `bcost[j]`

Let `r` be a vertex such that $v_r \rightarrow v_j$ is an edge, and

`bcost[r]+c(r,j)` is minimum

```
bcost[j] = bcost(r) + c[r,j)
d[j]=r
```

`p[1]=1; p[k]=n;`

<u>***for***</u> `j=k-1` <u>***to***</u> `2`

```
p[j]=d[p[j+1]]
```

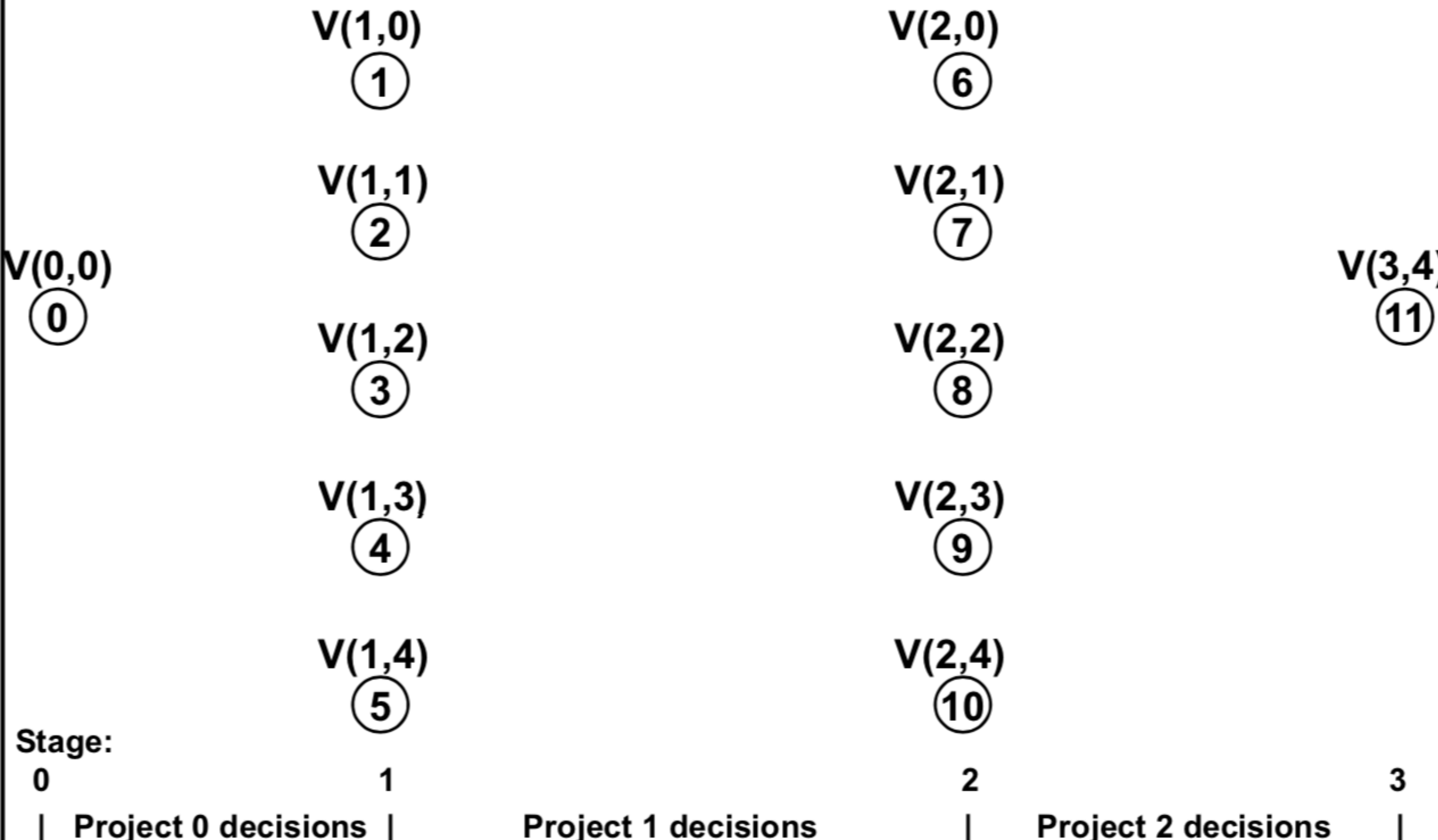
Ex: Build Multistage graph

src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Proj	0	1	2
Investment	Benefit	Benefit	Benefit
1	6	5	1
2	8	11	4
3	8	16	5
4	10	17	6

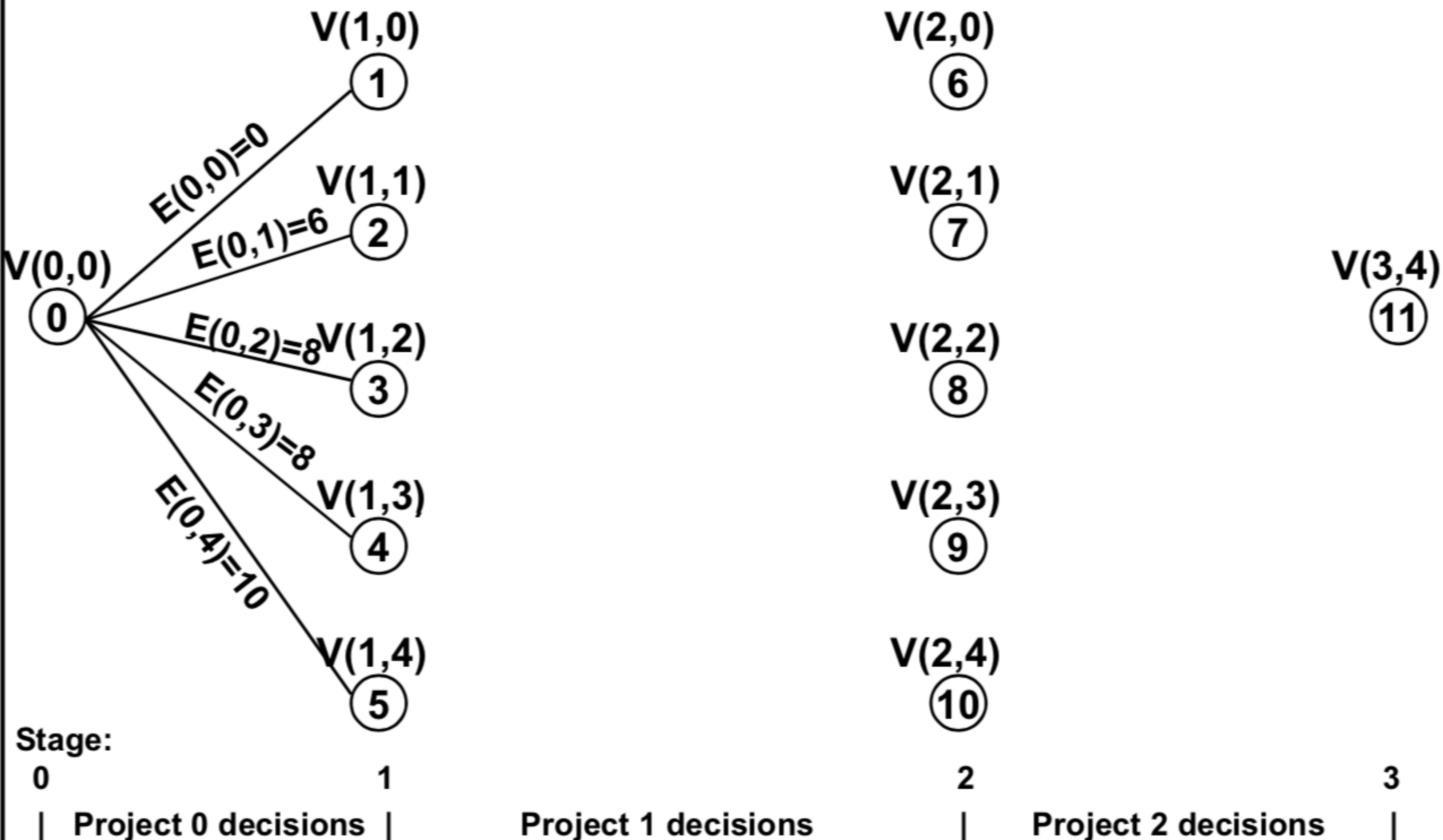
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



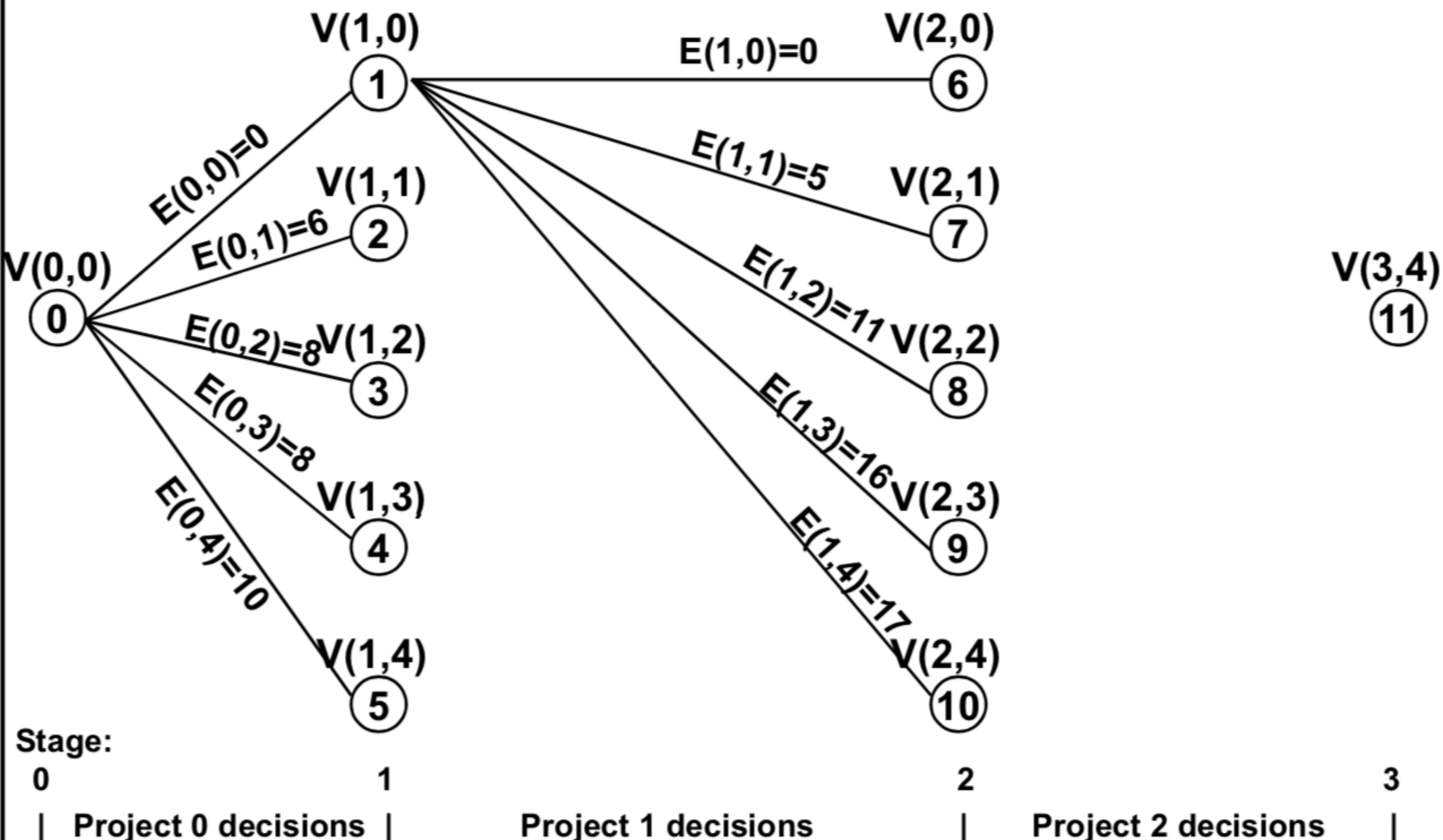
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



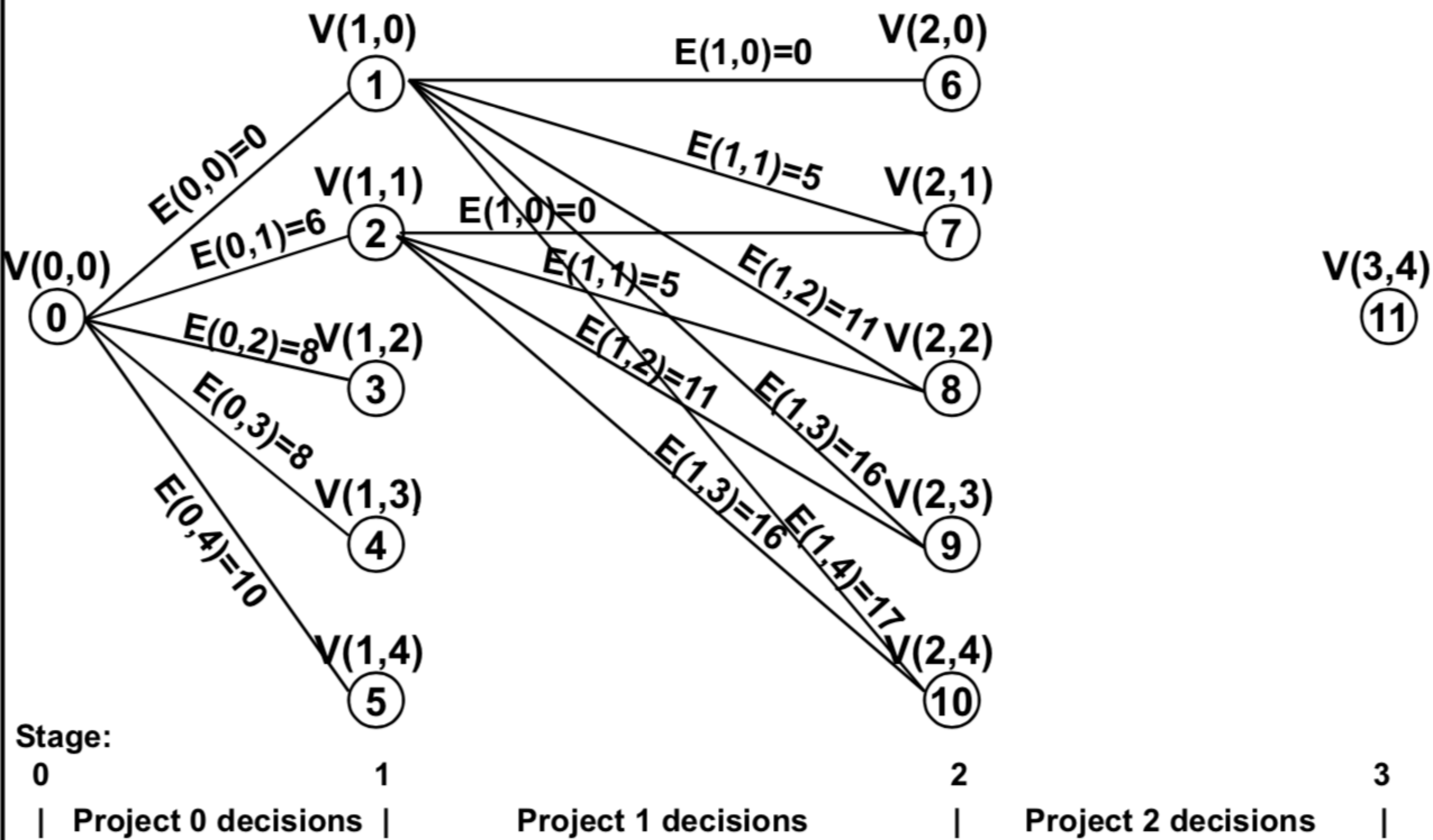
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



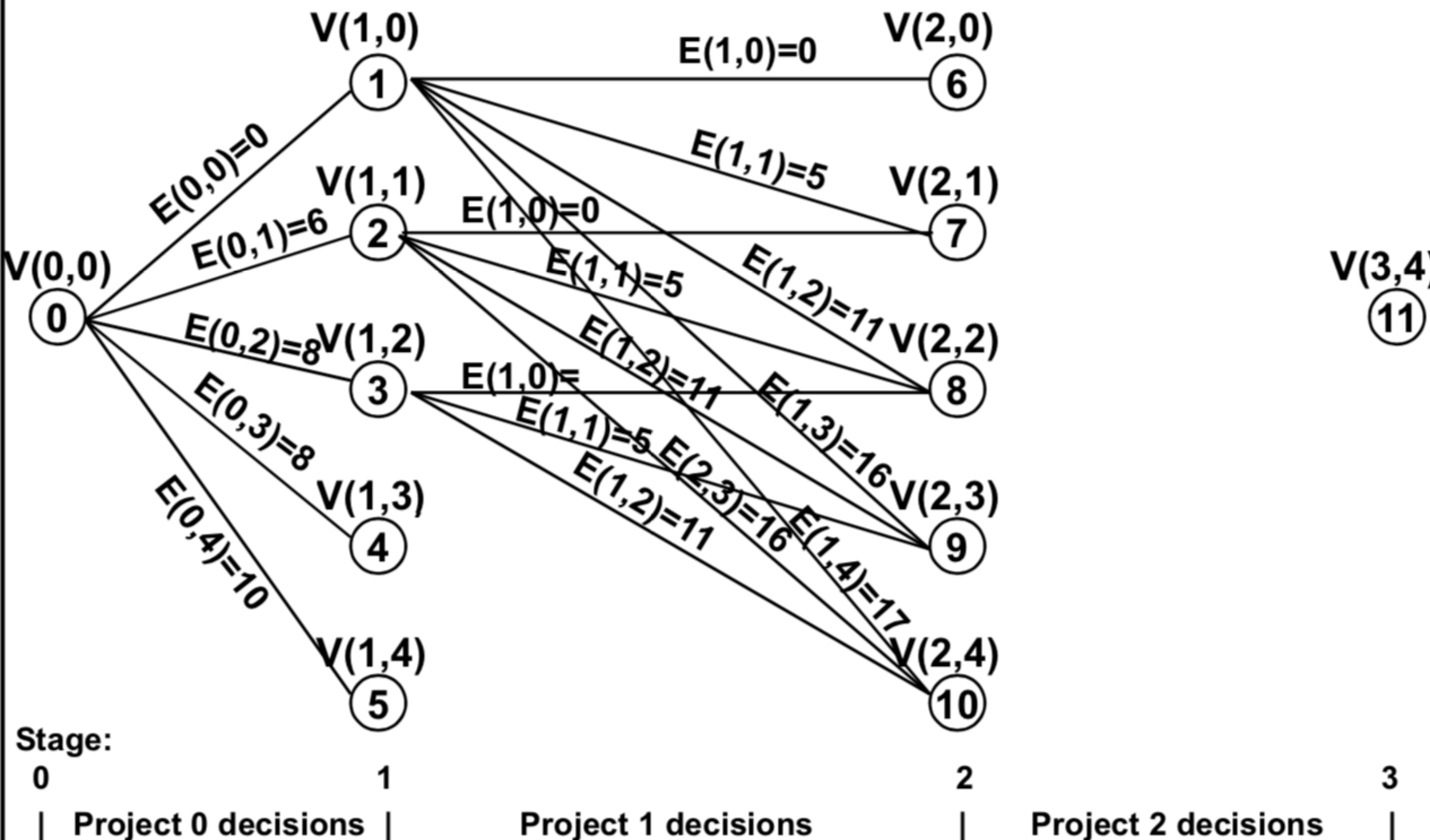
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



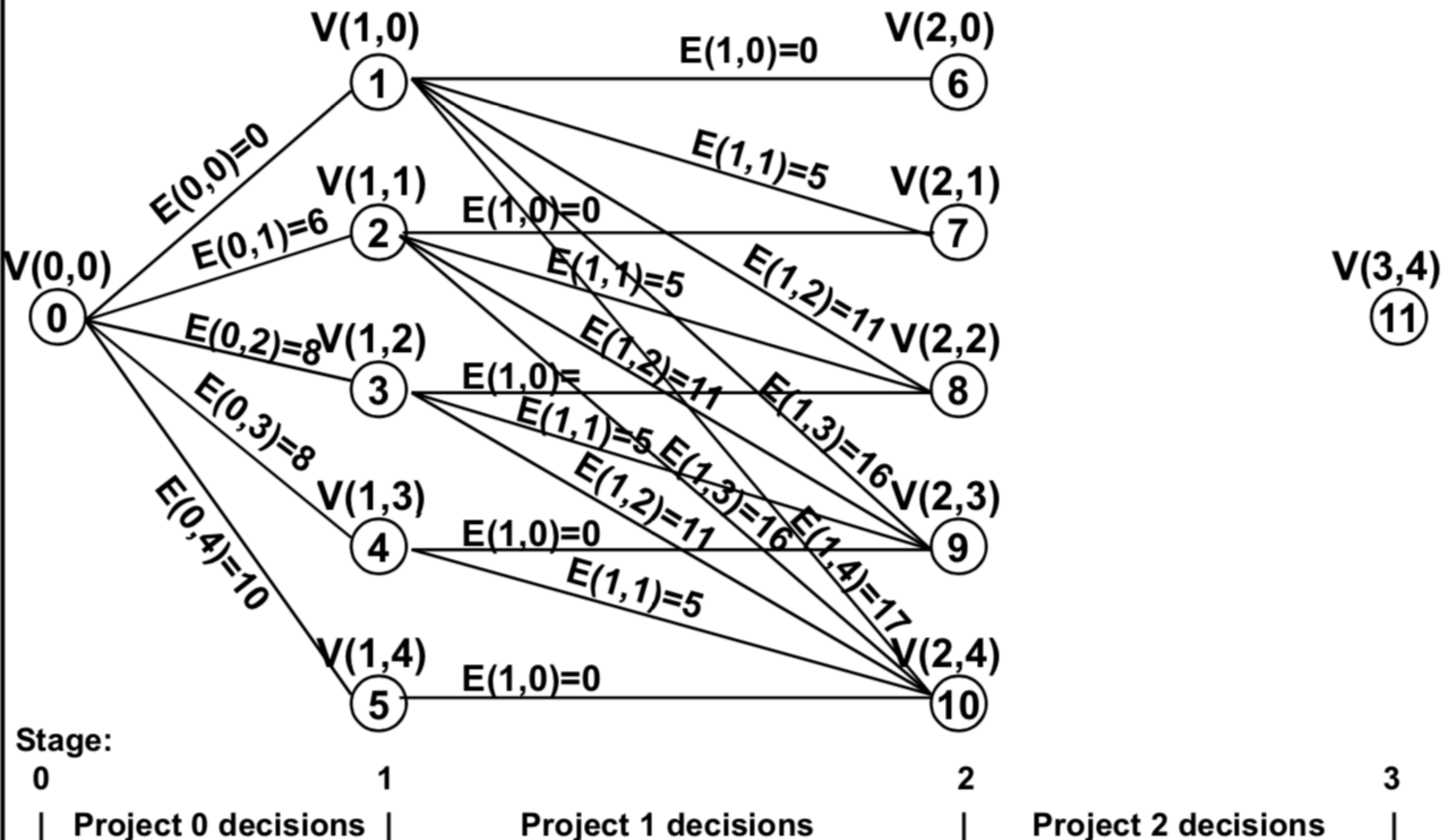
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



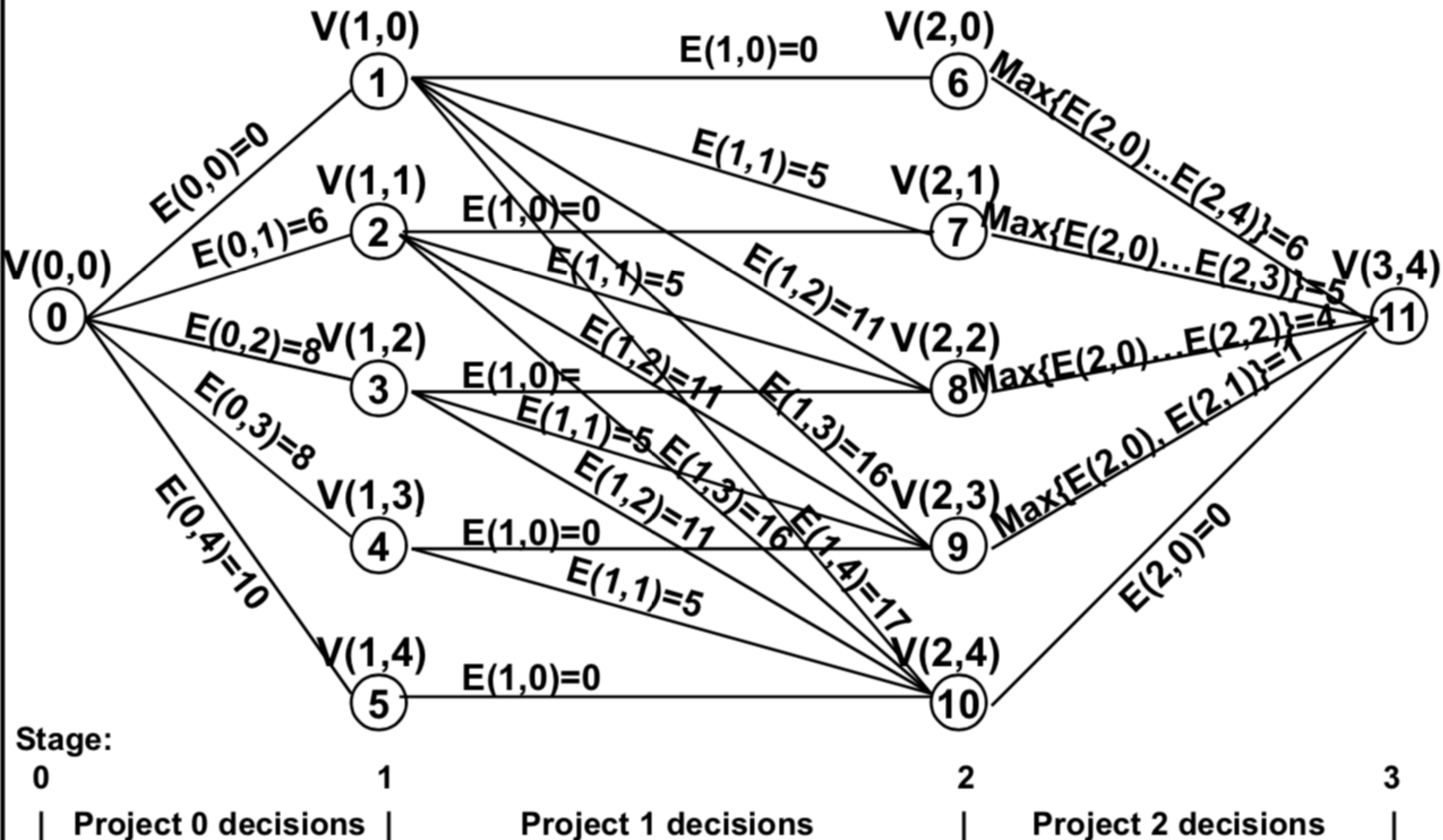
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



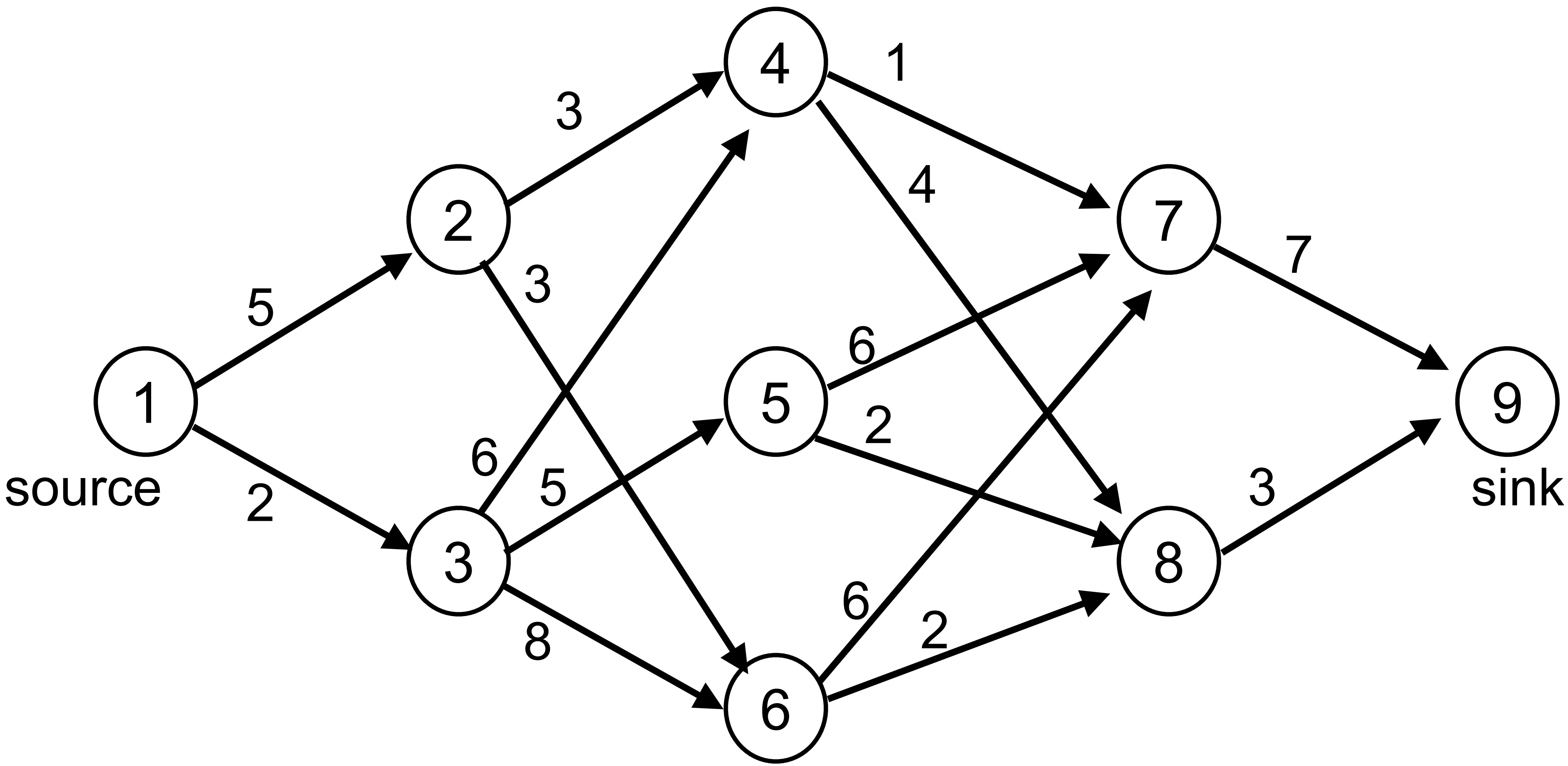
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MIT1_204S10_lec13.pdf

Dynamic programming graph: feasible



Ex 02: Find min cost path

- Using forward approach
- Using backward approach



Summary

- Multi stage graph
- Forward approach
- Backward approach