

Design and Analysis of Algorithms

L27: Multi-Stage Graphs Dynamic Programming

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

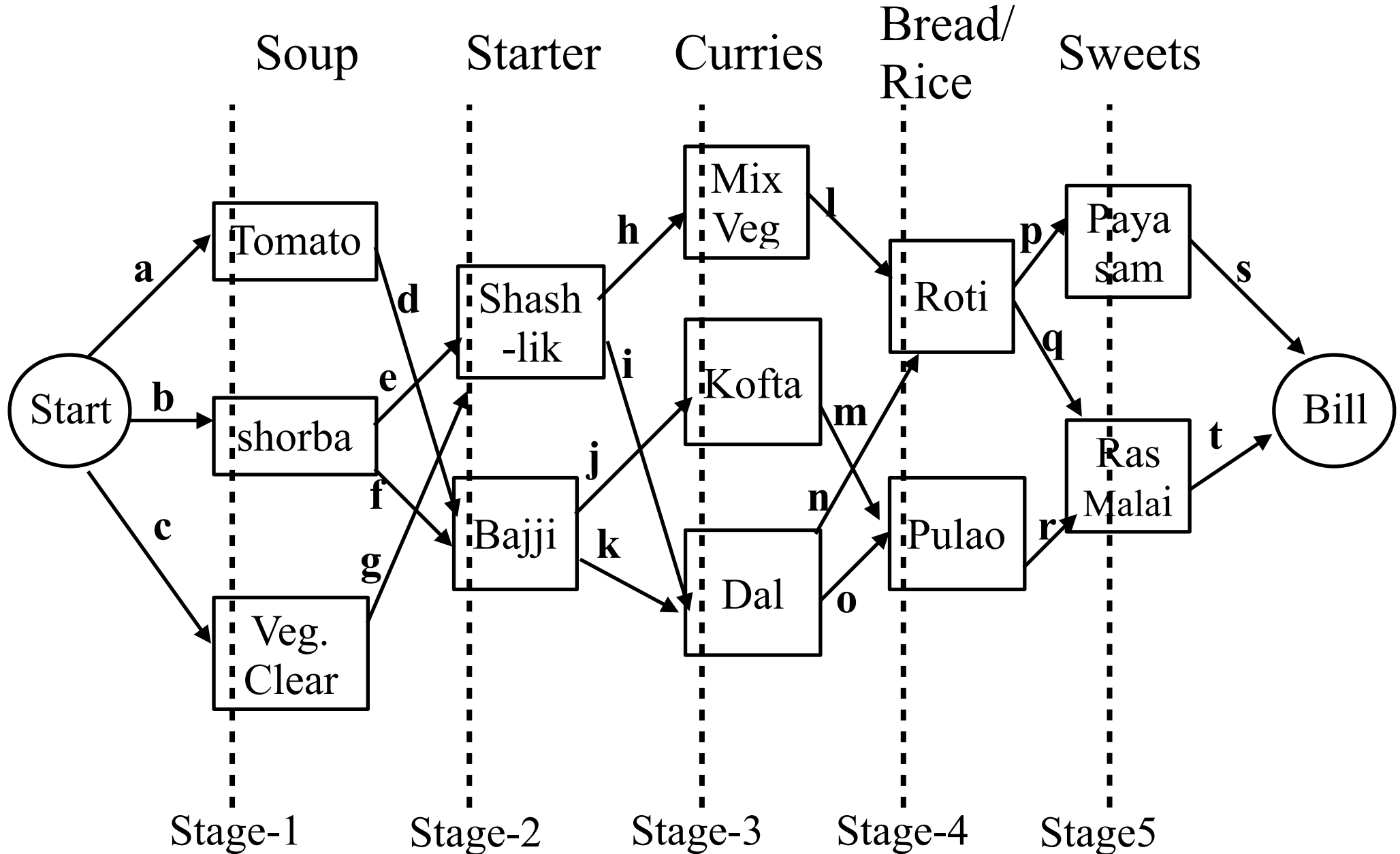
Resources

- Text book 2: Horowitz
 - Sec 5.1, 5.2, 5.4, 5.8, 5.9
- Text book 1: Levitin
 - Sec 8.2–8.4
- <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
- RI: Introduction to Algorithms
 - Cormen et al.

Consider Restaurant Ordering

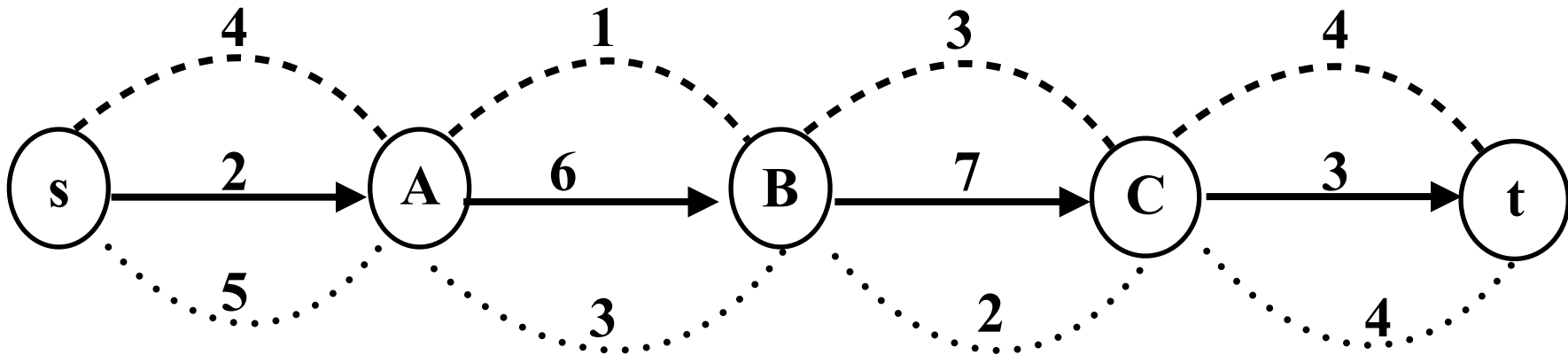
- Food order and serving
 - Soups
 - Starters
 - Main course (curries)
 - Breads/Rice
 - Sweets
 - Mouth freshners
- Each happens in stages
 - Want meal with minimum cost w/ 1 item in each stage
 - Have multiple choices in each stage.
 - Draw a multi-stage graph

Multi-Stage Graph: Restaurant



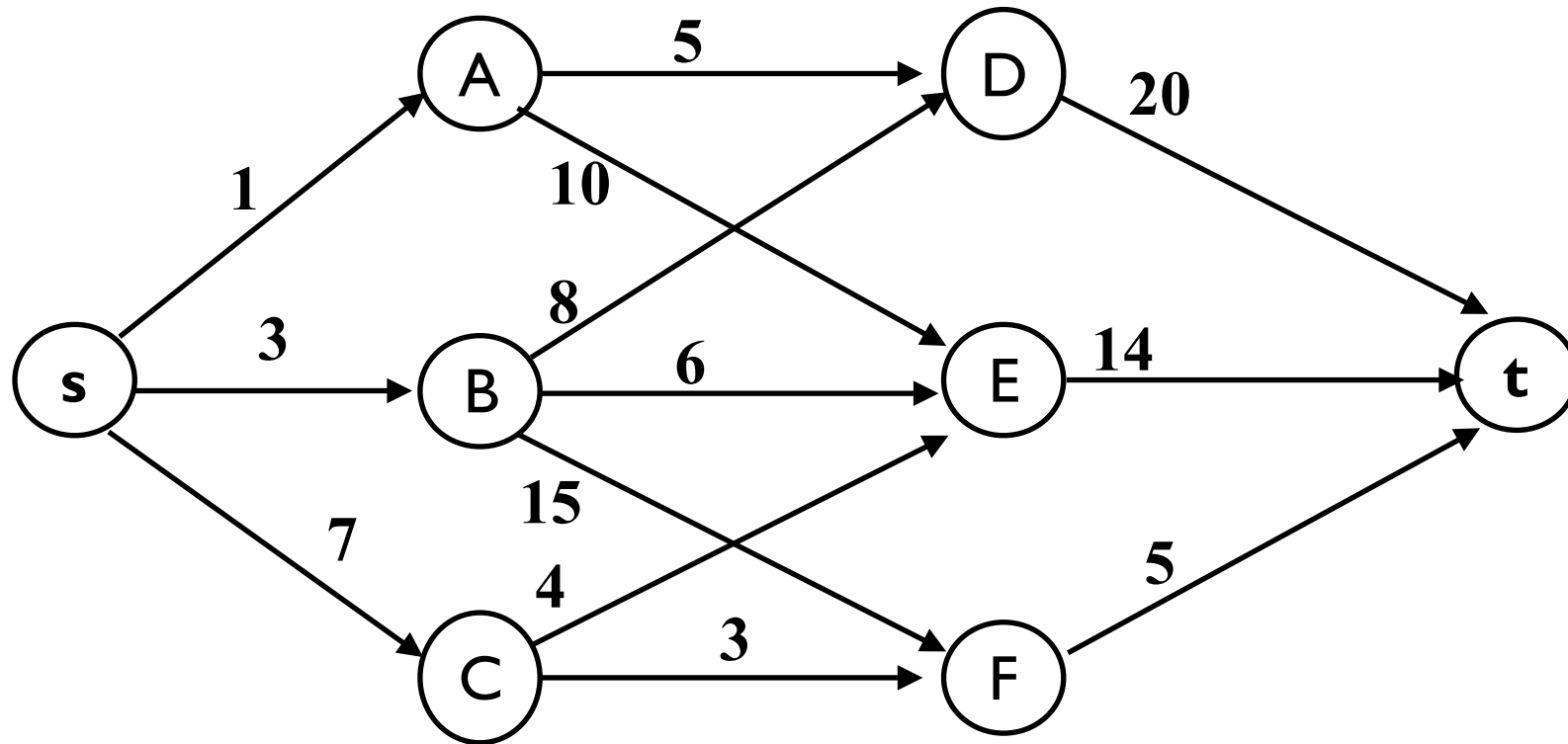
Simple Multi-Stage Graph

- Find shortest path from s to t



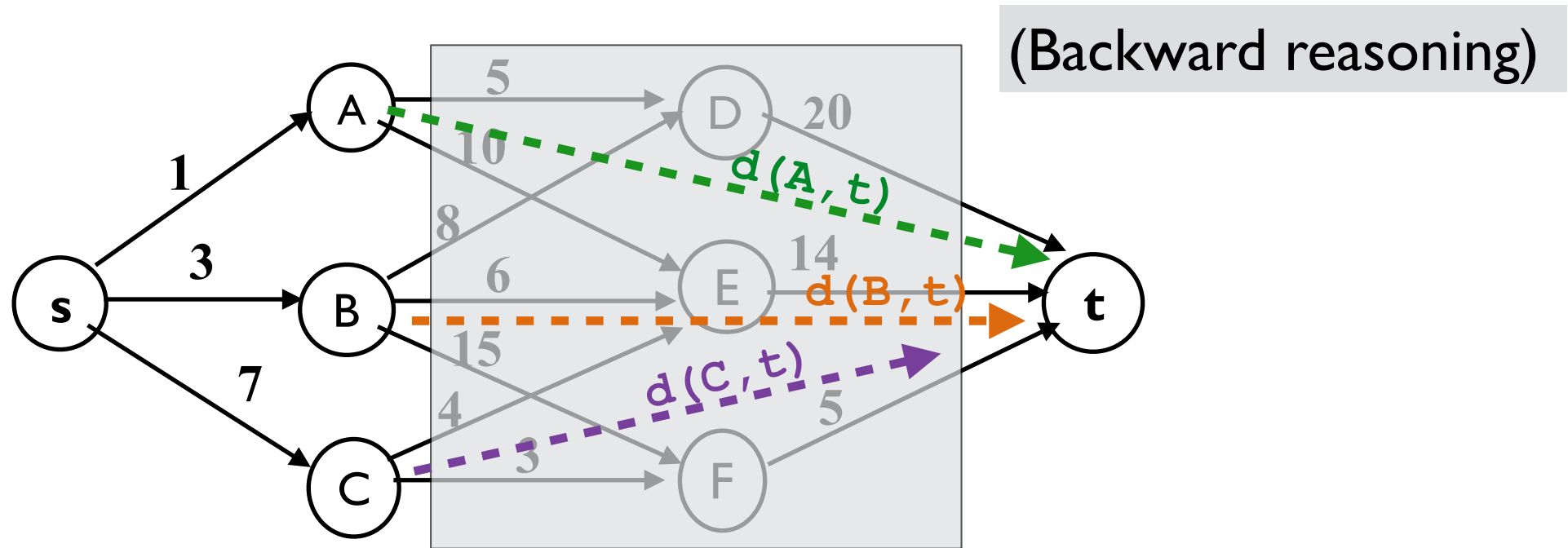
– Q: Does Greedy approach work?

Multistage Graph: Shortest Path



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

Dynamic Programming: Forward Approach



$$d(s, t) = \min\{1 + d(A, t), 3 + d(B, t), 7 + d(C, t)\}$$

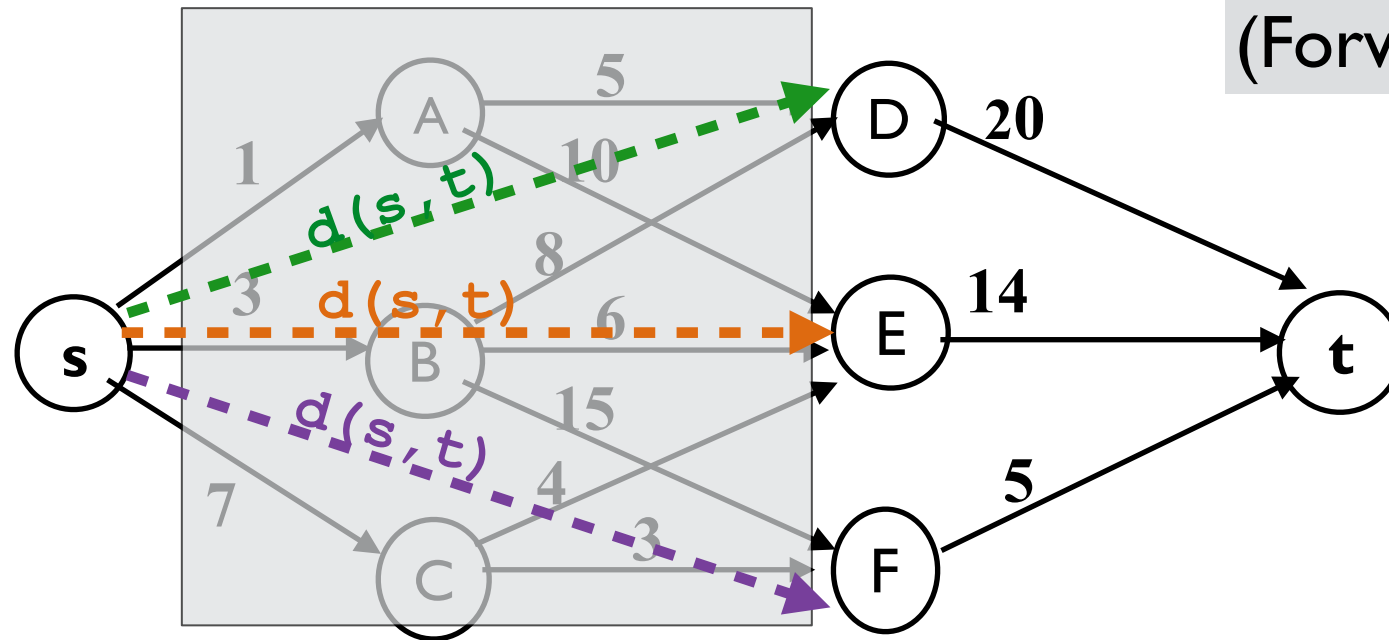
$$d(A, t) = \min\{5 + d(D, t), 10 + d(E, t)\} = \min(25, 24) = 24$$

$$d(B, t) = \min(8 + d(D, t), 6 + d(E, t), 15 + d(F, t)) \\ = \min\{8 + 20, 6 + 14, 15 + 5\} = 20$$

$$d(C, t) = \min\{4 + d(E, t), 3 + d(F, t)\} = \min\{18, 8\} = 8$$

$$d(s, t) = \min\{1 + 24, 3 + 20, 7 + 8\} = 15$$

Dynamic Programming: Backward Approach



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

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

$$d(s, E) = \min(d(s, A) + 10, d(s, B) + 6, d(s, C) + 4) \\ = \min\{1 + 10, 3 + 6, 7 + 4\} = 9$$

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

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

Dynamic Programming: Applications

- Resource allocation problem
- Consider the following scenario:
 - A team of 3 students are asked to complete 4 assignments.
 - Any student can choose to complete all 4 or none.
 - At a time, only one person will do assignment.
 - First P_1 , then P_2 , and then P_3 (in that order)
 - However, no assignment is to be done by 2 students
 - Duplicate (wasted) efforts to be avoided
 - All the 4 assignments need to be completed.
 - Depending upon assignments completed by a students, different marks are awarded as shown next

Resource (Assignment) Allocation

- Marks evaluation for assignment done by a person

Person → Assignments ↓	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

- Q: How to allocate assignments to team members so as to get maximum marks

Resource (Assignment) Allocation

- Possible allocations...
- P_1 : 0As:
 - P_2 :4A, P_3 :0A:
 - Marks: $0+10+0=10$
 - P_2 :3A, P_3 :1A,
 - Marks: $0+8+5=13$
 - P_2 :2A, P_3 :2As,
 - Marks: $0+7+5=12$
 - P_2 :1A, P_3 :3As,
 - Marks: $0+4+6=10$
 - P_2 :0A, P_3 :4As,
 - Marks: $0+0+6=6$

$P \rightarrow$ $A \downarrow$	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

Resource (Assignment) Allocation

- Possible allocations...
- $P_1: 1A$:
 - $P_2: 3A, P_3: 0A$,
 - Marks: $2+8+0=10$
 - $P_2: 2A, P_3: 1A$ s,
 - Marks: $2+7+5=14$
 - $P_2: 1A, P_3: 2A$ s,
 - Marks: $2+4+5=11$
 - $P_2: 0A, P_3: 3A$ s,
 - Marks: $2+0+6=8$

P → A ↓	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 : 2As.
 - P_2 :2A, P_3 :0A:
 - Marks: $5+7+0=12$
 - P_2 :1A, P_3 :1A,
 - Marks: $5+4+5=14$
 - P_2 :0A, P_3 :2As,
 - Marks: $5+0+5=10$

P → A ↓	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 : 3As
 - P_2 :1A, P_3 :0A:
 - Marks=7+4+0=11
 - P_2 :0A, P_3 :1A:
 - Marks=7+0+5=12
- P_1 :4As:
 - P_2 :0A, P_3 :0A
 - Marks=8

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

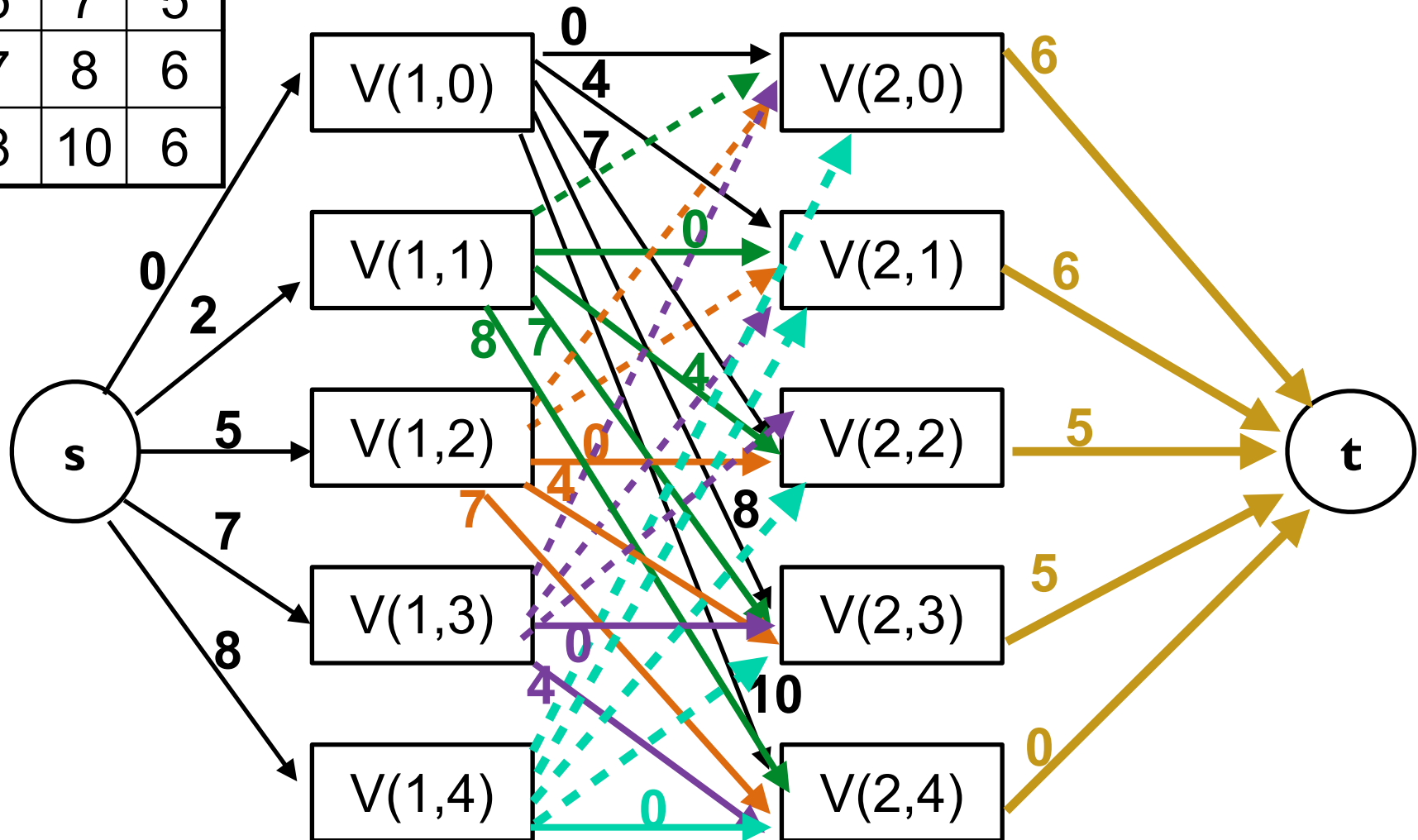
Resource (Assignment) Allocation

- Construction of Multistage graph
- The graph has 4 stages
 - Stage 1: Start
 - Stage 2: P_1 does some assignments
 - Stage 3: P_2 - some remaining assignments
 - Stage 4: P_3 - all remaining assignments
 - The end stage: all assignments are done
- 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 num of assignments done.
 - $1 \leq i < 3$; and $0 \leq j \leq 4$
- Start, end stage has one vertex each

$P \rightarrow$ $A \downarrow$	P1	P2	P3
1	2	4	5
2	5	7	5
3	7	8	6
4	8	10	6

Multistage Graph

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



Q: Find max marks using DP Forward approach?

Q: Find max marks using DP Backward approach?

DP Forward approach: Steps

- Generate multi-stage graph in forward direction
 - Start at source node s
 - Compute $V(i, j)$ and edge cost as graph is built
 - Keep track of predecessor $P(i)$ of each node that yields highest $V(i, j)$
 - Eliminates non-optimal subsequences (pruning)
 - Eliminate infeasible edges/nodes as graph is built
 - Construct solution by tracing back from sink t to source s using predecessor $P(i)$ variable

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[1:k] is a minimum 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  $v_j \rightarrow v_r$  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

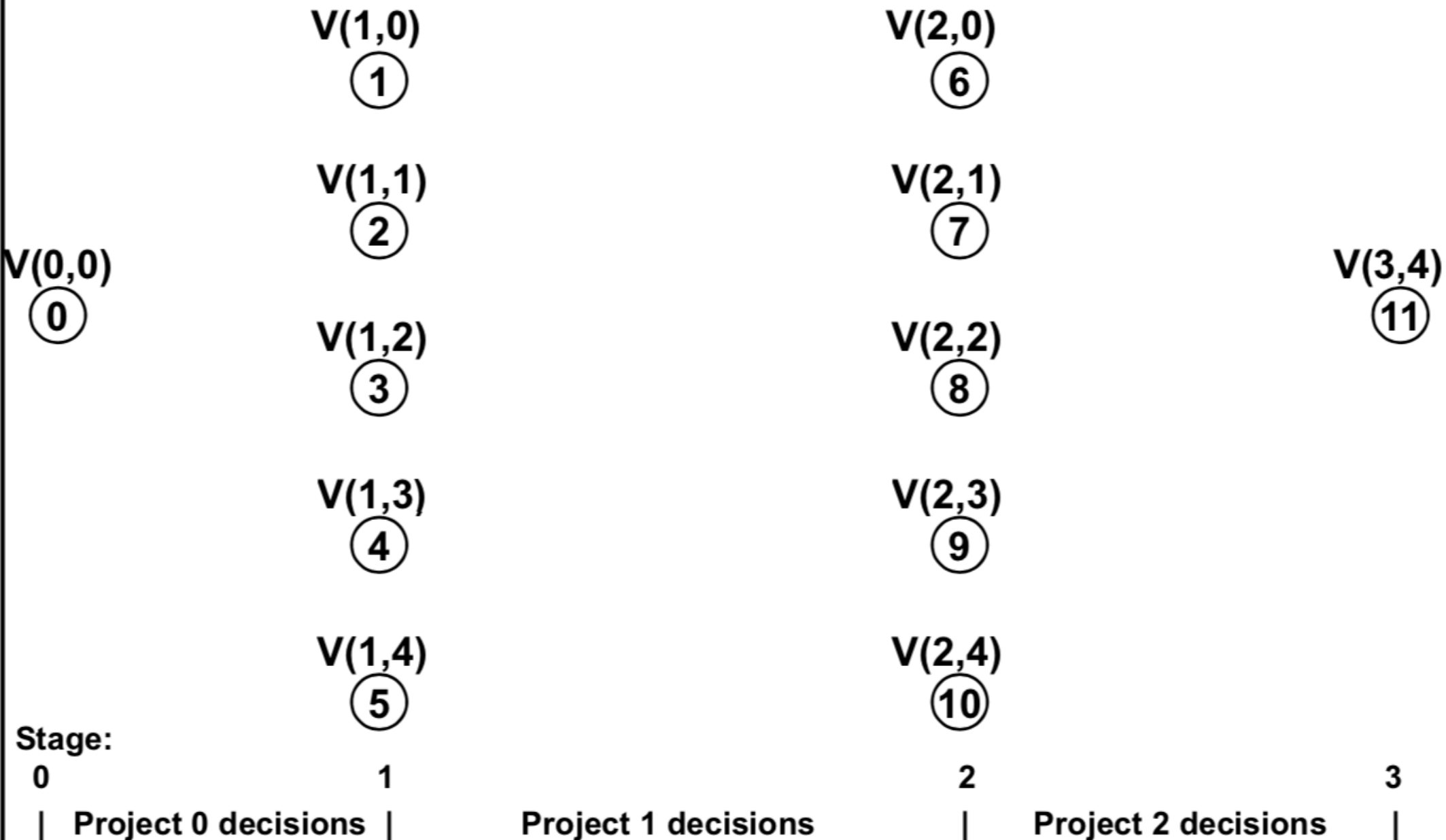
```
Algo: BGraph (Graph G, int k, int p[])  
// i/p k-stage graph n vertices indexed in order of stages.  
//      edge  $c(i, j)$  is cost of edge  $v_i \rightarrow v_j$   
//       $p[1:k]$  is a minimum cost path  
    float bcost[maxsize]; int d[maxsize], r;  
    bcost[n]=0.0  
    for j=2 to n // 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;  
    for j=k-1 to 2  
        p[j]=d[p[j+1]]
```

Ex: Build Multistage graph

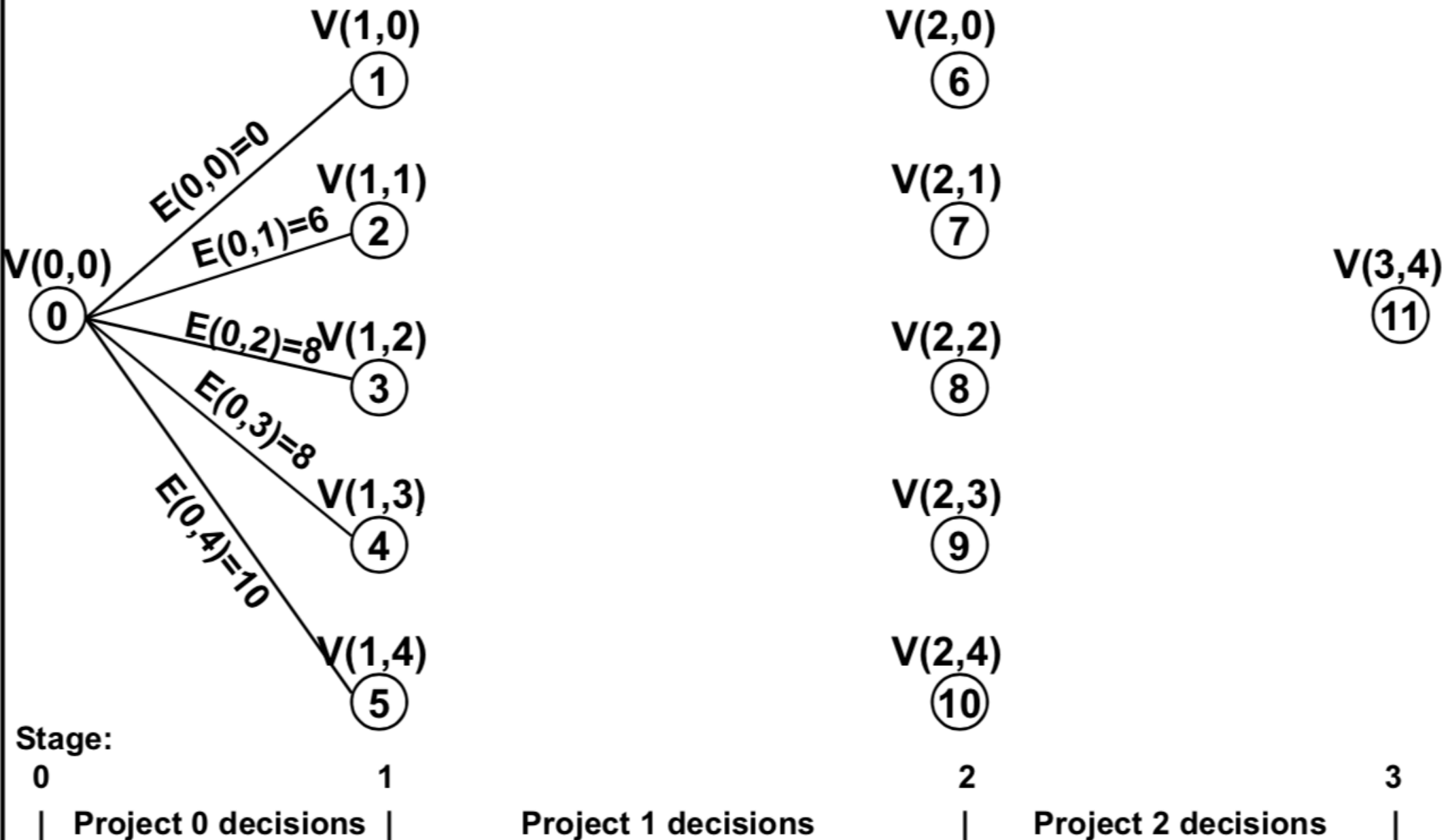
src: https://ocw.mit.edu/courses/civil-and-environmental-engineering/I-204-computer-algorithms-in-systems-engineering-spring-2010/lecture-notes/MITI_204S10_lec13.pdf

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

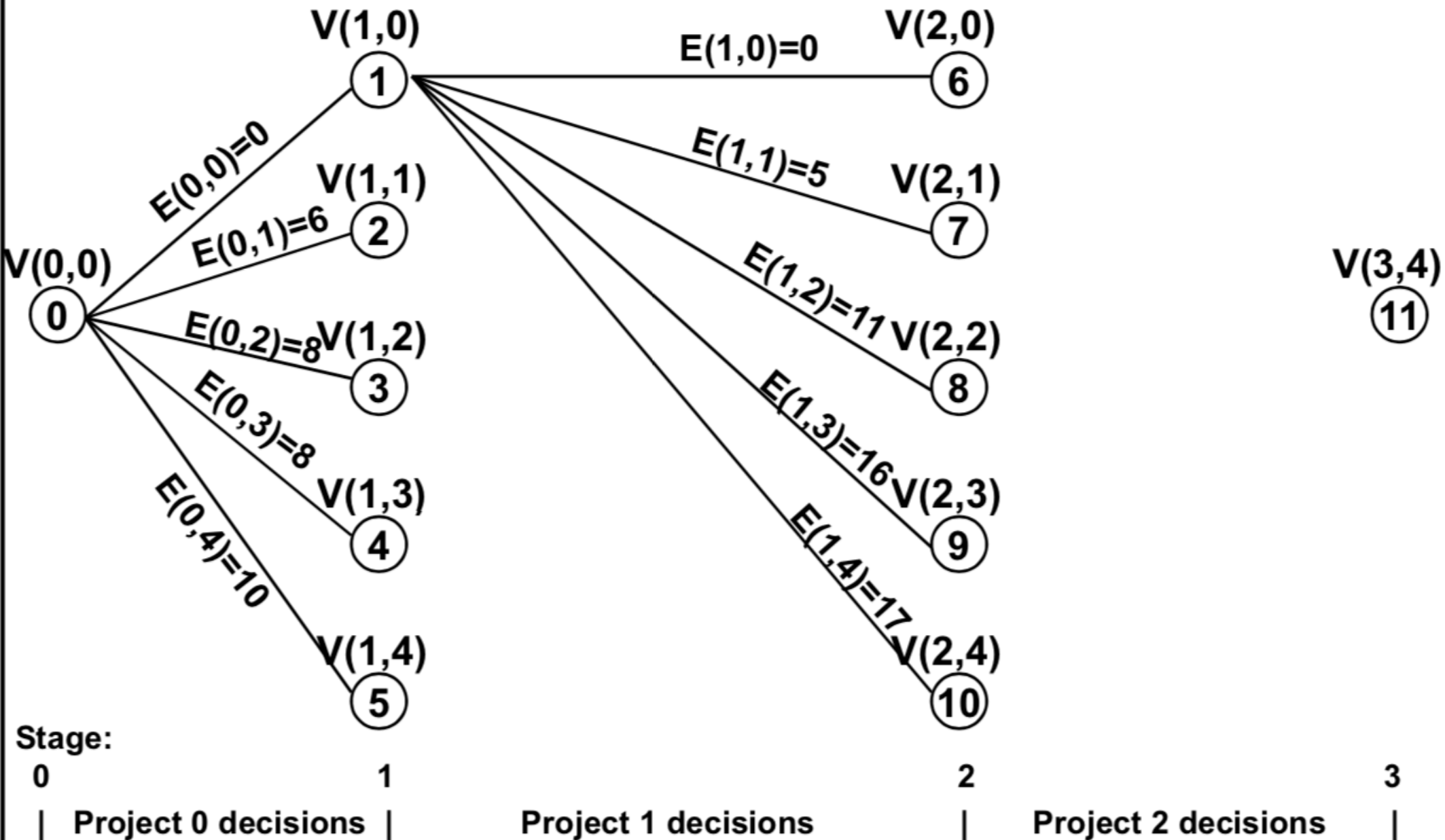
Dynamic programming graph: feasible



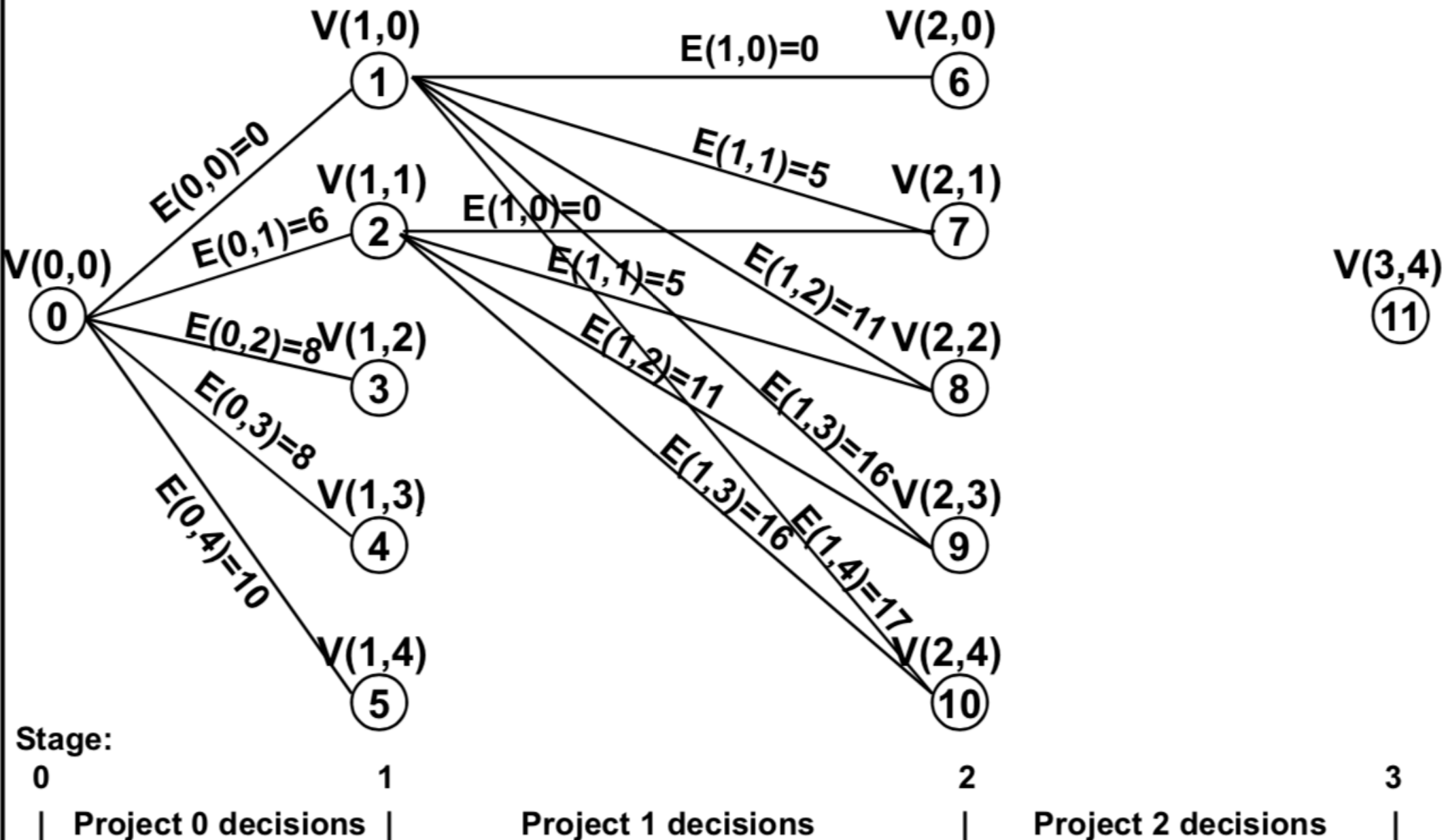
Dynamic programming graph: feasible



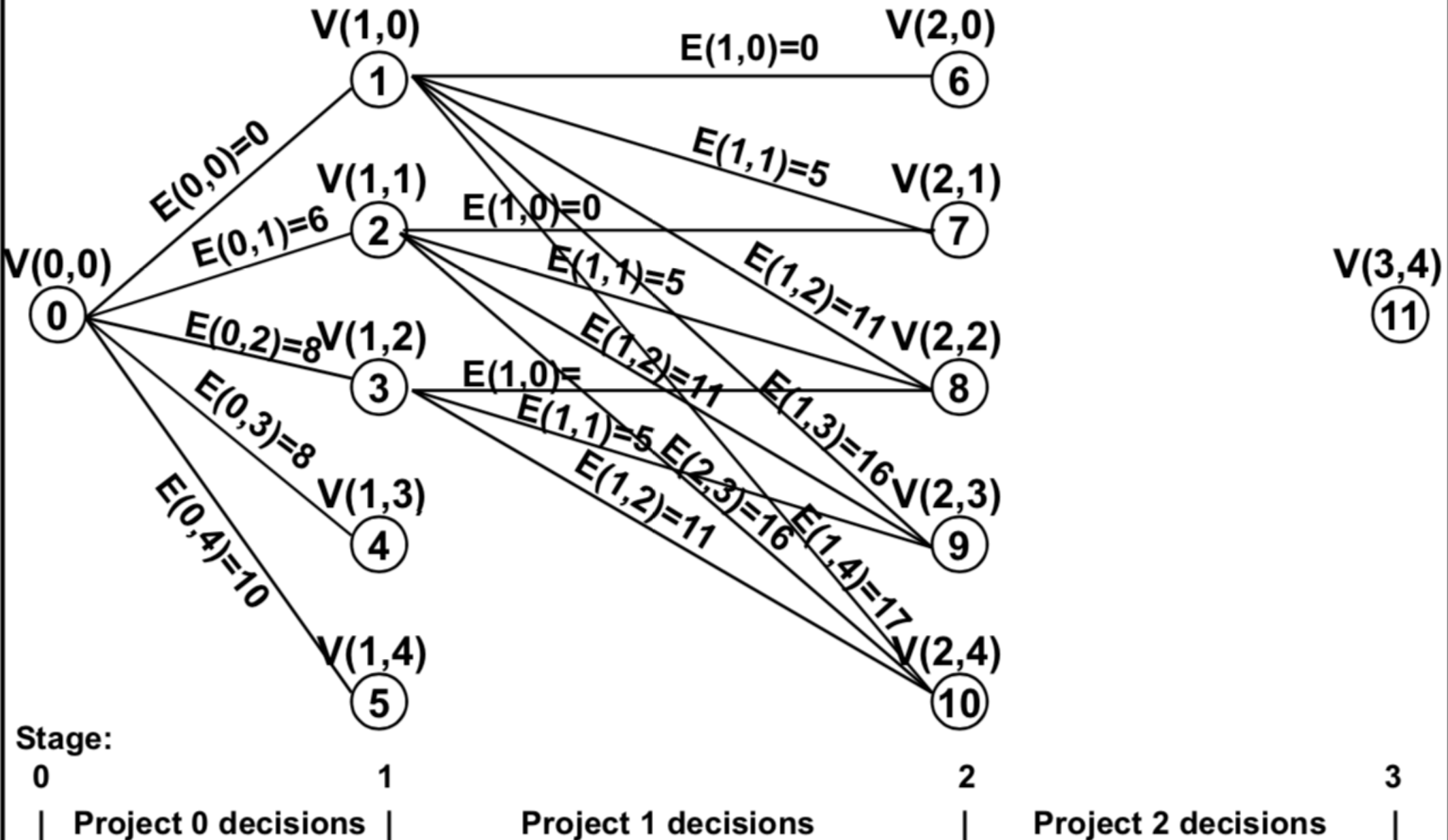
Dynamic programming graph: feasible



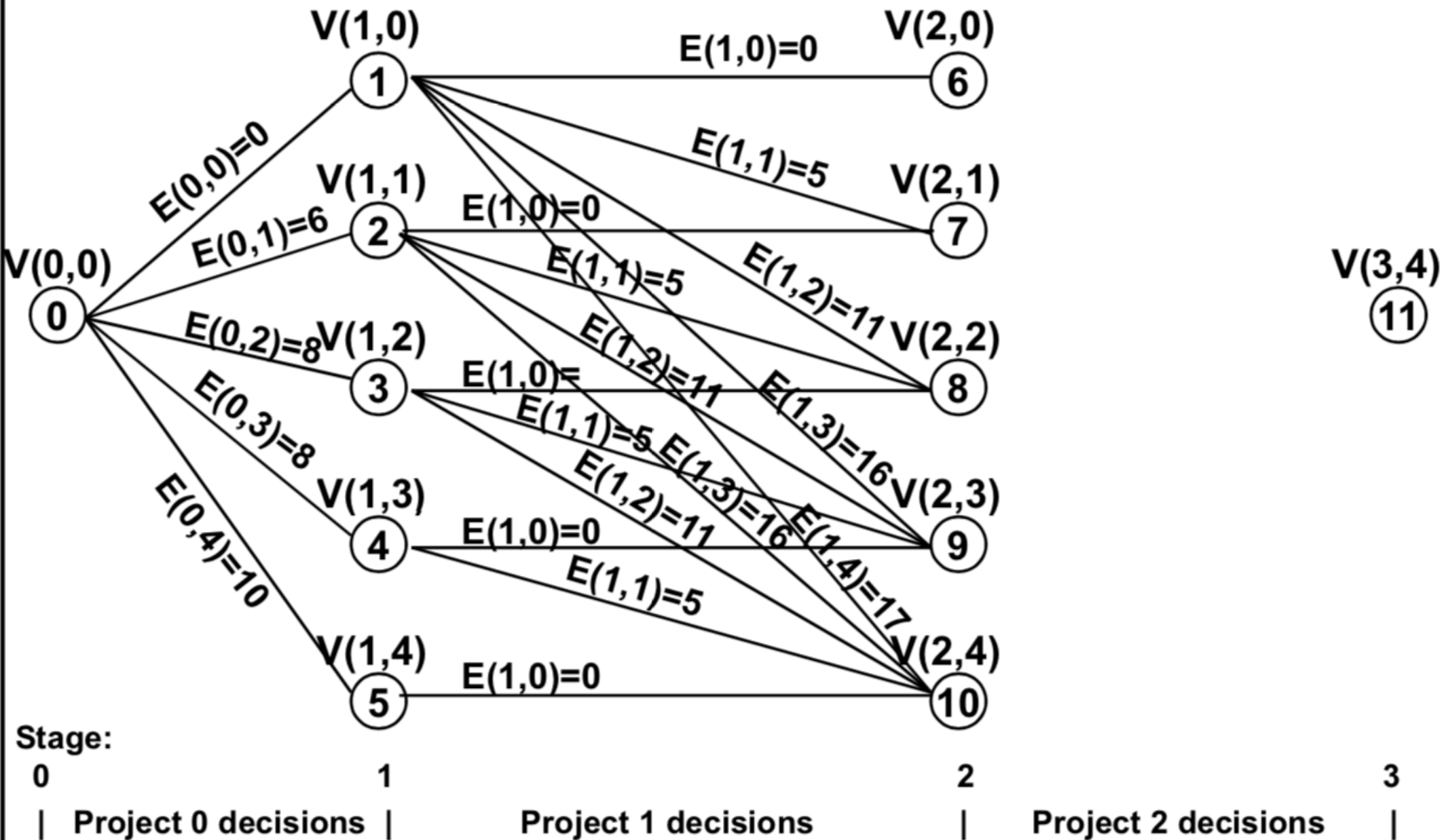
Dynamic programming graph: feasible



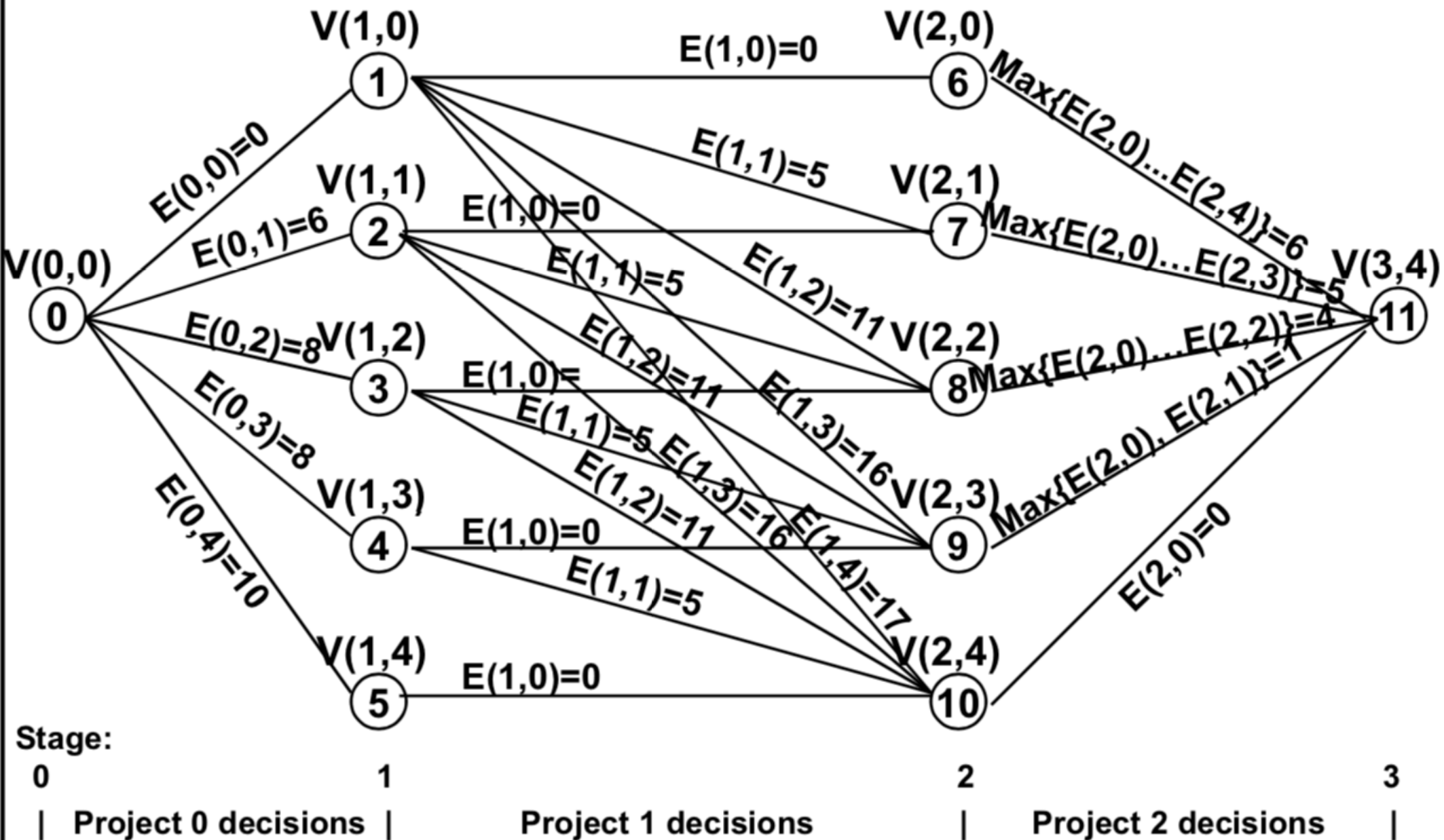
Dynamic programming graph: feasible



Dynamic programming graph: feasible

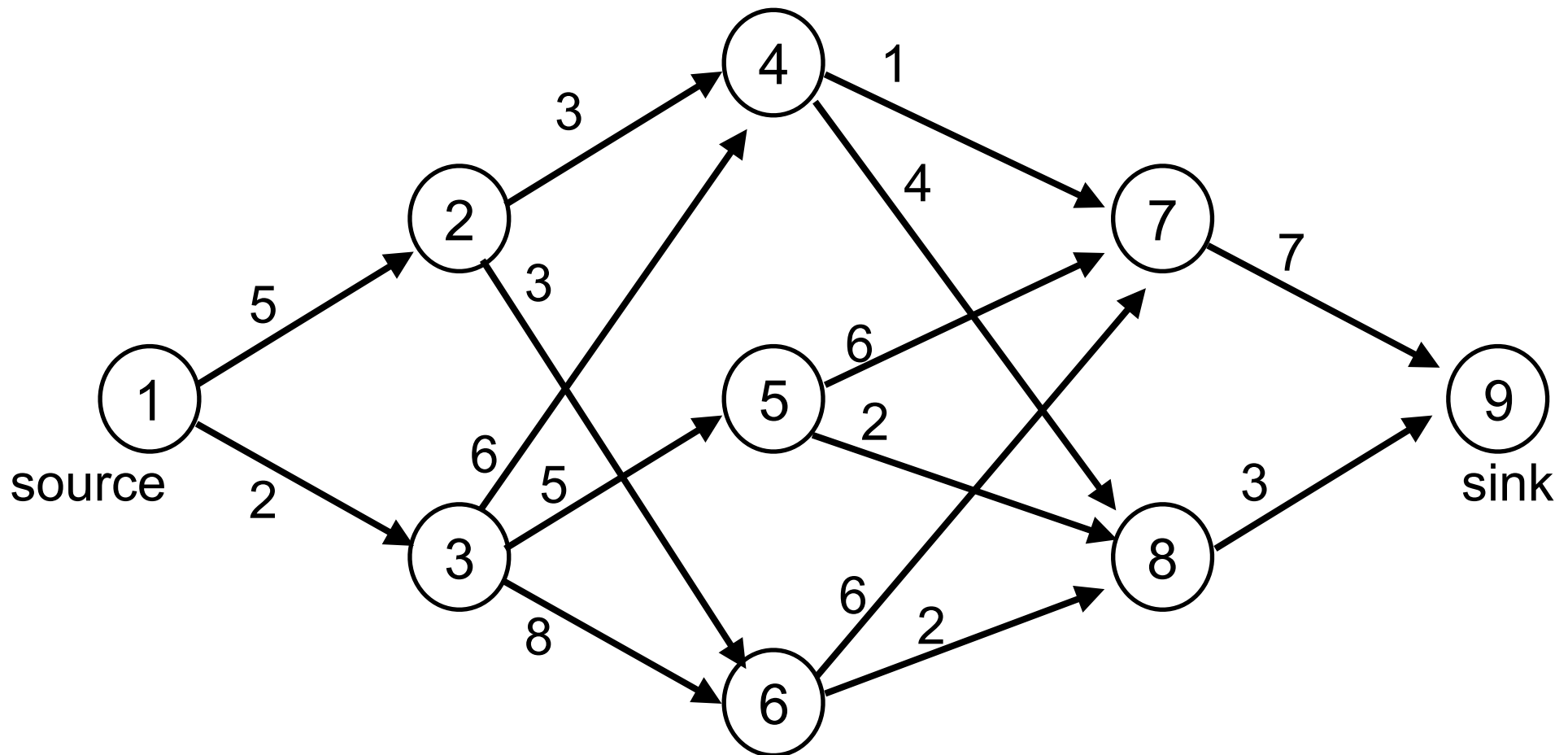


Dynamic programming graph: feasible



Ex 02: Find min cost path

- Using forward approach
- Using backward approach



Summary

- Multi stage graph
- Forward approach
- Backward approach