

CSC 226

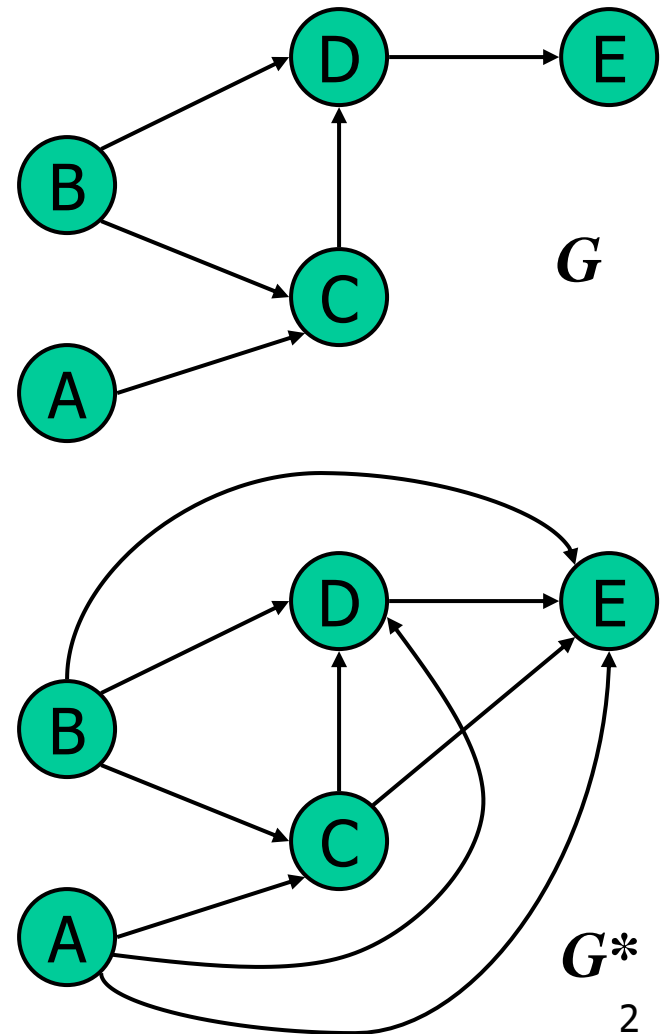
Algorithms and Data Structures: II

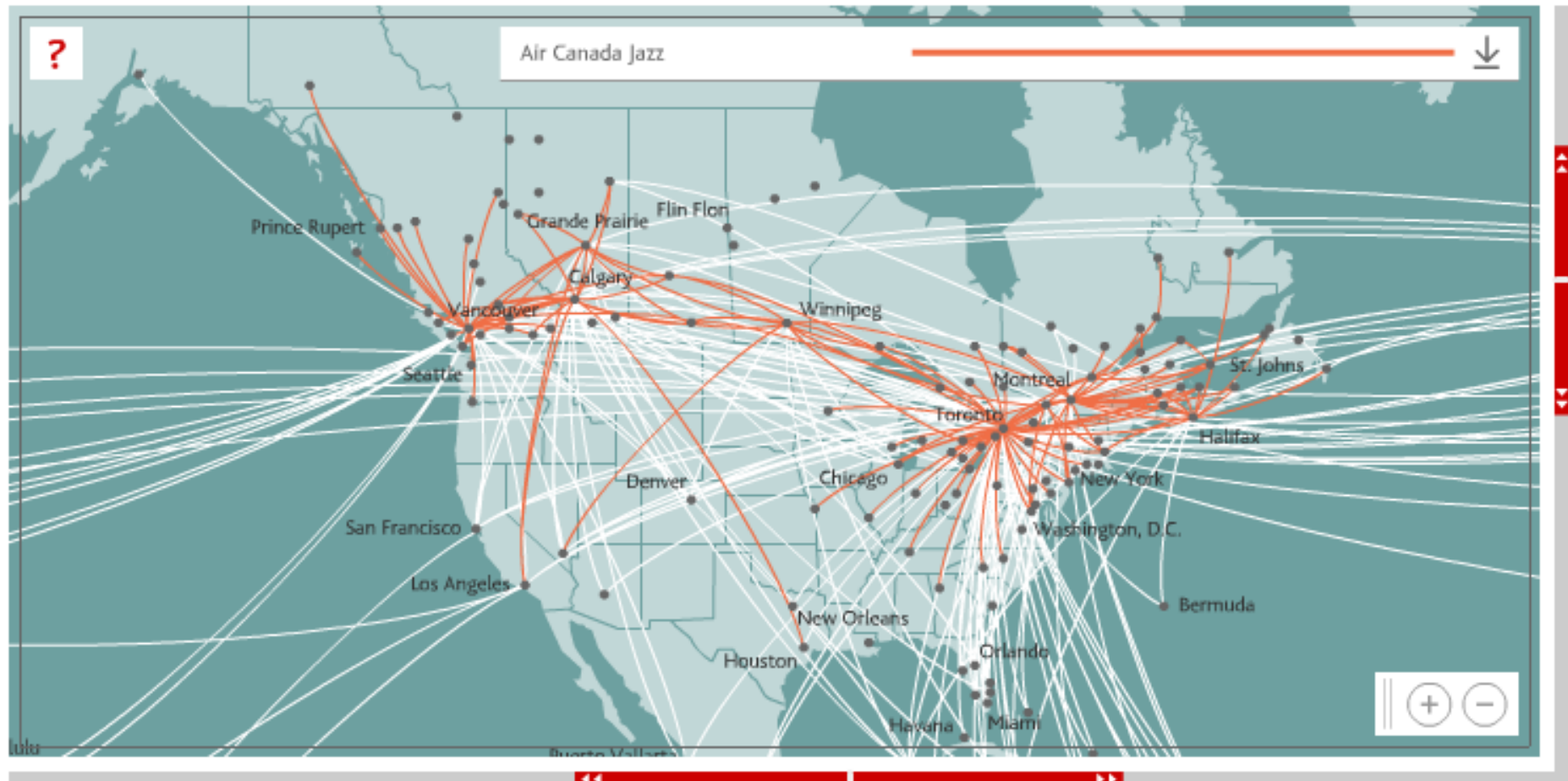
Rich Little

rlittle@uvic.ca

Transitive Closure

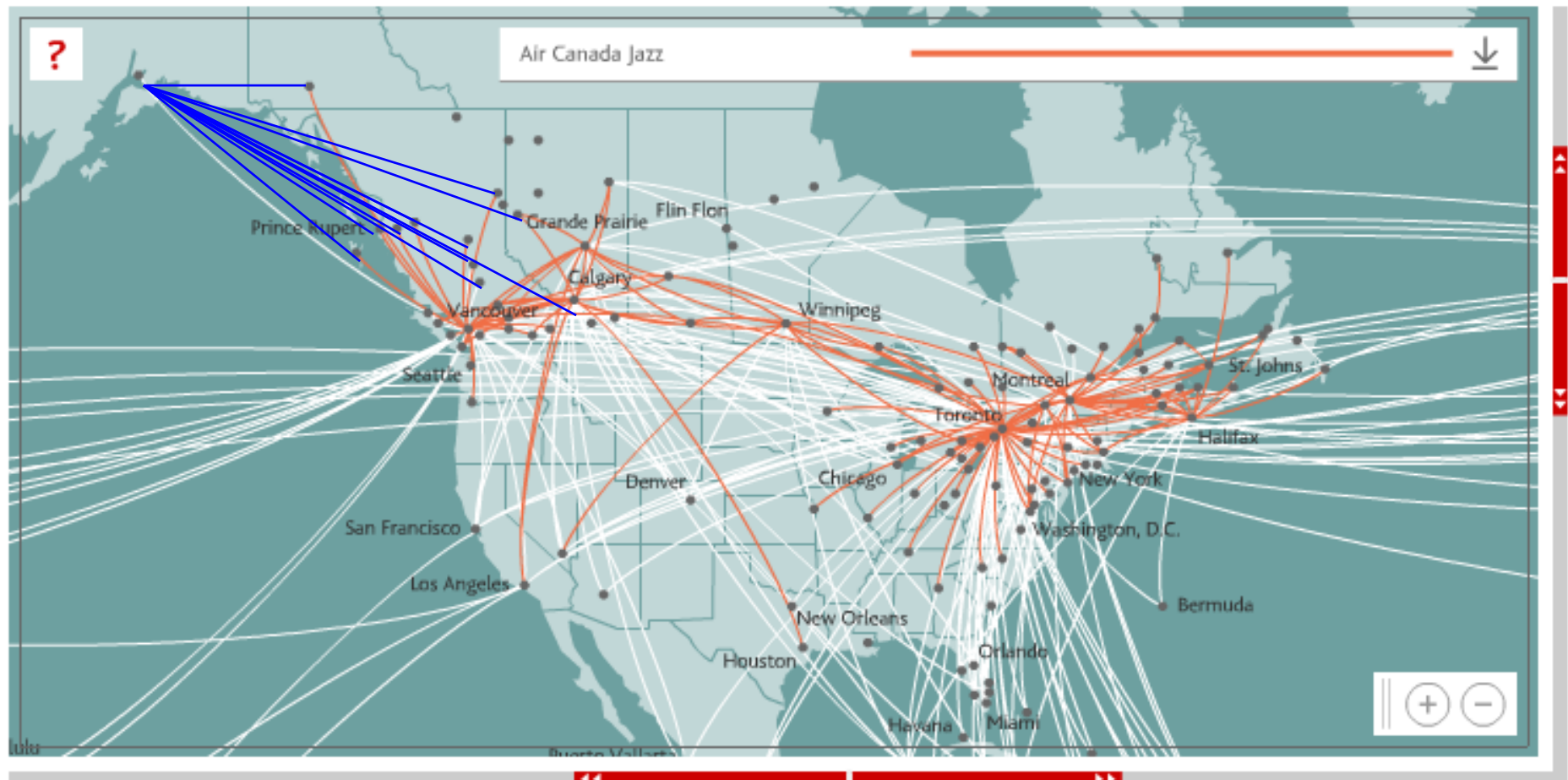
- Given a digraph G , the transitive closure of G is the digraph G^* such that
 - G^* has the same vertices as G
 - if G has a directed path from u to v ($u \neq v$), G^* has a directed edge from u to v
- The transitive closure provides reachability information about a digraph





[Exit Map](#)

Some routes or destinations served by Air Canada may not appear on this map due to seasonal service. Routes are subject to change based on demand or seasonal requirements. All rights reserved ©2003.



[Exit Map](#)

Some routes or destinations served by Air Canada may not appear on this map due to seasonal service. Routes are subject to change based on demand or seasonal requirements. All rights reserved ©2003.

Transitive Closure Algorithms

- The transitive closure graph G^* of a graph G can be constructed with four basic algorithms:
 - DFS or BFS for each vertex in the graph
 - Identify SCC and transitive closure on DAG
 - Floyd-Warshall's algorithm
 - Matrix multiplication
- Time complexity of transitive closure is basically $O(n^3)$

Transitive Closure using Floyd-Warshall's Algorithm

- Floyd-Warshall's algorithm constructs transitive closures with a simple, dynamic programming algorithm
- The algorithm is identical to Floyd's all-pairs-shortest-path algorithm
- In it we try to take advantage of the fact that some of the paths between vertices overlap.
 - i.e. don't follow the same path over and over again.

Algorithm Design Technique: Dynamic Programming

- The dynamic programming technique is used mainly for optimization problems, where we wish to find the “best” way of doing something.
- The number of “best” ways is often exponential and thus the brute-force way of solving the problem (e.g., enumerating all solutions and selecting the “best” one) is exponential.
- If the problem has a certain “structure”, then we can exploit that and arrive at a polynomial time algorithm.

Algorithm Design Technique

Dynamic Programming

- **Simple subproblems**

- There is a way to break down the problem into simple subproblems of similar structure
- The subproblems can easily be defined with a few indices (i.e., i, j, k)

- **Subproblem optimality**

- An optimal solution to the global problem is a composition of optimal subproblem solutions using a simple combining operation

- **Subproblem overlap**

- Optimal solutions to unrelated subproblems can have subproblems in common. Such overlap is recognized and improves the efficiency significantly
- This is particularly important to dynamic programming algorithms, because it takes advantage of *memorization*

Classic Dynamic Programming Problems

- Invented by Richard Bellman 1940-53
- 0-1 Knapsack problem (Integer Programming)
 - Pack a knapsack full of items and maximize the value or utility and stay under a maximum weight

$$\max \sum_{k \in S} b_k \text{ subject to } \sum_{k \in S} w_k \leq W$$

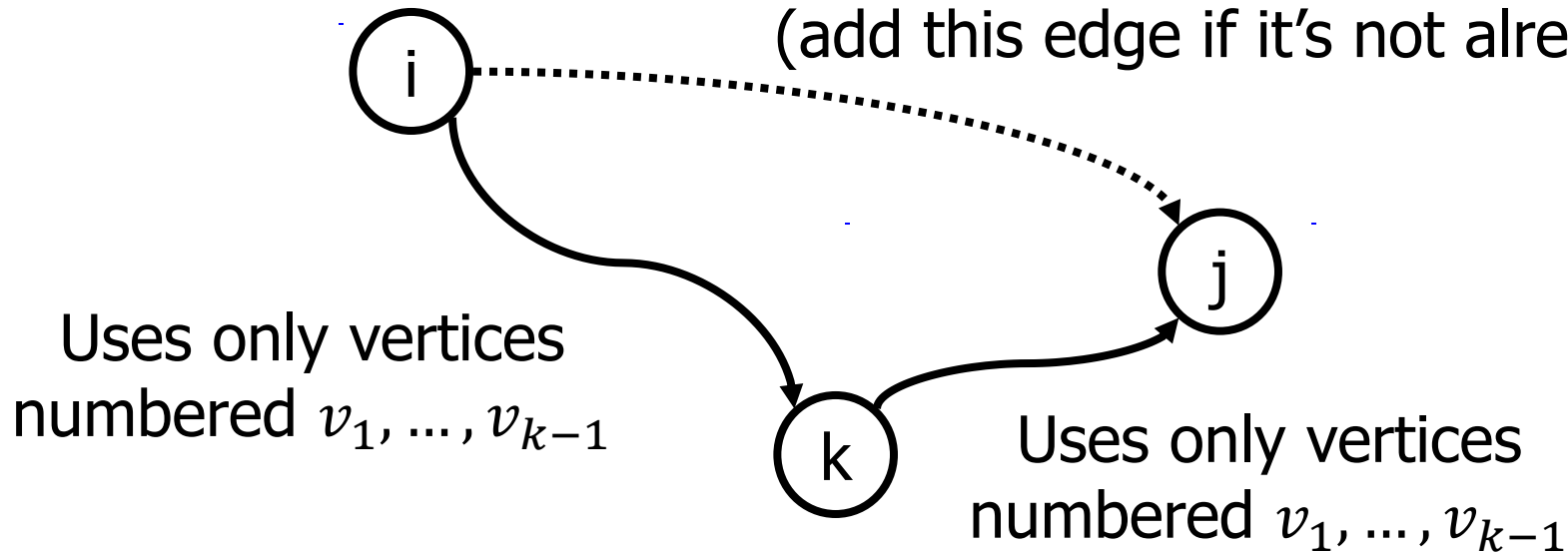
- Computing binomials
- Longest common subsequence
- Floyd-Warshall algorithm
- Floyd's all pairs shortest path algorithm
- Earley parsing algorithm
- General path finding
- Kleene's algorithm

Floyd-Warshall Transitive Closure



- Idea #1: Number the vertices v_1, v_2, \dots, v_n .
- Idea #2: Consider paths that use only vertices numbered v_1, v_2, \dots, v_k , as intermediate vertices:

Uses only vertices numbered v_1, \dots, v_k
(add this edge if it's not already in)



The Floyd-Warshall Algorithm

- Let G be a digraph with n vertices and m edges.
- Number the vertices, v_1, v_2, \dots, v_n , and let $G_0 = G$
- In the k^{th} iteration we construct G_k by adding to G_{k-1} , edge (v_i, v_j) if G_{k-1} contains (v_i, v_k) and (v_k, v_j)
- **Lemma:** For $i = 1, \dots, n$, digraph G_k has an edge (v_i, v_j) if and only if digraph G has a directed path from v_i to v_j , whose intermediate vertices are in the set $\{v_1, \dots, v_k\}$. In particular, $G_n = G^*$, the transitive closure of G .

Floyd-Warshall's Algorithm



Algorithm FloydWarshall(G):

Input: digraph G with n vertices

Output: transitive closure G^* of G

Let v_1, v_2, \dots, v_n be an arbitrary numbering of vertices in G

for $k \leftarrow 1$ **to** n **do**

$G_k \leftarrow G_{k-1}$

for $i \leftarrow 1$ **to** n , $i \neq k$ **do**

for $j \leftarrow 1$ **to** n , $j \neq i, k$ **do**

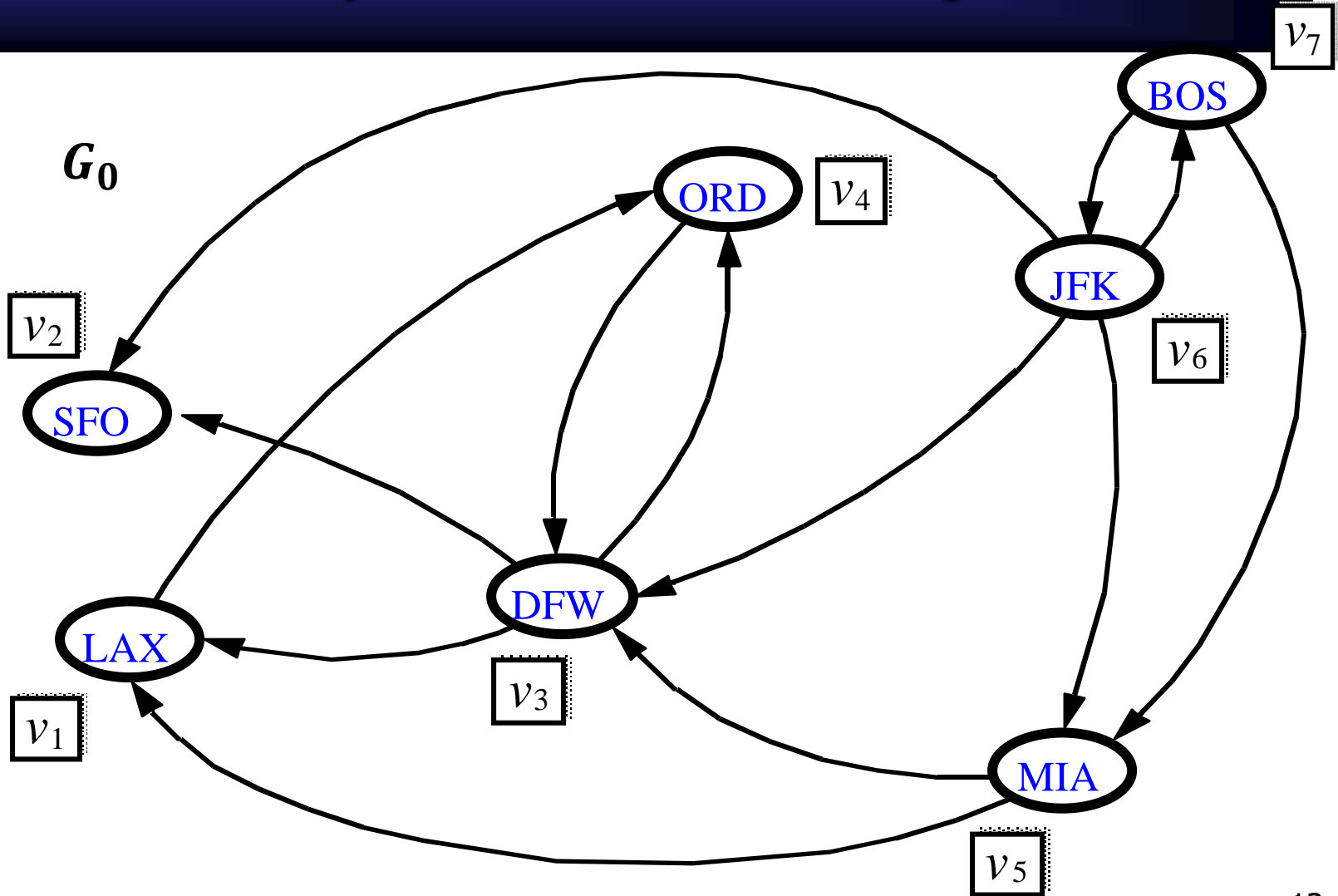
if (v_i, v_k) and (v_k, v_j) are in G_{k-1} **then**

if (v_i, v_j) is not in G_k **then**

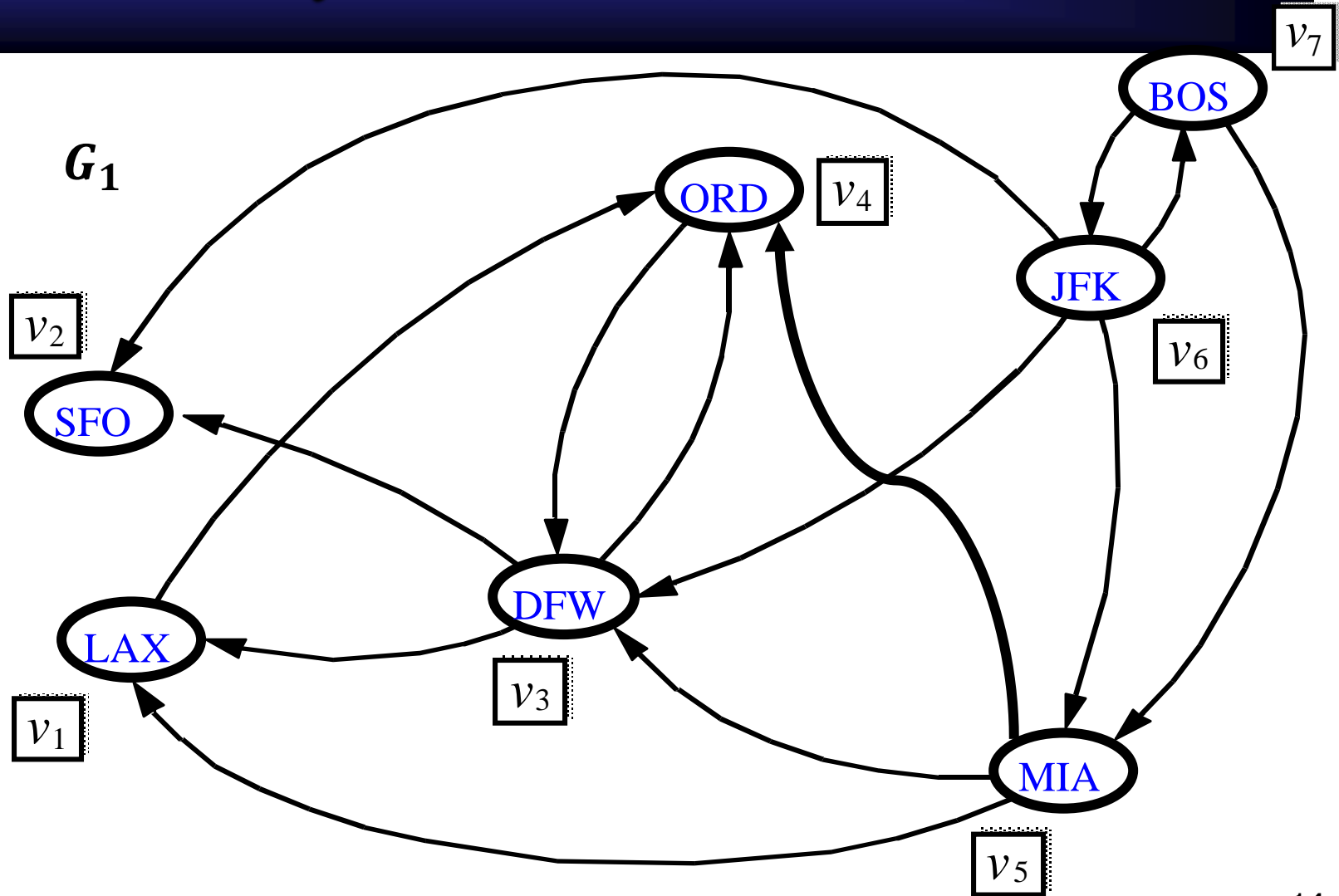
add (v_i, v_j) to G_k

return G_n

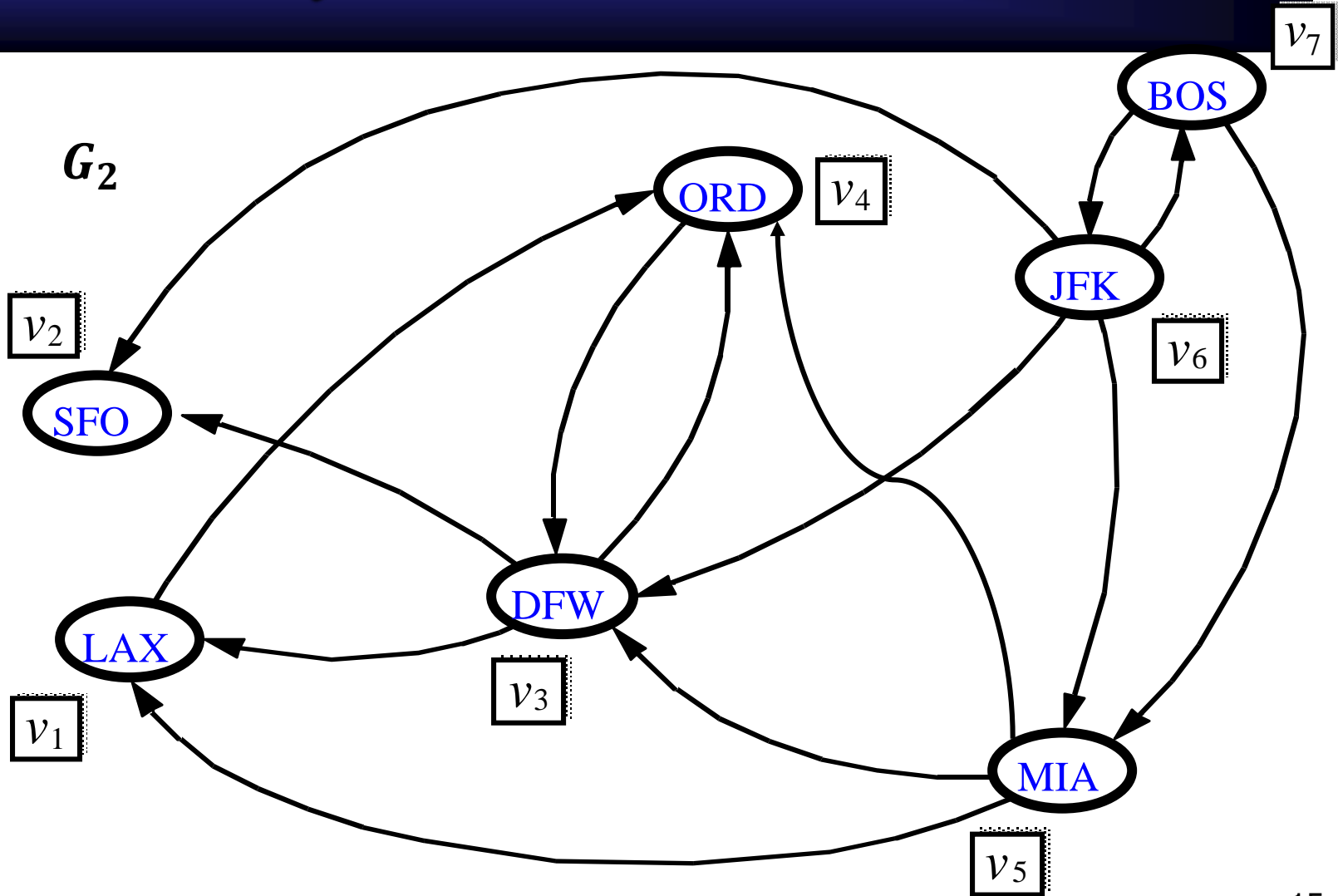
Floyd-Warshall Example



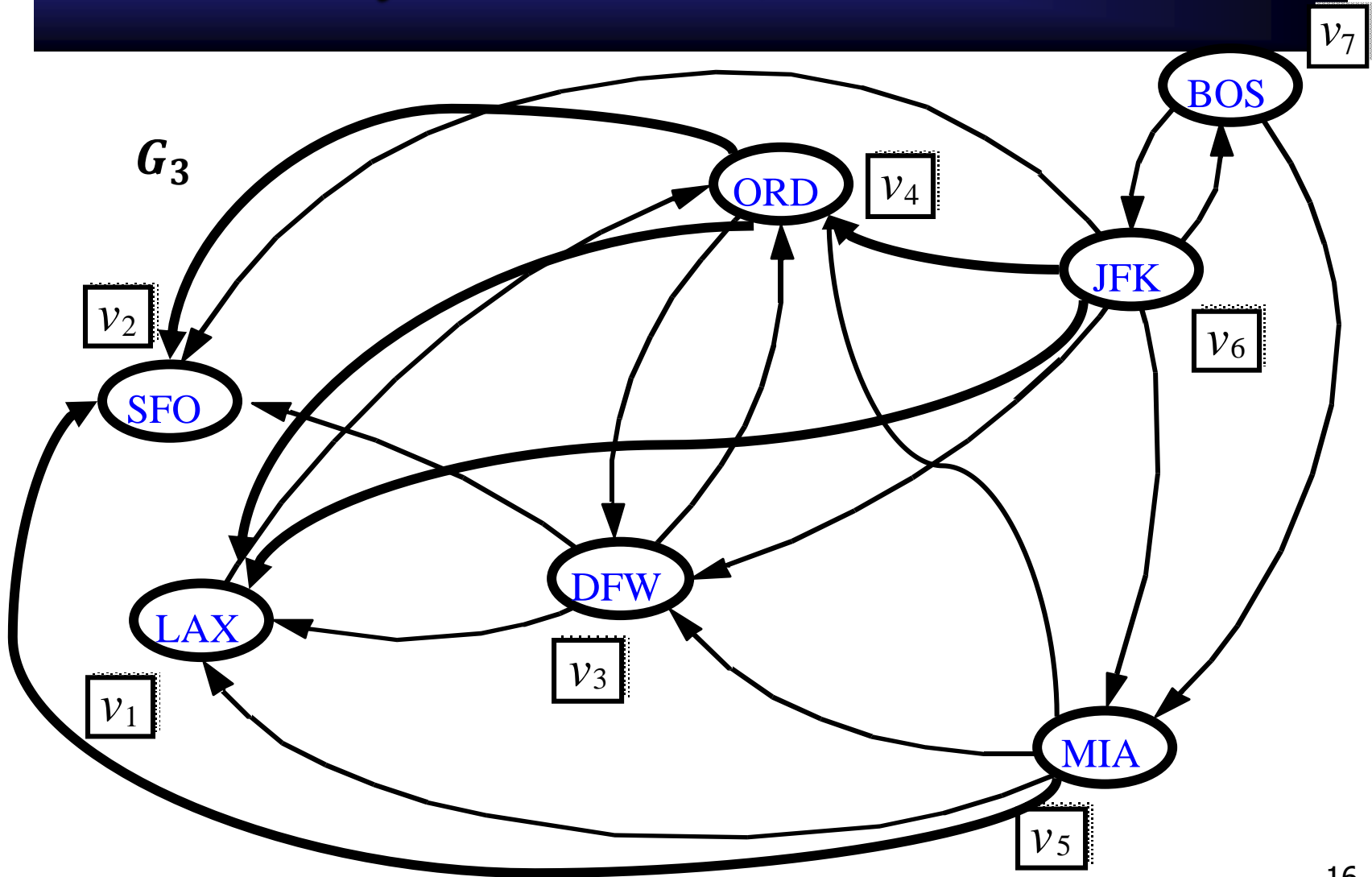
Floyd-Warshall, Iteration 1



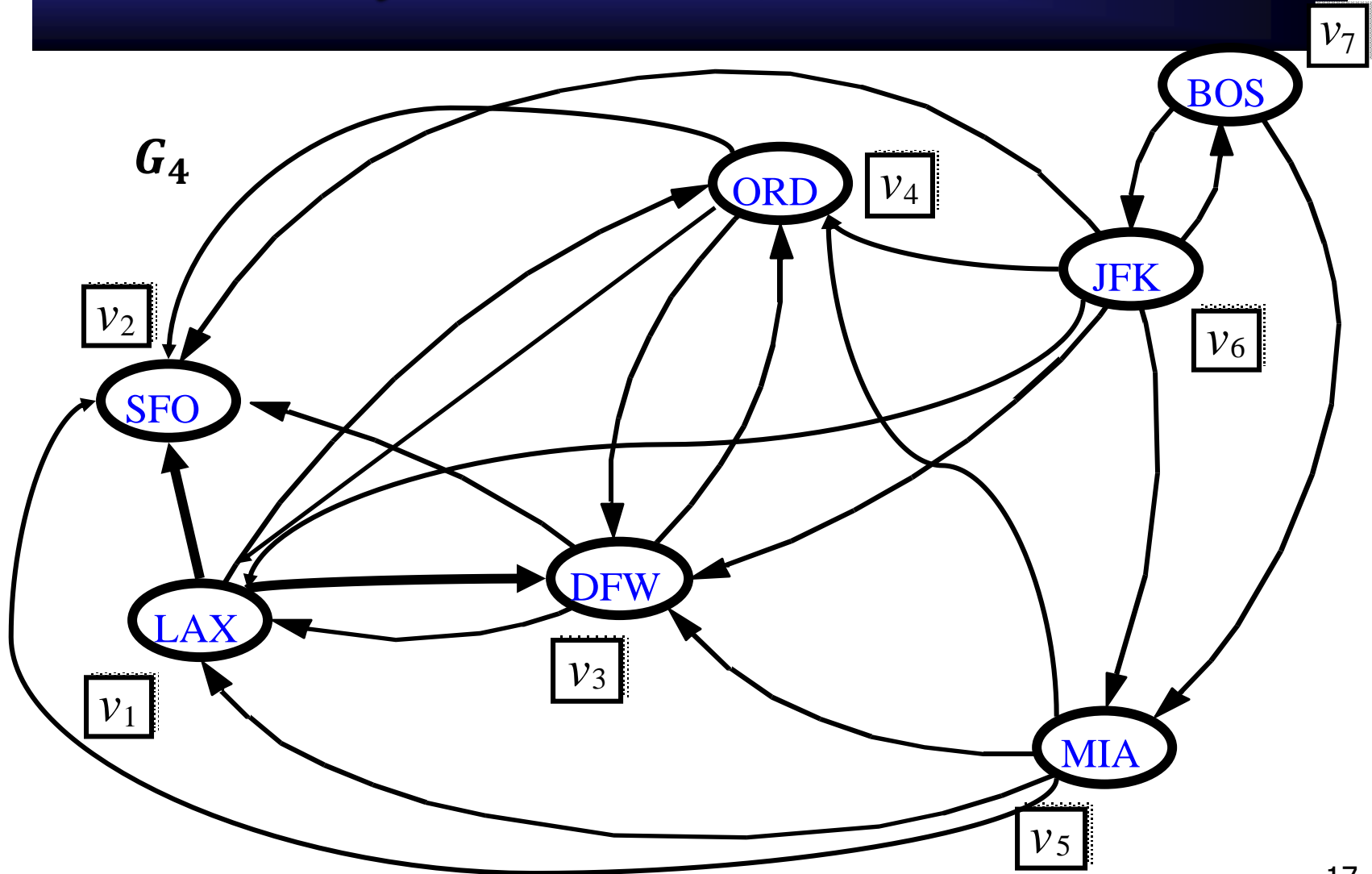
Floyd-Warshall, Iteration 2



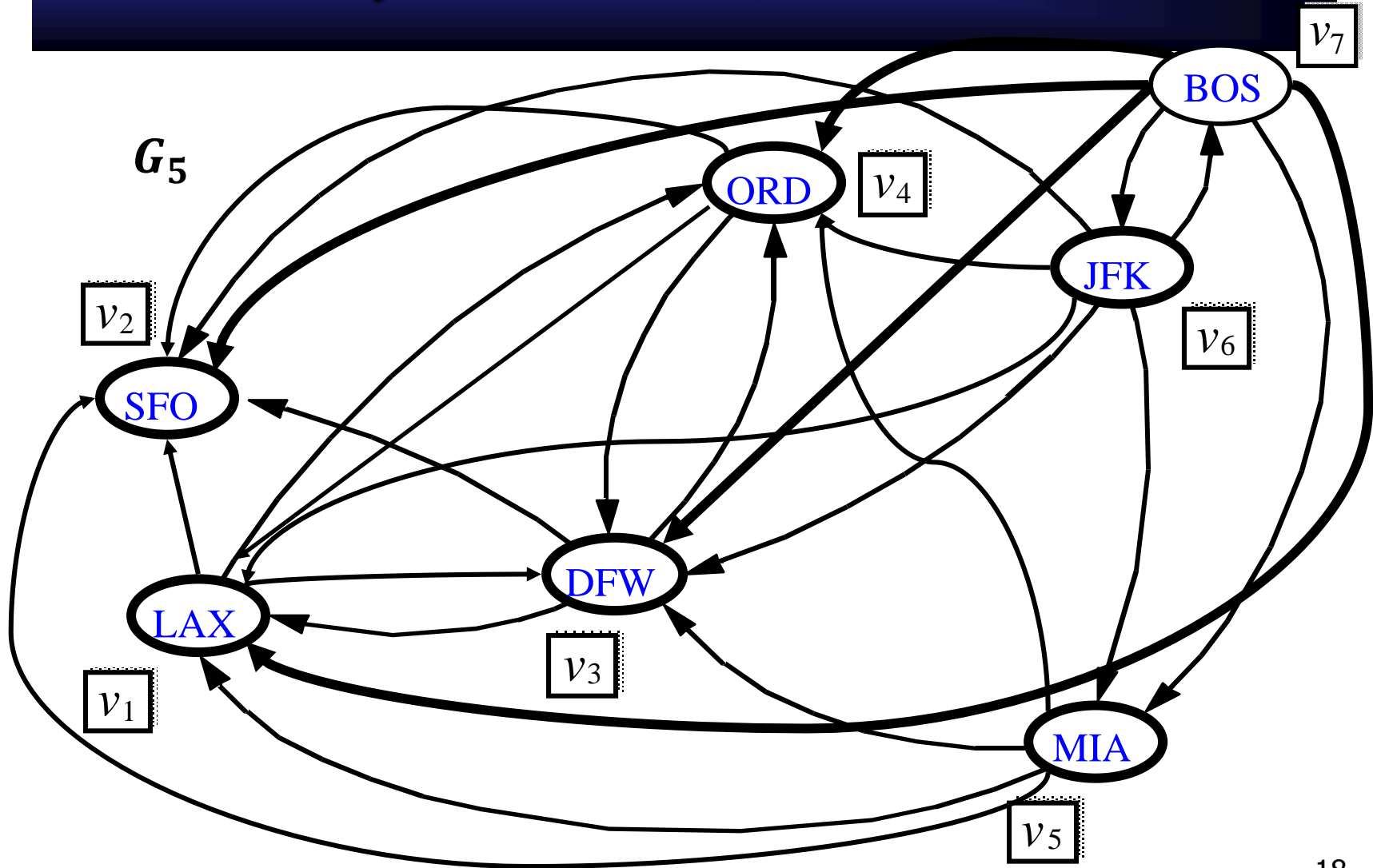
Floyd-Warshall, Iteration 3



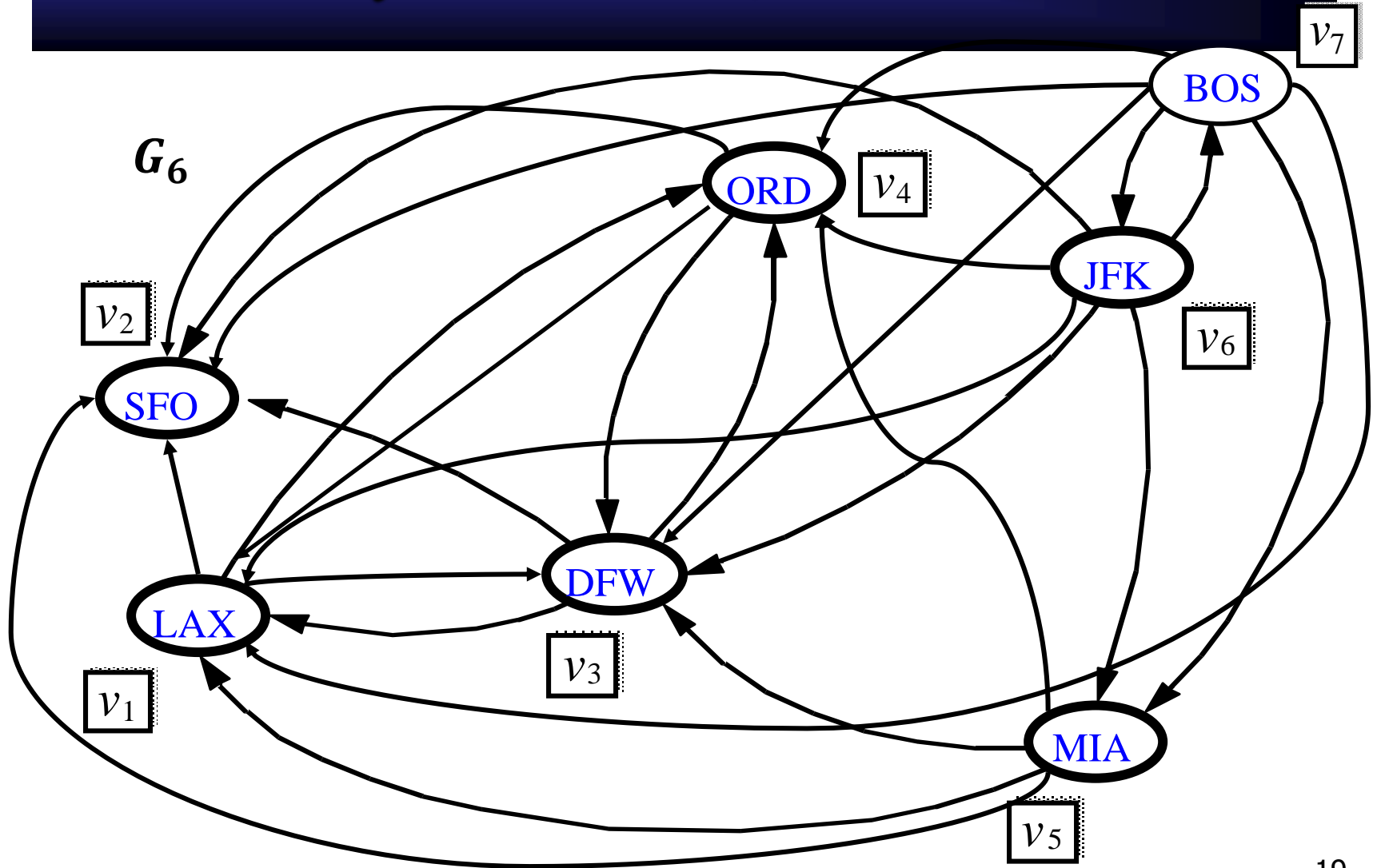
Floyd-Warshall, Iteration 4



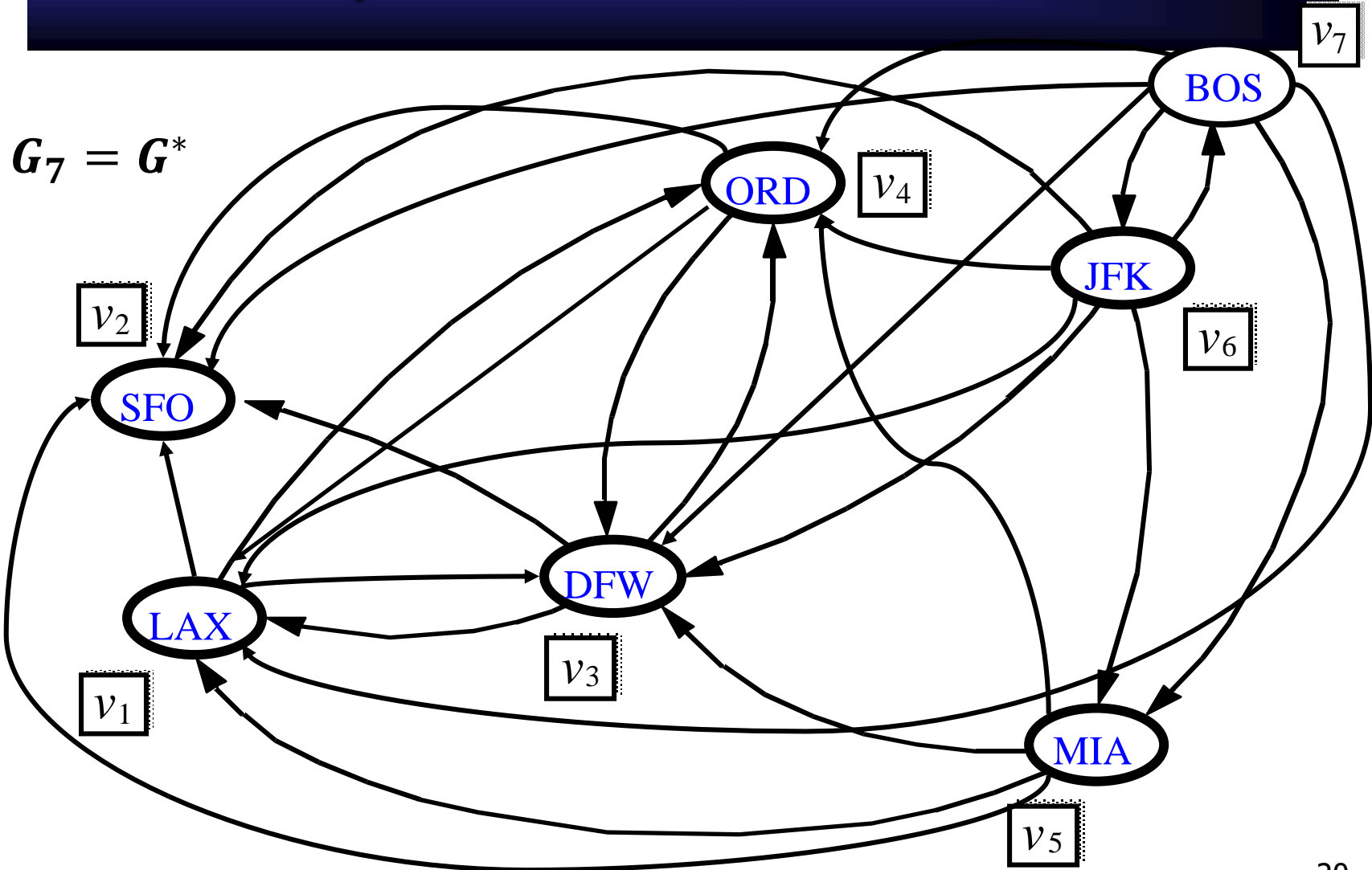
Floyd-Warshall, Iteration 5



Floyd-Warshall, Iteration 6



Floyd-Warshall, Conclusion

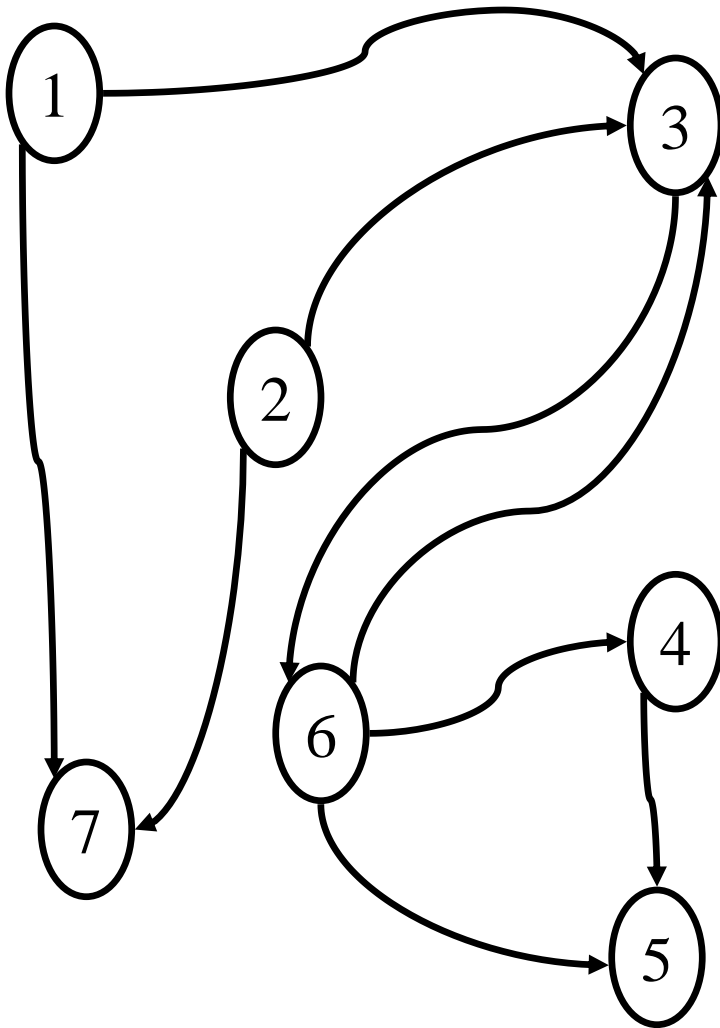


Floyd-Warshall Adj. Matrix

- We can use a 0-1 Matrix to run the Floyd-Warshall algorithm on a graph.

```
for each edge  $(i, j)$  in  $G$  do  $M[i, j] \leftarrow 1$  end  
for  $k \leftarrow 1$  to  $n$  do  
  for  $i \leftarrow 1$  to  $n$  do  $\backslash$  start vertex in  $G_{k-1}$   
    if  $i \neq k$  then  
      for  $j \leftarrow 1$  to  $n$  do  $\backslash$  end vertex in  $G_{k-1}$   
        if  $j \neq k$  and  $j \neq i$  then  
          if  $M[i, k] = 1$  and  $M[k, j] = 1$  then  
             $M[i, j] \leftarrow 1$ 
```

Warshall Example



Adjacency Matrix

	1	2	3	4	5	6	7
1			1				1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							

Adj Matrix	1	2	3	4	5	6	7
1			1				1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							23

k=1	1	2	3	4	5	6	7
1			1				1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							24

- For each pair i, j
 - Add a directed edge from vertex i to vertex j if there exists an edge from vertex i to vertex 1 and an edge from vertex 1 to vertex j .
- After this step, directed edges are added for all directed paths of length 2 in G that visit internal vertex 1.

i	k=1	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							26

k=1		1	2	3	4	5	6	7
i	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							27

k=1		1	2	3	4	5	6	7
i	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							28

i	k=1	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							29

k=1		1	2	3	4	5	6	7
i	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							30

i	k=1	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							31

i	k=1	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							32

- For each pair i, j : Add a directed edge from vertex i to vertex j if there exists one from vertex i to vertex 2 and one from vertex 2 to vertex j .
- After this step, directed edges are added for all directed paths of all directed paths of length 2 and 3 that visit as internal vertices only 1 and 2.

		k=2						
		1	2	3	4	5	6	7
i	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							34

i	k=2	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							35

i	k=2	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							36

k=2		1	2	3	4	5	6	7
i	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							37

k=2		1	2	3	4	5	6	7
i	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							38

k=2	1	2	3	4	5	6	7
1			1				1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							39

i

i	k=3	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							40

j

i

k=3	1	2	3	4	5	6	7
1			1				1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							41

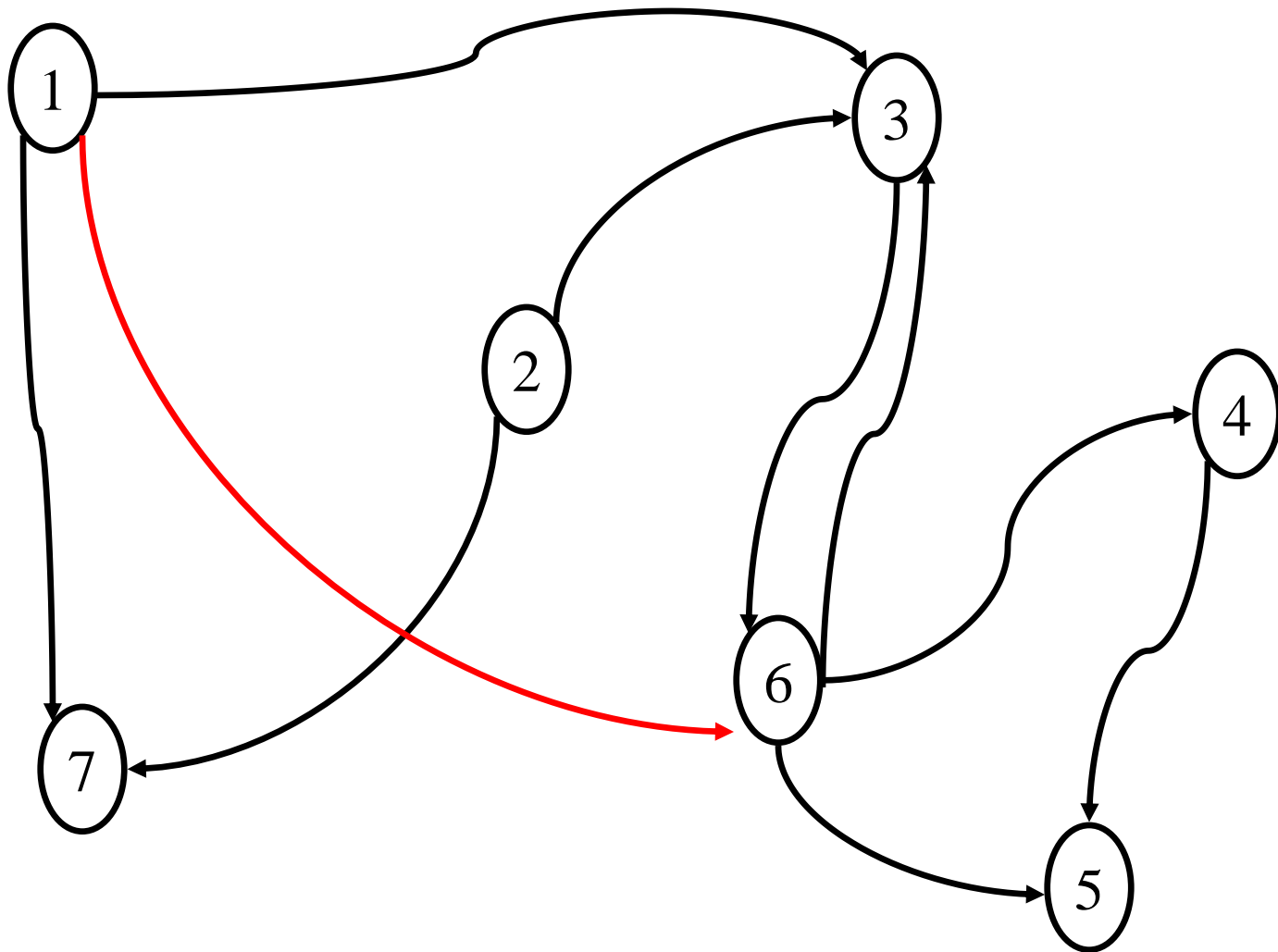
		j						
k=3	1	2	3	4	5	6	7	
i	1		1				1	
2			1				1	
3						1		
4					1			
5								
6			1	1	1			
7							42	

		j						
k=3	1	2	3	4	5	6	7	
i	1		1				1	
2			1				1	
3						1		
4					1			
5								
6			1	1	1			
7							43	

		j					
k=3	1	2	3	4	5	6	7
i	1		1				1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							44

		j						
i	k=3	1	2	3	4	5	6	7
	1			1				1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							45

		j						
i	k=3	1	2	3	4	5	6	7
	1			1			1	1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							46



		j						
i	k=3	1	2	3	4	5	6	7
	1			1			1	1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							48

k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							49

		j						
k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							50

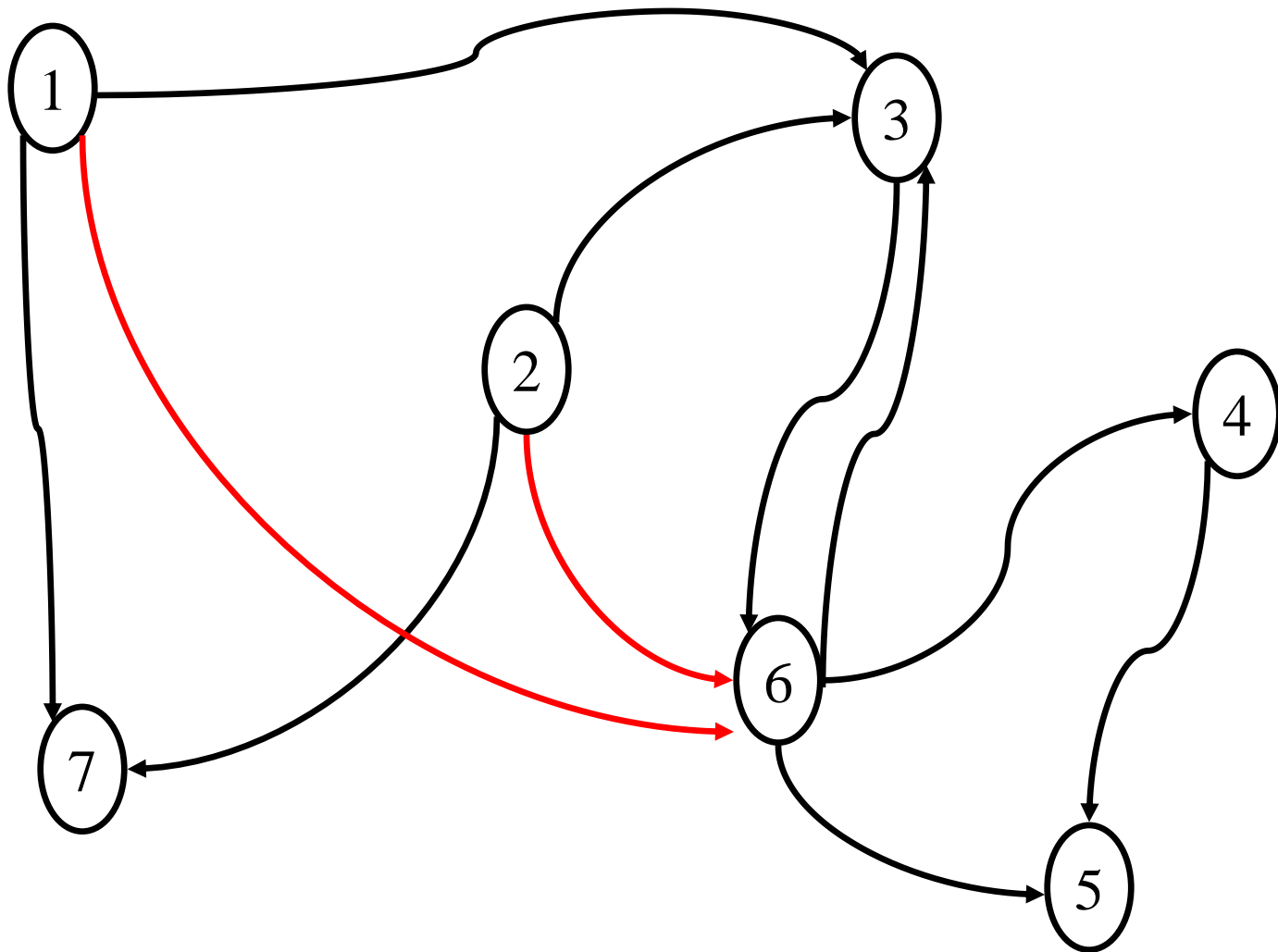
		j					
k=3	1	2	3	4	5	6	7
i	1		1			1	1
	2		1				1
	3					1	
	4				1		
	5						
	6		1	1	1		
	7						51

		j						
k=3	1	2	3	4	5	6	7	
i	1			1			1	1
	2			1				1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							52

		j					
k=3	1	2	3	4	5	6	7
i	1		1			1	1
	2		1				1
	3					1	
	4				1		
	5						
	6		1	1	1		
	7						53

		j					
k=3	1	2	3	4	5	6	7
i	1		1			1	1
	2		1				1
	3					1	
	4				1		
	5						
	6		1	1	1		
	7						54

		j					
k=3	1	2	3	4	5	6	7
i	1		1			1	1
	2		1			1	1
	3					1	
	4				1		
	5						
	6		1	1	1		
	7						55



		j						
k=3	1	2	3	4	5	6	7	
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							57

k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							58

k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							59

k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							60

		j						
k=3	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							61	

j

k=3	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							62

i

		j						
k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							63

		j						
k=3	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							64	

		j					
k=3	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							65
i							

k=3		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1	1	
	7							66

- After this step, directed edges are added for all directed paths of length 2, 3, and 4 in G that visit as internal vertices only 1,2 and 3.

k=4	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							68

k=4		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							69

k=4		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							70

i	k=4	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							71

k=4		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							72

k=4		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							73

		j					
k=4	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							74

j

k=4	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							75

i

		j						
k=4	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							76	

i

		j						
k=4	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							77	
i								

		j						
k=4	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							78	
i								

		j						
k=4	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							79	

k=4		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							80

- After this step, directed edges are added for all directed paths of length 2, 3, 4, and 5 in G that visit as internal vertices only 1, 2, 3 and 4.

k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							82

k=5		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							83

i \ k=5	1	2	3	4	5	6	7
	1		1			1	1
	2		1			1	1
	3					1	
	4				1		
	5						
	6		1	1	1		
	7						84

i	k=5	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							85

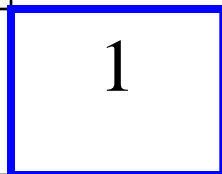
i	k=5	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							86

		j					
k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							87

j

k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							88

i



		j						
k=5	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
i 4					1			
5								
6			1	1	1			
7							89	

		j					
k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
i 4					1		
5							
6			1	1	1		
7							90

		j					
k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							91

i	k=5	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							92

k=5		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							93

		j						
k=5	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							94	

i

j

k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							95

i

		j						
k=5	1	2	3	4	5	6	7	
1			1			1	1	
2			1			1	1	
3						1		
4					1			
5								
6			1	1	1			
7							96	

		j						
k=5		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							97

		j					
k=5	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							98
i							

k=5		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							99

- After this step, directed edges are added for all directed paths of length 2, 3, 4, 5 and 6 in G that visit as internal vertices only 1, 2, 3, 4 and 5.

k=6	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							101

k=6		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							102

j

k=6	1	2	3	4	5	6	7
i	1		1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							103

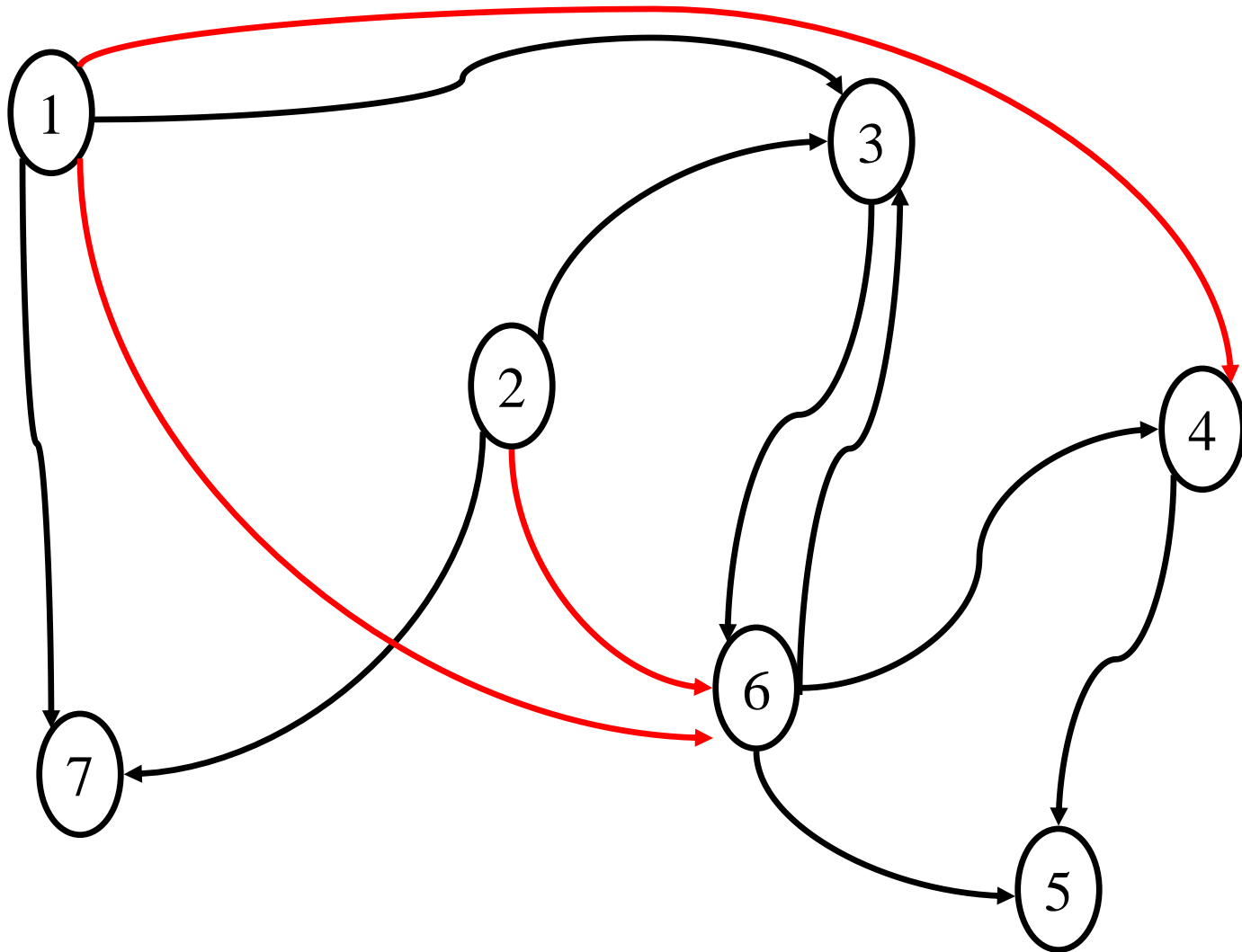
		j						
i	k=6	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							104

		j						
i	k=6	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							105

		j						
k=6		1	2	3	4	5	6	7
i	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							106

		j						
i	k=6	1	2	3	4	5	6	7
	1			1			1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							107

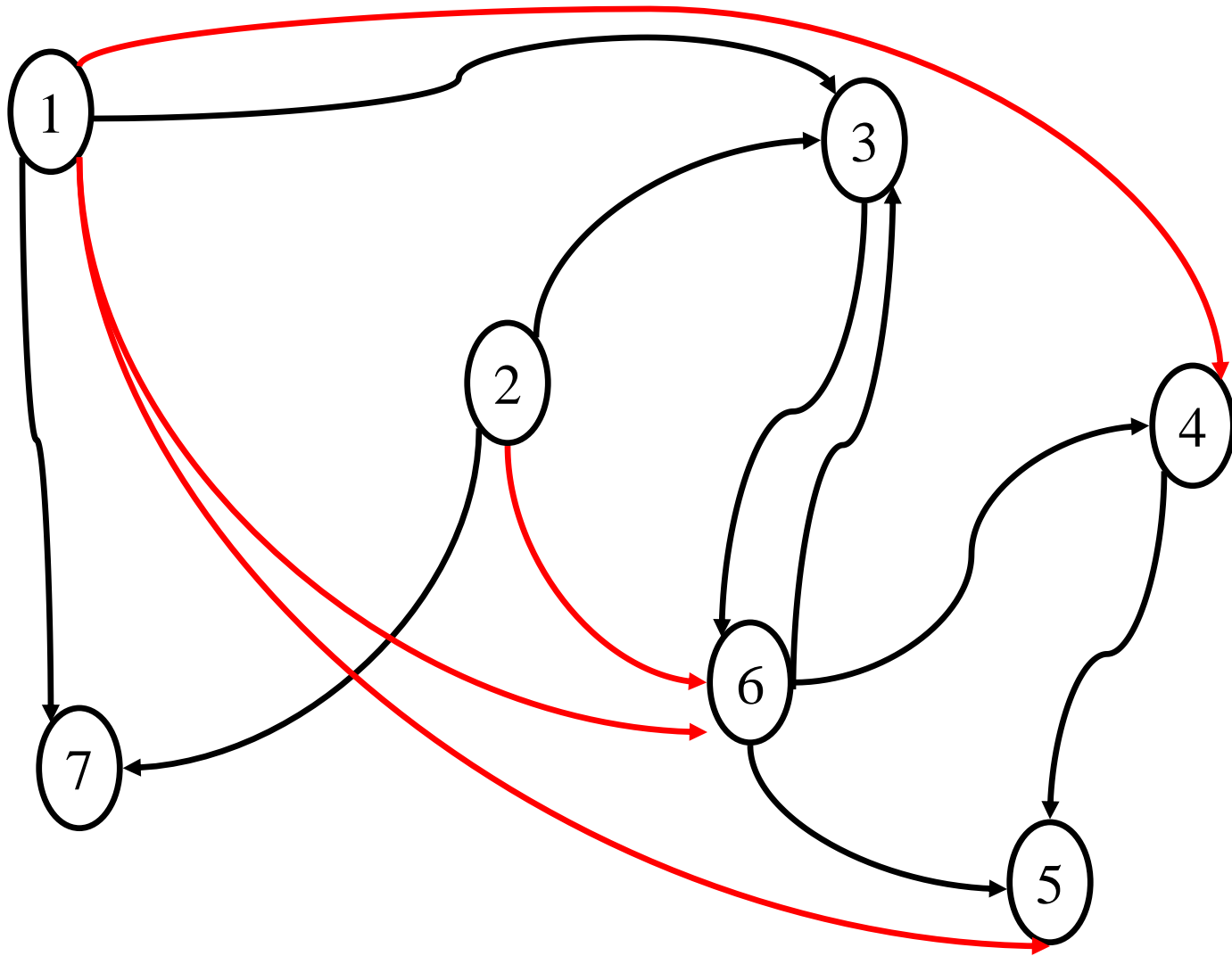
		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1		1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							108



		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1		1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							110

		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1		1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							111

		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							112



		j						
k=6	1	2	3	4	5	6	7	
i	1			1	1	1	1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							114

		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							115

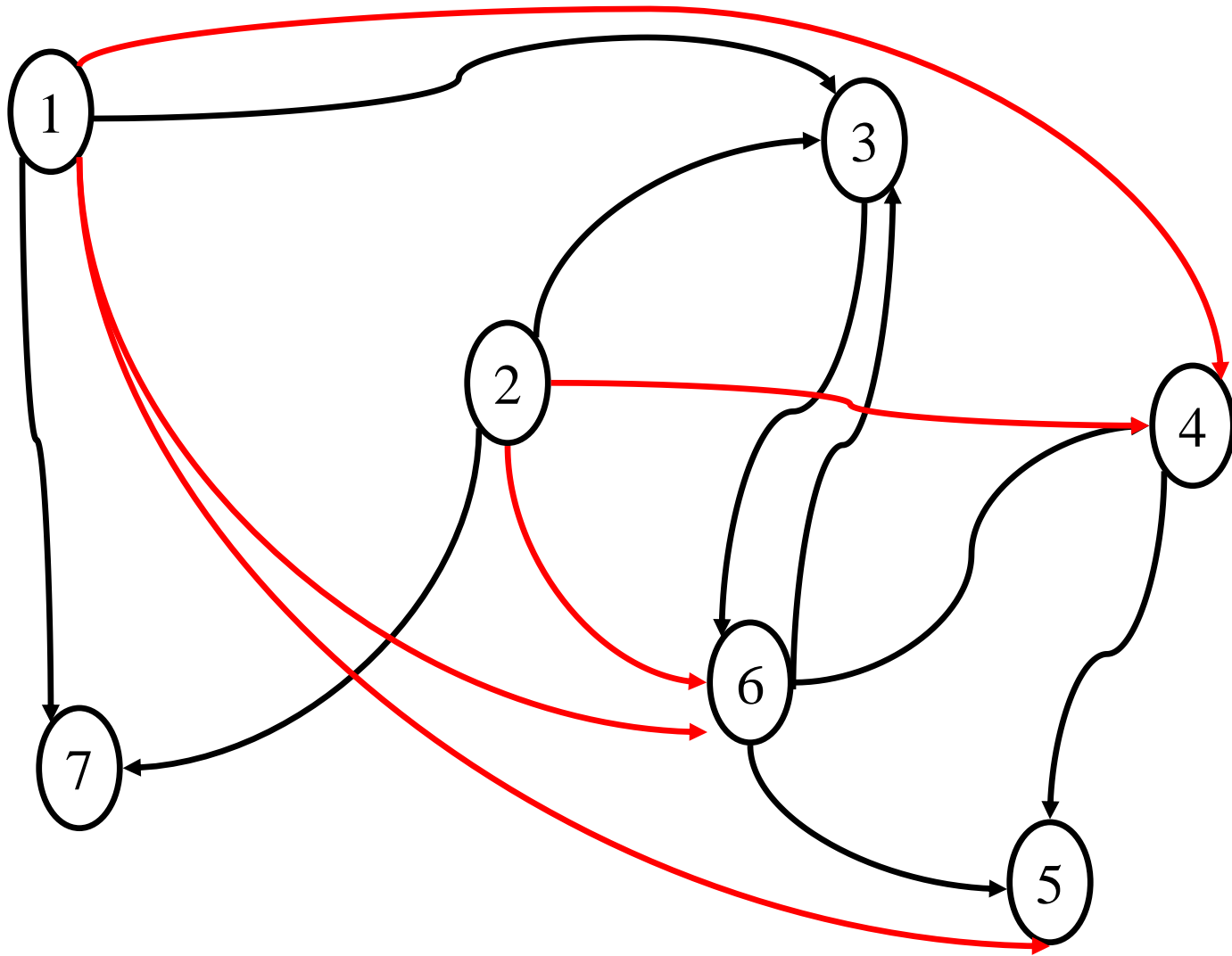
		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							116

		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							117

		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							118

		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1			1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							119

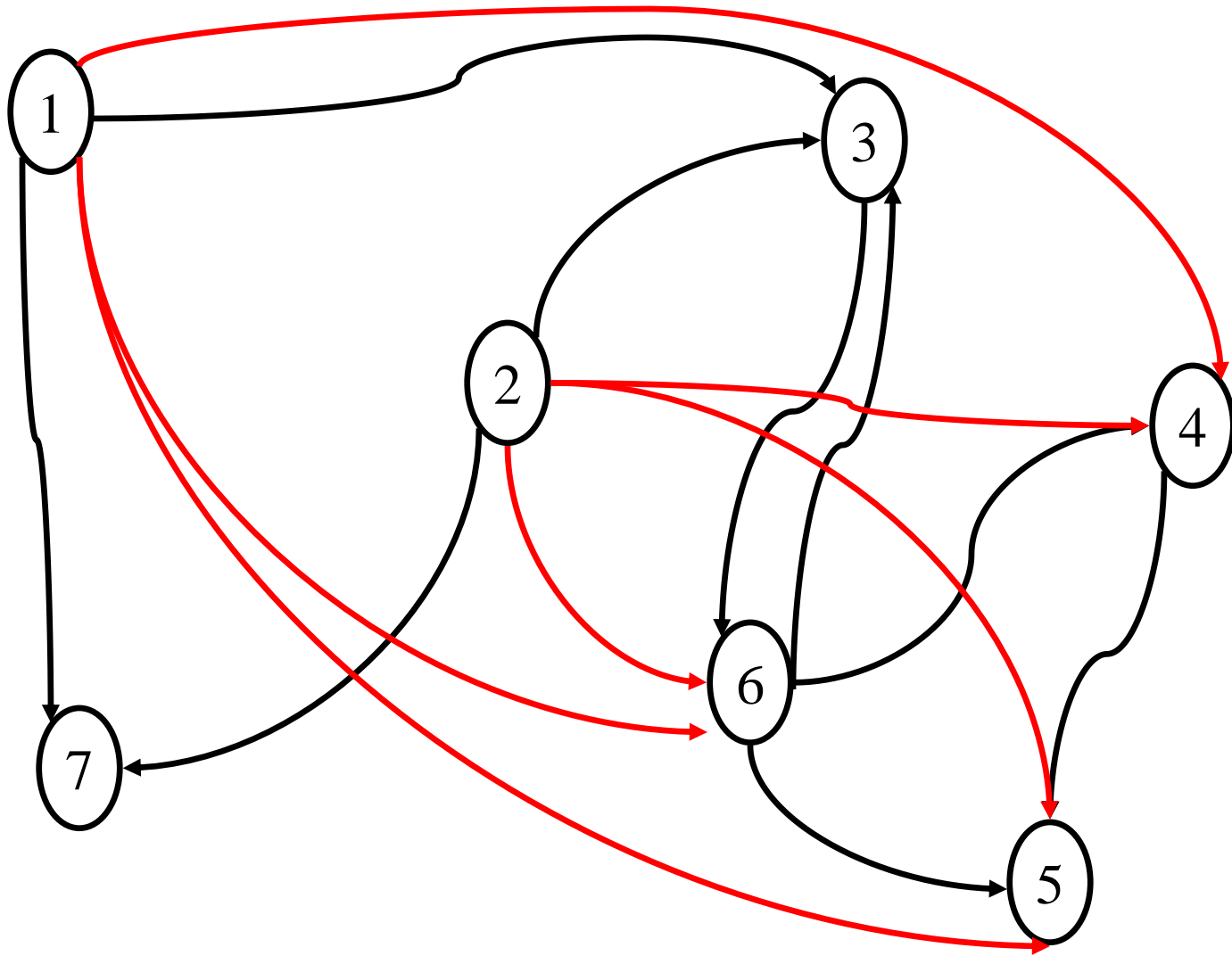
		j						
k=6	1	2	3	4	5	6	7	
1			1	1	1	1	1	
2			1	1		1	1	
3						1		
4					1			
5								
6			1	1	1			
7							120	



		j						
k=6	1	2	3	4	5	6	7	
1			1	1	1	1	1	
2			1	1		1	1	
3						1		
4					1			
5								
6			1	1	1			
7							122	

		j						
i	k=6	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1		1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							123

		j						
k=6	1	2	3	4	5	6	7	
1			1	1	1	1	1	
2			1	1	1	1	1	
3						1		
4					1			
5								
6			1	1	1			
7							124	



		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3						1	
4					1		
5							
6			1	1	1		
7							126

k=6		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3						1	
	4					1		
	5							
	6			1	1	1		
	7							127

		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
i 3						1	
4					1		
5							
6			1	1	1		
7							128

j

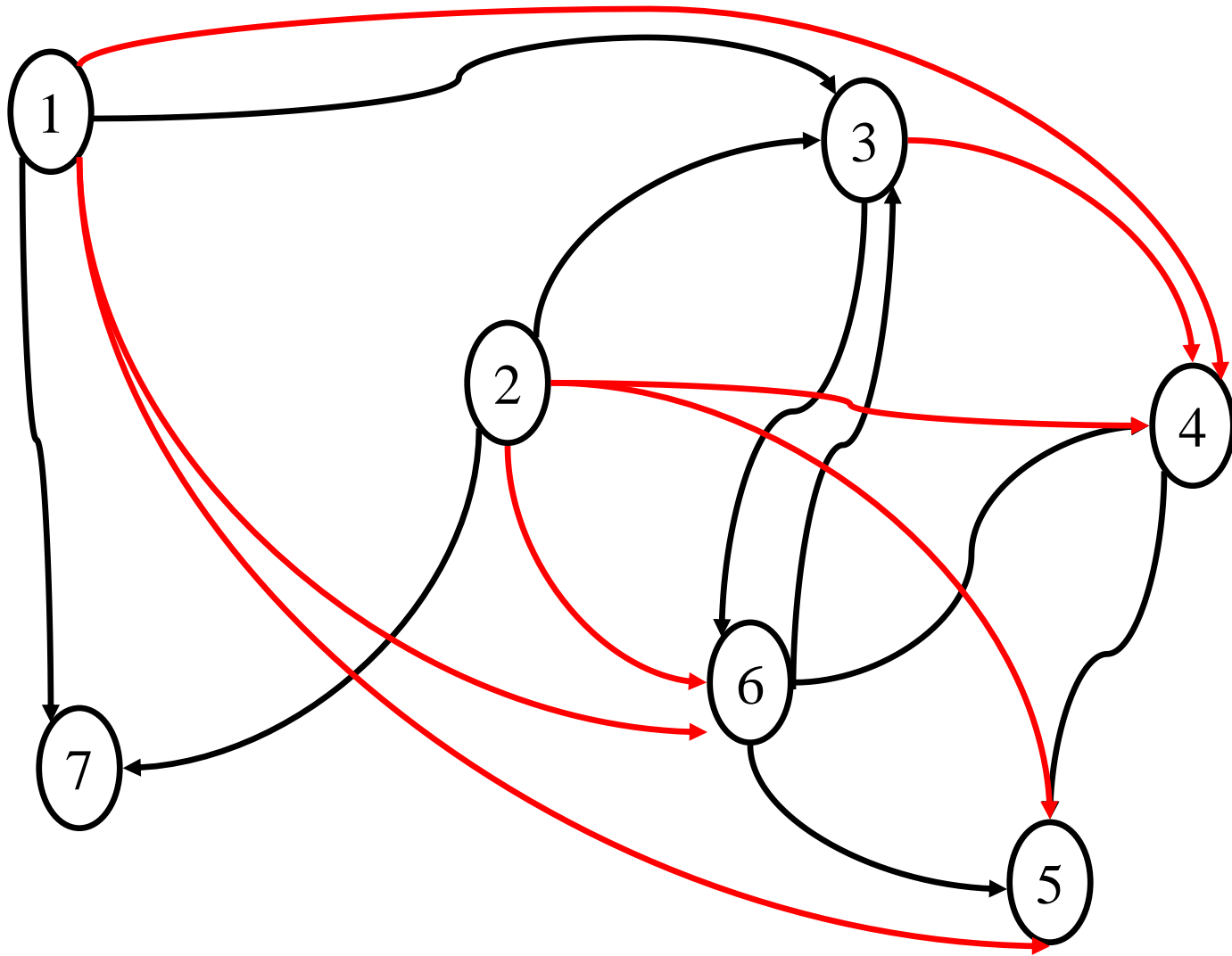
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3						1	
4					1		
5							
6			1	1	1		
7							129

i

		j						
k=6	1	2	3	4	5	6	7	
1			1	1	1	1	1	
2			1	1	1	1	1	
3						1		
4					1			
5								
6			1	1	1			
7							130	

		j						
k=6	1	2	3	4	5	6	7	
1			1	1	1	1	1	
2			1	1	1	1	1	
3						1		
4					1			
5								
6			1	1	1			
7							131	

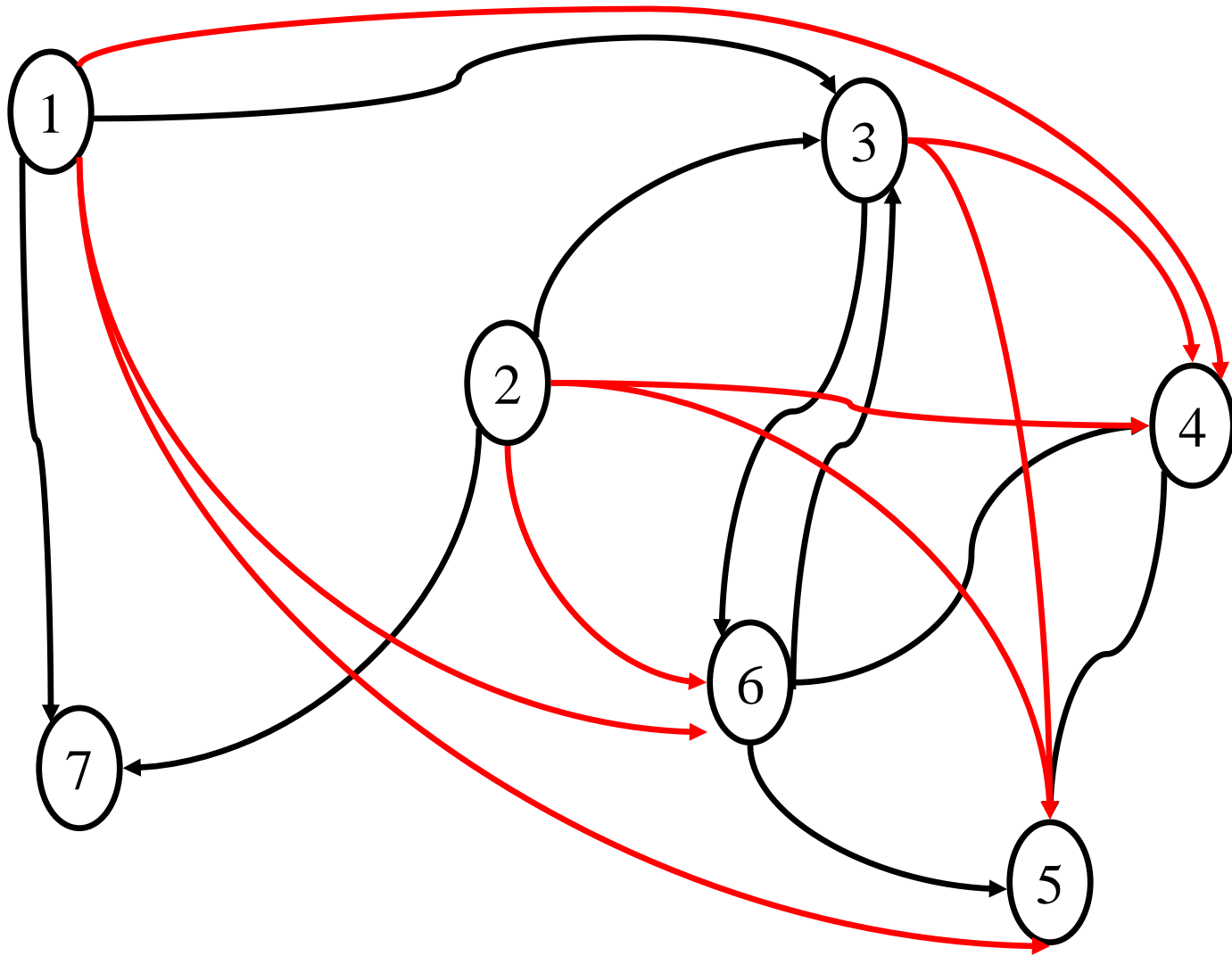
		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1		1	
4					1		
5							
6			1	1	1		
7							132



		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
i 3				1		1	
4					1		
5							
6			1	1	1		
7							134

		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
i 3				1		1	
4					1		
5							
6			1	1	1		
7							135

		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
i 3				1	1	1	
4					1		
5							
6			1	1	1		
7							136



		j					
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	
4					1		
5							
6			1	1	1		
7							138

k=6		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							139

i	k=6	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							140

k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	
4					1		
5							
6			1	1	1		
7							

i

- After this step, directed edges are added for all directed paths of length 2, 3, 4, 5, 6 and 7 in G that visit as internal vertices only 1, 2, 3, 4, 5 and 6.

k=7	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	
4					1		
5							
6			1	1	1		
7							143

k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							144

		j						
i	k=7	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							145

		j						
i	k=7	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							146

		j						
i	k=7	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							147

		j						
i	k=7	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							148

		j						
k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							149

i \ k=7	1	2	3	4	5	6	7
	1		1	1	1	1	1
	2		1	1	1	1	1
	3			1	1	1	
	4				1		
	5						
	6		1	1	1		
	7						150

		j						
k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							151

		j					
k=7	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	
4					1		
5							
6			1	1	1		
7							152

		j					
k=7	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	
4					1		
5							
6			1	1	1		
7							153

		j						
k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							154

		j					
k=7	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1	1	1	
4					1		
5							
6			1	1	1		
7							155

k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							156

i	k=7	1	2	3	4	5	6	7
	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							157

k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							158

k=7		1	2	3	4	5	6	7
i	1			1	1	1	1	1
	2			1	1	1	1	1
	3				1	1	1	
	4					1		
	5							
	6			1	1	1		
	7							159