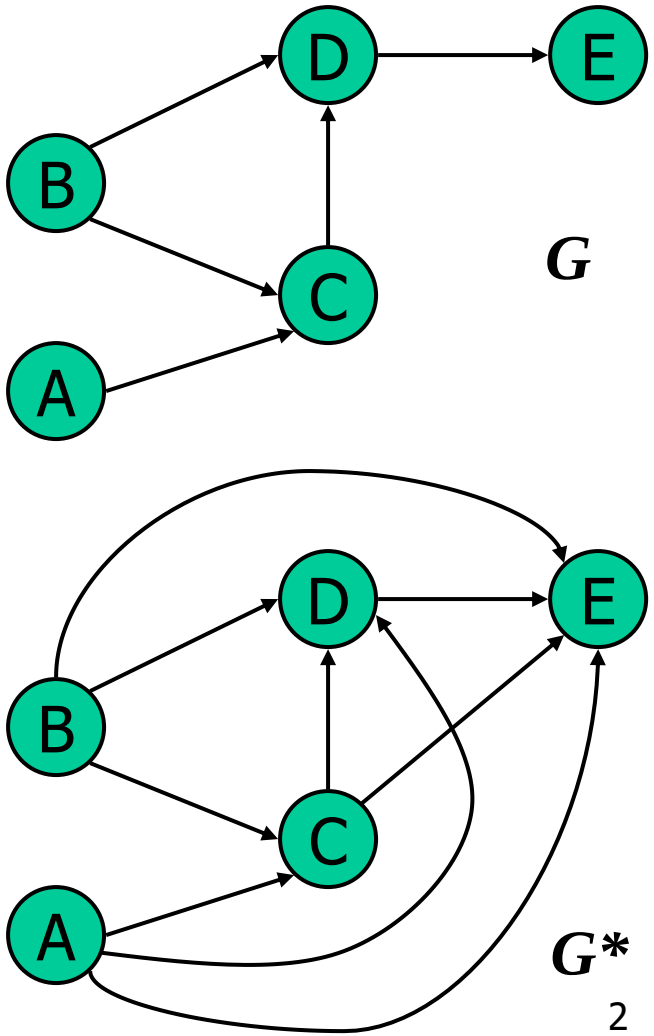
CSC 226

Algorithms and Data Structures: II

Rich Little
rlittle@uvic.ca

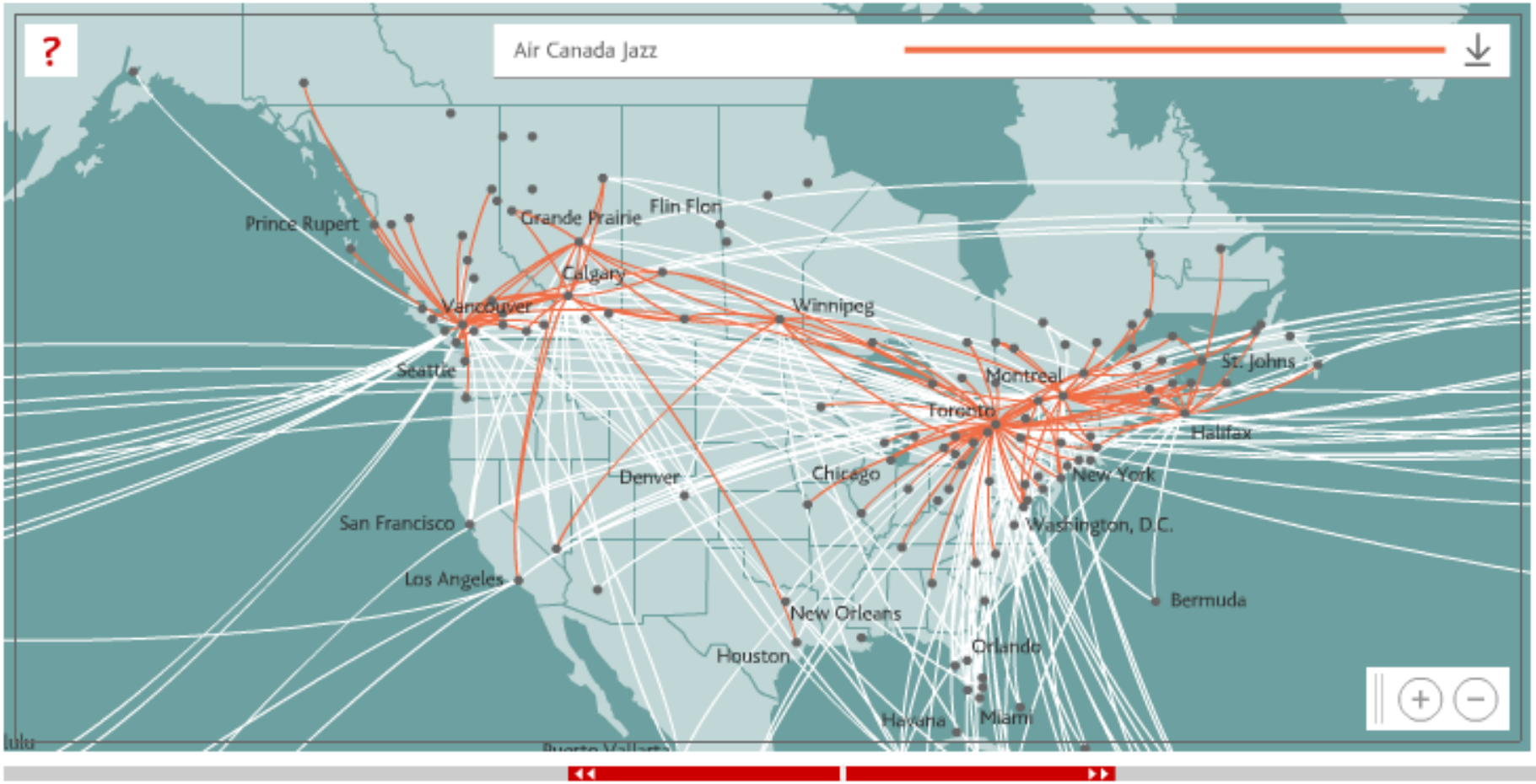
Transitive Closure

- 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*** ≠ ***v***), ***G**** has a directed edge from ***u*** to ***v***
- The transitive closure provides reachability information about a digraph



AIR CANADA

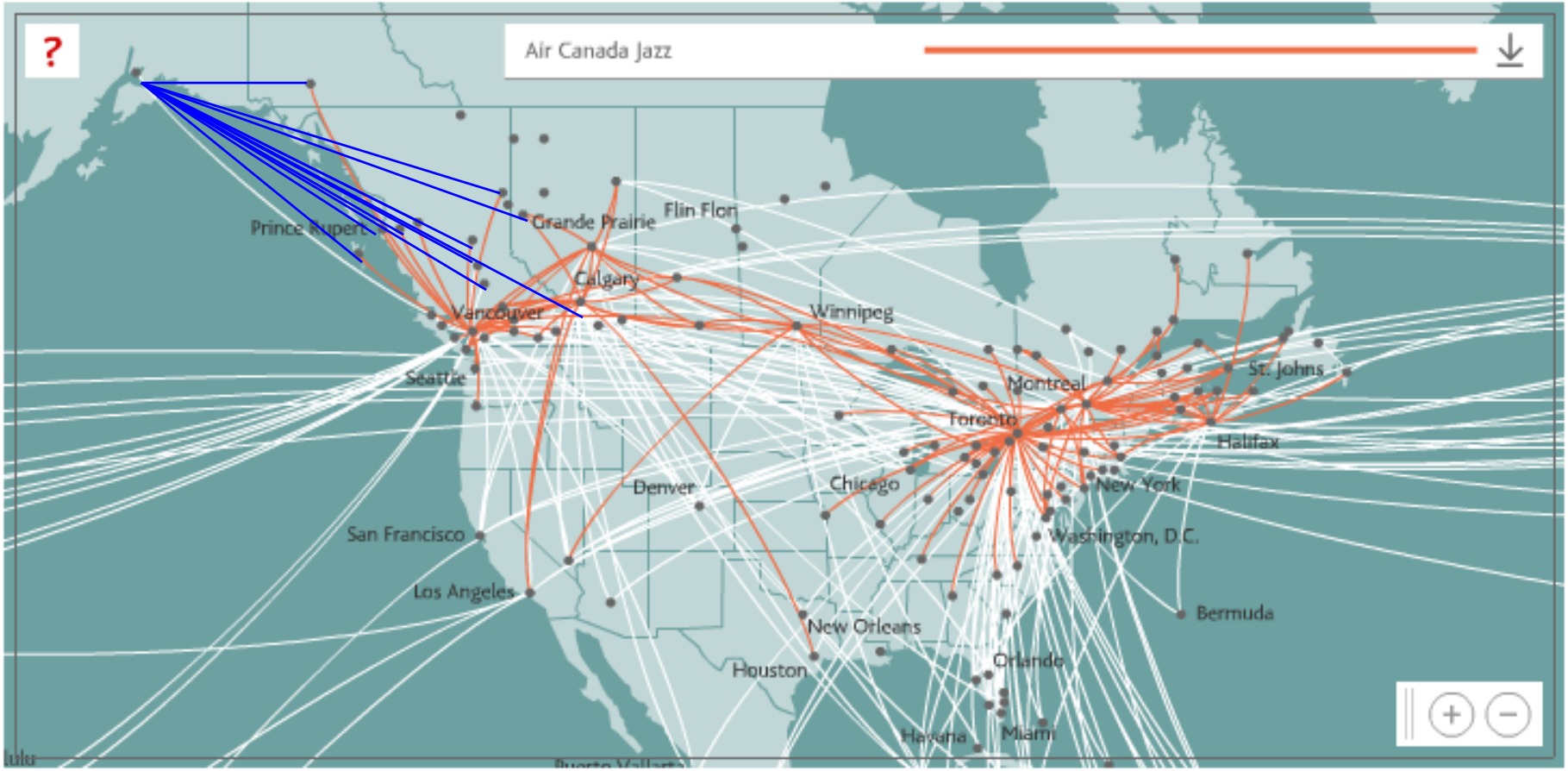
Select a map > World · Canada · United States · Europe · Asia / Pacific · Latin America



Exit Map

AIR CANADA

Select a map > World · Canada · United States · Europe · Asia / Pacific · Latin America



Exit Map

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

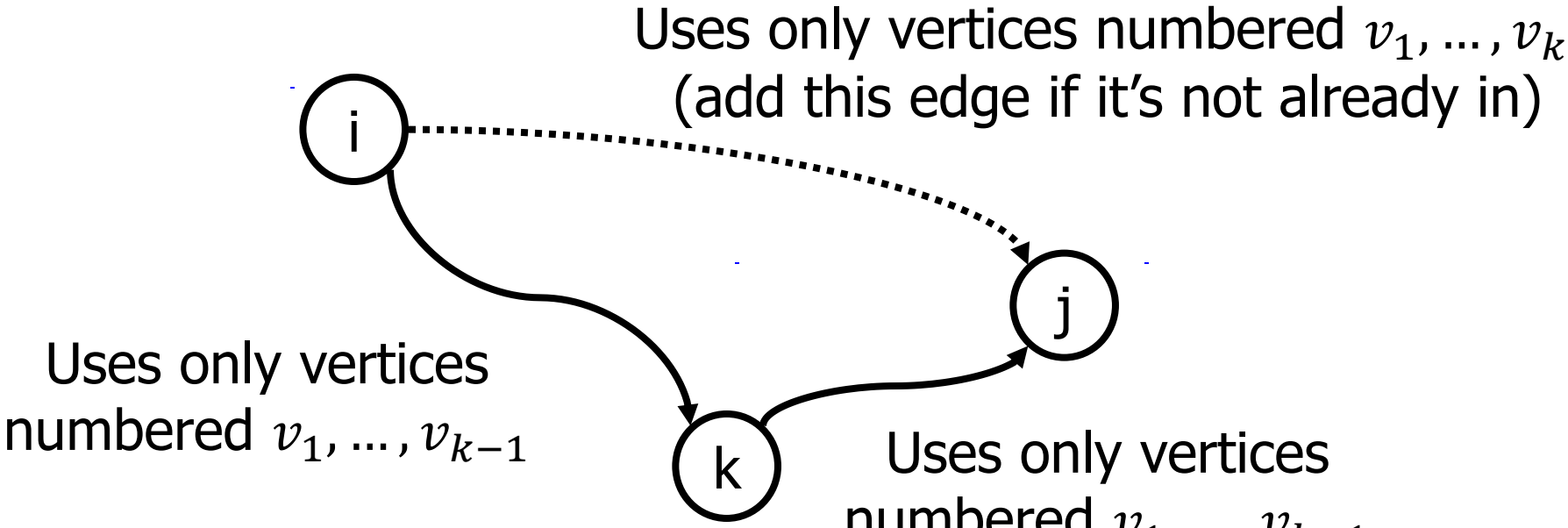
$$\max \sum_{k \in S} b_k \text{ subject to } \sum_{k \in S} w_k \leq \mathrm{W}$$

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

http://en.wikipedia.org/wiki/Dynamic_programming

Floyd-Warshall Transitive Closure

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



The Floyd-Warshall Algorithm

- Let G be a digraph with n vertices and m edges.
- Number the vertices, $v_1, v_2, \ldots, v_n$, 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 $i = 1, \ldots, n$, 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 $\{v_1, \ldots, v_k\}$. 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 $v_1, v_2, \ldots, v_n$ 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**

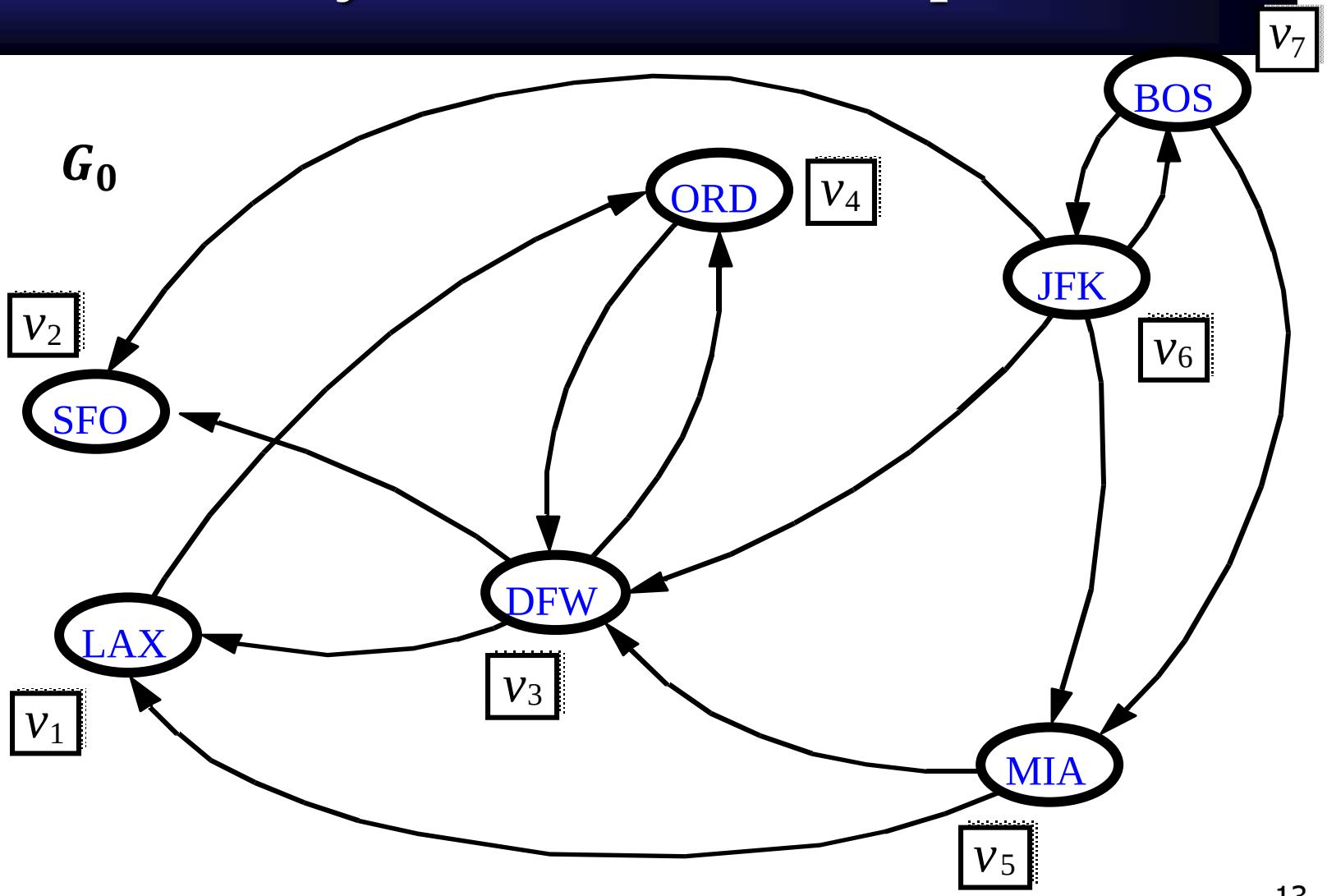
      if (v_i, v_k) and (v_k, v_j) are in G_{k-1} **then**

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

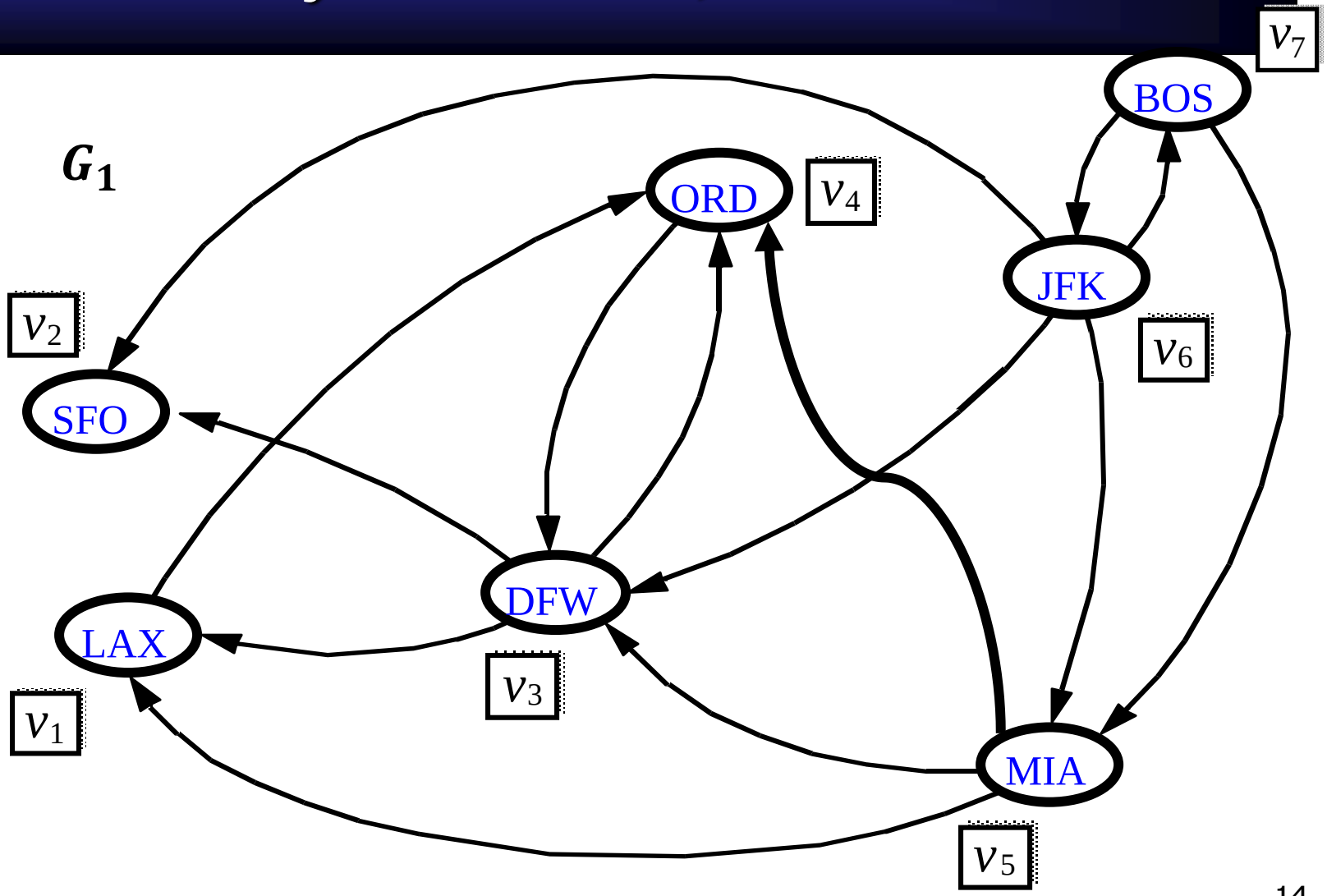
          add (v_i, v_j) to G_k

return G_n

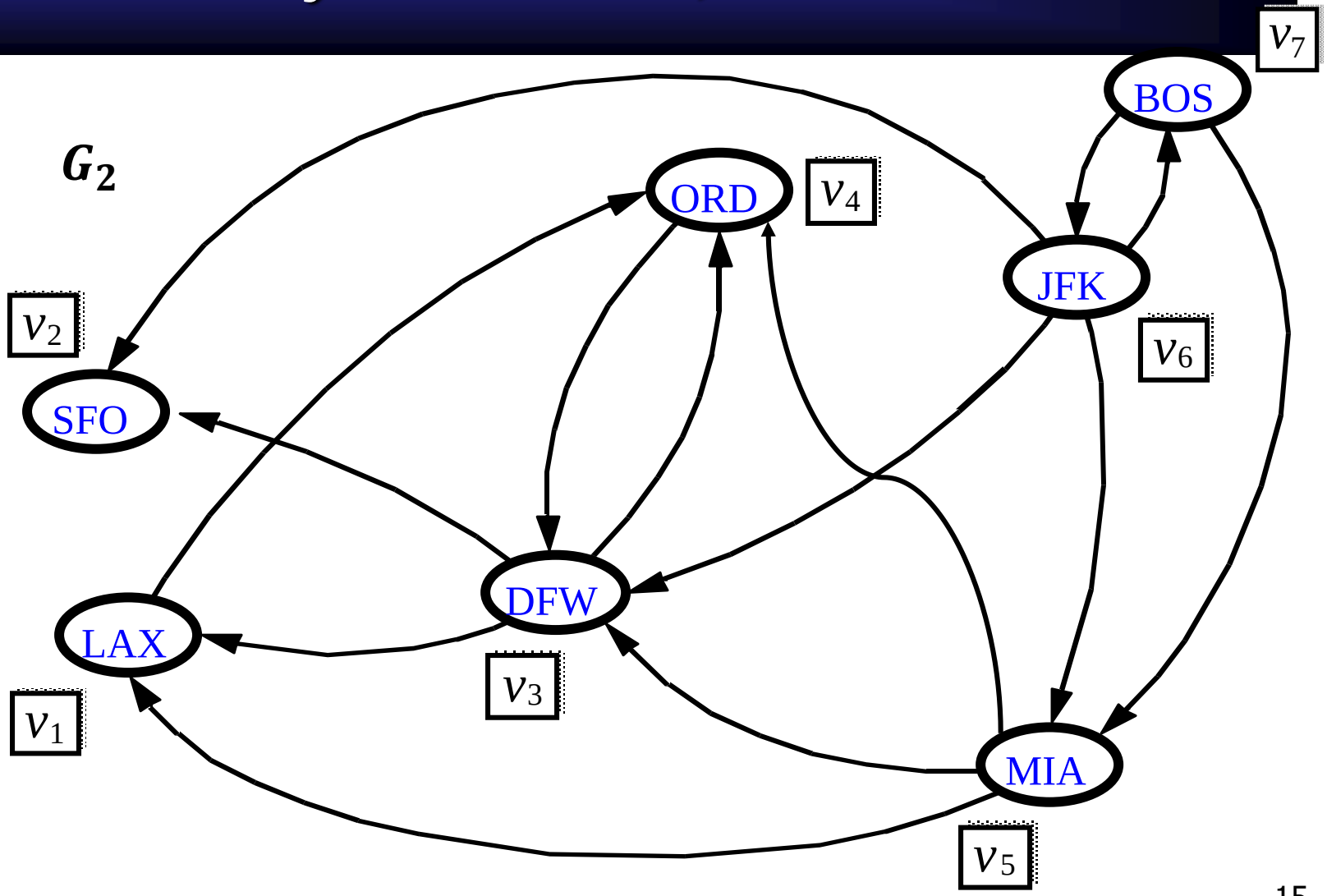
Floyd-Warshall Example



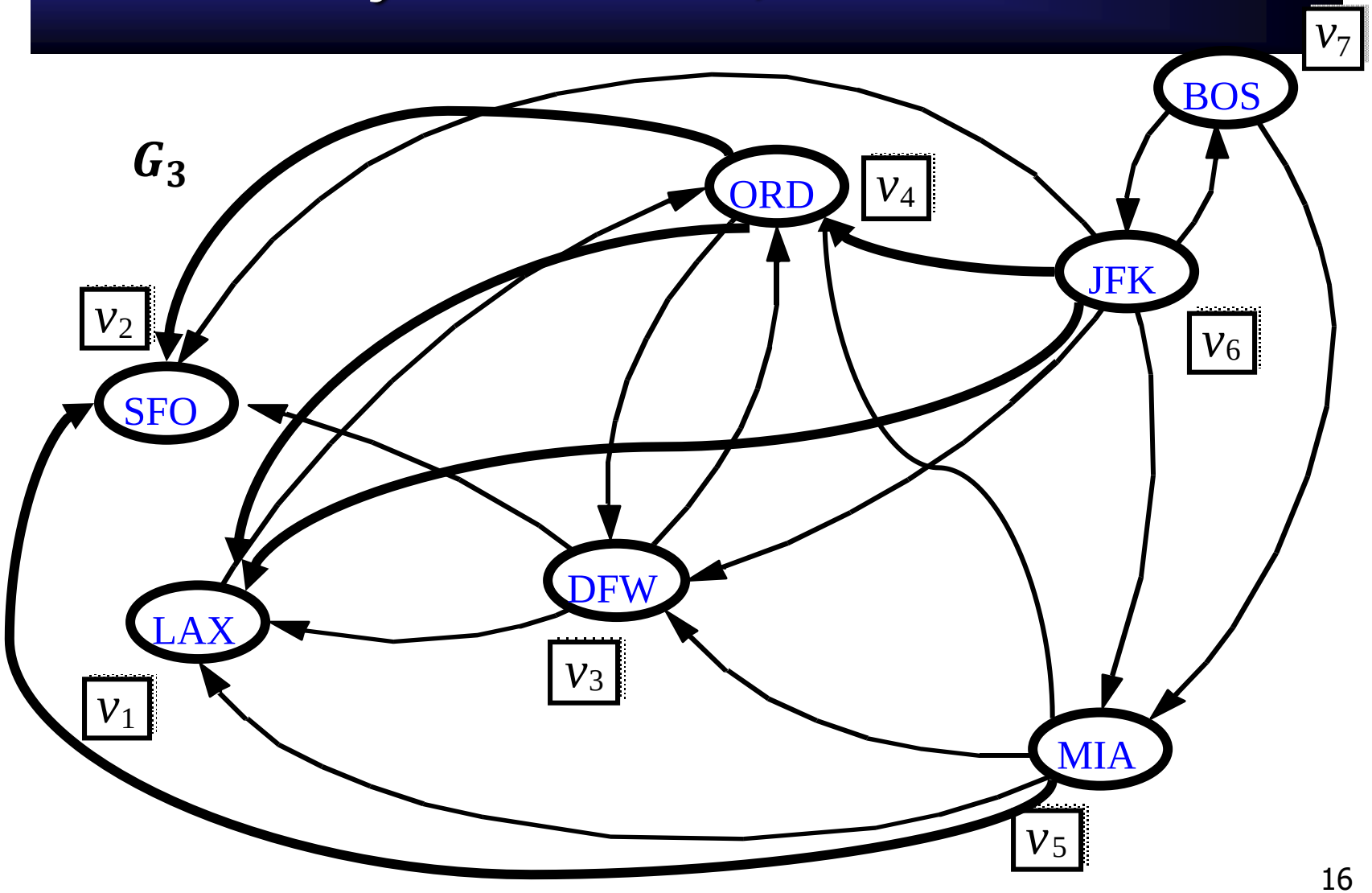
Floyd-Warshall, Iteration 1



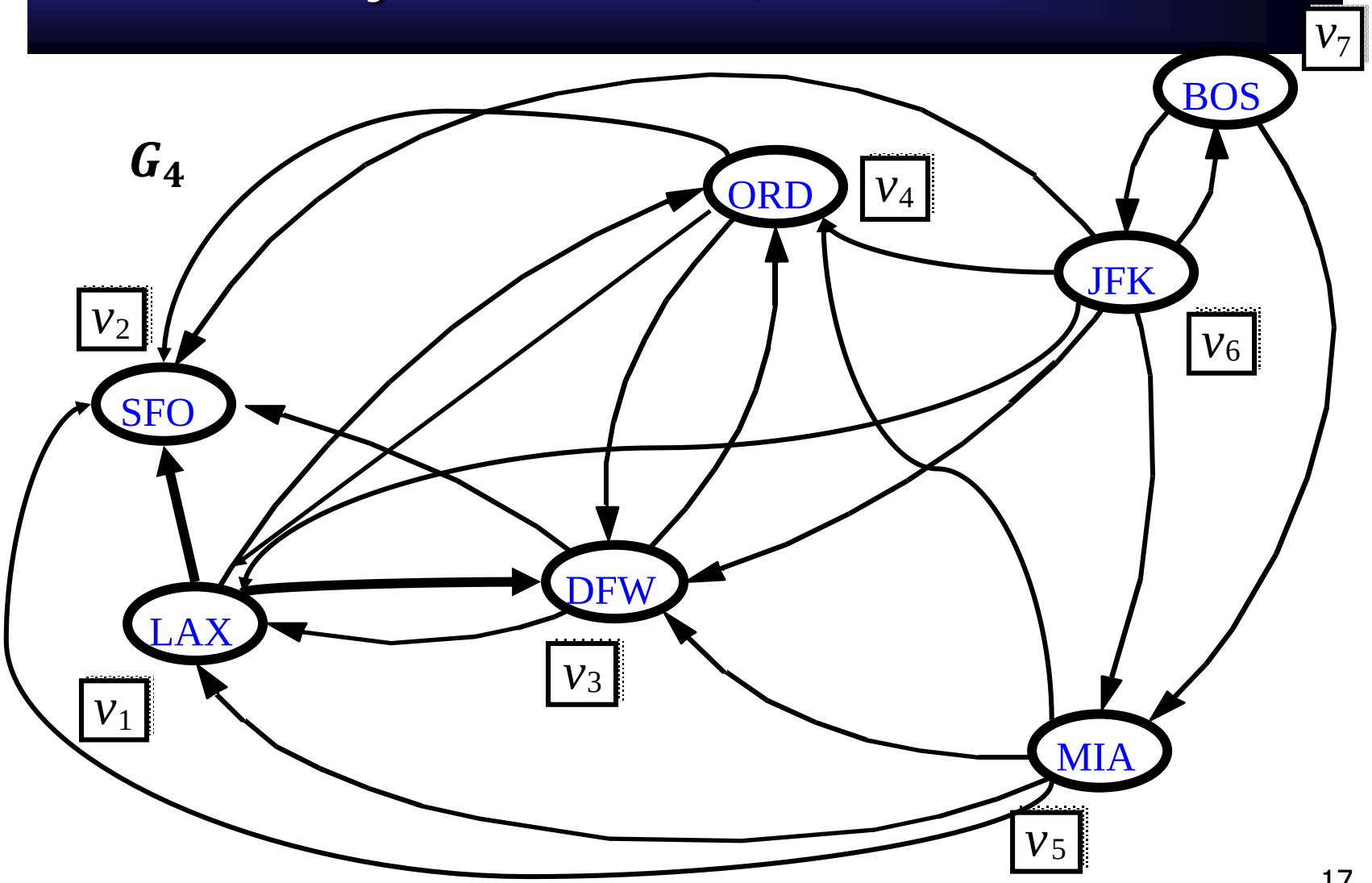
Floyd-Warshall, Iteration 2



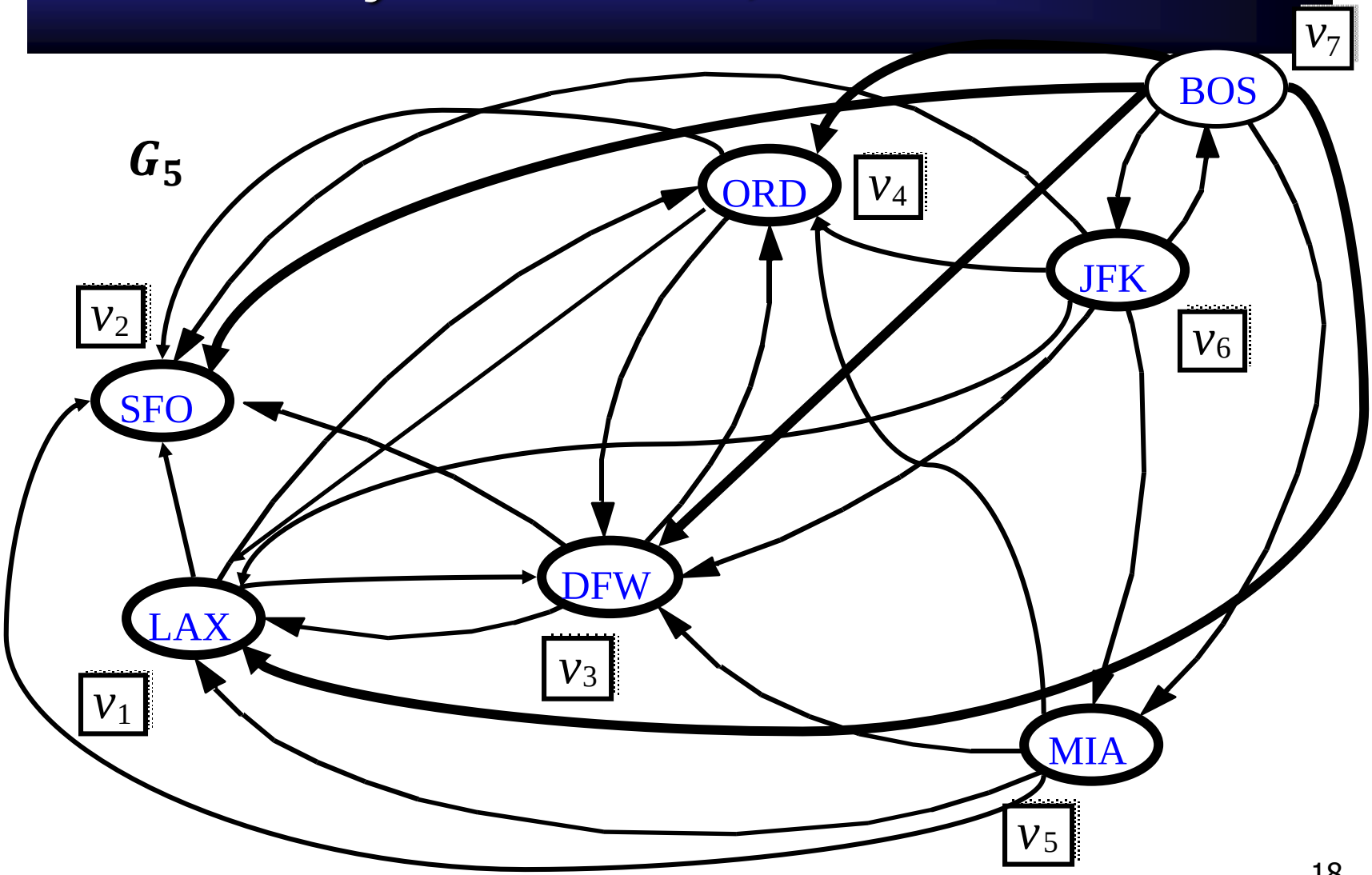
Floyd-Warshall, Iteration 3



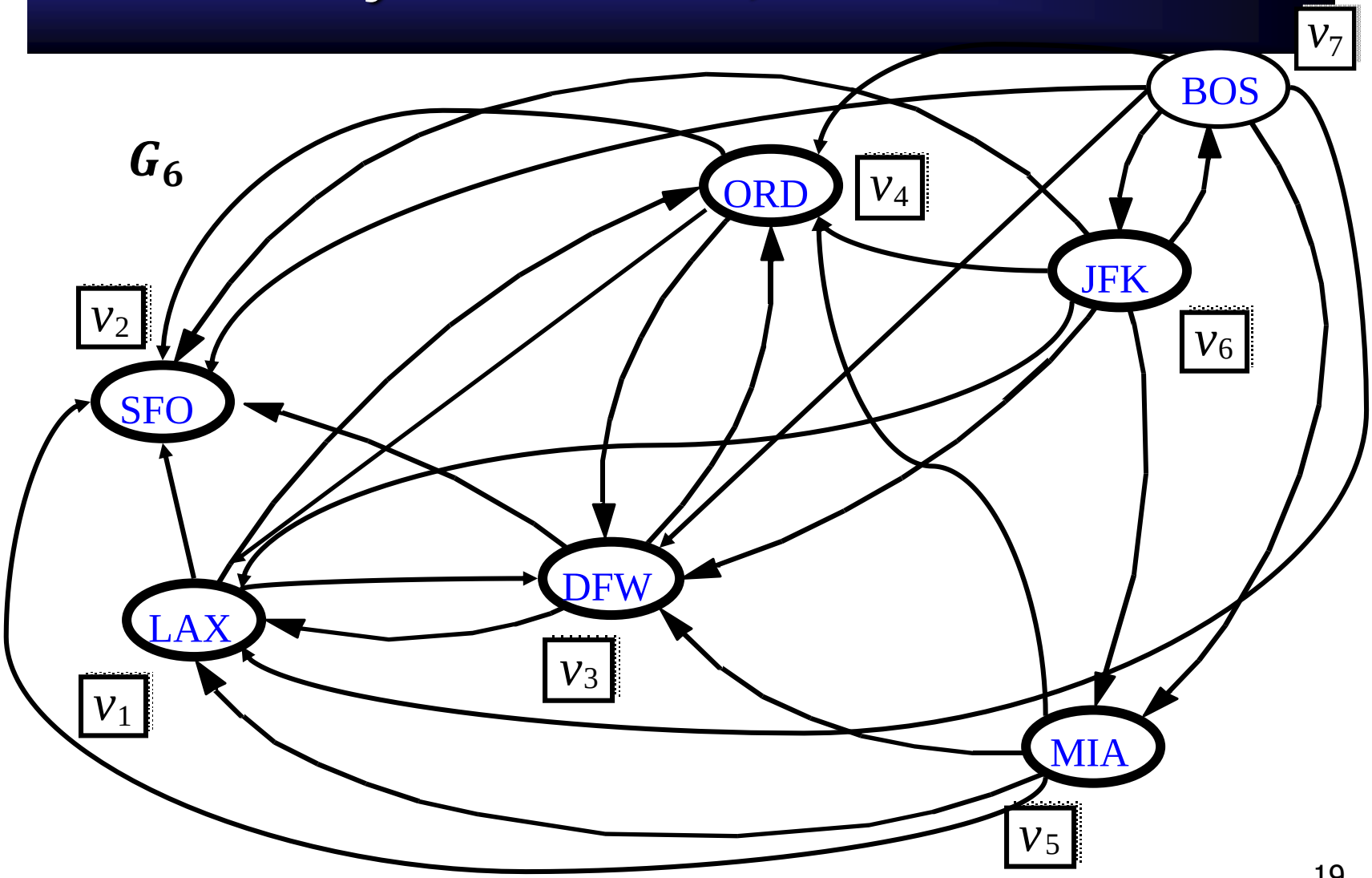
Floyd-Warshall, Iteration 4



Floyd-Warshall, Iteration 5

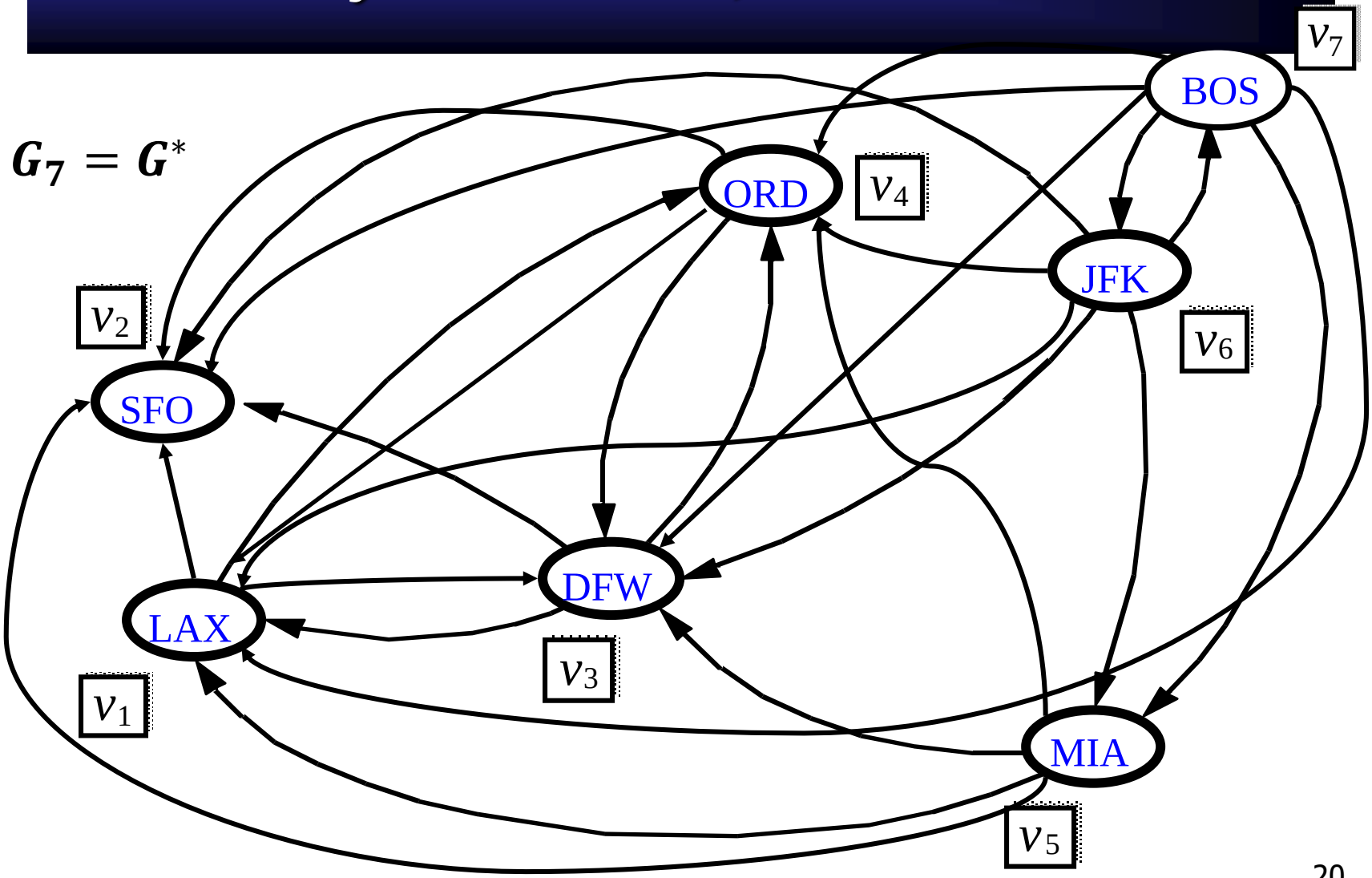


Floyd-Warshall, Iteration 6



Floyd-Warshall, Conclusion

$G_7 = G^*$

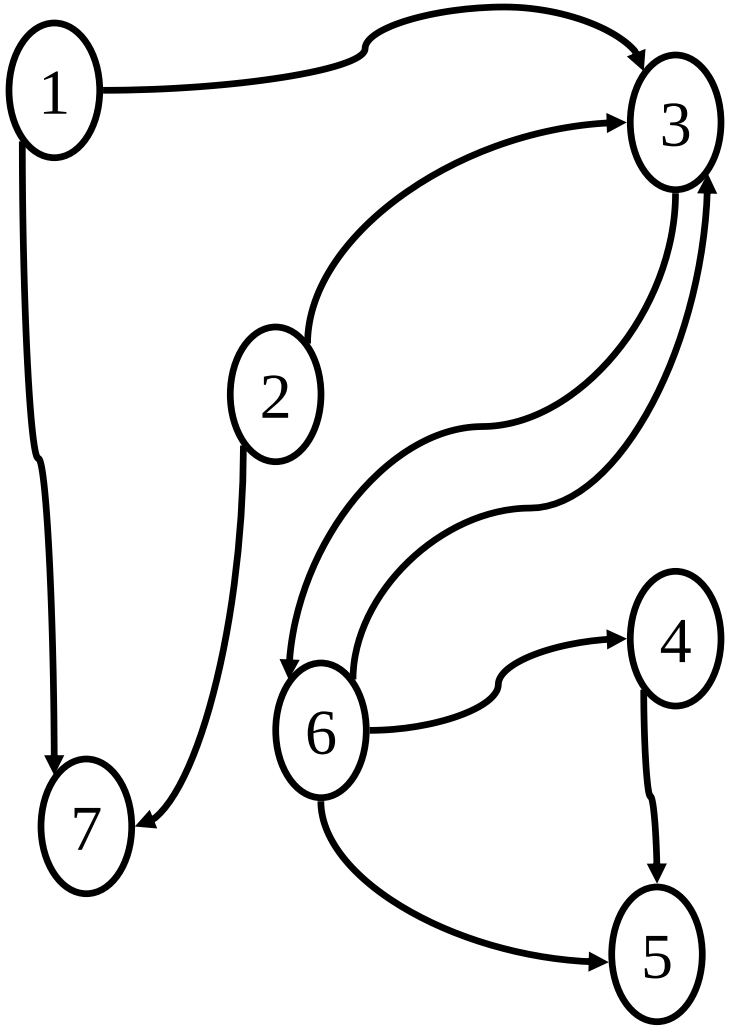


Floyd-Warshall Adj. Matrix

- 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** \\ start vertex in G_{k-1}
    if $i \neq k$ **then**
      for $j \leftarrow 1$ **to** n **do** \\ 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							

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							

- 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.

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							

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							

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							

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							

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							

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							

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							

i

- 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.

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							

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							

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							

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							

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							

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							

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							

i \ j	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							

j

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							

i

i \ j	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							

j

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							

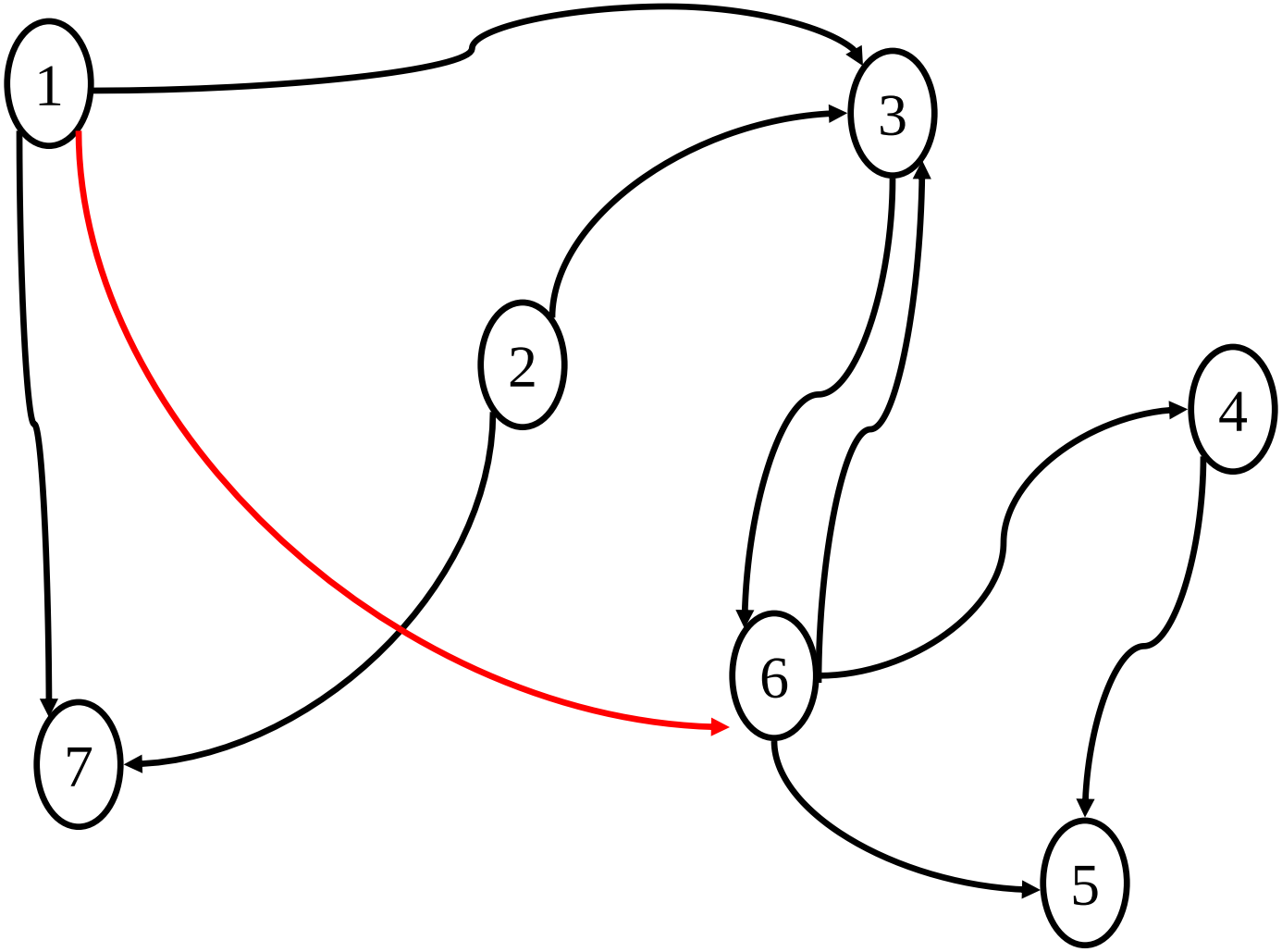
i

j

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							

i

i \ j	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							


1
3
2
4
6
7
5

j

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							

i

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							

j

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							

i

j

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							

i

	j						
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							

i

	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							

j

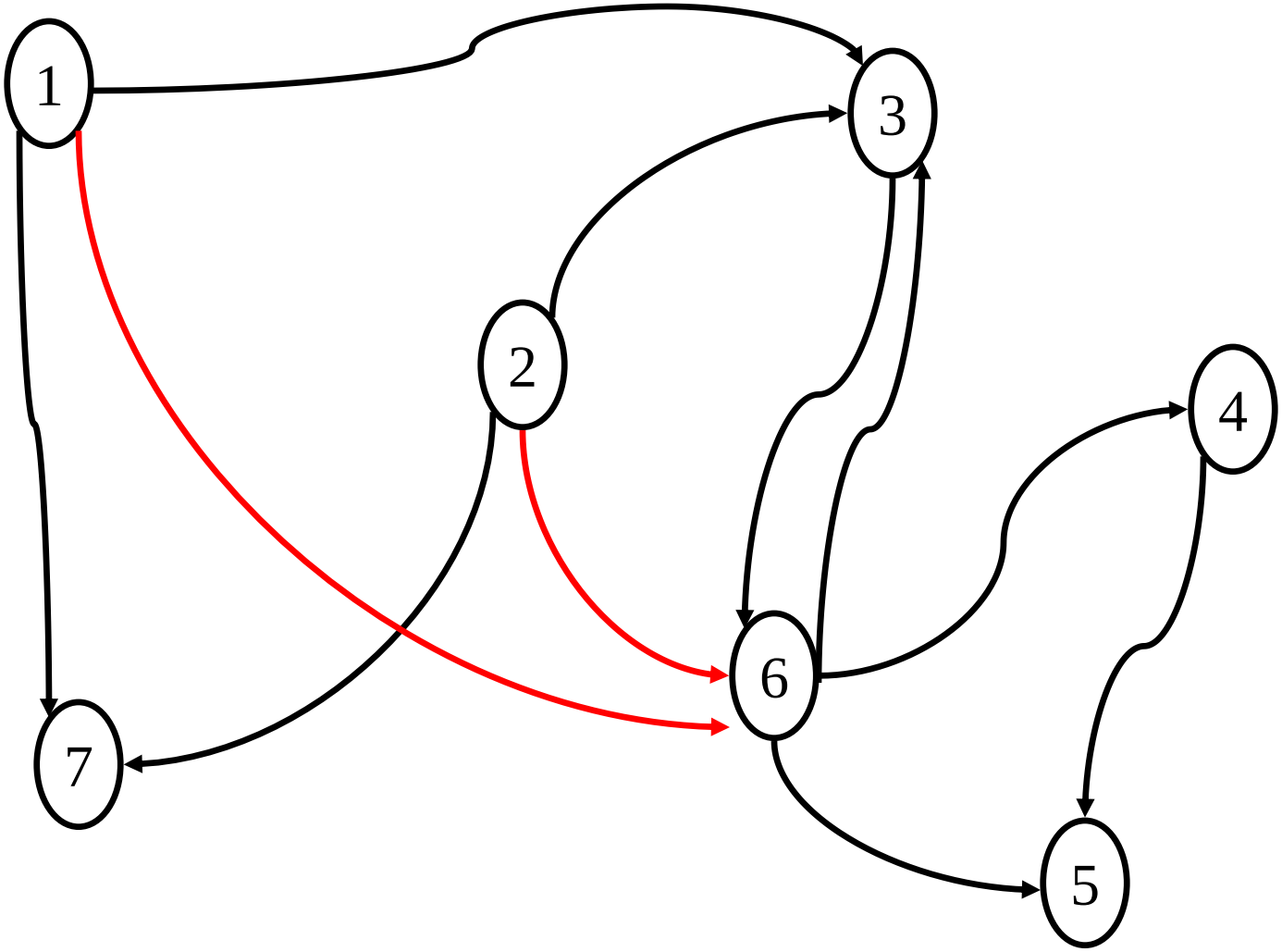
k=3	1	2	3	4	5	6	7
1			1			1	1
2			1				1
3						1	
4					1		
5							
6			1	1	1		
7							

i

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							

i


1
3
2
4
6
7
5

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							

i

i

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							

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							

i

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							

i

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							

i

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							

i

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							

i

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							

i

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							

i

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	1	
7							

i

- 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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

i

- 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							

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							

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							

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							

i

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							

	j							
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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

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							

i

- 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							

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							

j

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							

i

j

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							

i

j

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							

i

	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							

j

j

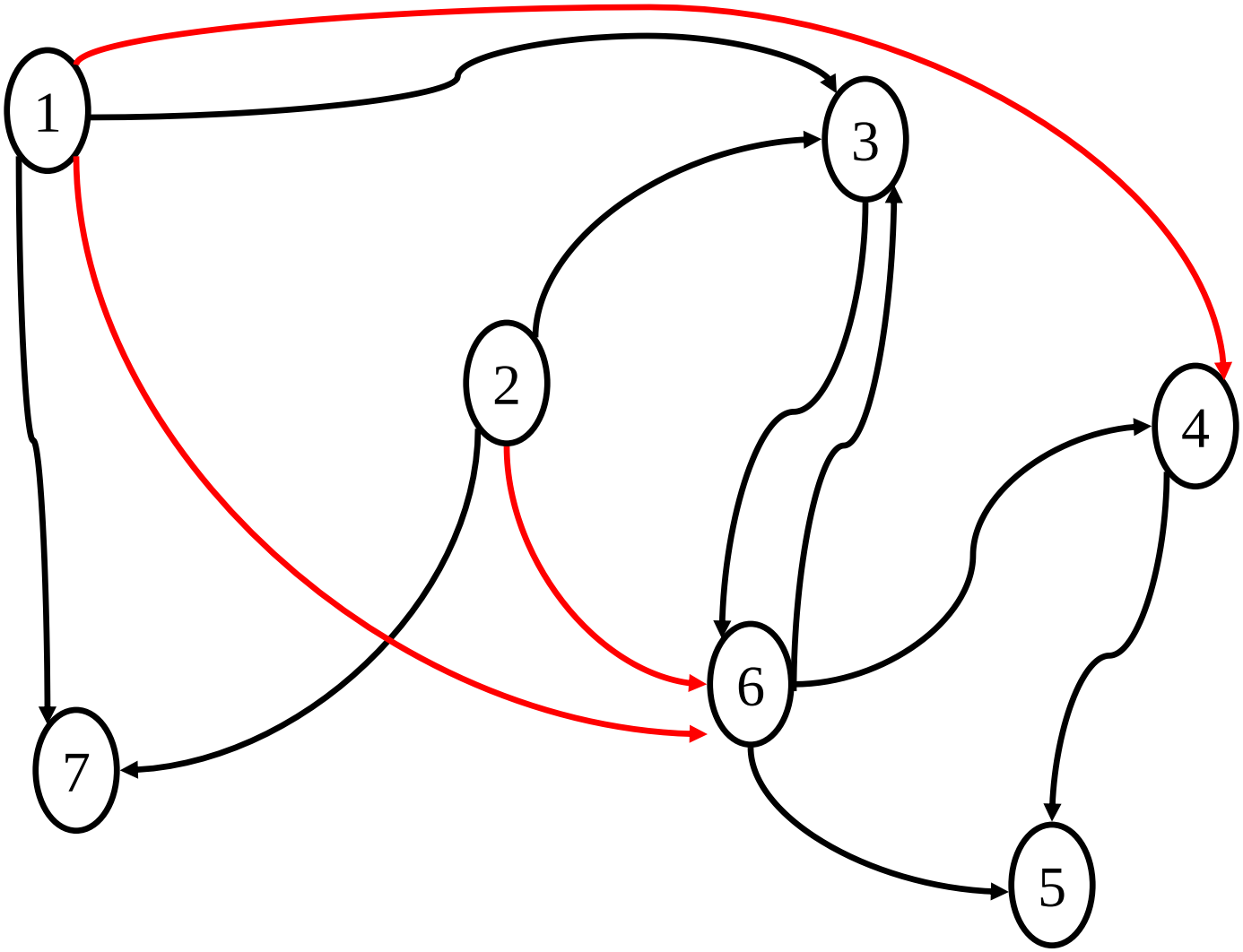
k=6	1	2	3	4	5	6	7
1			1			1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							

i

j

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							

i


1
3
2
4
6
7
5

j

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							

i

j

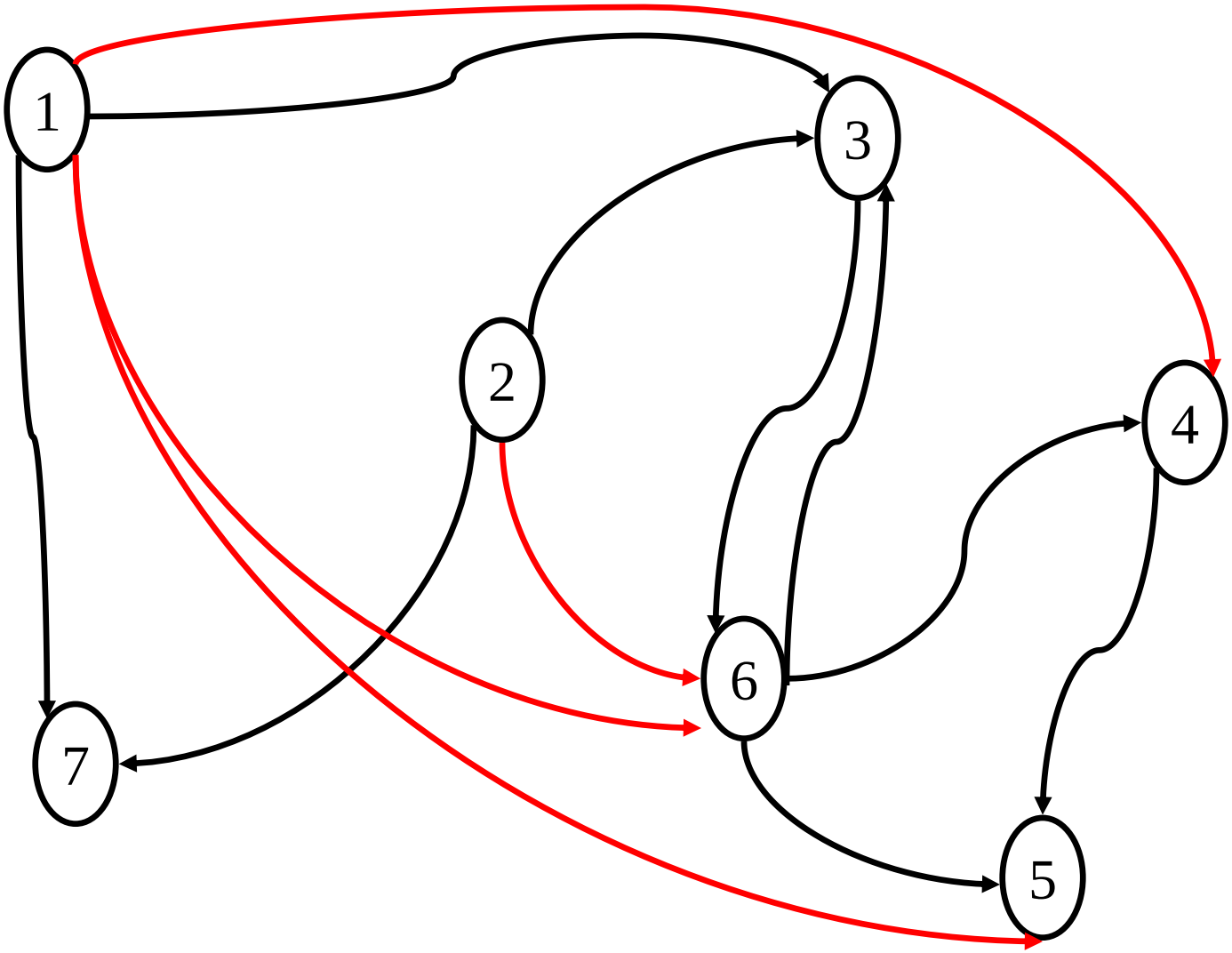
k=6	1	2	3	4	5	6	7
1			1	1		1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							

i

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							

i


1
3
2
4
6
7
5

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							

i

i \ 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							

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							

i

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							

i

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							

i

j

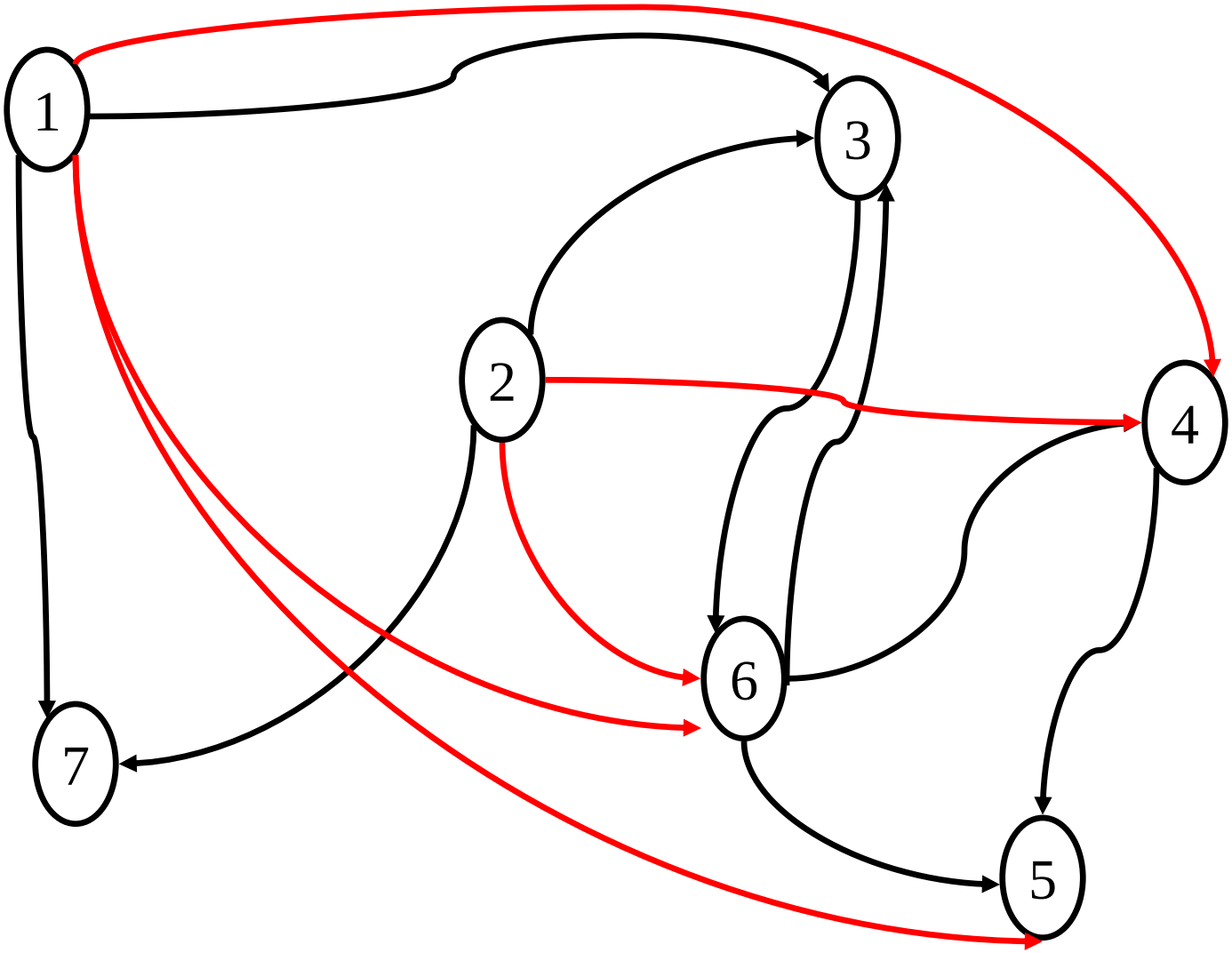
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1			1	1
3						1	
4					1		
5							
6			1	1	1		
7							

i

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							

i


1
3
2
4
6
7
5

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							

i

j

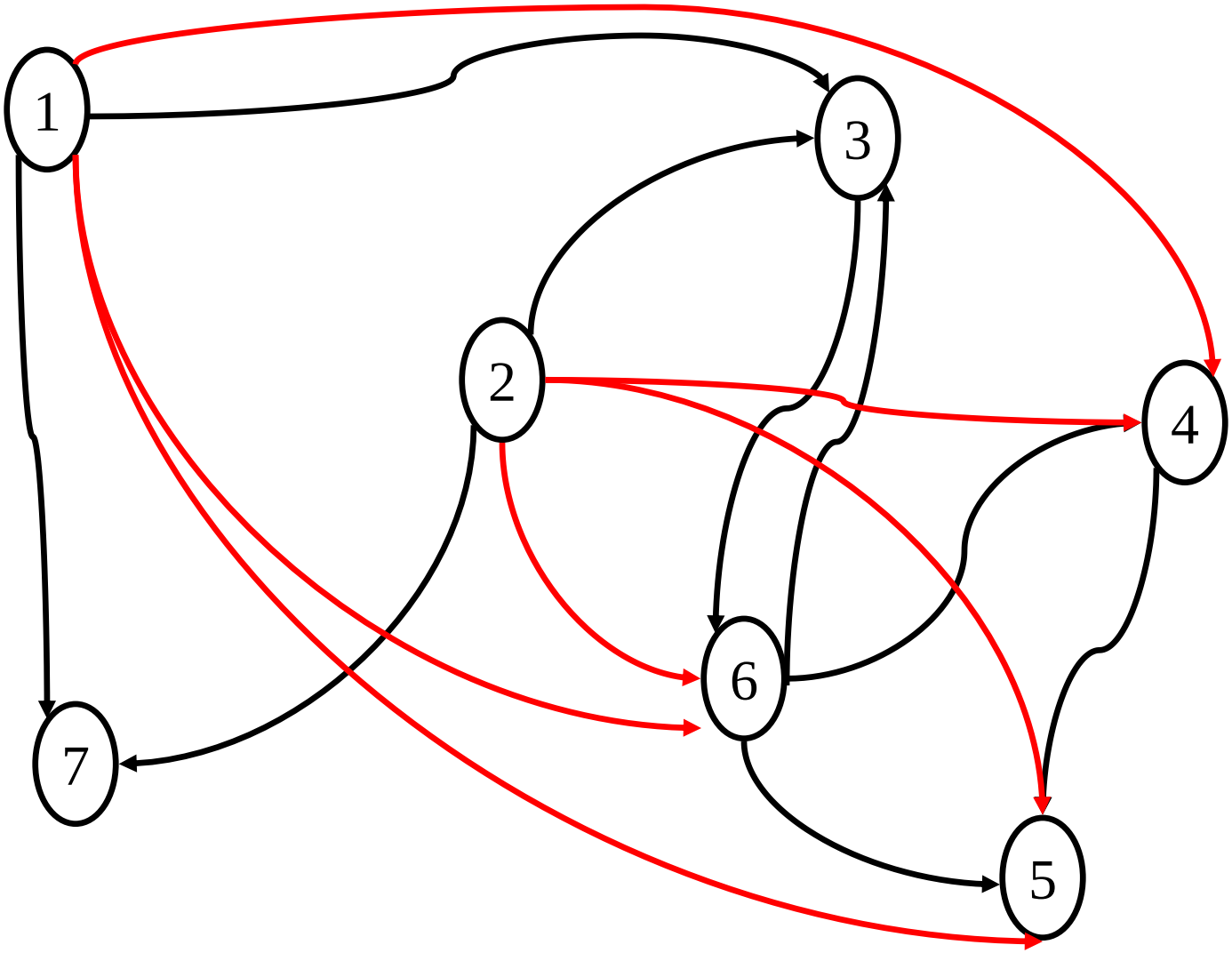
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1		1	1
3						1	
4					1		
5							
6			1	1	1		
7							

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							

i


1
3
2
4
6
7
5

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							

i

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							

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							

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							

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							

i

j

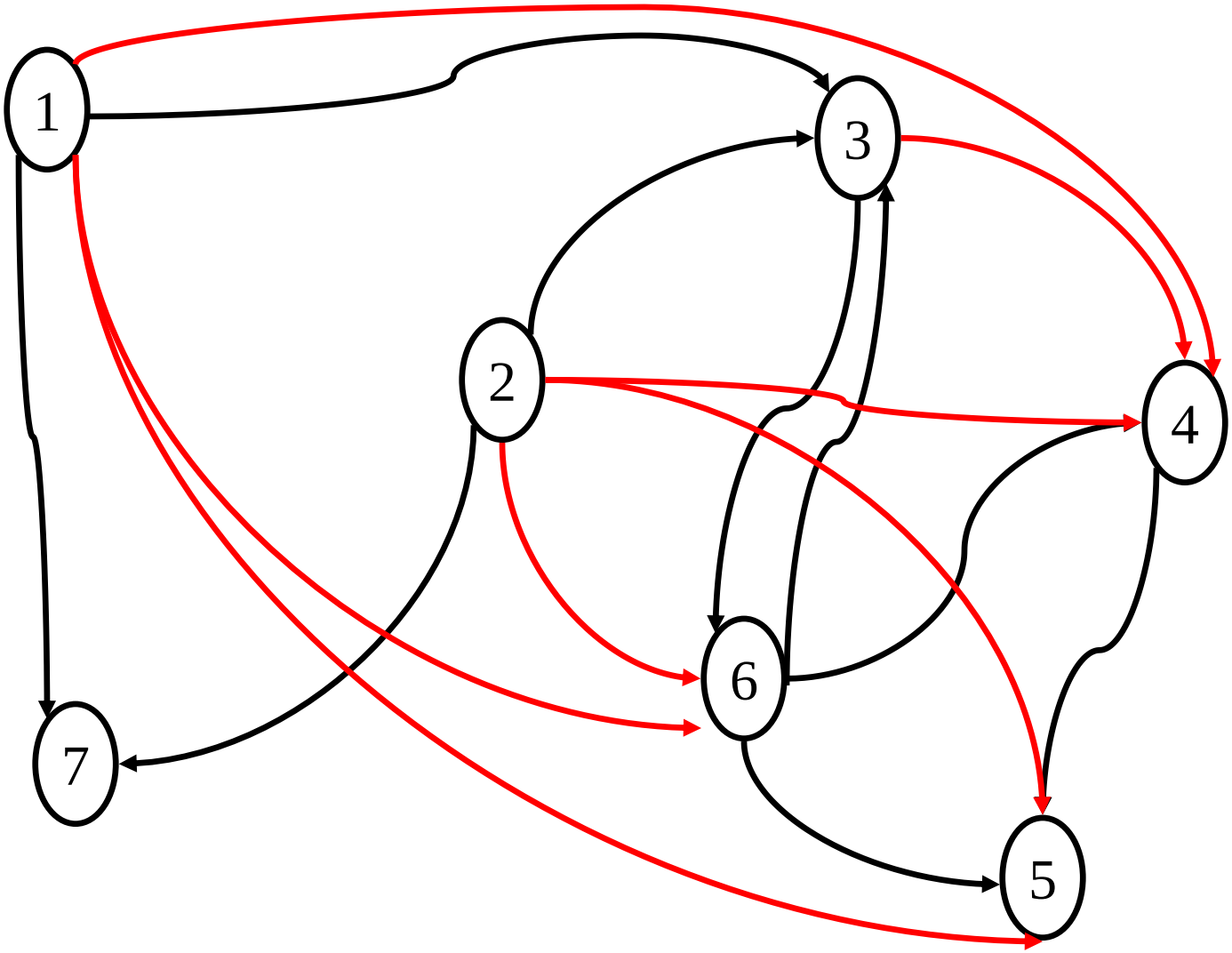
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3						1	
4					1		
5							
6			1	1	1		
7							

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		1	
4					1		
5							
6			1	1	1		
7							

i


1
3
2
4
6
7
5

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							

i

j

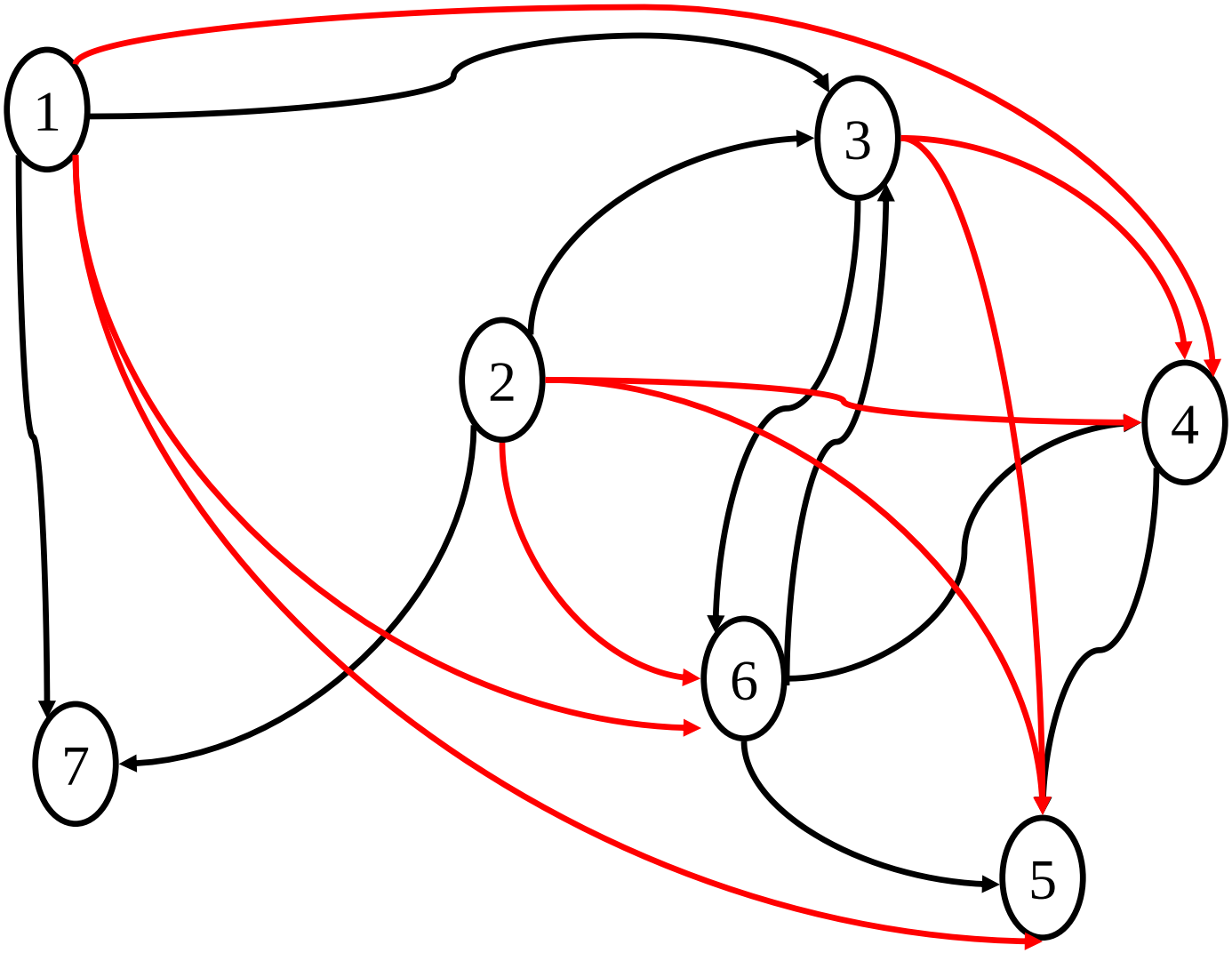
k=6	1	2	3	4	5	6	7
1			1	1	1	1	1
2			1	1	1	1	1
3				1		1	
4					1		
5							
6			1	1	1		
7							

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	1	1	
4					1		
5							
6			1	1	1		
7							

i


1
3
2
4
6
7
5

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							

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							

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							

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							

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							

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							

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							

i

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							

i

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							

i

		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							

	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							

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							

	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							

i

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							

i

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							

i

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							

i

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

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							

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							

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							

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							

i