SHORTEST PATHS AND RELATED PROBLEMS

- Shortest path problems are pervasive in network optimization.
- A key objective of any traveler, passenger, or carrier is to move from point *a* to point *b* along a *shortest, cheapest,* or *most comfortable path.*
- Associating flow costs with arcs in a network, the user seeks the minimum cost path from *a* to *b*.
- In economic terms, there is a supply of one or more units at node *a*, a demand for these units at node *b*, and a link flow cost assigned to each link in the network.
- In the context of computer networks, the goal is to send packets of information from *a* to *b* with minimum delay.
- Shortest path problems are often important subproblems for more complex network problems.

TYPES OF SHORTEST PATH PROBLEMS

- For a given network G with arc costs given by the matrix $\mathbf{D} = [d(i, j)]$, there are four shortest path problems of general interest that we consider.

  (SP1) Find the shortest path from a specific origin s to a specific destination t;

  (SP2) Find the shortest paths from a specific origin s to all other nodes;

  (SP3) Find the shortest paths between all pairs of nodes;

  (SP4) Find the second, third, and so on shortest paths.

- The distance entries $d(i, j)$ can be positive, negative, or zero provided that there is no cycle whose total cost is negative. If a negative cycle did exist, costs could be minimized by traversing it infinitely often.

SHORTEST PATHS FROM A GIVEN NODE

- An algorithm for this problem will solve the O-D path problem as well.
- Two algorithmic strategies
    - Label correcting
    - Label setting
- Label correcting methods consider all labels as temporary until the final step when they all become permanent.
- Label-setting methods designate one or more labels as permanent at each step of the procedure.

A LABEL CORRECTING ALGORITHM

Step 0. Definitions

$\ell(v)$ is the length of the current "shortest path" from node 0 to node v.

p(v) is the predecessor of v in the current "shortest path" to this node.

d(i, k) is the length of arc i - $k \in$ A.

T is a list of nodes to be scanned.

Step 1. Initialization

$$\ell(v) = \begin{cases} 0 & \text{if } v = 0 \\ \infty & \text{otherwise.} \end{cases}$$

p(v) = 0 for all v.

node 0 is the first element on list T.

A LABEL CORRECTING ALGORITHM — CONTINUED

Step 2. Basic Computation

Select the top element i from T. Scan element i. That is, for every node k such that $i - k \in A$, perform the following test:

If $\ell(i) + d(i, k) < \ell(k)$ then

(a) $\ell(k) = \ell(i) + d(i, k)$

$p(k) = i$, and

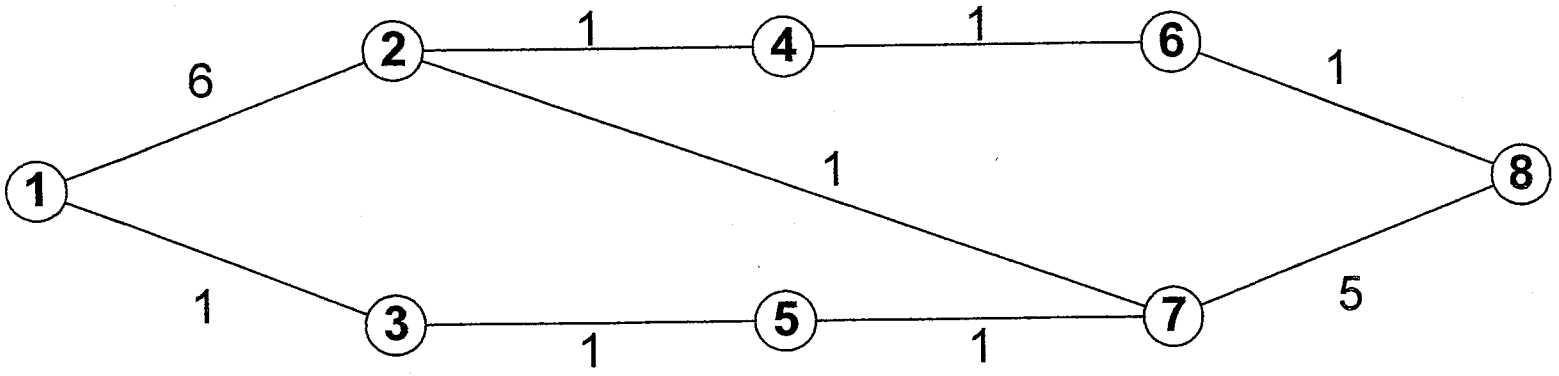
(b) place k at bottom of T, if it is not already on the list.

Step 3. Reducing the List Size

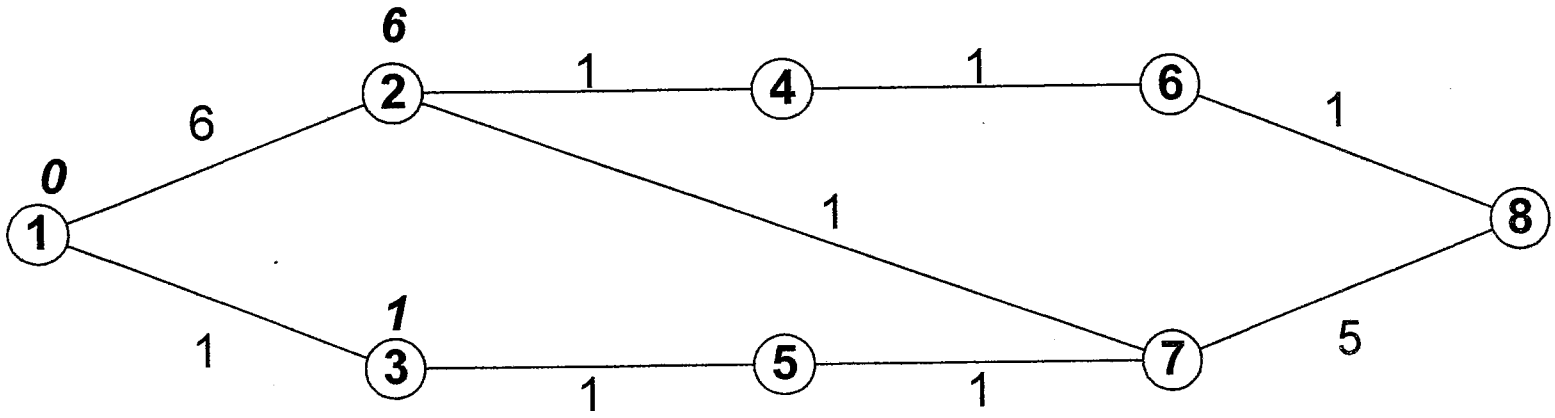
Remove (or cross out) node i from the list. Terminate the procedure if the list T is now empty. Otherwise, go to Step 2.

LABEL-CORRECTING EXAMPLE

Find the shortest path from node 1 to all other nodes in the undirected network below.

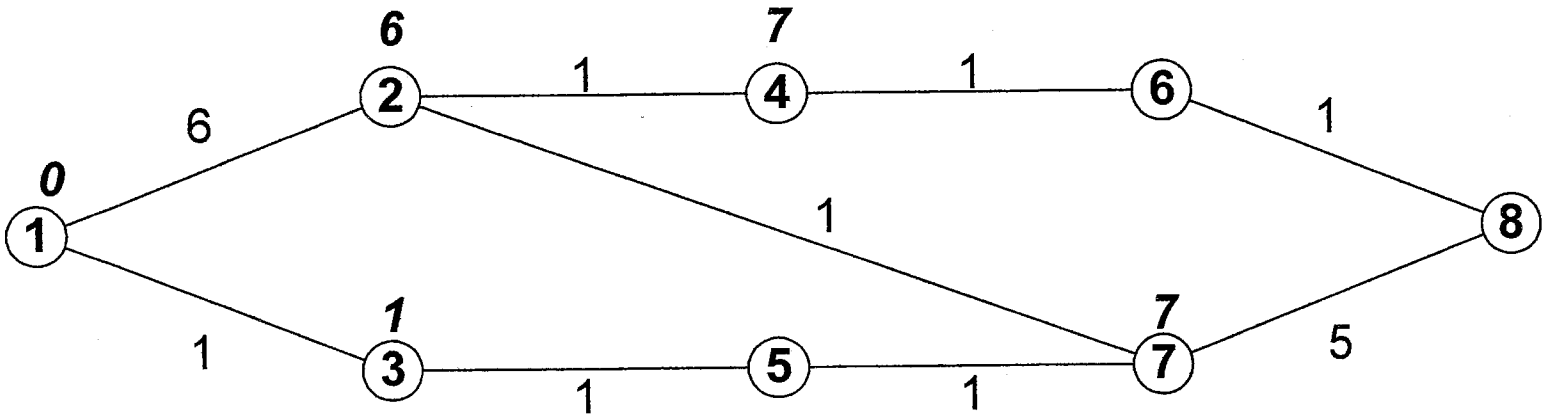


Scan node 1.

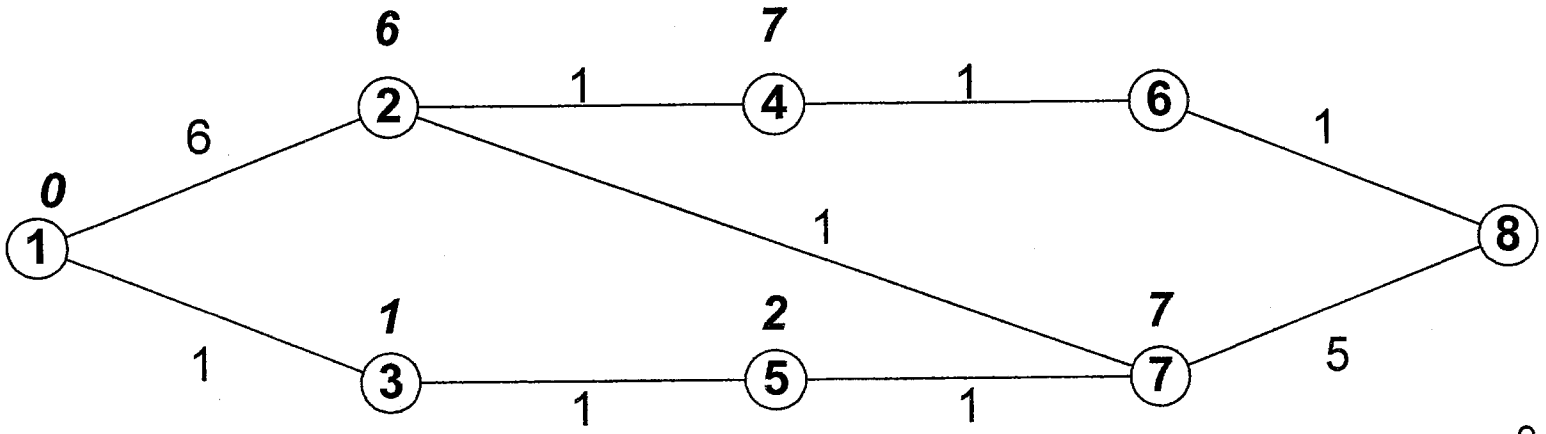


LABEL-CORRECTING EXAMPLE

Scan node 2.

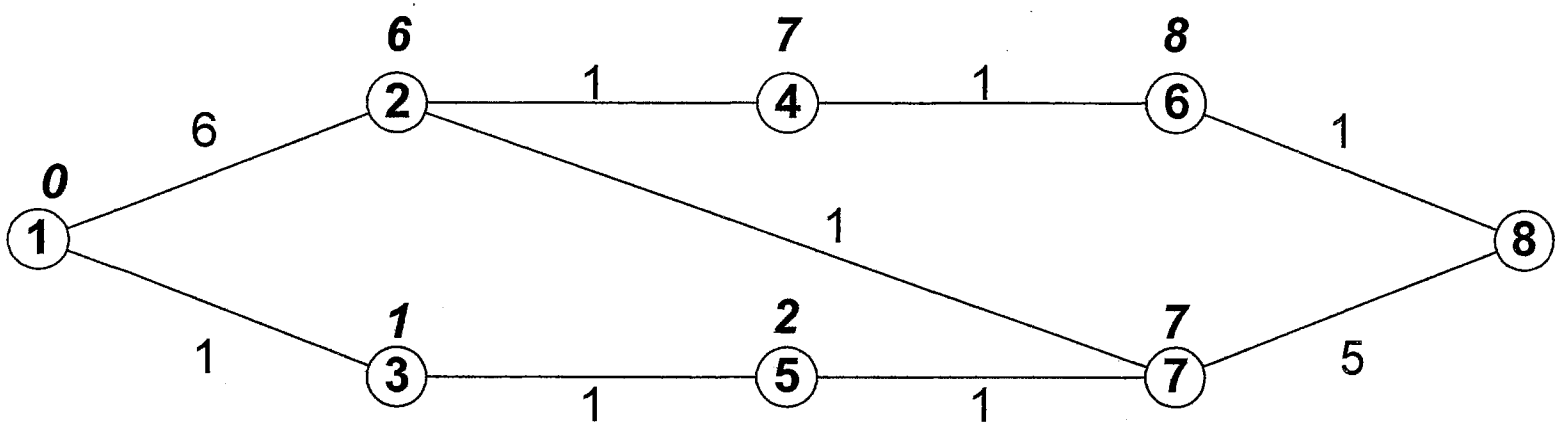


Scan node 3.

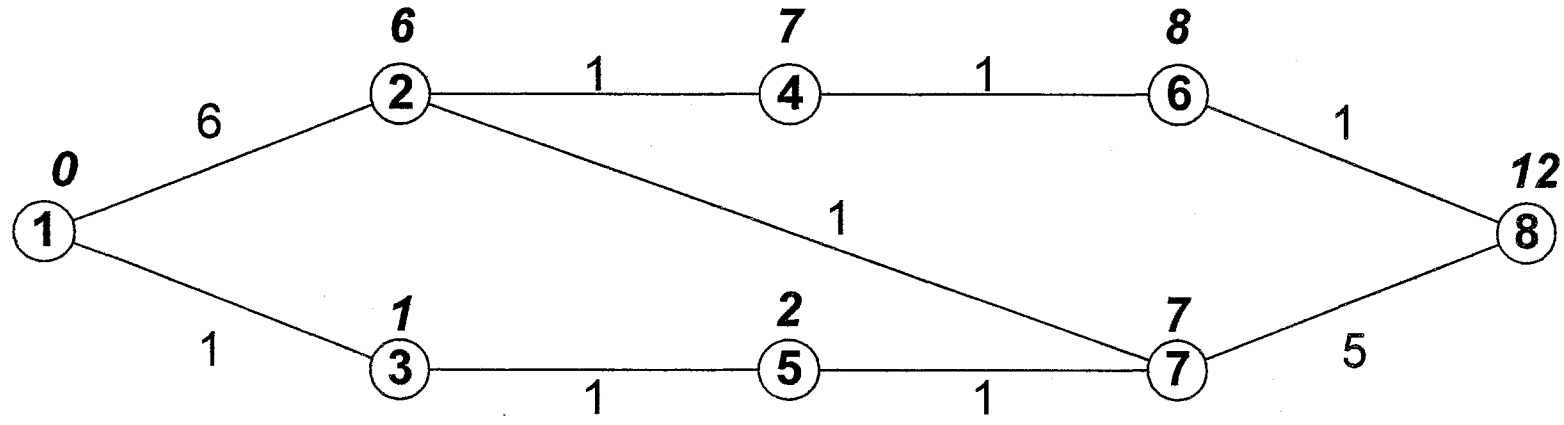


LABEL CORRECTING EXAMPLE

Scan node 4.

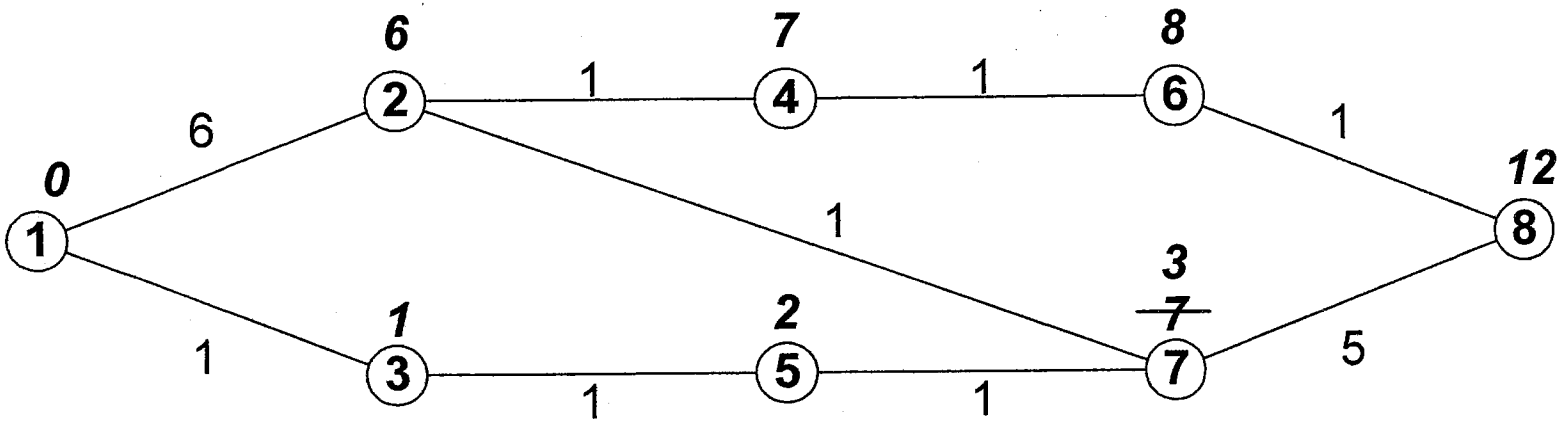


Scan node 7.

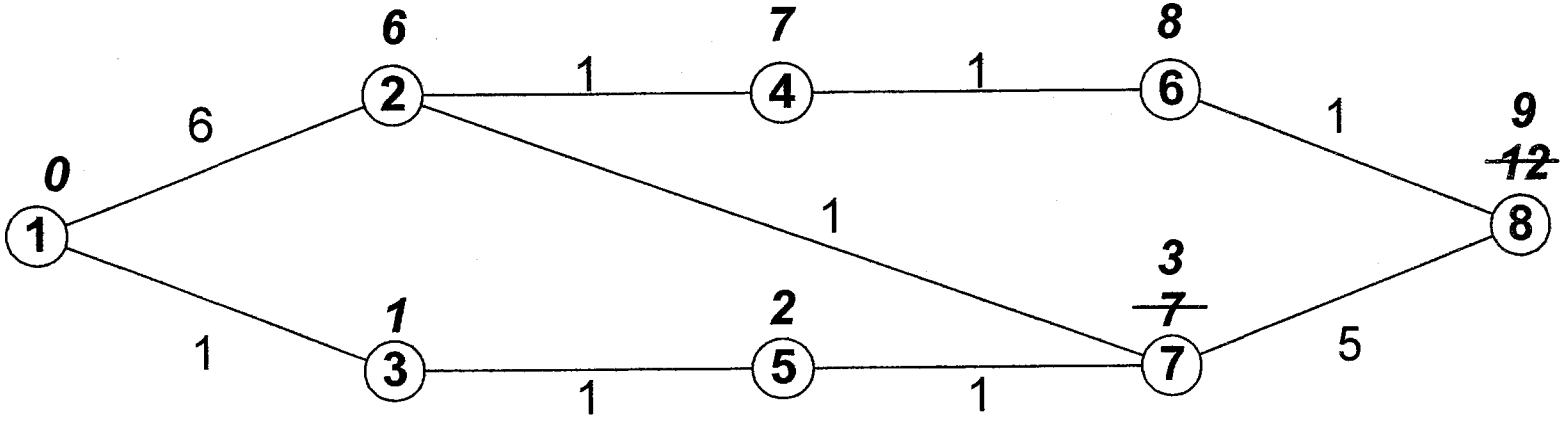


LABEL CORRECTING EXAMPLE

Scan node 5.

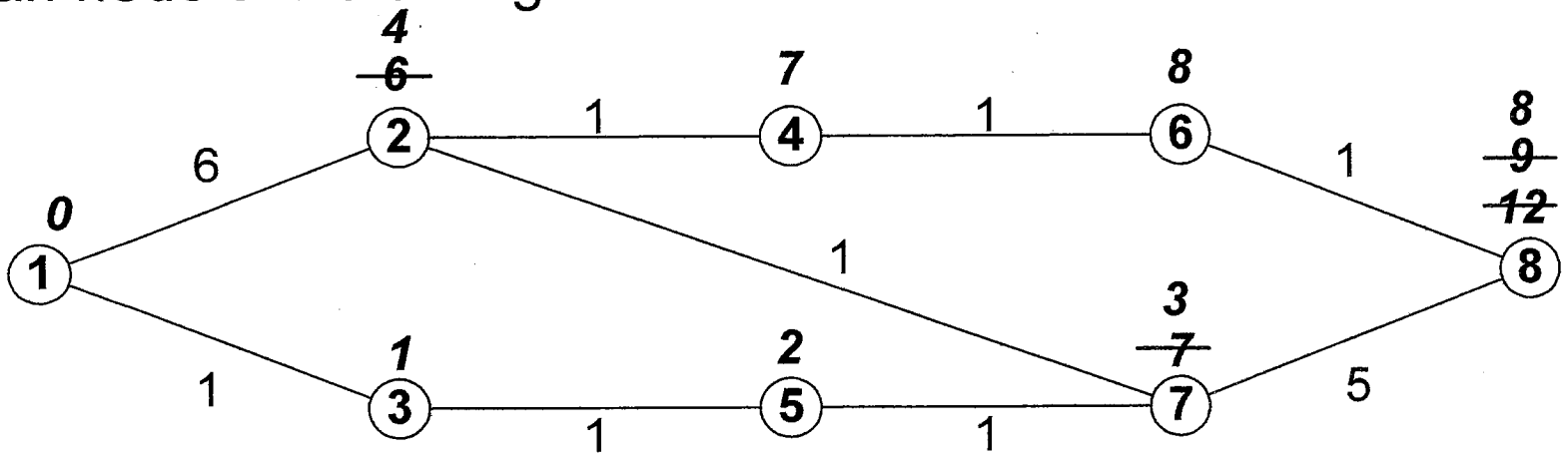


Scan node 6.

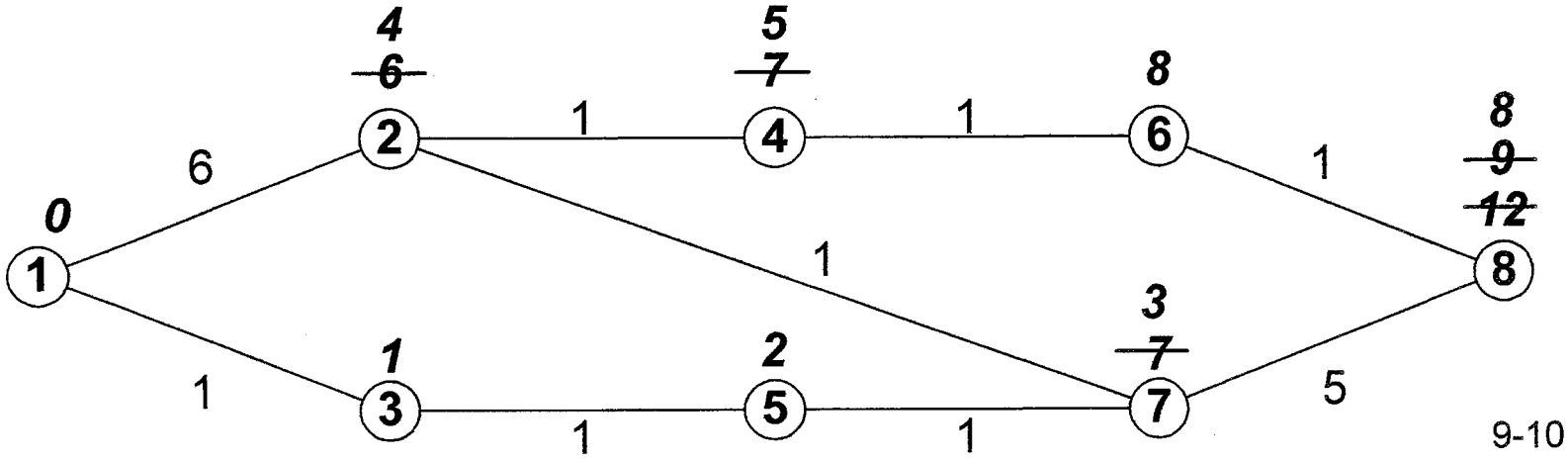


LABEL CORRECTING EXAMPLE

Scan node 8. No change. Scan node 7.

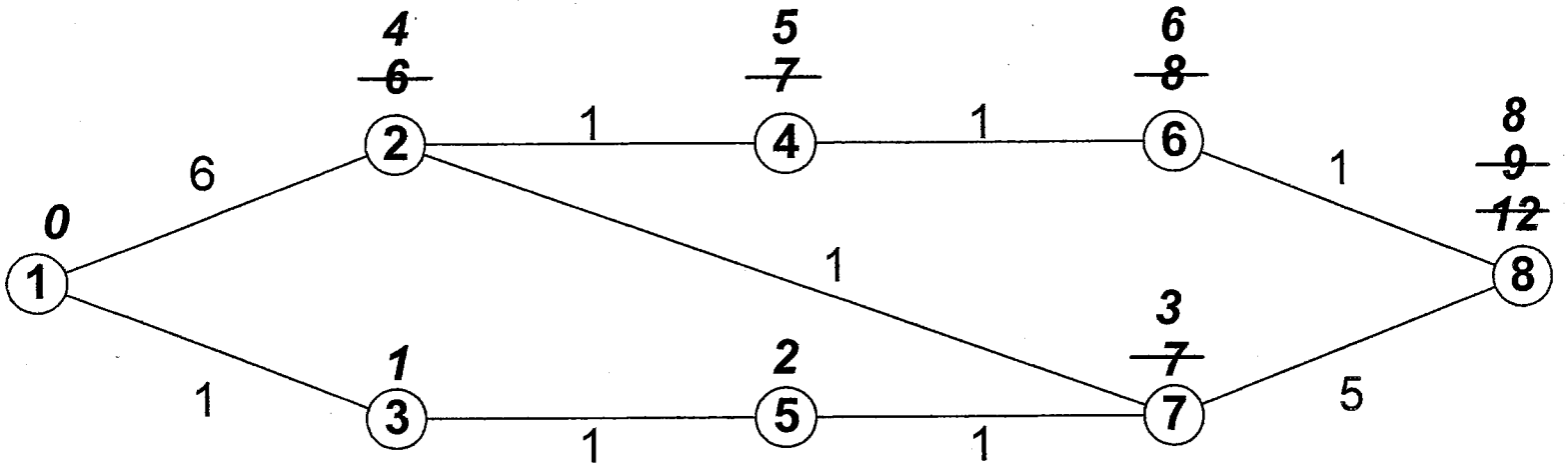


Scan node 2.

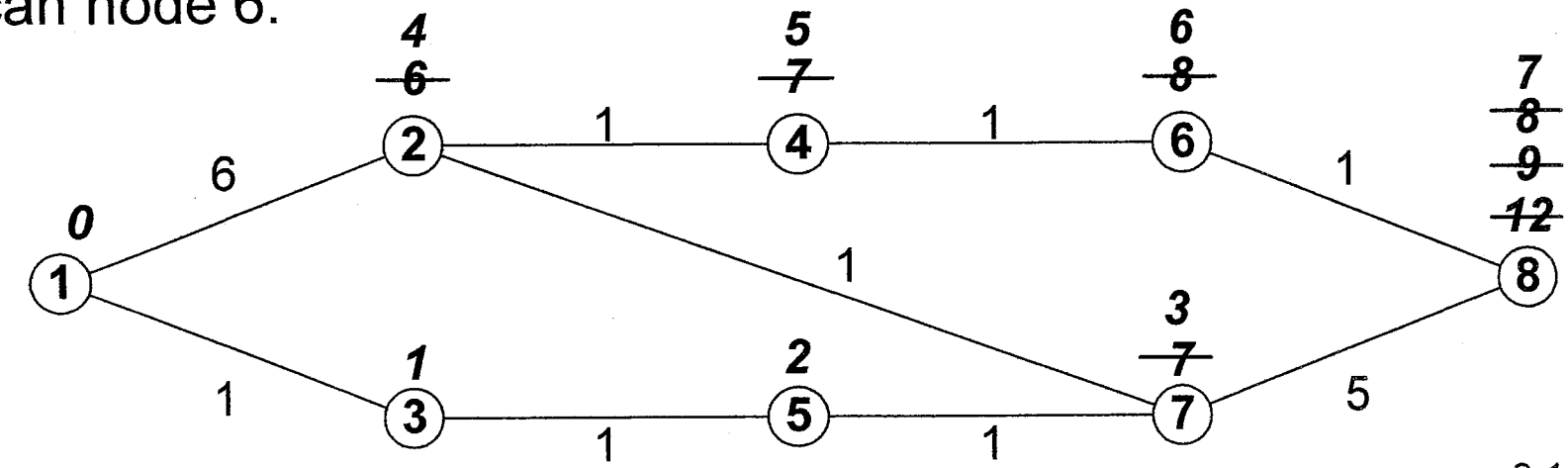


LABEL CORRECTING EXAMPLE

Scan node 8. No change. Scan node 4.



Scan node 6.



Scan node 8. No change. **Finished**

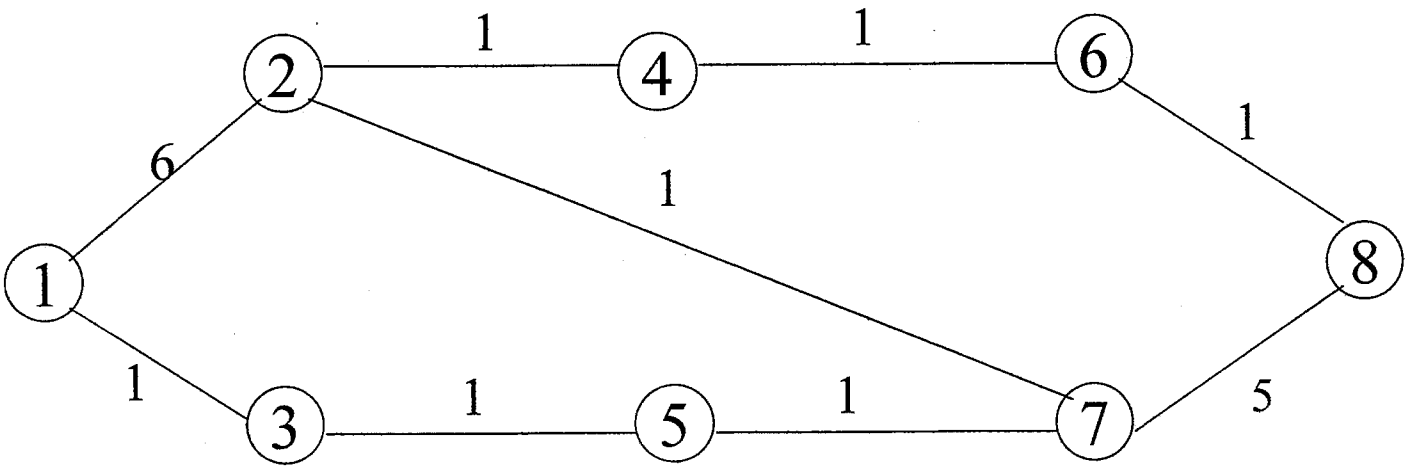
COMMENTS ON THE LABEL CORRECTING APPROACH

- Step 2 implements the fundamental (DP) recursion

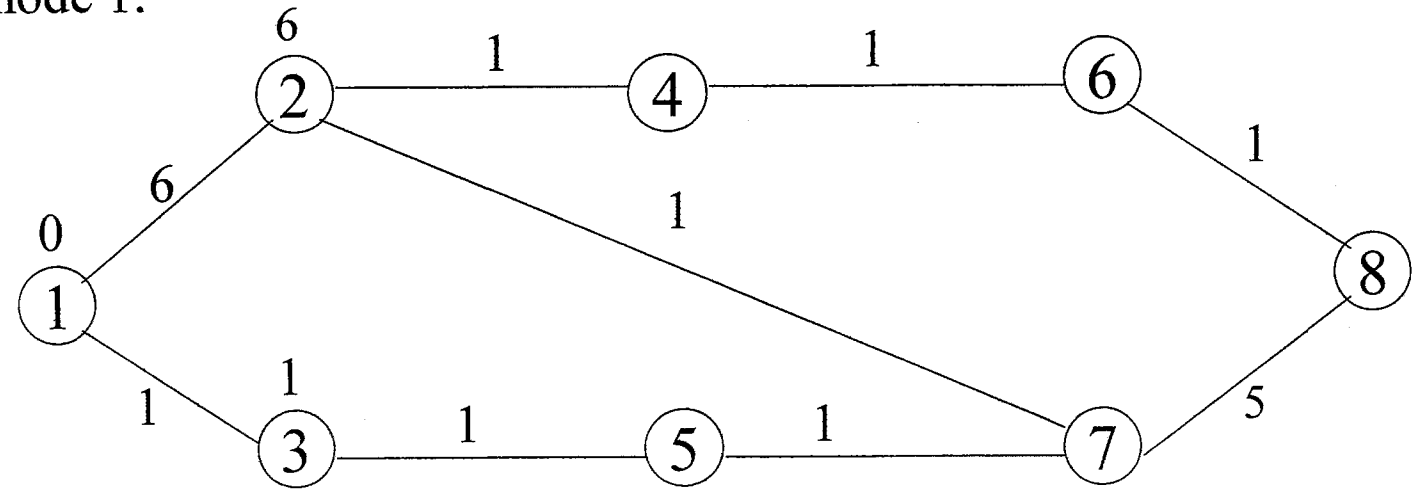
$$\ell^{k+1}(i) \leftarrow \min_{j} \{ \ell^{k}(j) + d(j, i) \}.$$

- The recursion states that the shortest path from the origin to node i using $k+1$ or fewer arcs is composed of some final arc j - i plus a shortest path to the predecessor j of node i using k or fewer arcs.
- The label-correcting procedure terminates with correct shortest path lengths.
- The algorithm requires at most $O(n^3)$ additions and comparisons.
- The label-setting procedure is known as Dijkstra's algorithm.

Pape's Algorithm

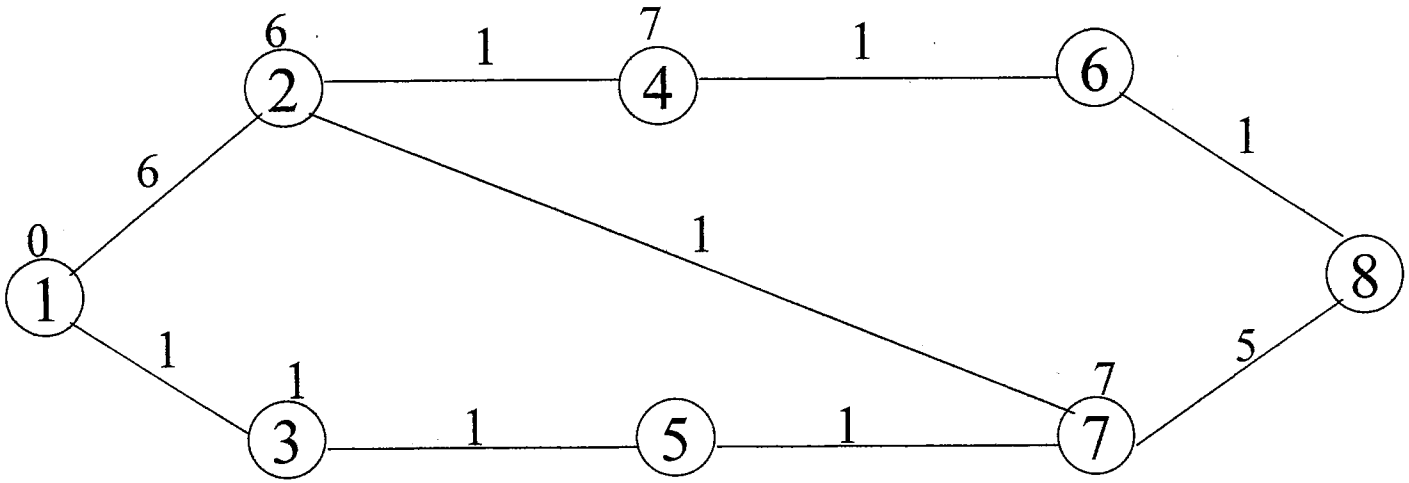


Scan node 1.

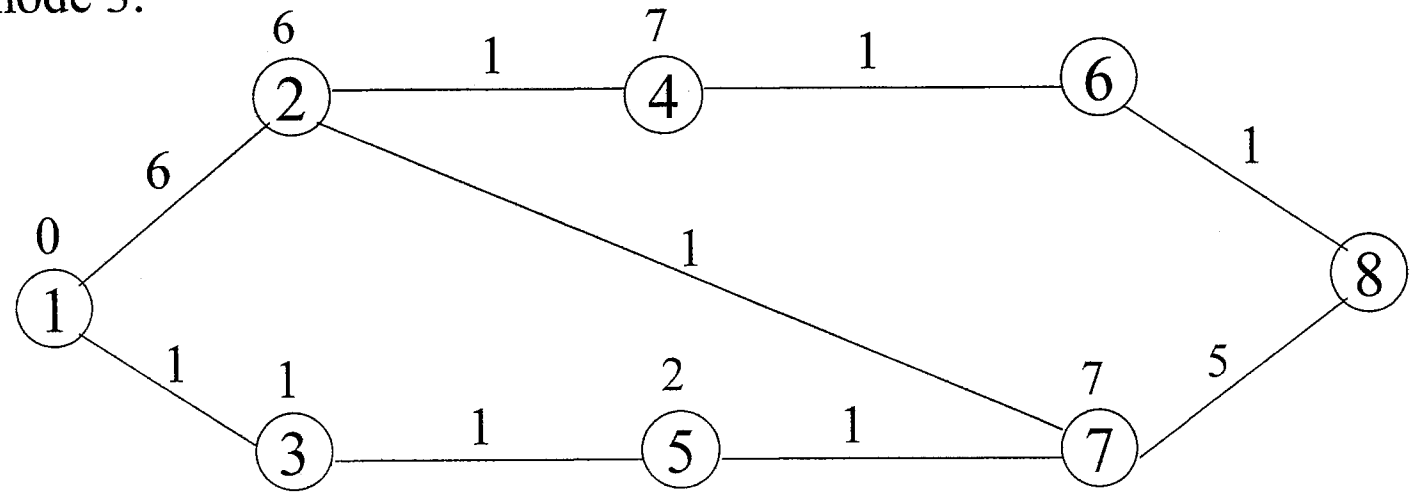


Pape's Algorithm

Scan node 2.

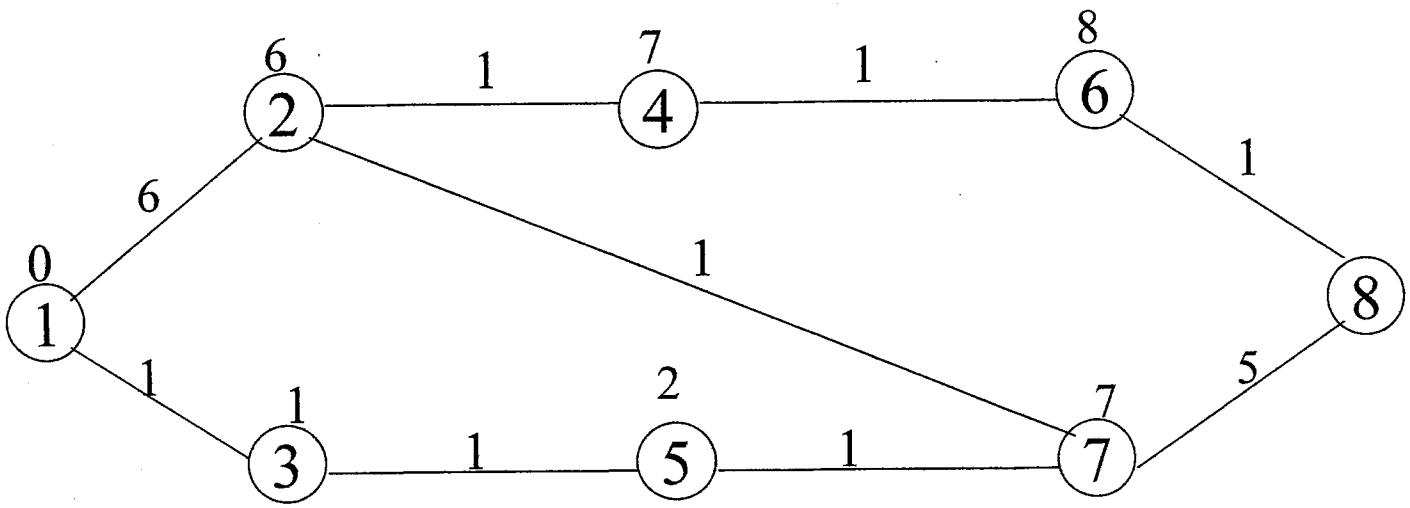


Scan node 3.

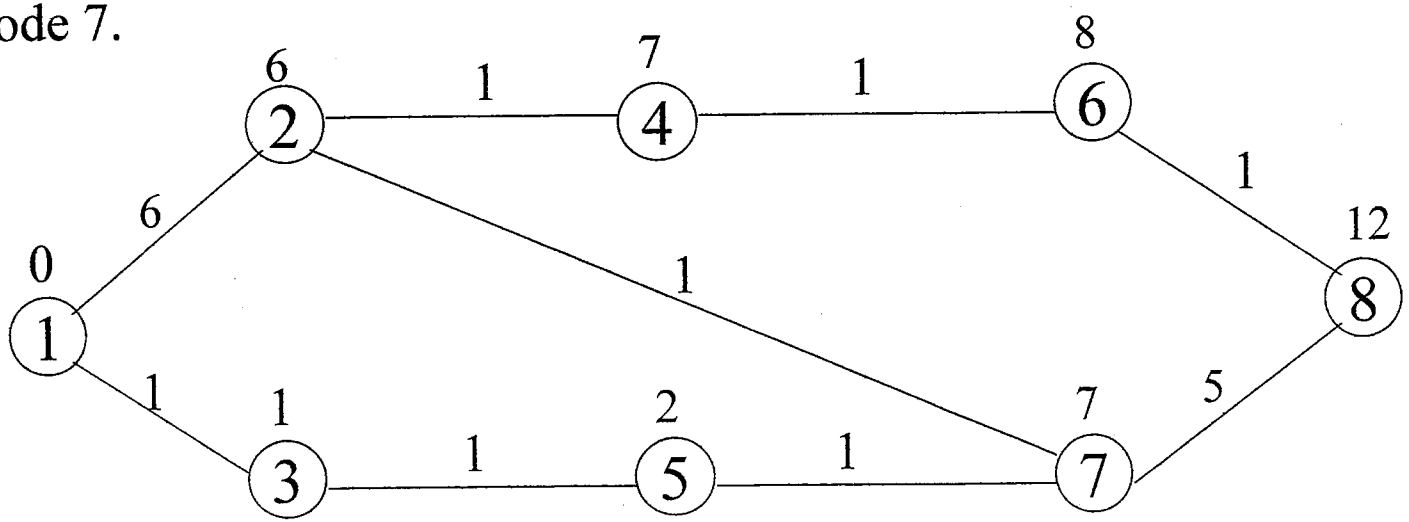


Pape's Algorithm

Scan node 4.

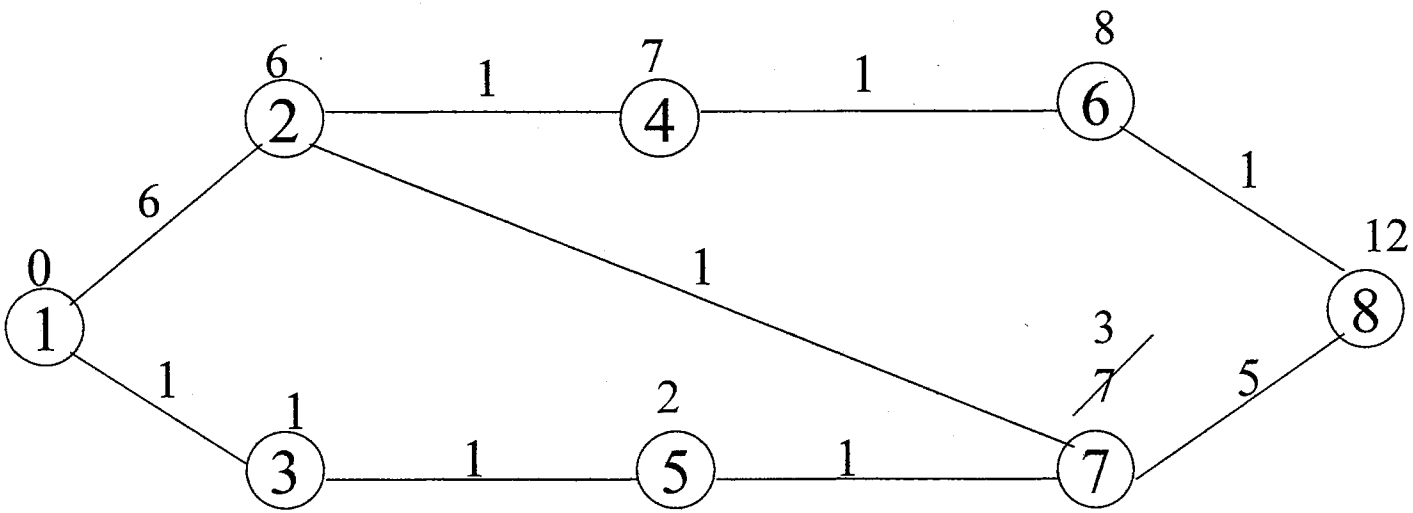


Scan node 7.

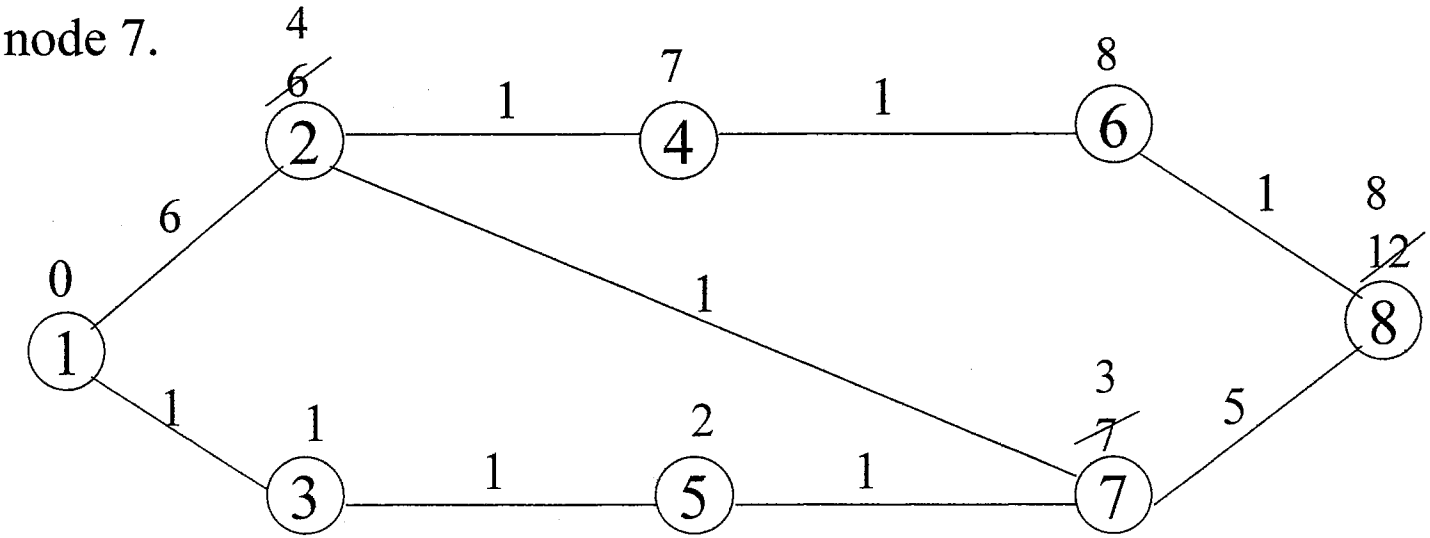


Pape's Algorithm

Scan node 5.

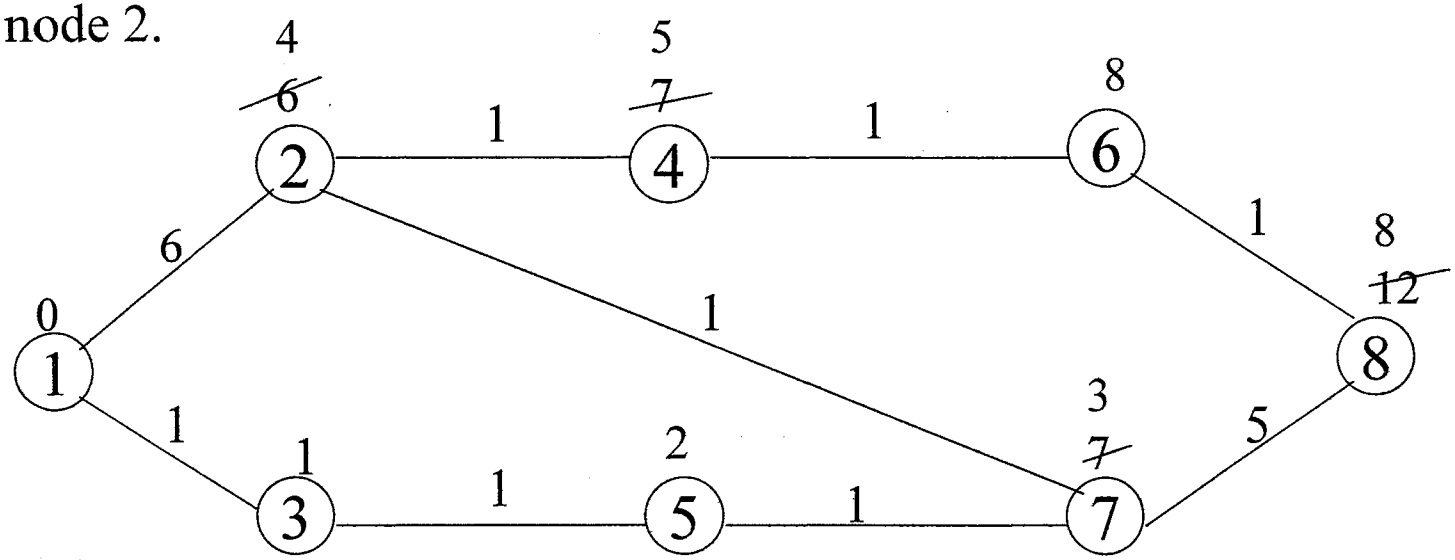


Scan node 7.

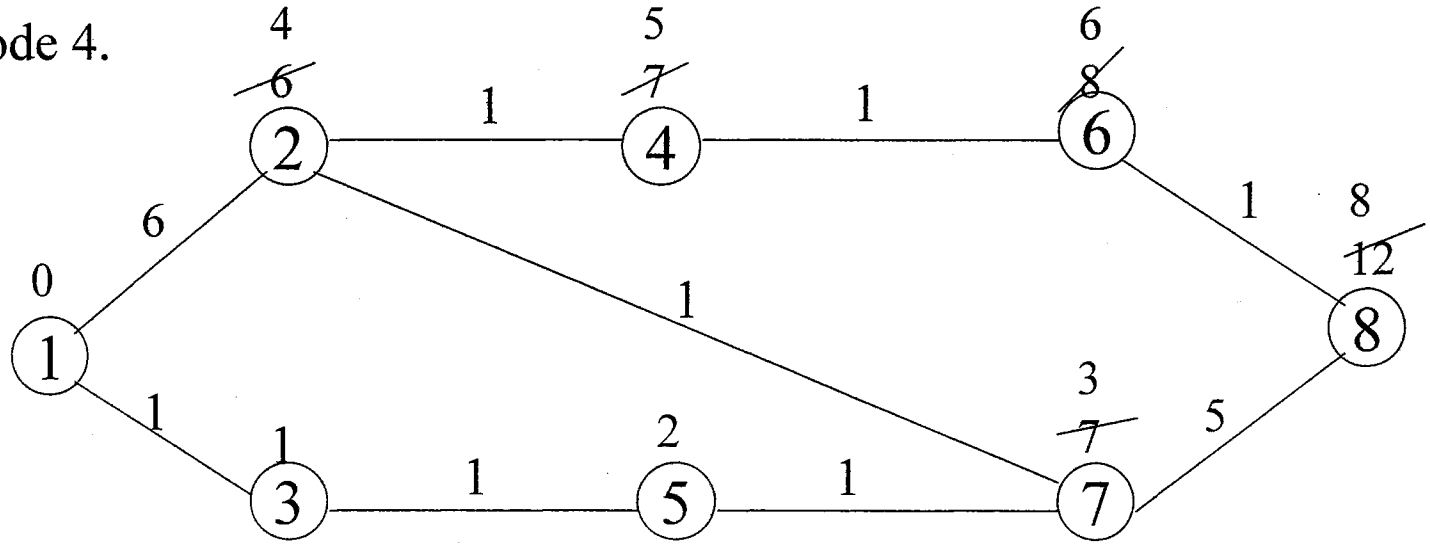


Pape's Algorithm

Scan node 2.

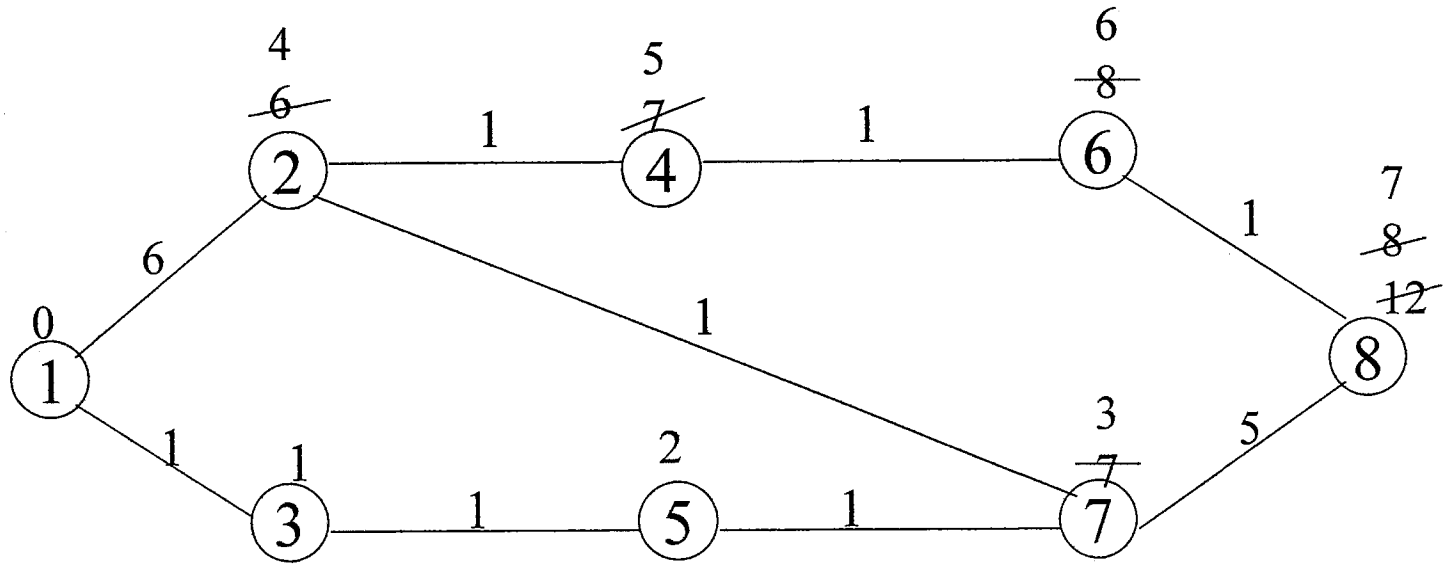


Scan node 4.



Pape's Algorithm

Scan node 6.



Scan node 8. No change. Finished.

DIJKSTRA'S ALGORITHM

Step 1. Initialization

$S = \{1\}$, $R = N - S$, $\ell(1) = 0$, $\ell(j) = d(1, j)$ for $j \neq 1$, and $p(j) = 1$ for all j.

Step 2. Assign a Permanent Label

Let $\ell(j^*) = \min_{j \in R}[\ell(j)]$. Set $R \leftarrow R - \{j^*\}$ and $S \leftarrow S + \{j^*\}$.

If $R = \varnothing$, terminate.

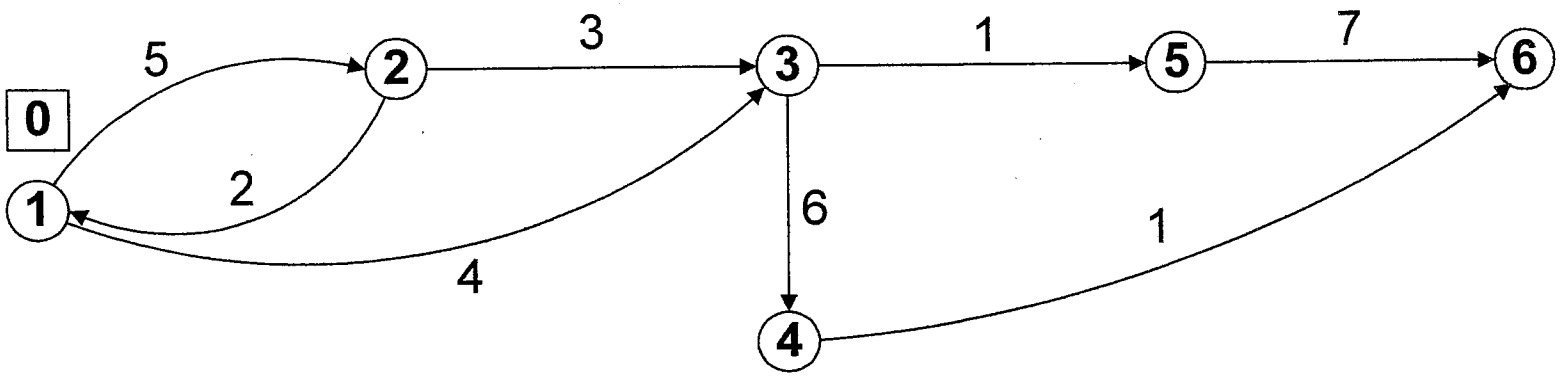
Step 3. Update All Labels

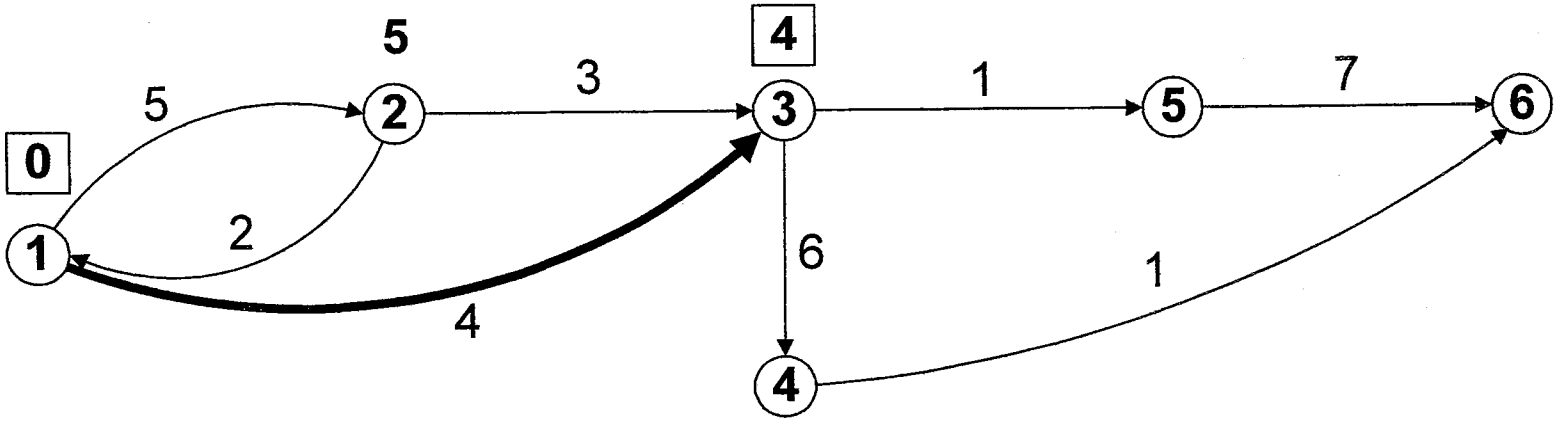
For all $j \in R$, let $\ell(j) = \min[\ell(j), \ell(j^*) + d(j^*, j)]$.

If $\ell(j) = \ell(j^*) + d(j^*, j)$, then $p(j) = j^*$.

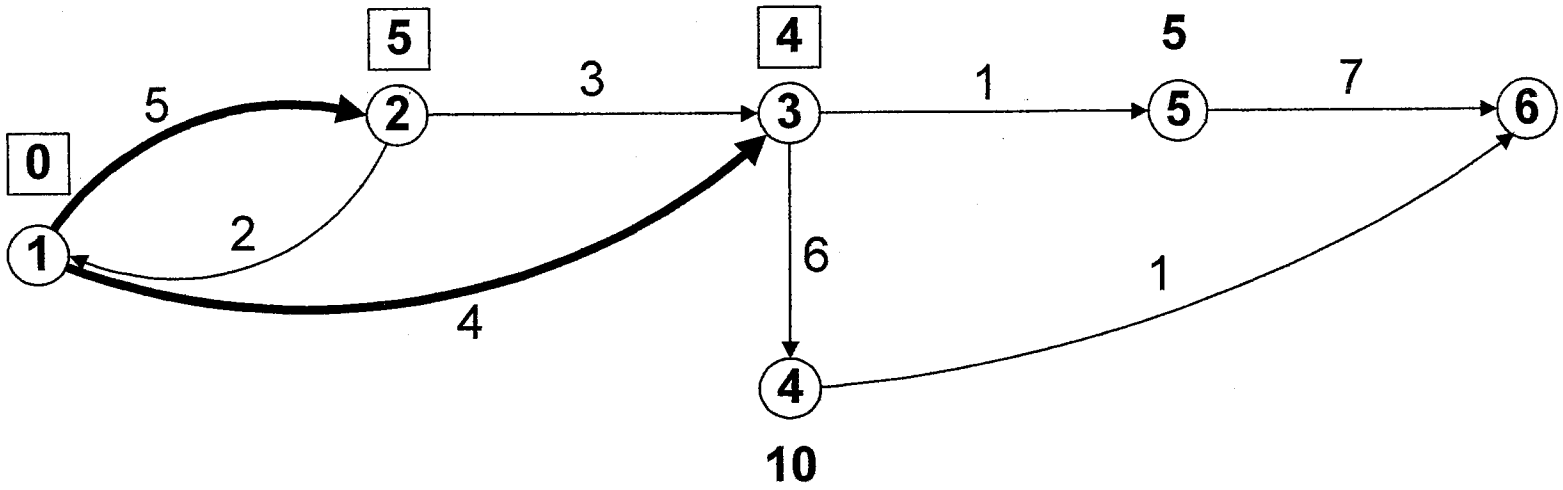
Return to Step 2.

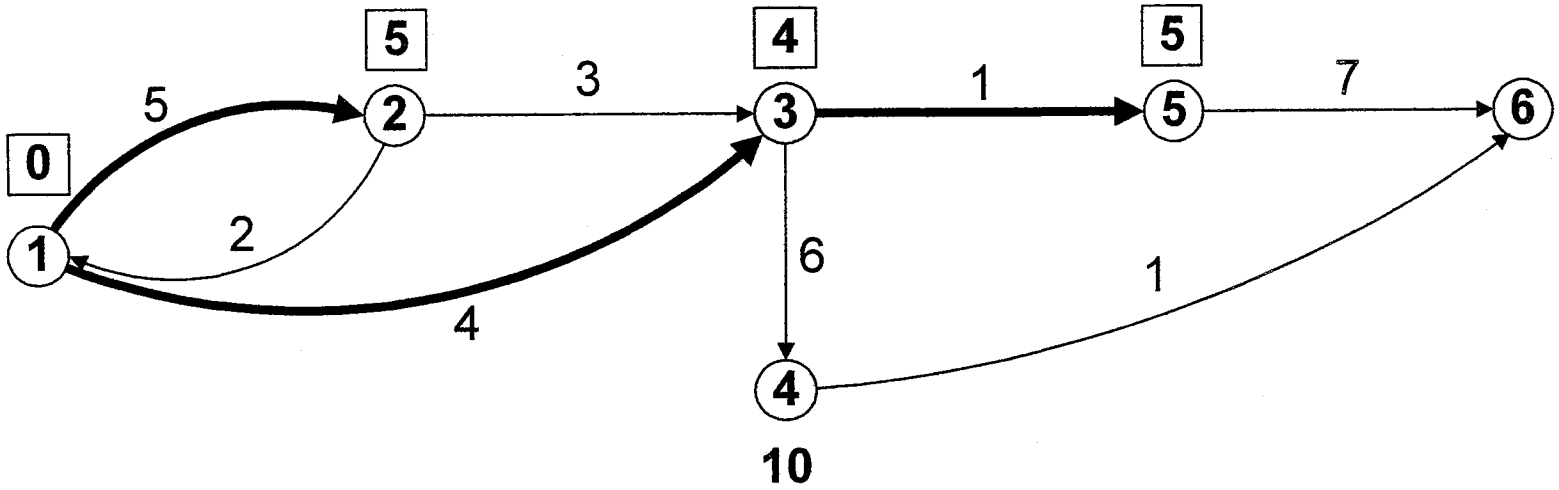
DIJKSTRA EXAMPLE

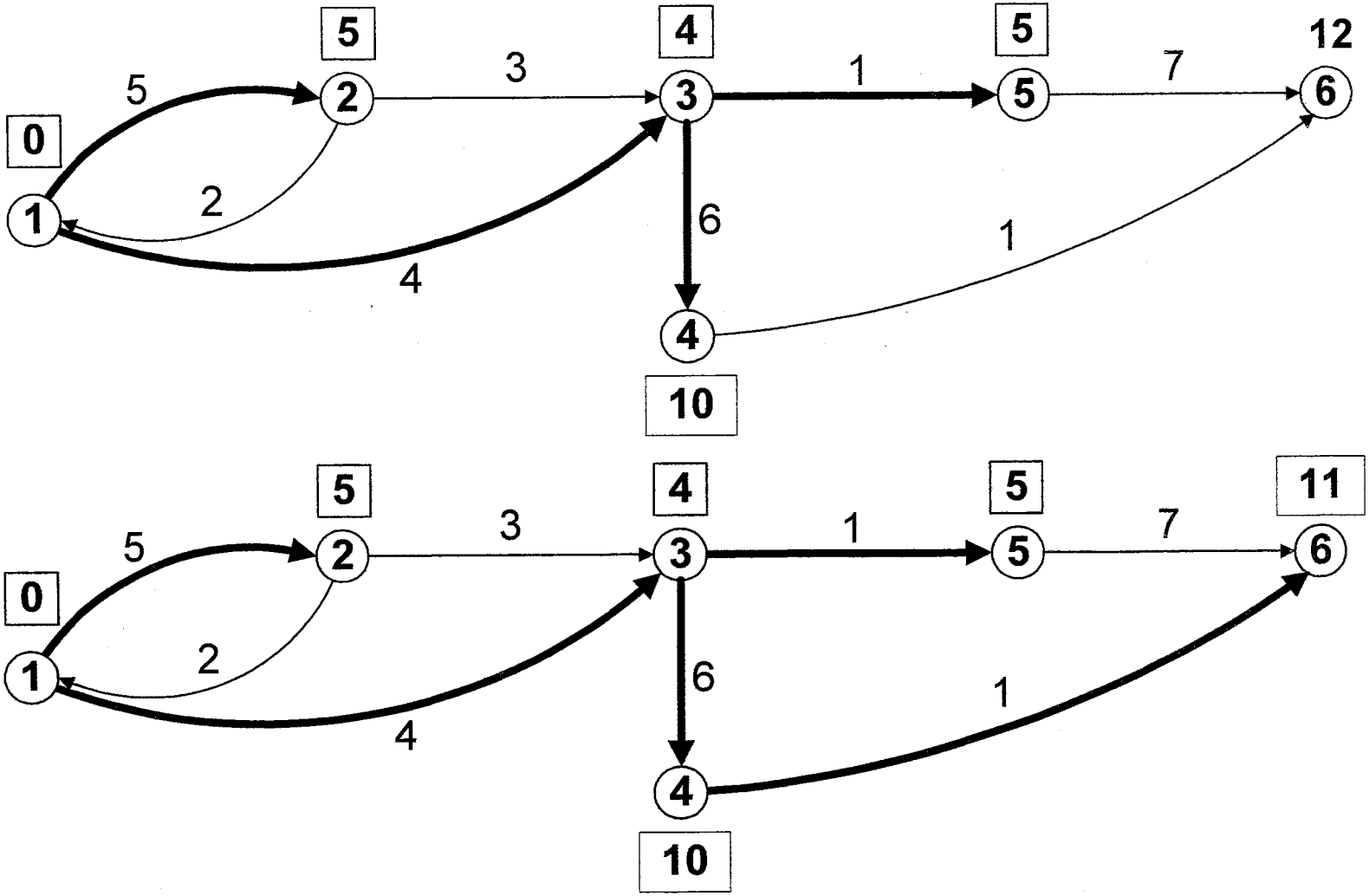




DIJKSTRA EXAMPLE

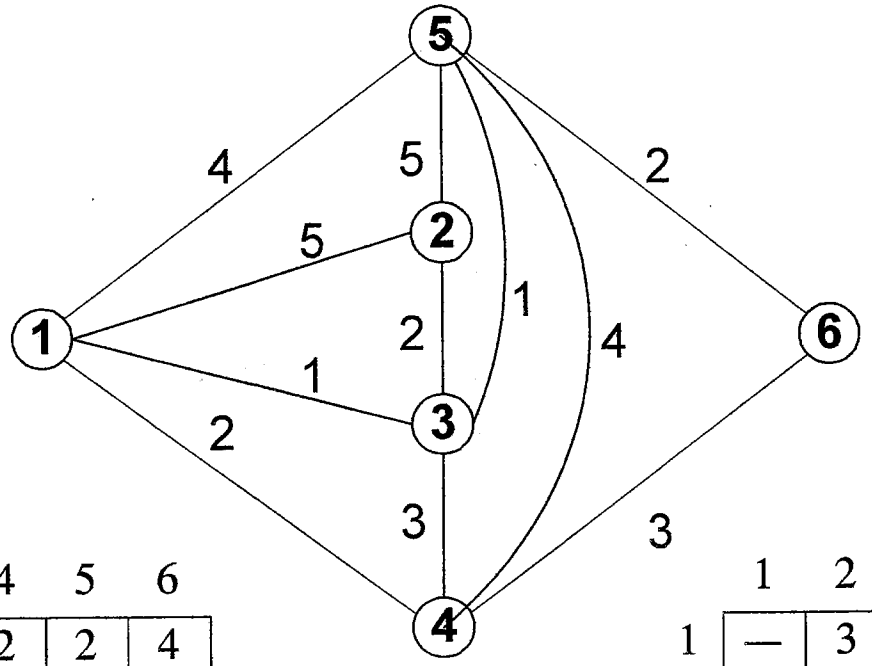






- Dijkstra's algorithm terminates with correct shortest path lengths when applied to problems with nonnegative arc lengths. The algorithm requires $O(n^2)$ additions and comparisons.

ALL-PAIRS SHORTEST PATH PROBLEM



	1	2	3	4	5	6
1	—	3	1	2	2	4
2	3	—	2	5	3	5
3	1	2	—	3	1	3
4	2	5	3	—	4	3
5	2	3	1	4	—	2
6	4	5	3	3	2	—

Distance Matrix

	1	2	3	4	5	6
1	—	3	3	4	3	3
2	3	—	3	3	3	3
3	1	2	—	4	5	5
4	1	3	3	—	5	6
5	3	3	3	4	—	6
6	5	5	5	4	5	—

Follower Matrix

Floyd's Algorithm

Step 0. Definitions

$f_i(j, k)$ = the length of the shortest path from node j to node k using only nodes 1, …, i as intermediate nodes.

$P_i(j, k)$ = the node immediately following node j on the shortest path from node j to node k using only nodes 1, …, i as intermediate nodes.

Step 1. Initialization

$$\left.\begin{aligned} f_0(j, k) &= d(j, k) \\ P_0(j, k) &= k \end{aligned}\right\} \text{for all j and k.}$$

Set i = 1.

Floyd's Algorithm -- continued

Step 2. Check for Negative Cycles

Compute $f_{i-1}(i, j) + f_{i-1}(j, i)$ for all j. If, for some j, this sum is negative, then stop since a negative cycle has been identified.

Otherwise, go to step 3.

Step 3. Update Labels

Let $f_i(j, k) = \min [f_{i-1}(j, k), f_{i-1}(j, i) + f_{i-1}(i, k)]$ for all j and k.

For each j and k,

$$P_i(j, k) = \begin{cases} P_{i-1}(j, k) & \text{if the first term is the minimum.} \\ P_{i-1}(j, i) & \text{otherwise.} \end{cases}$$

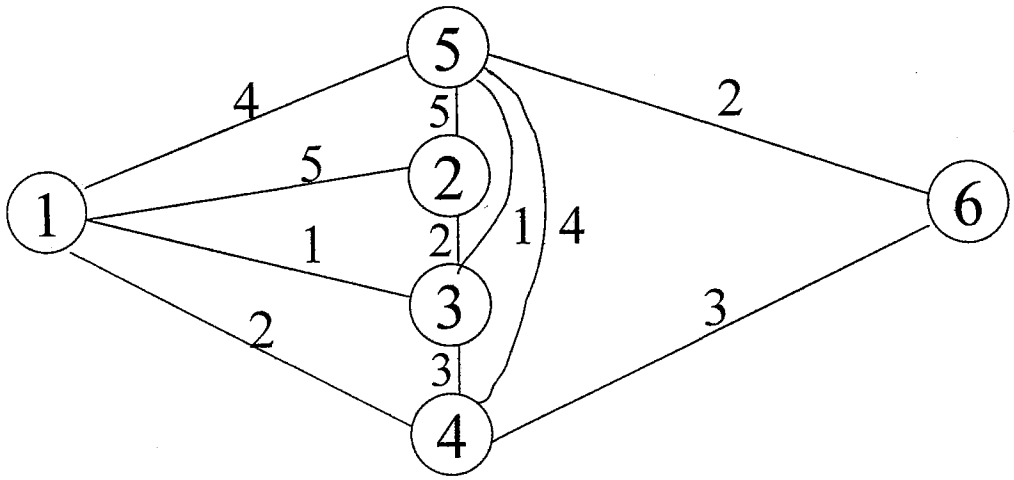
In case of ties, $P_i(j, k) = P_{i-1}(j, k)$.

Floyd's Algorithm -- continued

Step 4. Stopping Criterion

If i = n, then stop. Otherwise, increment i by 1 and return to step 2.

Sample Problem



Floyd's Algorithm -- continued

i = 0

F_0 =

	1	2	3	4	5	6
1	-	5	1	2	4	∞
2	5	-	2	∞	5	∞
3	1	2	-	3	1	∞
4	2	∞	3	-	4	3
5	4	5	1	4	-	2
6	∞	∞	∞	3	2	-

P_0 =

	1	2	3	4	5	6
1	-	2	3	4	5	6
2	1	-	3	4	5	6
3	1	2	-	4	5	6
4	1	2	3	-	5	6
5	1	2	3	4	-	6
6	1	2	3	4	5	-

Floyd's Algorithm -- continued

i = 1

F_1 =

	1	2	3	4	5	6
1	-	5	1	2	4	∞
2	5	-	2	(7)	5	∞
3	1	2	-	3	1	∞
4	2	(7)	3	-	4	3
5	4	5	1	4	-	2
6	∞	∞	∞	3	2	-

P_1 =

	1	2	3	4	5	6
1	-	2	3	4	5	6
2	1	-	3	(1)	5	6
3	1	2	-	4	5	6
4	1	(1)	3	-	5	6
5	1	2	3	4	-	6
6	1	2	3	4	5	-

Floyd's Algorithm -- continued

$i = 2 \quad F_2 = F_1$

$P_2 = P_1$

$i = 3$

$F_3 =$

	1	2	3	4	5	6
1	-	(3)	1	2	(2)	∞
2	(3)	-	2	(5)	(3)	∞
3	1	2	-	3	1	∞
4	2	(5)	3	-	4	3
5	(2)	(3)	1	4	-	2
6	∞	∞	∞	3	2	-

$P_3 =$

	1	2	3	4	5	6
1	-	(3)	3	4	(3)	6
2	(3)	-	3	(3)	(3)	6
3	1	2	-	4	5	6
4	1	(3)	3	-	5	6
5	(3)	(3)	3	4	-	6
6	1	2	3	4	5	-

Floyd's Algorithm -- continued

i = 4

F_4 =

	1	2	3	4	5	6
1	-	3	1	2	2	(5)
2	3	-	2	5	3	(8)
3	1	2	-	3	1	(6)
4	2	5	3	-	4	3
5	2	3	1	4	-	2
6	(5)	(8)	(6)	3	2	-

P_4 =

	1	2	3	4	5	6
1	-	3	3	4	3	(4)
2	3	-	3	3	3	(3)
3	1	2	-	4	5	(4)
4	1	3	3	-	5	6
5	3	3	3	4	-	6
6	(4)	(4)	(4)	4	5	-

Floyd's Algorithm -- continued

i = 5

F_5 =

	1	2	3	4	5	6
1	-	3	1	2	2	(4)
2	3	-	2	5	3	(5)
3	1	2	-	3	1	(3)
4	2	5	3	-	4	3
5	2	3	1	4	-	2
6	(4)	(5)	(3)	3	2	-

P_5 =

	1	2	3	4	5	6
1	-	3	3	4	3	(3)
2	3	-	3	3	3	(3)
3	1	2	-	4	5	(5)
4	1	3	3	-	5	6
5	3	3	3	4	-	6
6	(5)	(5)	(5)	4	5	-

Floyd's Algorithm -- continued

i = 6

F_6 =

	1	2	3	4	5	6
1	-	3	1	2	2	4
2	3	-	2	5	3	5
3	1	2	-	3	1	3
4	2	5	3	-	4	3
5	2	3	1	4	-	2
6	4	5	3	3	2	-

P_0 =

	1	2	3	4	5	6
1	-	3	3	4	3	3
2	3	-	3	3	3	3
3	1	2	-	4	5	5
4	1	3	3	-	5	6
5	3	3	3	4	-	6
6	5	5	5	4	5	-

LONGEST (CRITICAL) PATHS IN A DIRECTED ACYCLIC GRAPH

- Assume that nodes are numbered such that all directed arcs i-j have $i < j$. This can be ensured using on the order of n^2 computations in the worst case.

- A Longest Path Algorithm for Directed Acyclic Graphs

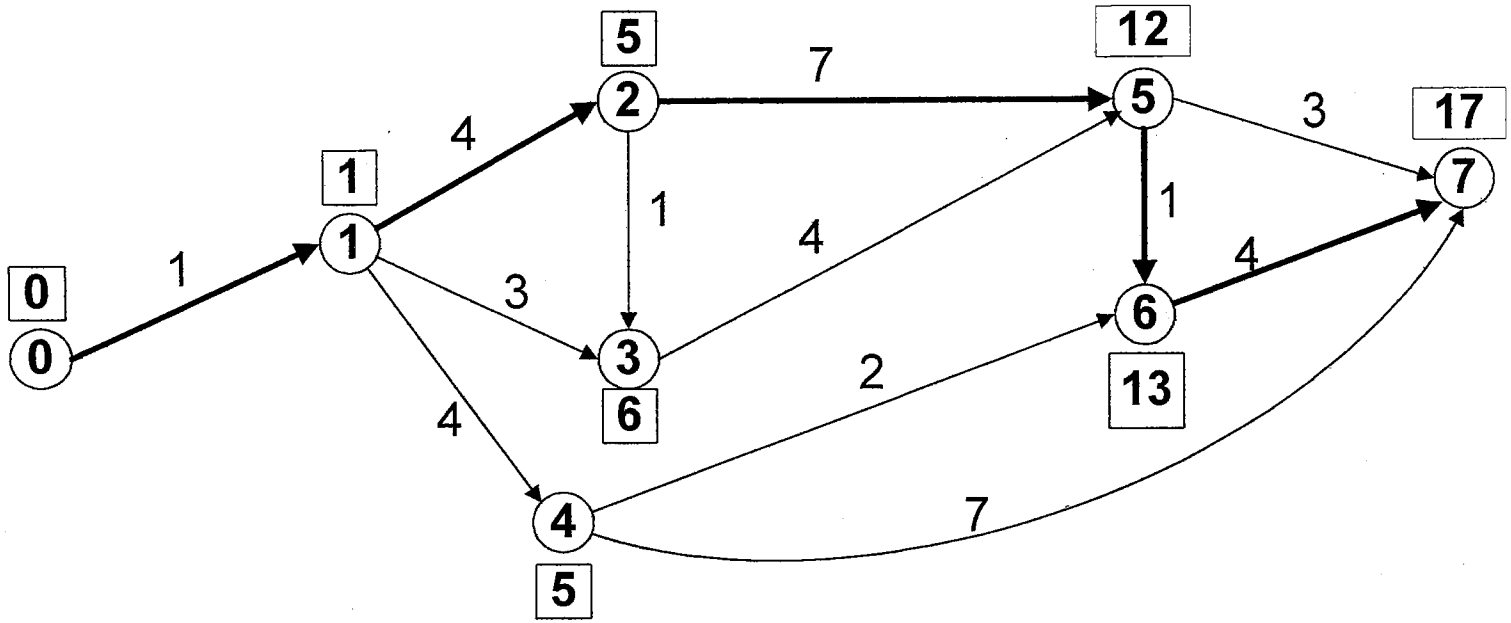
Step 1. Initialization

$$\ell(0) \leftarrow 0.$$

Step 2. Computation

For $i = 1, 2, \ldots, n$, let $\ell(i) \leftarrow \max_{j:\, j-i \in A} \{\ell(j) + d(j, i)\}$.

LONGEST PATH EXAMPLE



$\ell(0) = 0$

$\ell(1) = 1$

$\ell(2) = 5$

$\ell(3) = \max\{4, 6\} = 6$

$\ell(4) = 5$

$\ell(5) = \max\{12, 10\} = 12$

$\ell(6) = \max\{13, 7\} = 13$

$\ell(7) = \max\{12, 17, 15\} = 17$