MTM4501-Operations Research

Gökhan Göksu, PhD

Week 3

Course Content

- Definition of OR and Its History
- Decision Theory and Models
- Network Analysis
  - Shortest Path Algorithms
- Inventory Management Models
- Queue Models

Shortest Path Algorithms

- Shortest path algorithms determine the shortest route between the source and destination in a transportation network.
- Algorithms of these kind are used to determe directions on online applications such as Mapquest or Google Maps.
- Source and destination nodes
  - **single source shortest path algorithms** are used when we need to find the shortest path(s) from a source node to all other nodes.
  - **all pairs shortest path algorithms** are used when we need to find the shortest paths between each pair of nodes.

Shortest Path Algorithms: Dijkstra's Algorithm

In this section, two algorithms will be introduced for both cyclic (in other words, containing loops) and acyclic networks.

1. Dijkstra's algorithm
2. Floyd's algorithm

Dijkstra's algorithm is designed to determine the shortest paths between the source node and every other node in the network. The Floyd's algorithm, on the other hand, is more general as it allows the determination of the shortest path between any two nodes in the network.

Dijkstra's Algorithm: The calculations of this algorithm proceed from node i to the "immediate" succeeding node j using a special labeling procedure. Let u_i be the shortest distance from node 1 to the node i and let d_{ij} (≥ 0) be defined as the length of the link (i, j). Then the label for node j will be as follows:

$$[u_j, i] = [u_i + d_{ij}, i], \quad d_{ij} \geq 0$$

If it is the source node, it is marked as $[0, -]$.

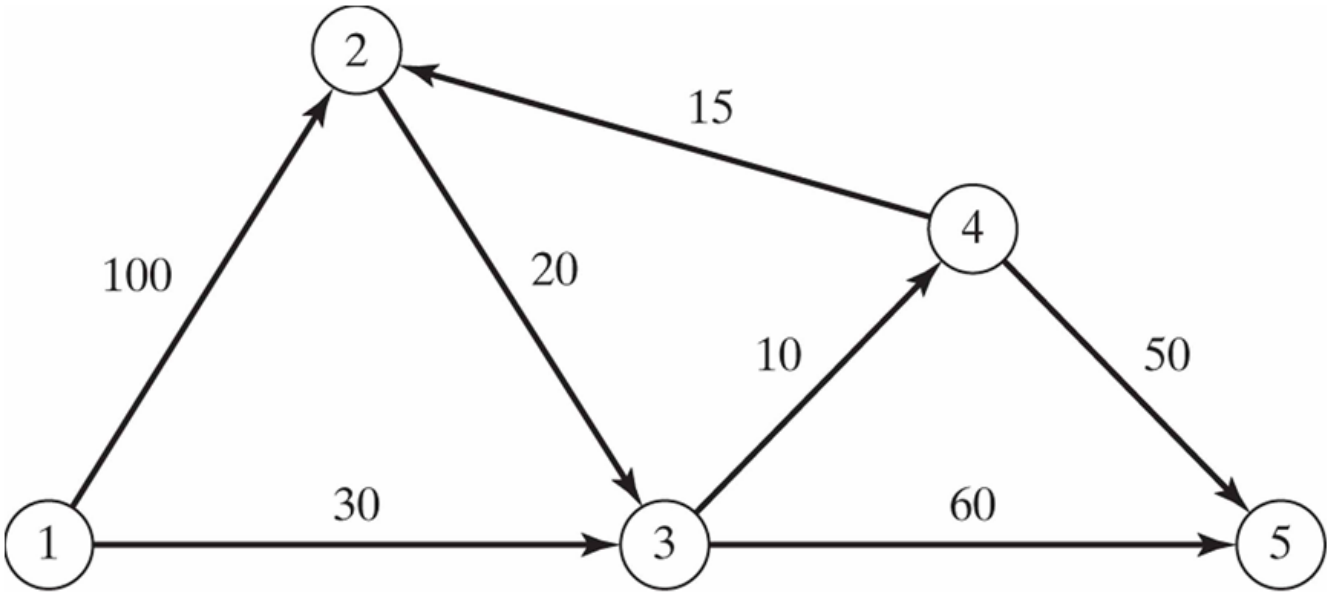
In this algorithm, node labels are of two types: *temporary* and *permanent*. The temporary label can be replaced with another label if a shorter path to the same node is found. At the point when it becomes clear that no better way can be found, the status of the temporary label becomes permanent.

Dijkstra's Algorithm

The steps of the algorithm are summarized below:

- **Step 0:** Label the source node (node 1) with permanent label $[0, -]$. Set $i = 1$.
- **Step** i**:**
  - **(a)** *Provided that j is not permanently tagged*, compute *temporary* labels $[u_i + d_{ij}, i]$ for each node j reachable from node i. If node j is already labeled with $[u_j, k]$ within another node k and $u_i + d_{ij} < u_j$, then replace $[u_j, k]$ with $[u_i + d_{ij}, i]$.
  - **(b)** Stop, if all nodes have persistent labels. Otherwise, among all *temporary* tags, choose the one with the shortest distance $(= u_r)$ of $[u_r, s]$ (in case of tie, choose any at random). Set $i = r$ and repeat step i.

Dijkstra's Algorithm



Example

The network in the figure shows the roads between the 1^{st} *city (*1^{st} *node) and the other four cities (*2^{nd} *node to* 5^{th} *node) and their distances in km. Determine the shortest route from the* 1^{st} *city to the remaining four cities.*

Dijkstra's Algorithm

- **Iteration 0:** Assign *permanent label* $[0, -]$ to node 1.
- **Iteration 1:** From node 1, which was last permanently tagged, nodes 2 and 3 can be reached. Thus, the list of labeled nodes (temporary and permanent) becomes as the following.

  When we look at the labels of the 2^{nd} and 3^{rd} nodes, $[100, 1]$ and $[30, 1]$, which have two temporary labels, it is seen that the 3^{rd} node gives the shorter distance ($u_3 = 30$). Therefore, the status of the 3^{rd} node is changed permanently.
- **Iteration 2:** From the 3^{rd} node, the 4^{th} and 5^{th} nodes can be reached and the list of labeled nodes is generated as follows:

  The status of the temporary tag $[40, 3]$ at the 3^{rd} node is changed permanently ($u_4 = 40$).

Dijkstra's Algorithm

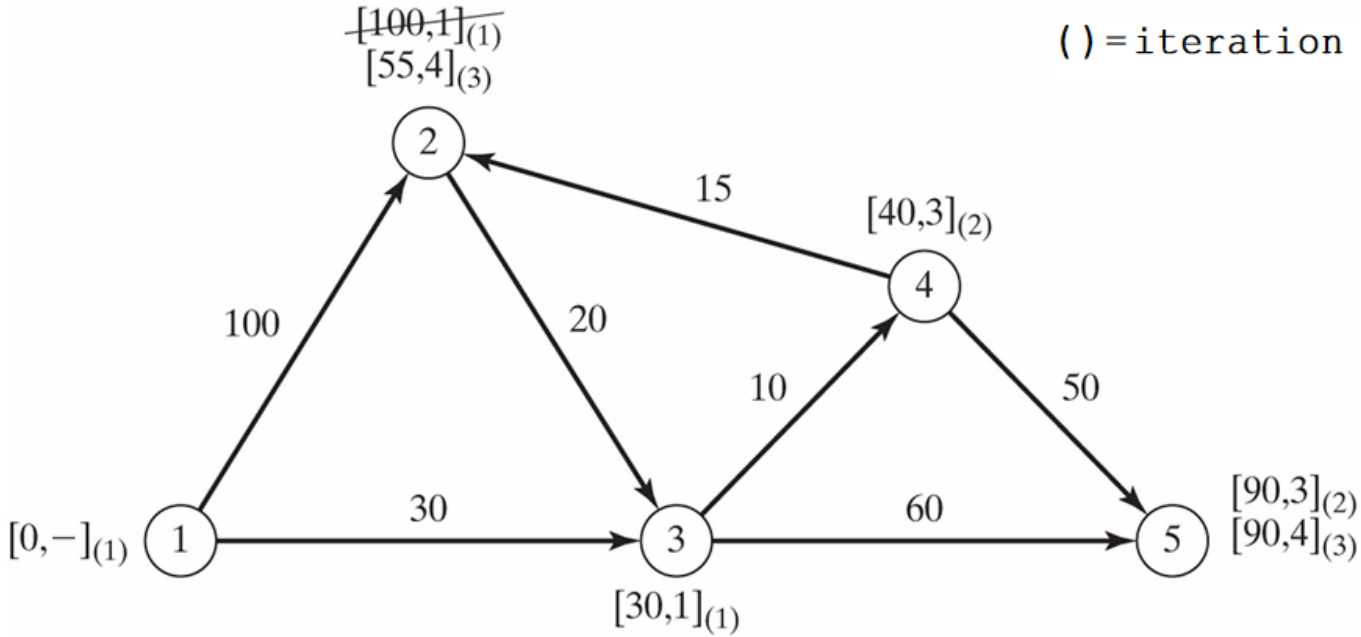
- **Iteration 3:** From the 4^{th} node, the 2^{nd} and 5^{th} nodes can be reached. Thus, the list of labeled nodes is updated as follows.

  In the 2^{nd} iteration, the 2^{nd} node's temporary label $[100, 1]$ is changed to $[55, 4]$ in the 3^{rd} iteration, indicating that a shorter path exists for the 4^{th} node. Moreover, in the 3^{rd} iteration, the 5^{th} node has two alternatives with the same distance ($u_5 = 90$).

- **Iteration 4:** Only the 3^{rd} node can be reached from the 2^{nd} node. However, the 3^{rd} node is permanently tagged and cannot be relabeled. The new list of labels remains the same except that the label in the 2^{nd} node persists in the 3^{rd} iteration. This leaves the 5^{th} node as the only temporary label. Since the 5^{th} node does not go to other nodes, its status is converted to permanent and the process is completed.

Dijkstra's Algorithm

The algorithm's calculations can be performed more easily over the network.



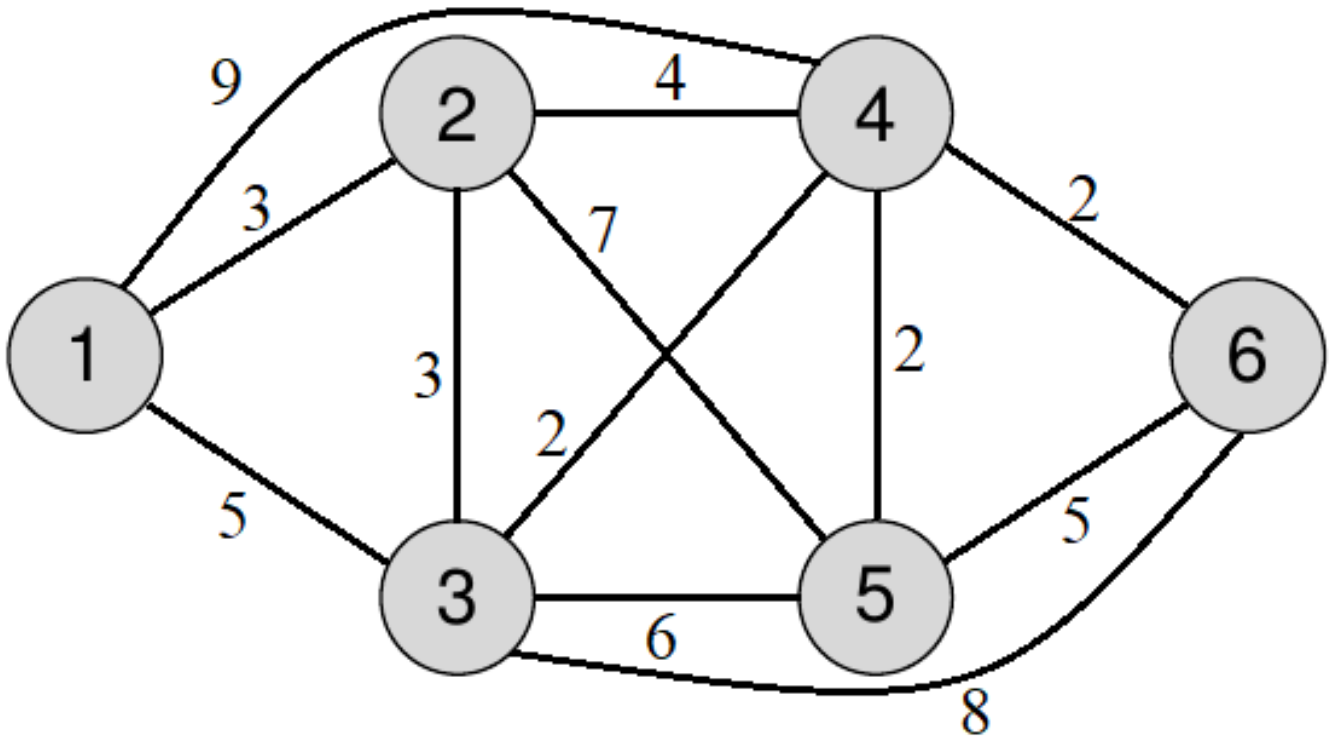
To determine the shortest path between the 1^{st} node and another node in the network, one starts from the desired destination node and moves backwards through the nodes using the information provided by the permanent labels. For example, the following order determines the shortest path from the 1^{st} node to the 2^{nd} node.

Thus, the desired path is and the total length is km.

Dijkstra's Algorithm

Example

The network in the figure shows the roads between the 1^{st} *city (*1^{st} *node) and the other five cities (*2^{nd} *node to* 6^{th} *node) and their distances in km. Determine the shortest route from the* 1^{st} *city to the remaining five cities.*



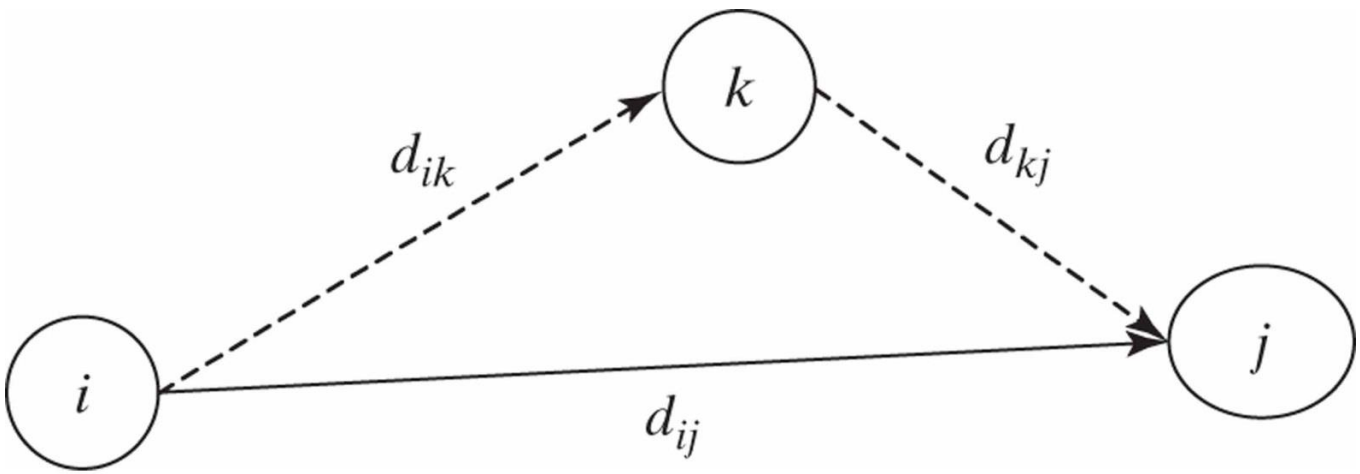
Floyd's Algorithm

Floyd's Algorithm: Floyd's algorithm is more general than Dijkstra's algorithm, because it determines the shortest path between any two nodes in the network. In this algorithm, the network with n nodes is expressed as a square matrix with n rows and n columns. The (i, j) element of the matrix is the distance d_{ij} from node i to the node j; d_{ij} is finite if i is directly connected to j, otherwise it is infinite.

For given nodes i, j and k, if it is shorter to reach k by starting from i and passing through j, i.e.

$$d_{ij} + d_{jk} < d_{ik}$$

then, replacing the direct path from i to k with the indirect path $i \rightarrow j \rightarrow k$ gives the optimal solution.



This **triple operation** is systematically applied to the network using the steps on the next slide.

Floyd's Algorithm

- **Step 0:** The starting distance matrix D_0 and the node sequence matrix S_0 are defined as below. Diagonal elements are marked with $(-)$ to indicate that they are blocked. It is determined as $k = 1$.

D_0	1	2	...	j	...	n
1	$-$	d_{12}	...	d_{1j}	...	d_{1n}
2	d_{21}	$-$	...	d_{2j}	...	d_{2n}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	d_{i1}	d_{i2}	...	d_{ij}	...	d_{in}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
n	d_{n1}	d_{n2}	...	d_{nj}	...	$-$

S_0	1	2	...	j	...	n
1	$-$	2	...	j	...	n
2	1	$-$	...	j	...	n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
i	1	2	...	j	...	n
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\ddots$	$\vdots$
n	1	2	...	j	...	$-$

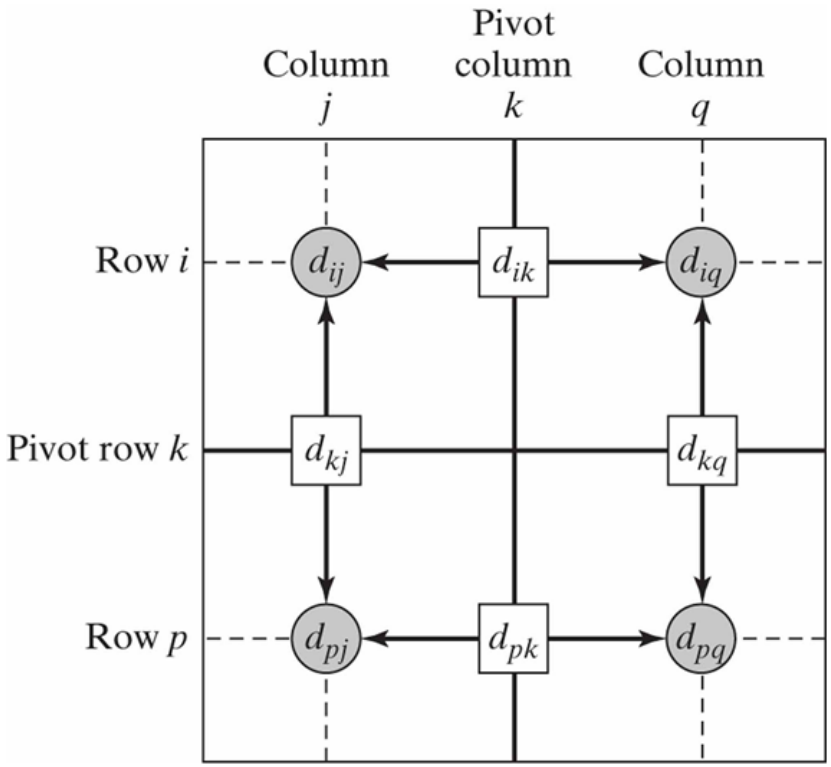
- **General Step k:** Row k and column k are defined as the key (pivot) row and the key (pivot) column. The triple operation is applied to each d_{ij} element in D_{k-1} for all i and j. If the condition

$$d_{ik} + d_{kj} < d_{ij}, \quad (i \neq k, j \neq k \text{ and } i \neq j)$$

is satisfied, make the following changes:

  - **(a)** In D_{k-1}, d_{ij} is replaced by $d_{ik} + d_{kj}$ to form D_k.
  - **(b)** In S_{k-1}, s_{ij} is replaced by k to form S_k. Set $k = k + 1$. If $k = n + 1$, stop; else repeat step k.

Floyd's Algorithm



D_{k-1}, as shown in the figure, step k in the algorithm can be explained more clearly. Here, row k and column k define the current pivot row and column. Row i represents any of the rows $1, 2, ..., k-1$, and row p represents any of the row $k+1, k+2, ..., n$. Similarly, column j represents any of the columns $1, 2, ..., k-1$. In the *triple operation*, if the sum of the elements represented by squares in the key row and key column is less than the corresponding intersection element represented by a circle; the sum of the key distances is written instead of the intersection distance.

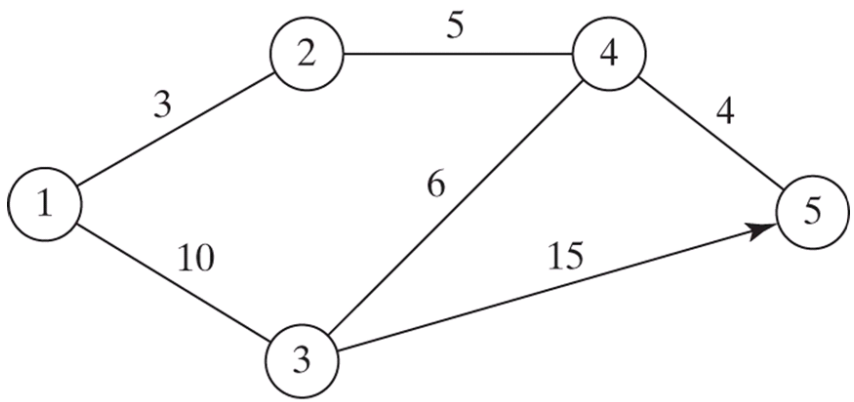
Floyd's Algorithm

After n steps, we determine the shortest path between nodes i and j from the matrices D_n and S_n using the following rules:

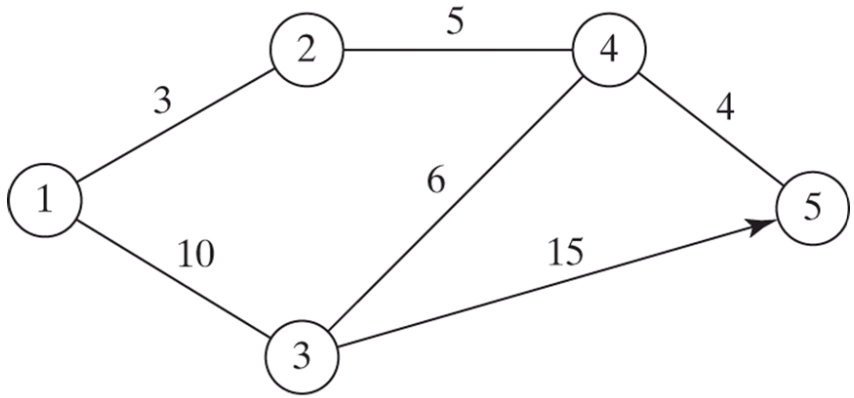
1. In matrix D_n; d_{ij} gives the shortest path between nodes i and j.
2. In the S_n matrix; the intermediate node $k = s_{ij}$ that gives the path $i \to k \to j$ is determined. If $s_{ik} = k$ and $s_{kj} = j$, stop; all intermediate nodes of the path have been found. Otherwise, the procedure is repeated between nodes i and k and between nodes k and j.

Example

Find the shortest paths between both nodes for the network in the figure. Distances are given on the arcs in km. Link $(3, 5)$ *is directional with no traffic from node* 5 *to node* 3. *All the other arcs allow two-way traffic.*



Floyd's Algorithm

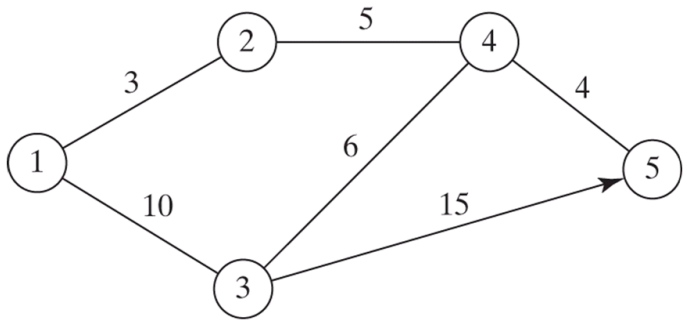


- **Iteration 0:** The matrices D_0 and S_0 give the initial representation of the network. Since there is no traffic from node 5 to node 3, D_0 is symmetric except that $d_{53} = \infty$.

D_0	1	2	3	4	5
1					
2		–	ⓞ	5	∞
3		ⓞ	–	6	15
4		5	6	–	4
5		∞	∞	4	–

S_0	1	2	3	4	5
1	–	2	3	4	5
2	1	–	③	4	5
3	1	②	–	4	5
4	1	2	3	–	5
5	1	2	3	4	–

Floyd's Algorithm



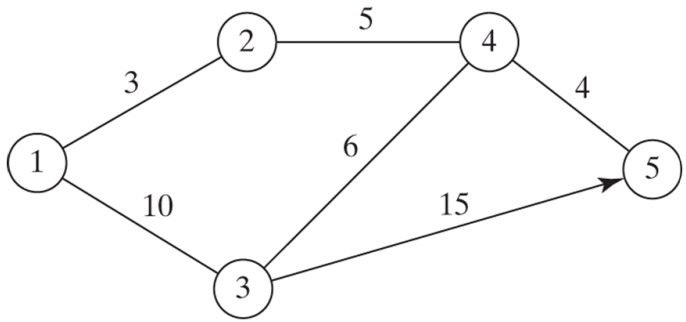
- **Iteration 1:** To obtain D_1 and S_1 from D_0 and S_0,
  1. Replacing d_{23} with $d_{21} + d_{13} = 3 + 10 = 13$ and setting $s_{23} = 1$,
  2. Replacing d_{32} with $d_{31} + d_{12} = 10 + 3 = 13$ and setting $s_{32} = 1$,

  is required. These changes are shown in **bold** in the D_1 and S_1 matrices.

D_1	1	2	3	4	5
1	–		10	⊗	∞
2					
3	10		–	6	15
4	⊗		6	–	4
5	∞		∞	4	–

S_1	1	2	3	4	5
1	–	2	3	④	5
2	1	–	**1**	4	5
3	1	**1**	–	4	5
4	①	2	3	–	5
5	1	2	3	4	–

Floyd's Algorithm

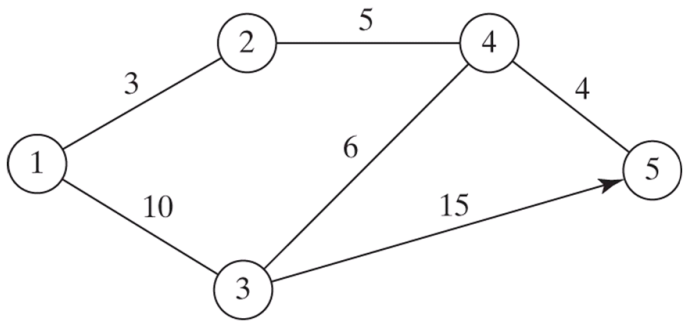


- **Iteration 2:** $k = 2$ is determined as shown by red rows and columns in D_1. *Triple operation* is applied to the elements circled in the cell in D_1 and S_1. The changes are shown in **bold** in D_2 and S_2.

D_2	1	2	3	4	5
1	–	3		**8**	ⓞ
2	3	–		5	ⓞ
3					
4	**8**	5		–	4
5	∞	∞		4	–

S_2	1	2	3	4	5
1	–	2	3	**2**	⑤
2	1	–	1	4	⑤
3	1	1	–	4	5
4	**2**	2	3	–	5
5	1	2	3	4	–

Floyd's Algorithm

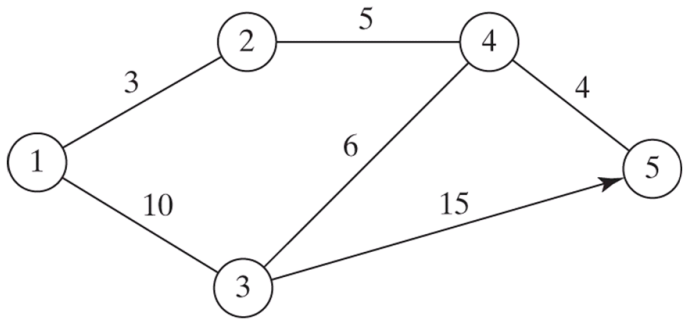


- **Iteration 3:** $k = 3$ is determined as shown by red rows and columns in D_2. The new matrices are given by D_3 and S_3.

D_3	1	2	3	4	5
1	–	3	10		(25)
2	3	–	(13)		(28)
3	10	(13)	–		15
4					
5	(∞)	(∞)	(∞)		–

S_3	1	2	3	4	5
1	–	2	3	2	(3)
2	1	–	(1)	4	(3)
3	1	(1)	–	4	(5)
4	2	2	3	–	5
5	(1)	(2)	(3)	4	–

Floyd's Algorithm



- **Iteration 4:** $k = 4$ is determined as shown by red rows and columns in D_3. The new matrices are given by D_4 and S_4.

D_4	1	2	3	4	5
1	–	3	10	8	
2	3	–	**11**	5	
3	10	**11**	–	6	
4	8	5	6	–	
5					

S_4	1	2	3	4	5
1	–	2	3	2	**4**
2	1	–	**4**	4	**4**
3	1	**4**	–	4	**4**
4	2	2	3	–	5
5	**4**	**4**	**4**	4	–

- **Iteration 5:** $k = 5$ is determined as shown by red rows and columns in D_4. There is no improvement in this iteration.

Floyd's Algorithm

D_4	1	2	3	4	5
1	–	3	10	8	
2	3	–	**11**	5	
3	10	**11**	–	6	
4	8	5	6	–	
5					

S_4	1	2	3	4	5
1	–	2	3	2	**4**
2	1	–	**4**	4	**4**
3	1	**4**	–	4	**4**
4	2	2	3	–	5
5	**4**	**4**	**4**	4	–

The final matrices D_4 and S_4 contain all the information to determine the shortest path between any two nodes in the network. For example, the shortest distance from node 1 to node 5 is $d_{15} = 12$.

To determine the path for this, the element (i, j) simply indicates a direct connection since $s_{ij} = j$. Otherwise i and j must be connected with at least another node. Since $s_{15} = 4$ and $s_{45} = 5$, the path is initially given as $1 \to 4 \to 5$. Since $s_{14} \neq 4$, the $(1, 4)$ segment is not a direct connection and intermediate nodes must be determined. Given $s_{14} = 2$ and $s_{24} = 4$, the path $1 \to 4$ is replaced by path $1 \to 2 \to 4$. Since $s_{12} = 2$ and $s_{24} = 4$, there is no other intermediate node.

The combined result gives the optimal path as $1 \to 2 \to 4 \to 5$. The length of this road is 12 km.

Linear Programing Formulation of the Shortest Path Algorithms

Consider the network with capacity $G = (N, A)$. Here N is the set of nodes, A is the set of arcs and the amount of flow is defined as

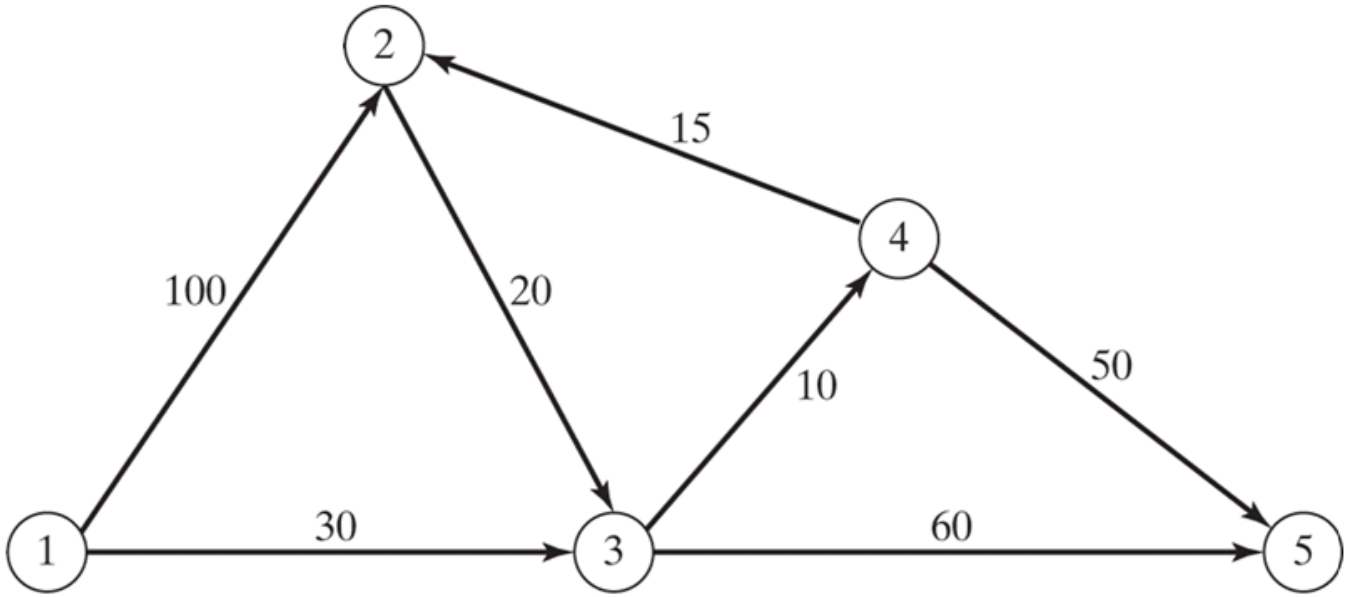
$$\begin{aligned} x_{ij} &= i \text{ flow amount from node } i \text{ to node } j \\ &= \begin{cases} 1, & \text{if arc } (i,j) \text{ is on the shortest path} \\ 0, & \text{otherwise} \end{cases} \\ c_{ij} &= i \text{ length of arc } (i,j). \end{aligned}$$

Accordingly, the linear programming formulation is as follows:

$$\min z = \sum_{(i,j)\in A} \sum c_{ij} x_{ij}$$

$$\text{subject to } \begin{pmatrix} \text{External input} \\ \text{into node } j \end{pmatrix} + \sum_{i \in N} x_{ij} = \begin{pmatrix} \text{External output} \\ \text{from node } j \end{pmatrix} + \sum_{j \in N} x_{jk}.$$

Linear Programing Formulation of the Shortest Path Algorithms

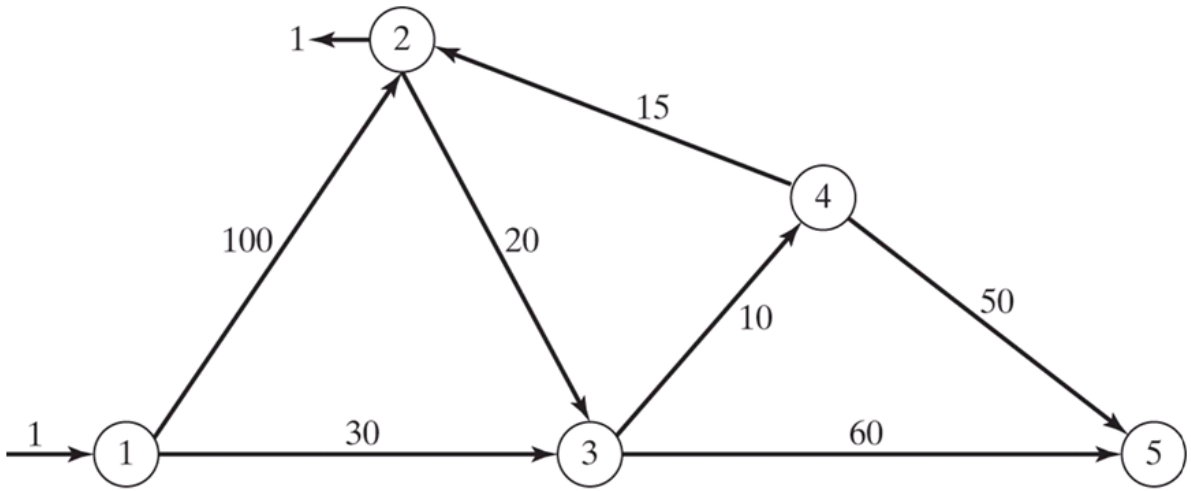


Example

Show the linear programming formulation that determines the shortest path from node 1 to node 2 in the network shown in the figure.

Linear Programing Formulation of the Shortest Path Algorithms

For this formulation, unit input flow to node 1 and unit output flow to node 2 are marked.



According to the flow conservation equations, the following equations are written for each node:

$$1 = x_{12} + x_{13} \quad \text{(Node 1)}$$

$$x_{12} + x_{42} = x_{23} + 1 \quad \text{(Node 2)}$$

$$x_{13} + x_{23} = x_{34} + x_{35} \quad \text{(Node 3)}$$

$$x_{34} = x_{42} + x_{45} \quad \text{(Node 4)}$$

$$x_{35} + x_{45} = 0 \quad \text{(Node 5)}$$

Linear Programing Formulation of the Shortest Path Algorithms

Accordingly, the linear programming problem is expressed as follows:

	x_{12}	x_{13}	x_{23}	x_{34}	x_{35}	x_{42}	x_{45}		
min $z =$	100	30	20	10	60	15	50		
Node 1	1	1						$=$	1
Node 2	-1		1			-1		$=$	-1
Node 3		-1	-1	1	1			$=$	0
Node 4				-1		1	1	$=$	0
Node 5					-1		-1	$=$	0

In this table, it is important to note that there are +1 and -1 in each column. This is a typical example of a network linear programming problem.

Optimal solution through a package program can be obtained as

$$z = 55,\ x_{13} = 1,\ x_{34} = 1,\ x_{42} = 1.$$

This means the shortest path from node 1 to node 2 is

$$1 \to 3 \to 4 \to 2$$

and this indicates that the distance is $z = 55$ km.