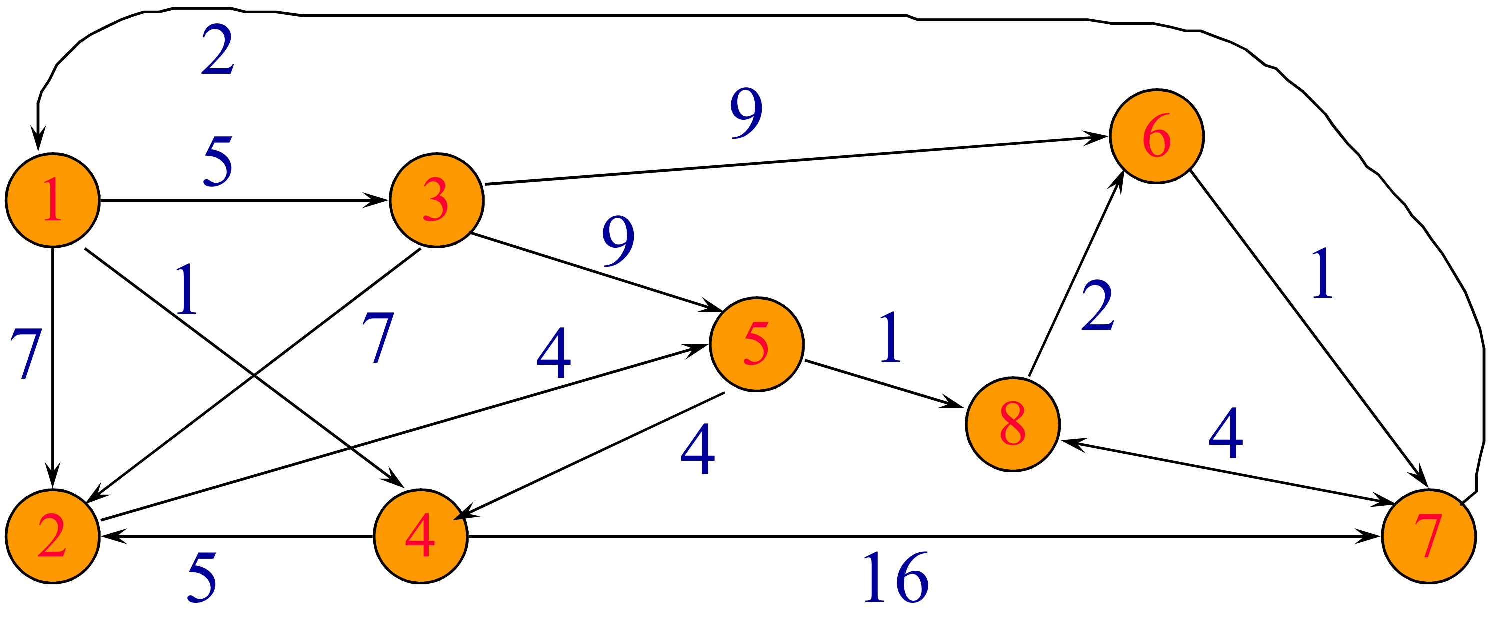
Shortest Paths
Part III

Prof. Ki-Hoon Lee
Dept. of Computer Engineering
Kwangwoon University

All-Pairs Shortest Paths

All-Pairs Shortest Paths

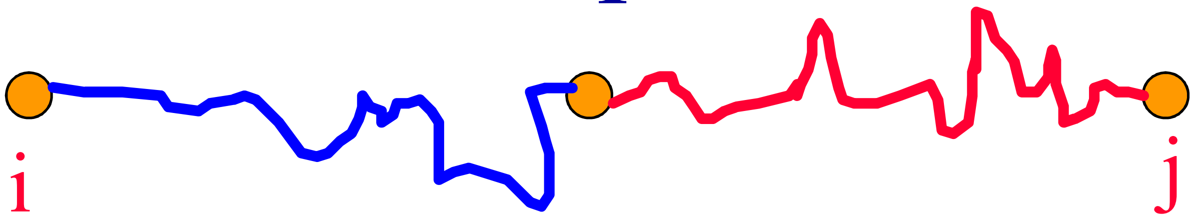
- Given an n-vertex directed weighted graph, find a shortest path from vertex i to vertex j for each of the n^2 vertex pairs (i, j).



Floyd's Algorithm

- Dynamic programming
- Time complexity is $O(n^3)$ time.
- Works so long as there is no cycle whose length is < 0.
- When there is a cycle whose length is < 0, some shortest paths aren't finite.
- Assume that $0 <$ vertex numbers $\leq n$

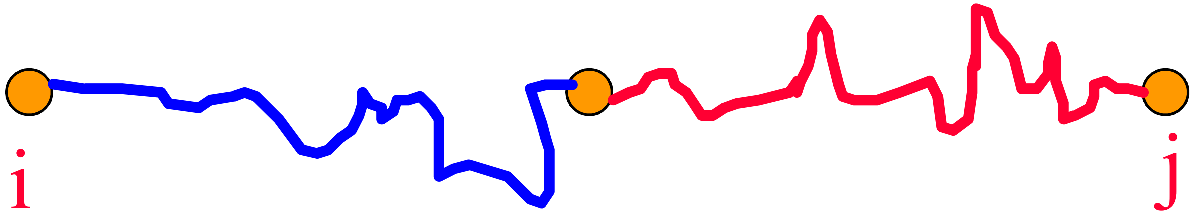
A Triple



- (i,j,k) denotes the problem of finding the shortest path from vertex i to vertex j that has no intermediate vertex larger than k.

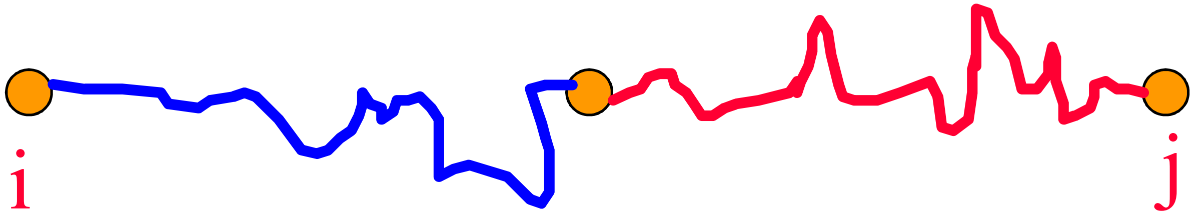
- (i,j,n) denotes the problem of finding the shortest path from vertex i to vertex j (with no restrictions on intermediate vertices).

Decision Sequence



- First decide the highest intermediate vertex (i.e., largest vertex number) on the shortest path from i to j.
- If the shortest path is i, 2, 6, 3, 8, 5, 7, j the first decision is that vertex 8 is an intermediate vertex on the shortest path and no intermediate vertex is larger than 8.
- Then decide the highest intermediate vertex on the path from i to 8, and so on.

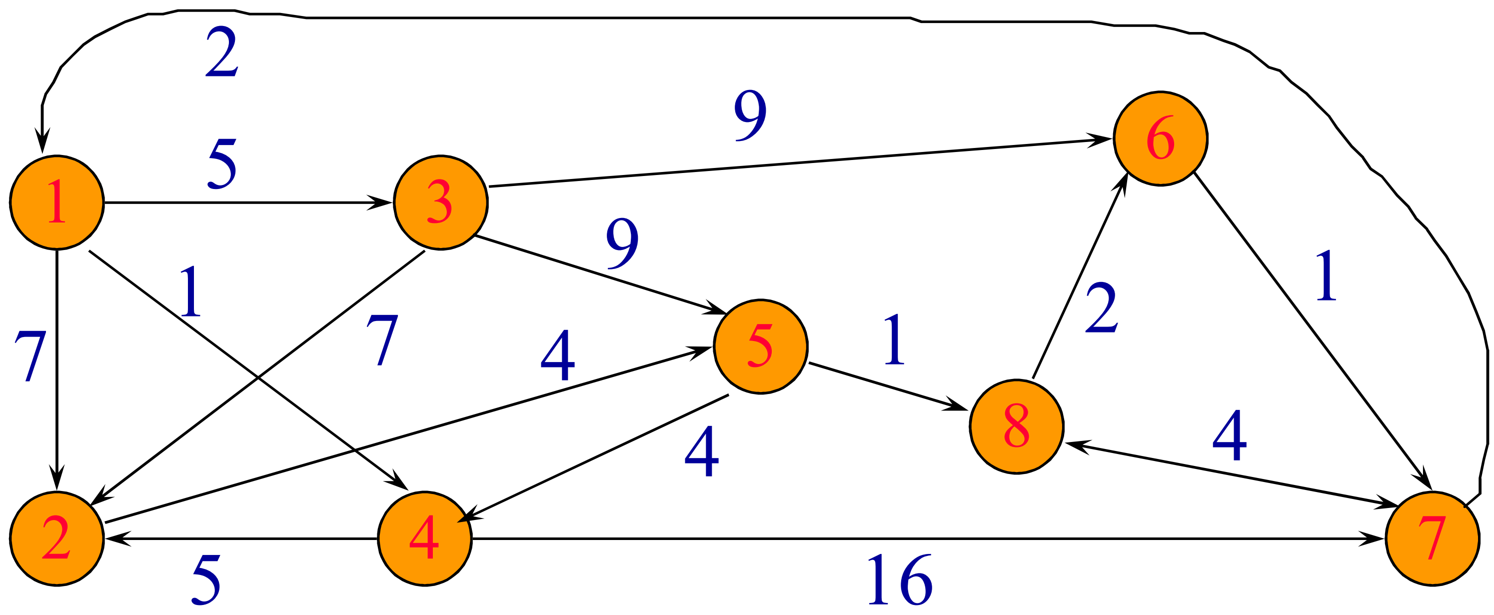
Cost Function



- Let $A^k[i][j]$ be the length of a shortest path from vertex i to vertex j that has no intermediate vertex larger than k.

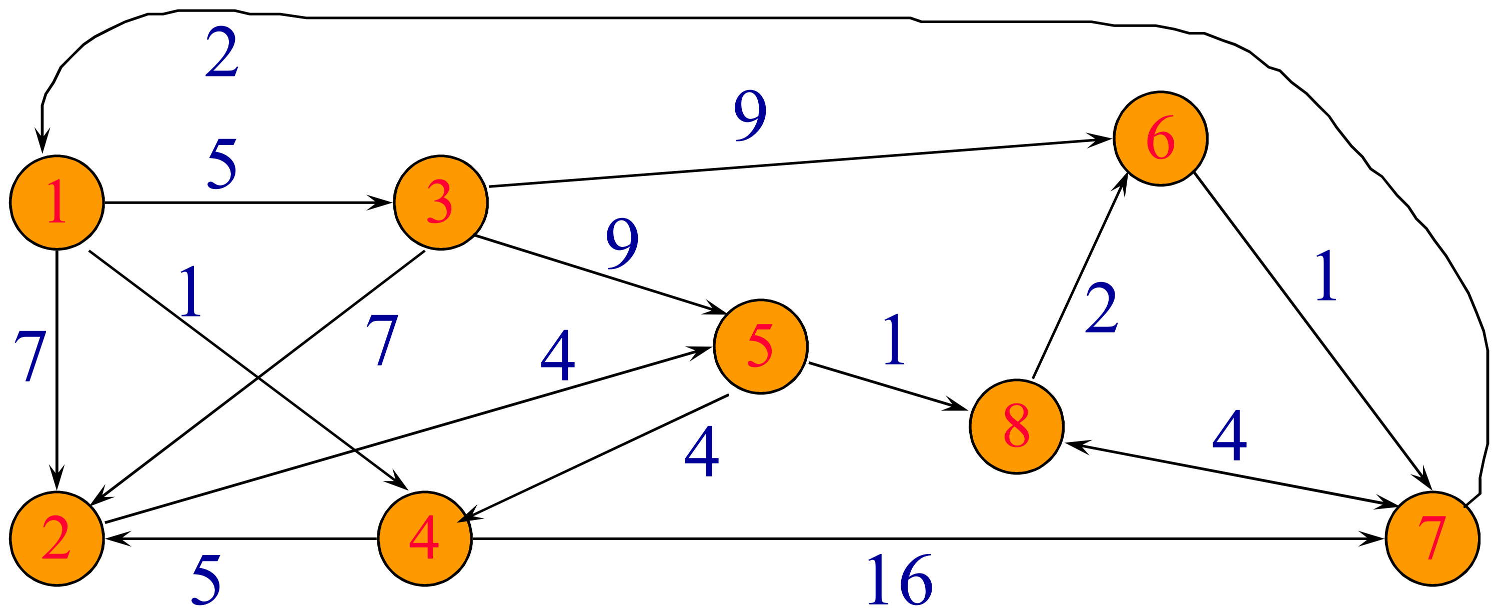
$A^n[i][j]$

- $A^n[i][j]$ is the length of a shortest path from vertex i to vertex j that has no intermediate vertex larger than n.
- No vertex is larger than n.
- Therefore, $A^n[i][j]$ is the length of a shortest path from vertex i to vertex j.



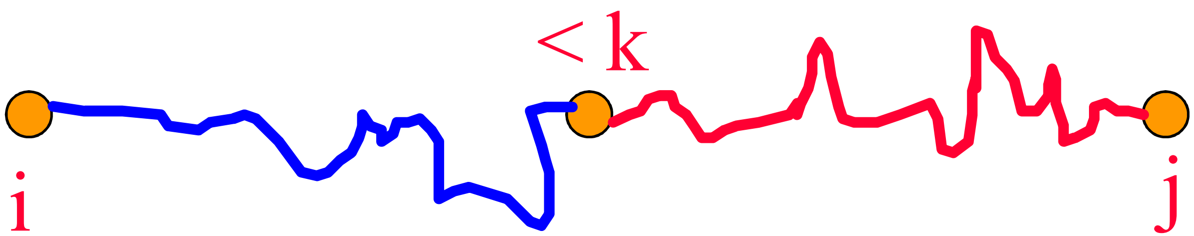
$A^0[i][j]$

- $A^0[i][j]$ is the length of a shortest path from vertex i to vertex j that has no intermediate vertex larger than 0.
  - Every vertex is larger than 0.
  - Therefore, $A^0[i][j]$ is the length of a single-edge path from vertex i to vertex j.

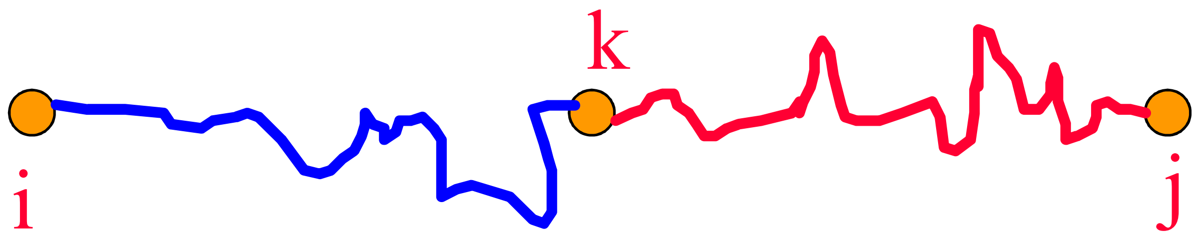


Recurrence for $A^k[i][j]$, $k > 0$

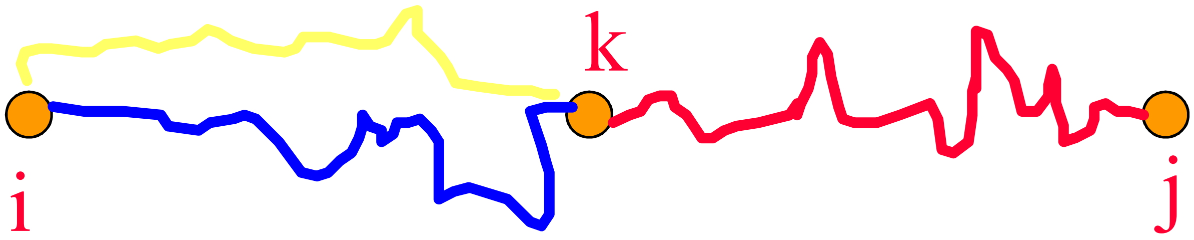
- The shortest path from vertex i to vertex j that has no intermediate vertex larger than k may or may not go through vertex k.
- If this shortest path does not go through vertex k, the largest permissible intermediate vertex is k-1. So the path length is $A^{k-1}[i][j]$.



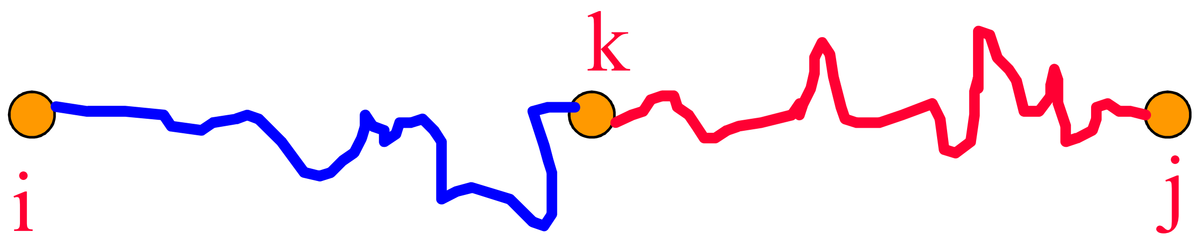
- Shortest path goes through vertex k.



- We may assume that vertex k is not repeated because no cycle has negative length.
- Largest permissible intermediate vertex on i to k and k to j paths is k-1.



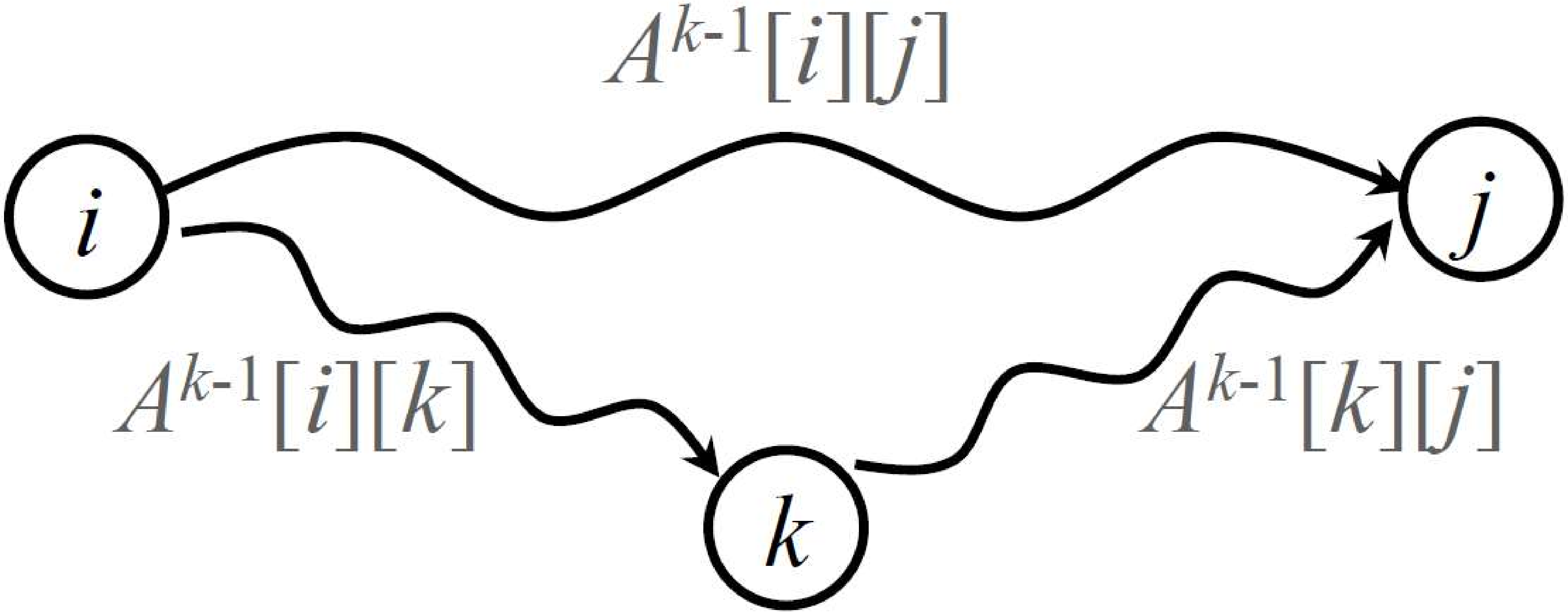
- i to k path must be a shortest i to k path that goes through no vertex larger than k-1.



- Similarly, k to j path must be a shortest k to j path that goes through no vertex larger than k-1.
- Therefore, length of i to k path is $A^{k-1}[i][k]$, and length of k to j path is $A^{k-1}[k][j]$.
- So, $A^{k}[i][j] = A^{k-1}[i][k] + A^{k-1}[k][j]$.

$$A^{0}[i][j] = \text{length}[i][j]$$

$$A^{k}[i][j] = \min\{A^{k-1}[i][j],$$
$$A^{k-1}[i][k] + A^{k-1}[k][j]\},\ \ k \geq 0$$



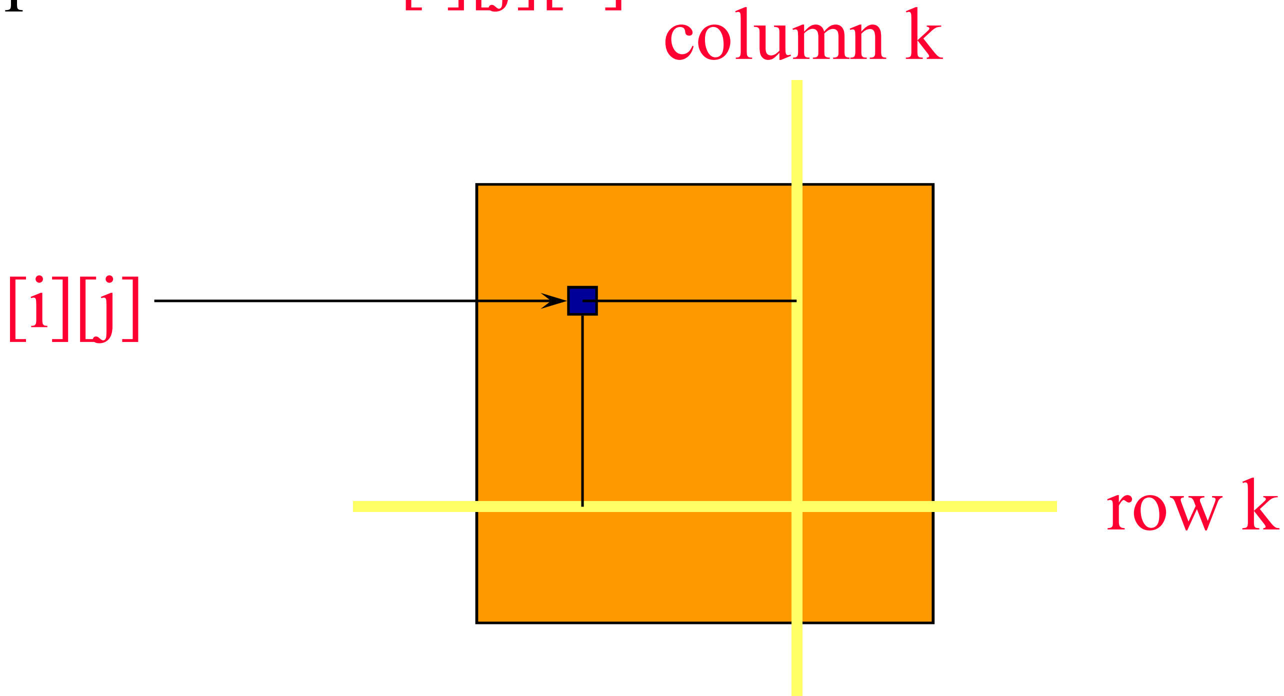
Floyd's Shortest Paths Algorithm

```
for (int k = 1; k <= n; k++)
   for (int i = 1; i <= n; i++)
      for (int j = 1; j <= n; j++)
         A[i][j][k] = min{A[i][j][k-1],
                          A[i][k][k-1] + A[k][j][k-1]};
```

- Time complexity is $O(n^3)$.
- $O(n^3)$ space is needed for A[*][*][*].

Space Reduction

- A[i][j][k] =
  min{A[i][j][k-1], A[i][k][k-1] + A[k][j][k-1]}
- When neither i nor j equals k, A[i][j][k-1] is used only in the computation of A[i][j][k].



- So A[i][j][k] can overwrite A[i][j][k-1].

Space Reduction

- $A[i][j][k] =$

  $\min\{A[i][j][k-1], A[i][k][k-1] + A[k][j][k-1]\}$
- When $i = k$, $A[i][j][k] = A[i][j][k-1]$.
  - $A[k][j][k] = \min\{A[k][j][k-1], A[k][k][k-1] + A[k][j][k-1]\}$

    $= \min\{A[k][j][k-1], 0 + A[k][j][k-1]\}$

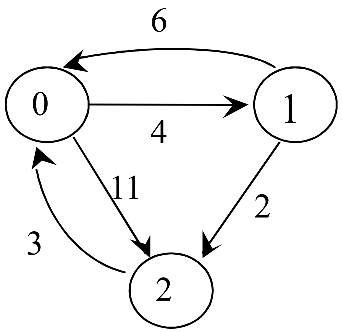
    $= A[k][j][k-1]$
- So, when $i = k$, $A[i][j][k]$ can overwrite $A[i][j][k-1]$.
- Similarly when $j = k$, $A[i][j][k]$ can overwrite $A[i][j][k-1]$.
- So, in all cases $A[i][j][k]$ can overwrite $A[i][j][k-1]$.

Floyd's Shortest Paths Algorithm

```
for (int k = 1; k <= n; k++)
  for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
      A[i][j] = min{A[i][j], A[i][k] + A[k][j]};
```

- Initially, A[i][j] = A[i][j][0].
- Upon termination, A[i][j] = A[i][j][n].
- Time complexity is $O(n^3)$.
- $O(n^2)$ space is needed for A[*][*].

Example



(a) Example digraph

$$A^0[2][1] = \min(A^{-1}[2][1], A^{-1}[2][0] + A^{-1}[0][1])$$

A^{-1}	0	1	2
0	0	4	11
1	6	0	2
2	3	∞	0

(b) A^{-1}

A^{0}	0	1	2
0	0	4	11
1	6	0	2
2	3	**7**	0

(c) A^{0}

$$A^1[0][2] = \min(A^0[0][2], A^0[0][1] + A^0[1][2])$$

A^{1}	0	1	2
0	0	4	6
1	6	0	2
2	3	7	0

(d) A^{1}

A^{2}	0	1	2
0	0	4	6
1	5	0	2
2	3	7	0

(e) A^{2}

$$A^2[1][0] = \min(A^1[1][0], A^1[1][2] + A^0[2][0])$$