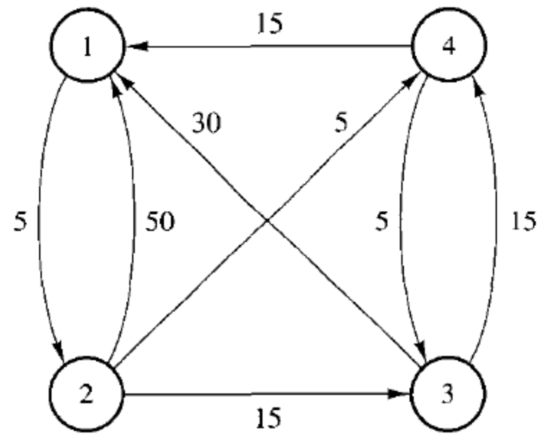
The All-Pairs Shortest-Path Problem



Problem definition :

- Let $G = \{V, E\}$ be a connected "directed" graph where V is the set of nodes and E is the set of edges.
- Each edge has an associated nonnegative length
- **Find the shortest path between all pairs of nodes in** G

We study a dynamic programming approach : Floyd's algorithm.

Floyd Algorithm

We have a distance matrix $L[i,j]$ that gives the length of each edge :

- $L[i,i] = 0$, $L[i,j] \geq 0$ if $i \neq j$,
- $L[i,j] = \infty$ if the edge (i,j) does not exist.

Algorithm Floyd($L[n, n]$)
  $D = L$
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        $D[i,j] = min(D[i,j], D[i,k] + D[k,j])$
  return D

Algorithm constructs a matrix D that gives the length of the shortest path between each pair of nodes.

D is initialized to L, the direct distances between nodes.

After each iteration k, D contains the length of the shortest paths that only use nodes in $\{1, 2, \ldots, k\}$ as intermediate nodes.

Floyd Algorithm

Algorithm Floyd$(L[n,n])$
  $D = L$
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        $D[i,j] = min(D[i,j], D[i,k] + D[k,j])$
  return D

At iteration k, the algo checks each pair of nodes (i,j) whether or not there exists a path from i to j passing through node k that is better than the present optimal path passing only through nodes in $\{1, 2, \ldots, k-1\}$.

If D_k represents the matrix D after the k-th iteration (so $D_0 = L$), the necessary check can be implemented by

$$D[i,j] = min(D[i,j], D[i,k] + D[k,j])$$

Execution of Floyd's algorithm

Algorithm Floyd($L[n, n]$)
  $D = L$
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        $D[i, j] = min(D[i, j], D[i, k] + D[k, j])$
  return D

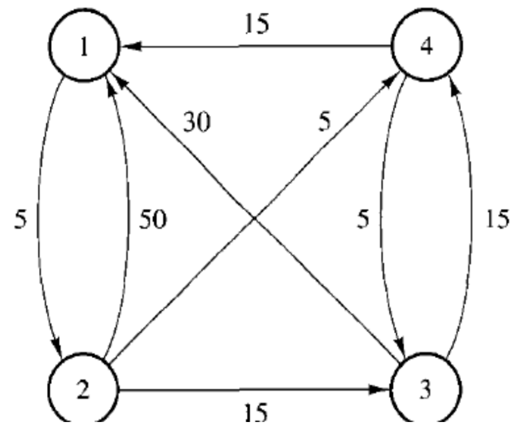
Base case $D = L$, the smallest problem instances

$$D_0 = L = \begin{pmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & \infty & 0 & 15 \\ 15 & \infty & 5 & 0 \end{pmatrix}$$

Execution of Floyd's algorithm

Algorithm Floyd($L[n, n]$)
  $D = L$
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        $D[i, j] = min(D[i, j], D[i, k] + D[k, j])$
  return D

For $k = 1$, compute the shortest path between each pair of nodes (i, j) when the path is allowed to pass through node 1.



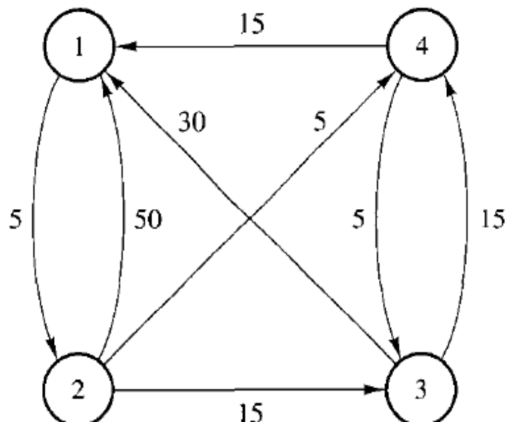
$$\begin{pmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & \infty & 0 & 15 \\ 15 & \infty & 5 & 0 \end{pmatrix}$$

$$D_1 = \begin{pmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{pmatrix}$$

Execution of Floyd's algorithm

Algorithm Floyd$(L[n,n])$
  $D = L$
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        $D[i,j] = min(D[i,j], D[i,k] + D[k,j])$
  return D

For $k = 2$, compute the shortest path between each pair of nodes (i, j) when the path is allowed to pass through nodes 1 and 2.



$$D_1 = \begin{pmatrix} 0 & 5 & \infty & \infty \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{pmatrix}$$

$$D_2 = \begin{pmatrix} 0 & 5 & 20 & 10 \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \end{pmatrix}$$

Execution of Floyd's algorithm

Algorithm Floyd$(L[n, n])$

$D = L$

for $(k = 1; k \leq n; k++)$

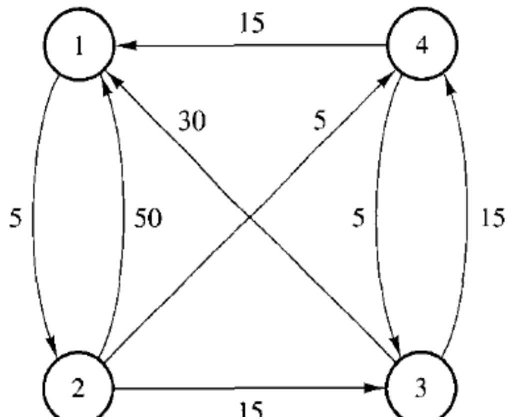
  for $(i = 1; i \leq n; i++)$

    for $(j = 1; j \leq n; j++)$

      $D[i, j] = min(D[i, j], D[i, k] + D[k, j])$

return D

For $k = 3$, compute the shortest path between each pair of nodes (i, j) when the path is allowed to pass through nodes $\{1, 2, 3\}$.



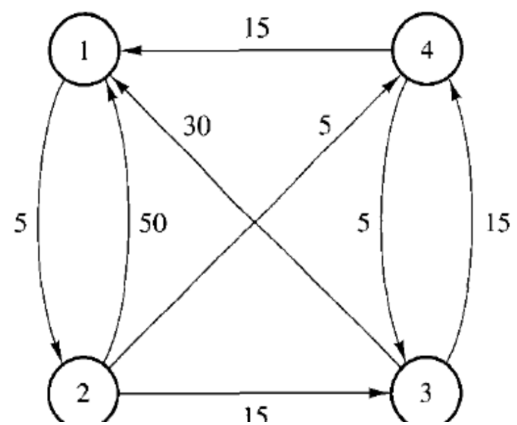
$$D_2 = \begin{pmatrix} 0 & 5 & 20 & 10 \\ 50 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 5 & 20 & 10 \\ 45 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \end{pmatrix}$$

Execution of Floyd's algorithm

Algorithm Floyd$(L[n,n])$
  $D = L$
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        $D[i,j] = min(D[i,j], D[i,k] + D[k,j])$
  return D

For $k = 4$, solution, compute the shortest path between each pair of nodes (i,j) when the path is allowed to pass through any nodes.



$$D_3 = \begin{pmatrix} 0 & 5 & 20 & 10 \\ 45 & 0 & 15 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{pmatrix}$$

$$D_4 = \begin{pmatrix} 0 & 5 & 15 & 10 \\ 20 & 0 & 10 & 5 \\ 30 & 35 & 0 & 15 \\ 15 & 20 & 5 & 0 \end{pmatrix}$$

Computing the shortest paths

- We want to know the shortest paths, not just their length.
- For that we create a new matrix P of size $n \times n$.
- Then use the following algorithm in place of the previous one :

Algorithm Floyd$(D[n, n])$
  Input : An array D of shortest path lenghts
  Output : The shortest path between every pair of nodes
  $P[n, n]$ an $n \times n$ array initialized to 0
  for $(k = 1; k \leq n; k++)$
    for $(i = 1; i \leq n; i++)$
      for $(j = 1; j \leq n; j++)$
        if $D[i, k] + D[k, j] < D[i, j]$ **then**
          $D[i, j] = D[i, k] + D[k, j]$
          $P[i, j] = k$;

Computing the shortest paths

- The matrix P is initialized to 0.
- When the previous algorithm stops, $P[i,j]$ contains the number of the last iteration that caused a change in $D[i,j]$.

$$P = \begin{pmatrix} 0 & 0 & 4 & 2 \\ 4 & 0 & 4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- If $P[i,j] = 0$, then the shortest path between i and j is directly along the edge (i,j).
- If $P[i,j] = k$, the shortest path from i to j goes through k.

Computing the shortest paths

$$P = \begin{pmatrix} 0 & 0 & 4 & 2 \\ 4 & 0 & 4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

- Look recursively at $P[i,k]$ and $P[k,j]$ to find other intermediate vertex along the shortest path.
- In the table above, since, $P[1,3] = 4$, the shortest path from 1 to 3 goes through 4. If we look recursively at $P[1,4]$ we find that the path between 1 and 4 goes through 2. Recursively again, if we look at $P[1,2]$ and $P[2,4]$ we find direct edge.
- Similarly if we look recursively to $P[4,3]$ we find a direct edge (because $P[4,3] = 0$). Then the shortest path from 1 to 3 is 1,2,4,3.