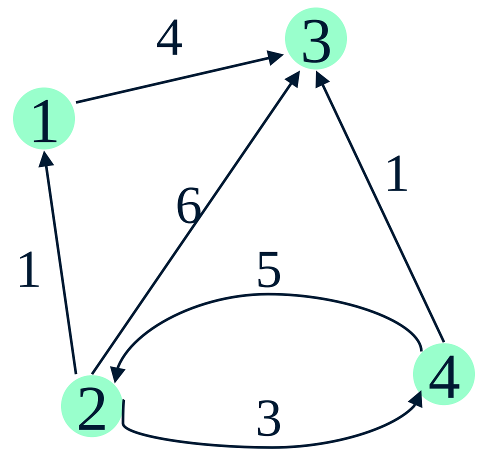
UNIT III DYNAMIC PROGRAMMING:

Dynamic Programming – Change-making Problem – Computing a Binomial Coefficient – All-pairs Shortest-paths Problem – Warshall's and Floyd's Algorithms – 0/1 Knapsack Problem

Floyd's Algorithm: All pairs shortest paths

Problem: **In a weighted (di)graph, find shortest paths between every pair of vertices**

Same idea: construct solution through series of matrices $D^{(0)}$, …, $D^{(n)}$ using increasing subsets of the vertices allowed as intermediate

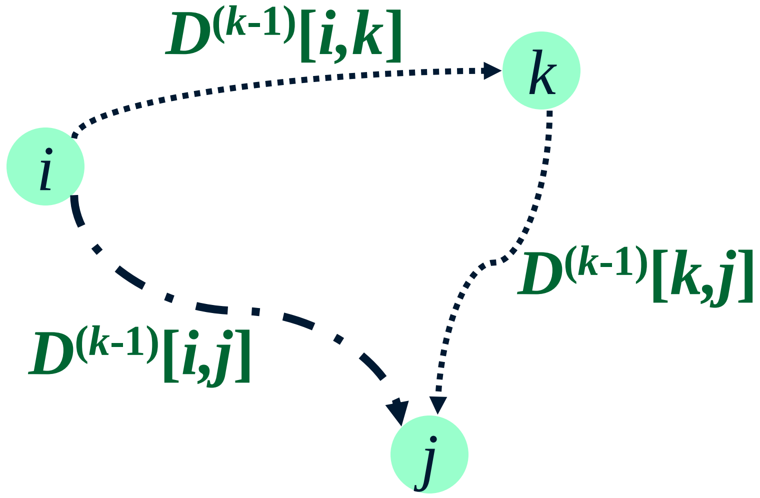


$$\begin{matrix} 0 & \infty & 4 & \infty \\ 1 & 0 & 4 & 3 \\ \infty & \infty & 0 & \infty \\ 6 & 5 & 1 & 0 \end{matrix}$$

Floyd's Algorithm (matrix generation)

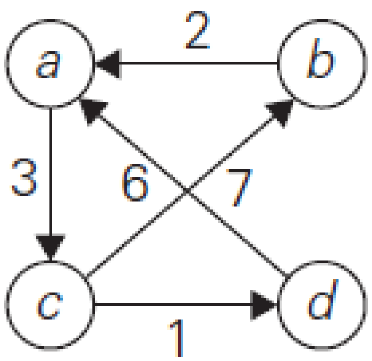
On the *k*-th iteration, the algorithm determines shortest paths between every pair of vertices *i, j* that use only vertices among 1,…,*k* as intermediate

$$D^{(k)}[i,j] = \min \{D^{(k-1)}[i,j],\ D^{(k-1)}[i,k] + D^{(k-1)}[k,j]\}$$



Initial condition?

Floyd's Algorithm (matrix generation)



(a)

$$W = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{bmatrix} \end{array}$$

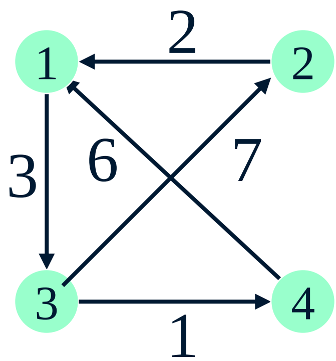
(b)

$$D = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} \begin{array}{cccc} a & b & c & d \end{array} \\ \begin{bmatrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ 7 & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{bmatrix} \end{array}$$

(c)

(a) Digraph. (b) Its weight matrix. (c) Its distance matrix.

Floyd's Algorithm (example)



$$D^{(0)} = \begin{matrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \infty & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \infty & 0 \end{matrix}$$

$$D^{(1)} = \begin{matrix} 0 & \infty & 3 & \infty \\ 2 & 0 & \mathbf{5} & \infty \\ \infty & 7 & 0 & 1 \\ 6 & \infty & \mathbf{9} & 0 \end{matrix}$$

$$D^{(2)} = \begin{matrix} 0 & \infty & 3 & \infty \\ 2 & 0 & 5 & \infty \\ \mathbf{9} & 7 & 0 & 1 \\ 6 & \infty & 9 & 0 \end{matrix}$$

$$D^{(3)} = \begin{matrix} 0 & \mathbf{10} & 3 & \mathbf{4} \\ 2 & 0 & 5 & \mathbf{6} \\ 9 & 7 & 0 & 1 \\ 6 & \mathbf{16} & 9 & 0 \end{matrix}$$

$$D^{(4)} = \begin{matrix} 0 & 10 & 3 & 4 \\ 2 & 0 & 5 & 6 \\ \mathbf{7} & 7 & 0 & 1 \\ 6 & 16 & 9 & 0 \end{matrix}$$

Floyd's Algorithm (example)

Solve the all-pairs shortest path problem for the digraph with the following weight matrix

$$\begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

$$D^{(0)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

Floyd's Algorithm (example)

$$D^{(0)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \infty \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \infty & \infty & \infty & 0 \end{bmatrix}$$

$$D^{(1)} = \begin{bmatrix} 0 & 2 & \infty & 1 & 8 \\ 6 & 0 & 3 & 2 & \mathbf{14} \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & \mathbf{5} & \infty & \mathbf{4} & 0 \end{bmatrix}$$

$$D^{(2)} = \begin{bmatrix} 0 & 2 & \mathbf{5} & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \mathbf{8} & 4 & 0 \end{bmatrix}$$

$$D^{(3)} = \begin{bmatrix} 0 & 2 & 5 & 1 & 8 \\ 6 & 0 & 3 & 2 & 14 \\ \infty & \infty & 0 & 4 & \infty \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & 8 & 4 & 0 \end{bmatrix}$$

$$D^{(4)} = \begin{bmatrix} 0 & 2 & \mathbf{3} & 1 & \mathbf{4} \\ 6 & 0 & 3 & 2 & \mathbf{5} \\ \infty & \infty & 0 & 4 & \mathbf{7} \\ \infty & \infty & 2 & 0 & 3 \\ 3 & 5 & \mathbf{6} & 4 & 0 \end{bmatrix}$$

$$D^{(5)} = \begin{bmatrix} 0 & 2 & 3 & 1 & 4 \\ 6 & 0 & 3 & 2 & 5 \\ \mathbf{10} & \mathbf{12} & 0 & 4 & 7 \\ \mathbf{6} & \mathbf{8} & 2 & 0 & 3 \\ 3 & 5 & 6 & 4 & 0 \end{bmatrix} = D$$

Floyd's Algorithm (pseudocode and analysis)

```
ALGORITHM  Floyd(W[1..n, 1..n])
    //Implements Floyd's algorithm for the all-pairs shortest-paths problem
    //Input: The weight matrix W of a graph with no negative-length cycle
    //Output: The distance matrix of the shortest paths' lengths
    D ← W //is not necessary if W can be overwritten
    for k ← 1 to n do
        for i ← 1 to n do
            for j ← 1 to n do
                D[i, j] ← min{D[i, j], D[i, k] + D[k, j]}
    return D
```

If $D[i,k] + D[k,j] < D[i,j]$ then $P[i,j] \leftarrow k$

Time efficiency: $\Theta(n^3)$

Space efficiency: Matrices can be written over their predecessors

Note: Works on graphs with negative edges but without negative cycles. Shortest paths themselves can be found, too. How?

Thank You