Design and Analysis of Algorithms

L28: Warshall & Floyd Algo

Dynamic Programming

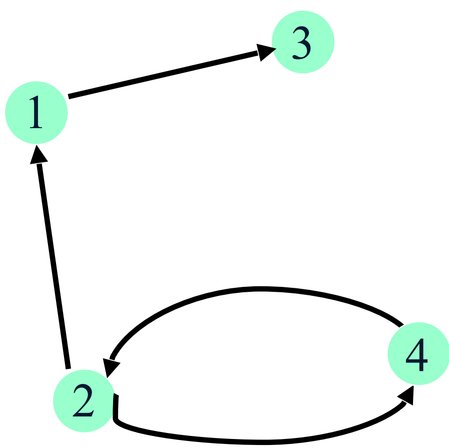
Dr. Ram P Rustagi
Sem IV (2020-Even)
Dept of CSE, KSIT
rprustagi@ksit.edu.in

Resources

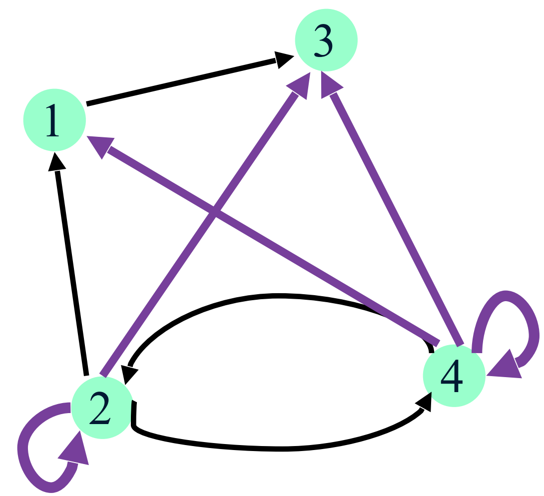
- Text book 1: Levitin
  - Sec **8.2**, 8.3, 8.4
- Text book 2: Horowitz
  - Sec 5.1, 5.2, 5.4, 5.8, 5.9
- R1: Introduction to Algorithms
    - Cormen et al.

Transitive Closure

- Computes the transitive closure of a relation
- Alternatively:
  - existence of all nontrivial paths in a digraph
- Example of transitive closure:



	1	2	3	4
1	0	0	1	0
2	1	0	0	1
3	0	0	0	0
4	0	1	0	0



	1	2	3	4
1	0	0	1	0
2	1	**1**	**1**	1
3	0	0	0	0
4	**1**	1	**1**	**1**

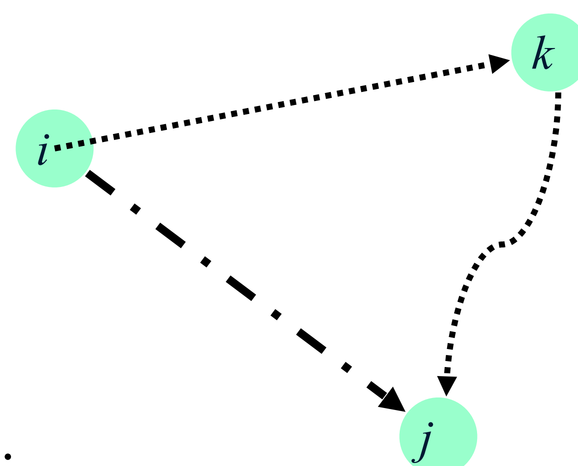
Warshall's Approach

- Constructs transitive closure T as the last matrix in the sequence of n-by-n matrices
  $R^{(0)}, \ldots, R^{(k)}, \ldots, R^{(n)}$ where
- $R^{(k)}[i,j]=1$ iff
  - there is nontrivial path from i to j
  - with only the first k vertices (numbered from 1 to k) are allowed as intermediate
- Note that
  - $R^{(0)} = A$ (adjacency matrix),
  - $R^{(n)} = T$ (transitive closure)

Warshall's algo: Recurrence

- On the k^{th} iteration,
  - the algo determines for every pair of vertices i, j
  - if a path exists from i to j
    - with just vertices 1, …, k allowed as intermediate

$$R^{(k)}[i,j] = \begin{cases} R^{(k-1)}[i,j] & \text{(path using just 1, …, k-1)} \\ \textbf{or} & \\ R^{(k-1)}[i,k] \ \textbf{and} \ R^{(k-1)}[k,j] & \text{(path from i to k and from k to j, using just 1, …, k-1)} \end{cases}$$



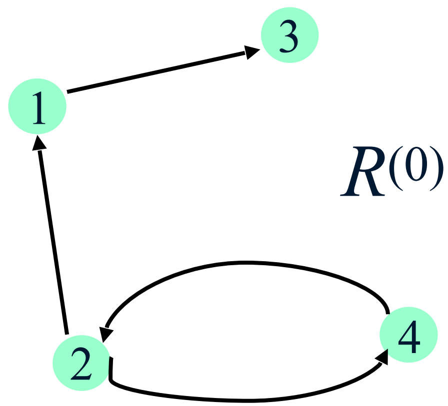
Warshall's algo: Matrix Generation

- Recurrence relating elements $R^{(k)}$ to elements of $R^{(k-1)}$ is:

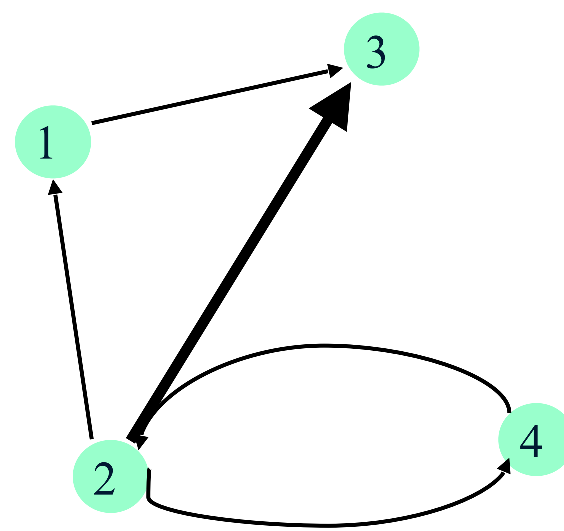
  R(k)[i,j]=R(k-1)[i,j] **or** (R(k-1)[i,k] and R(k-1)[k,j])

- It implies the following rules for generating $R^{(k)}$ from $R^{(k-1)}$:
  - Rule 1: If an element in row i and column j is 1 in $R^{(k-1)}$, it remains 1 in $R^{(k)}$
  - Rule 2: If an element in row i and column j is 0 in $R^{(k-1)}$,
    - it has to be changed to 1 in $R^{(k)}$ iff the element in its row i and column k and the element in its row k and column j are both 1's in $R^{(k-1)}$

Warshall’s algo: Example

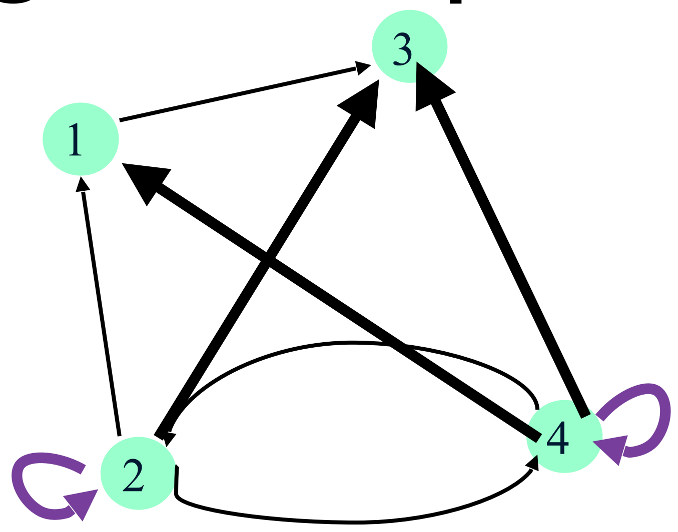


$$R^{(0)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix} \qquad R^{(1)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$



Warshall's algo: Example

$$R^{(1)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & \mathbf{1} & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{matrix}$$



$$R^{(2)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ \mathbf{1} & 1 & \mathbf{1} & \mathbf{1} \end{matrix}$$

$$R^{(3)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

No Change

$$R^{(4)} = \begin{matrix} 0 & 0 & 1 & 0 \\ 1 & \mathbf{1} & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \end{matrix}$$

Warshall's Algo: Analysis

```
Algo Warshall(A[1..n,1..n])
// i/p: Adjacency matrix A of a diagraph with n vertices
// o/p: Transitive closure of diagraph
R(0) ← A
for k←1 to n do
    for i←1 to n  do
        for j←1 to n do
            R(k)[i,j]←R(k-1)[i,j] OR
                (R(k-1)[i,k] AND R(k-1)[k,j])
return R(n)
```

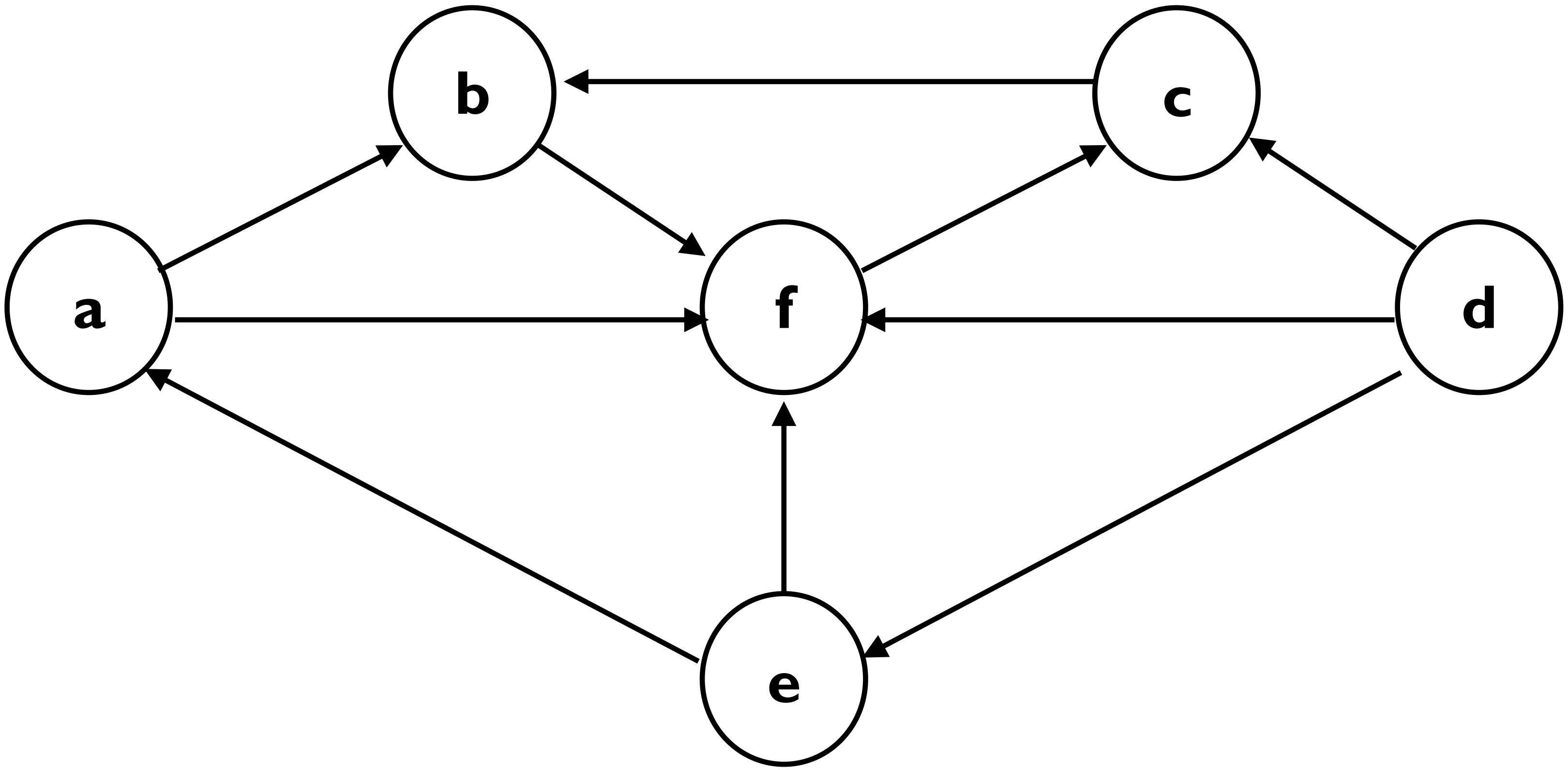
Time efficiency: $\Theta(n^3)$

Space efficiency:

Matrices can be written over their predecessors (with some care), so it's $\Theta(n^2)$.

Exercise:

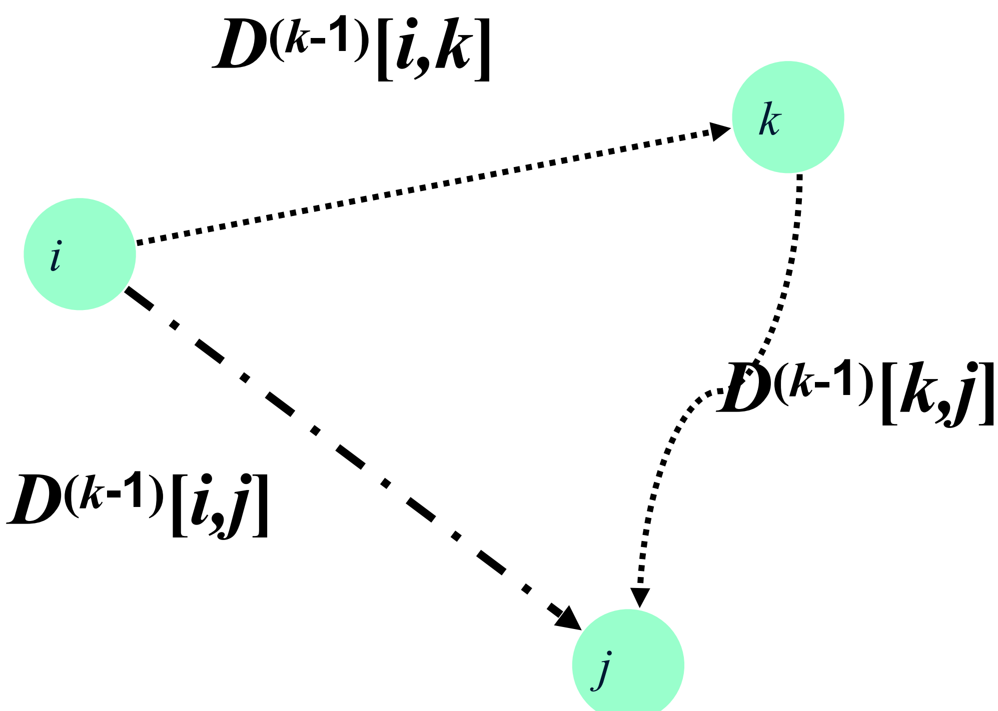
- Ex: Construct transitive closure for below graph



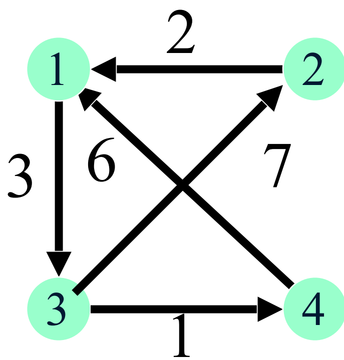
Floyd's Algorithm: Matrix Generation

- On the k^{th} iteration,
  - the algorithm determines shortest paths between every pair of vertices i, j that use only vertices among $1, \ldots, k$ as intermediate

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



Example: Floyd Algo



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

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

Floyd Algo: Analysis

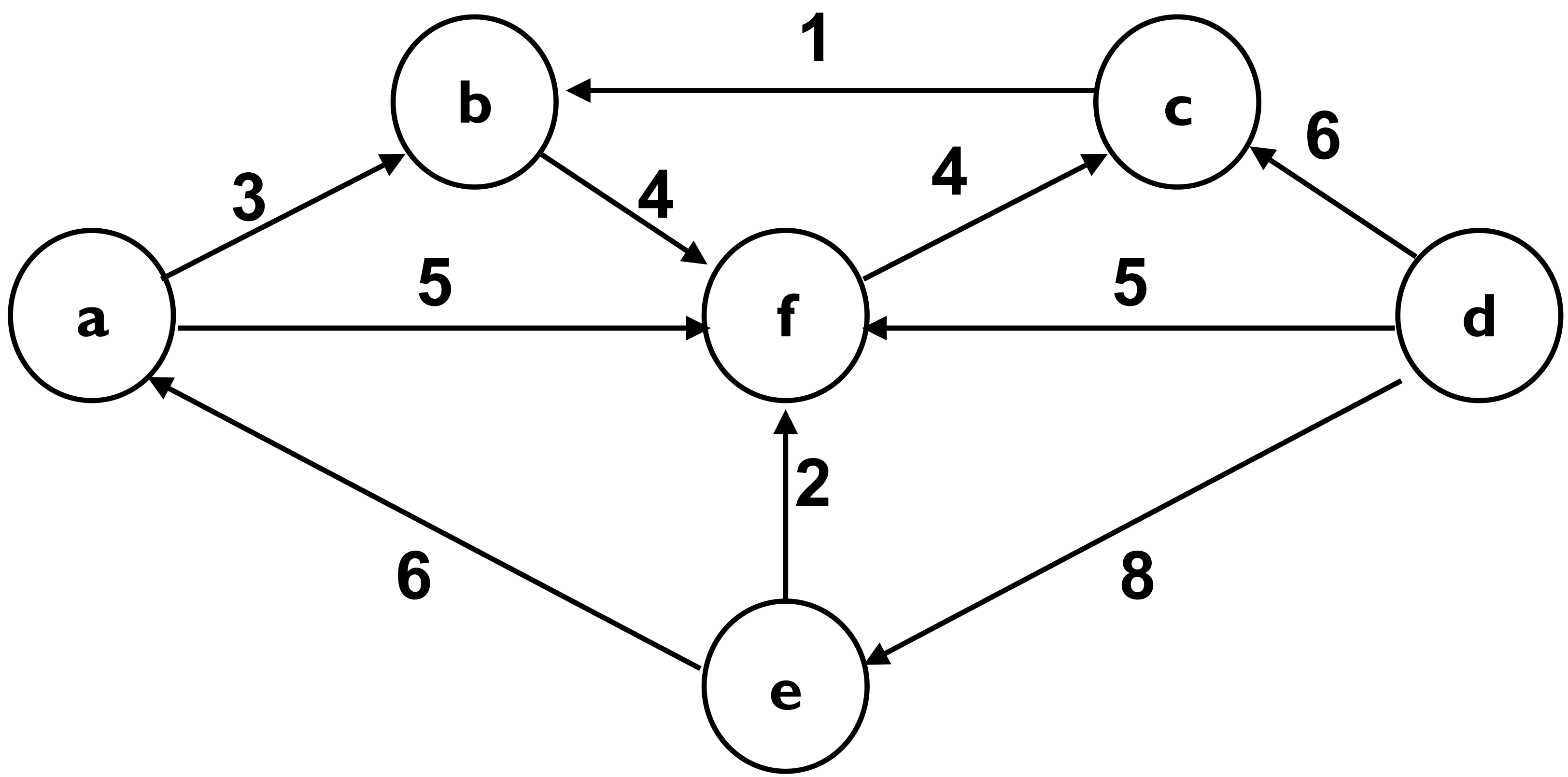
```
Algo Floyd(A[1..n,1..n])
// i/p: Weight matrix W of a diagraph A with n vertices
// o/p: Distance matrix of shortest path lengths
D ← W
for k←1 to n do
    for i←1 to n  do
        for j←1 to n do
            D[i,j]←min{D[i,j],D[i,k]+D[k,j]}
            if D[i,k]+D[k,j] < D[i,j]  then
                P[i,j] ← k
return D
```

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

Space efficiency: Matrices can be written over their predecessors (with some care), so it's $\Theta(n^2)$.

Exercise:

- Ex: Find all pair shortest distance for below graph



Summary

- Transitive closure
- Warshall Algorithm
- Floyd Algorithm