Data Structures and Algorithms Design

BITS Pilani
Hyderabad Campus

ONLINE SESSION -PLAN

Sessions(#)	List of Topic Title	Text/Ref Book/external resource
10	**Dynamic Programming - Design Principles and Strategy, Matrix Chain Product Problem, 0/1 Knapsack Problem, All-pairs Shortest Path Problem**	**T1: 5.3, 7.2**

The General Dynamic Programming Technique

- Applies to a problem that at first seems to require a lot of time (possibly exponential), provided we have:
  - **Simple subproblems:** the subproblems can be defined in terms of a few variables, such as j, k, l, m, and so on.
  - **Subproblem optimality:** the global optimum value can be defined in terms of optimal subproblems
  - **Subproblem overlap:** the subproblems are not independent, but instead they overlap (hence, should be constructed bottom-up).

All-Pairs Shortest Paths

- Problem: Given a weighted connected graph (undirected or directed), the ***all-pairs shortest paths problem*** asks to find the distances—i.e., the lengths of the shortest paths—from each vertex to all other vertices.

- Floyd's algorithm computes the distance matrix of a weighted graph with n vertices through a series of $n \times n$ matrices: $D(0), \ldots, D(k-1), D(k), \ldots, D(n)$.

All-Pairs Shortest Paths

- Each of these matrices contains the lengths of shortest paths with certain constraints on the paths considered for the matrix in question.
- The element $d(k)ij$ in the ith row and the jth column of matrix $D(k)$ $(i, j = 1, 2, \ldots, n, k = 0, 1, \ldots, n)$ is equal to the length of the shortest path among all paths from the ith vertex to the jth vertex with each intermediate vertex, if any, numbered not higher than k.
- In particular, the series starts with $D(0)$, which does not allow any intermediate vertices in its paths, hence, $D(0)$ is simply the weight matrix of the graph.

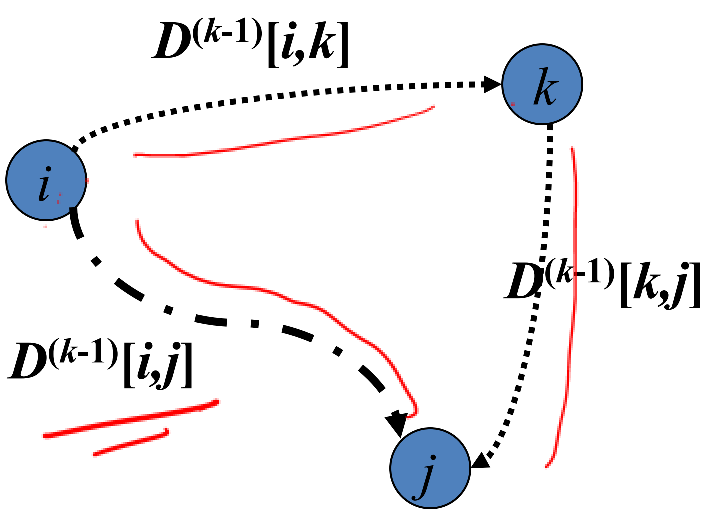
All-Pairs Shortest Paths

- The last matrix in the series, *D(n)*, contains the lengths of the shortest paths among all paths that can use all *n* vertices as intermediate and hence is nothing other than the distance matrix being sought.

All-Pairs Shortest Paths-Floyd's Algorithm

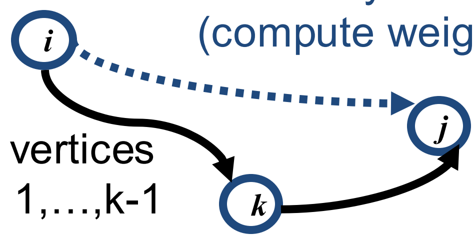
- On the k-th iteration, the algorithm determines shortest paths between every pair of vertices i, j that use only vertices among 1,…,k as intermediate

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



All-Pairs Shortest Paths-Floyd's Algorithm

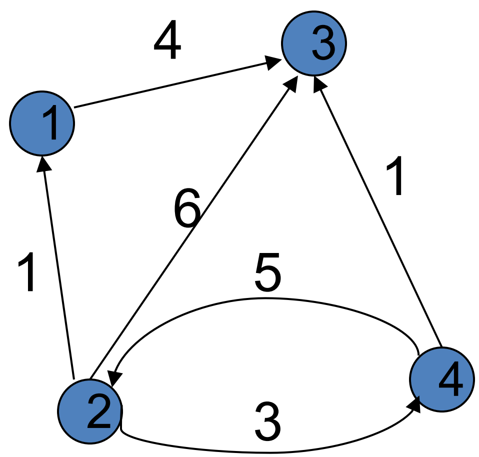
Uses only vertices numbered 1,…,k
(compute weight of this edge)



Uses only vertices
numbered 1,…,k-1

Uses only vertices
numbered 1,…,k-1

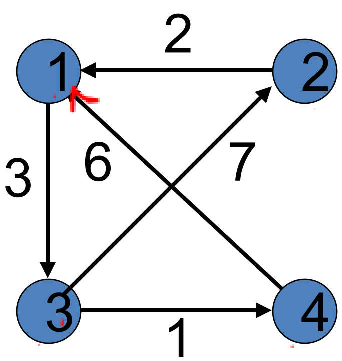
All-Pairs Shortest Paths



$D^{(0)}$

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \infty & 4 & \infty \\ 2 & 1 & 0 & 4 & 3 \\ 3 & \infty & \infty & 0 & \infty \\ 4 & 6 & 5 & 1 & 0 \end{array}$$

All-Pairs Shortest Paths



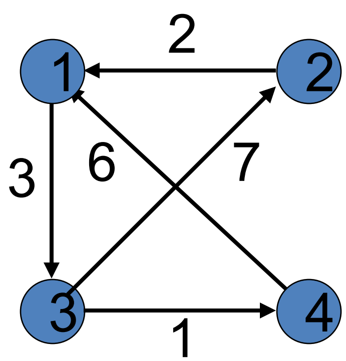
$$D^{(0)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \infty & 3 & \infty \\ 2 & 2 & 0 & \infty & \infty \\ 3 & \infty & 7 & 0 & 1 \\ 4 & 6 & \infty & \infty & 0 \end{array}$$

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

$(2,\cancel{1}) = 2$ $\quad(\cancel{1},3) = 3$ $\quad(2,3) = 5$

$(4,\cancel{1}) = 6$ $\quad(\cancel{1},3) = 3$ $\quad(4,3) = 9$

All-Pairs Shortest Paths



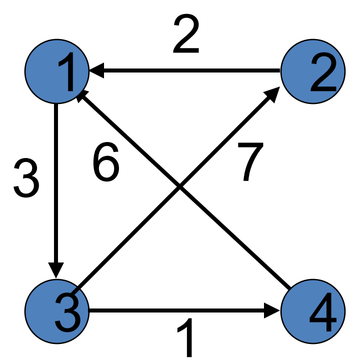
$$
D^{(1)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \infty & 3 & \infty \\ 2 & 2 & 0 & \mathbf{5} & \infty \\ 3 & \infty & 7 & 0 & 1 \\ 4 & 6 & \infty & \mathbf{9} & 0 \end{array}
$$

$(3,2) = 7 \quad (2,1) = 2 \quad \therefore (3,1) = 9$

$(3,2) = 7 \quad (2,3) = 5 \quad \therefore (3,3) = X$

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

All-Pairs Shortest Paths



$$D^{(2)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \infty & 3 & \infty \\ 2 & 2 & 0 & 5 & \infty \\ 3 & \mathbf{9} & 7 & 0 & 1 \\ 4 & 6 & \infty & 9 & 0 \end{array}$$

(4,3) = 9 (3,1) = 9 (4,1) = X

(4,3) = 9 (3,2) = 7 (4,2) = 16

(4,3) = 9 (3,4) = 1 (4,4) = X

(1,3) = 3 (3,1) = 9 (1,1) = X

(1,3) = 3, (3,2) = 7 (1,2) = 10

(1,3) = 3, (3,4) = 1 (1,4) = 4

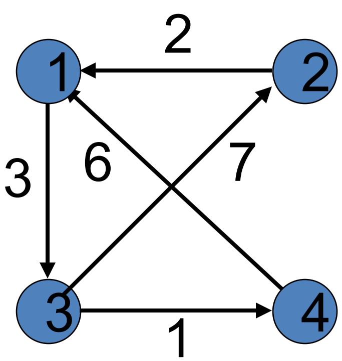
(2,3) = 5, (3,1) = 9 (2,1) = X

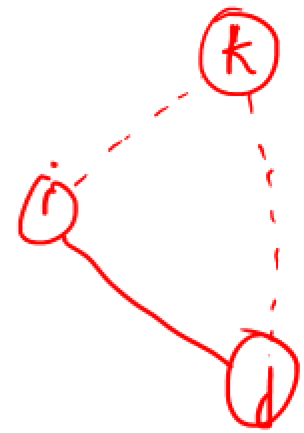
(2,3) = 5 (3,2) = 7 (2,2) = X

(2,3) = 5 (3,4) = 1 (2,4) = 6

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

All-Pairs Shortest Paths

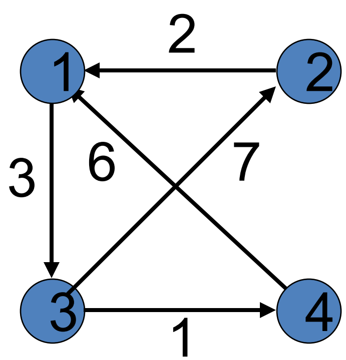




$$D^{(3)} = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & \mathbf{10} & 3 & \mathbf{4} \\ 2 & 2 & 0 & 5 & \mathbf{6} \\ 3 & 9 & 7 & 0 & 1 \\ 4 & 6 & \mathbf{16} & 9 & 0 \end{array}$$

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

All-Pairs Shortest Paths



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

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

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

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

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

All-Pairs Shortest Paths-Floyd's Algorithm

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*

All-Pairs Shortest Paths

- Find the distance between every pair of vertices in a weighted directed graph G.
- We can make n calls to Dijkstra's algorithm (if no negative edges), which takes O(n mlog n) time.
- Likewise, n calls to Bellman-Ford would take $O(n^2m)$ time.
- We can achieve $O(n^3)$ time using dynamic programming (similar to the Floyd-Warshall algorithm).

All-Pairs Shortest Paths-Negative Cycles

- Negative-weight edges may be present,
- But no negative-weight cycles. Why?

All-Pairs Shortest Paths-Negative Cycles

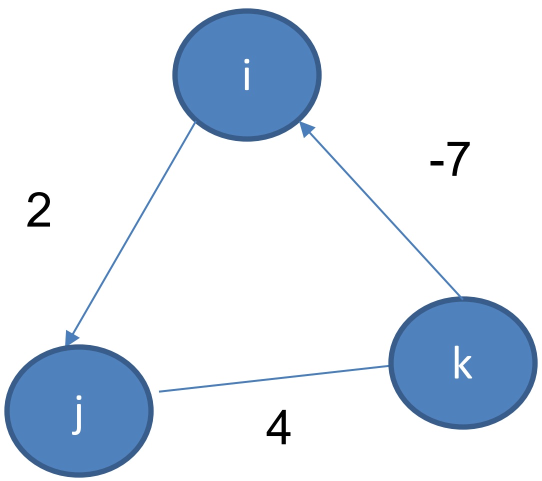
innovate achieve lead

- A negative cycle is a cycle whose edges sum to a negative value.
- There is no shortest path between any pair of vertices i, j which form part of a negative cycle, because path-lengths from i to j can be arbitrarily small (negative)

All-Pairs Shortest Paths-Negative Cycles

- Given a graph, suppose to have a cycle given by Nodes i , j, k of negative cost.

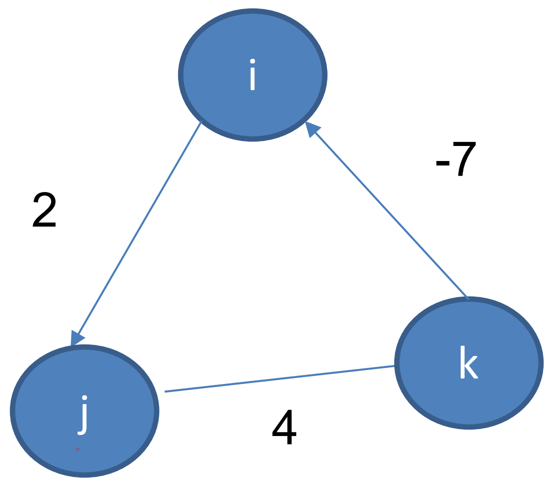
- Example:



- Suppose you want to find the shortest path between i and j

All-Pairs Shortest Paths-Negative Cycles

- You have P={(i,j)} with cost(P)=2.
- But you can loop through the negative cycle and have:



- P′={(i,j),(j,k),(k,i),(i,j)} with cost(P′)=1
- P″={(i,j),(j,k),(k,i),(i,j),(j,k),(k,i),(i,j)} with cost(P″)=0 and so on.

All-Pairs Shortest Paths-Negative Cycles

- Nevertheless, if there are negative cycles, the algorithm can be used to detect them.
- The intuition is as follows:
  - The Floyd's algorithm iteratively revises path lengths between all pairs of vertices (i,j), including where i=j
  - Initially, the length of the path (i,i) is zero;
  - A path {i,k,….i} can only improve upon this if it has length less than zero, i.e. denotes a negative cycle;
  - Thus, after the algorithm, (i,i) will be negative if there exists a negative-length path from i back to i.

All-Pairs Shortest Paths-Negative Cycles

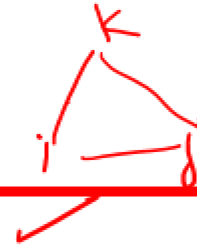
ALGORITHM *Floyd(W[1..n, 1..n])*
//Implements Floyd's algorithm for the all-pairs shortest-paths problem
//Input: The weight matrix W of a graph with no negative-length cycle
//Output: The distance matrix of the shortest paths' lengths
$D \leftarrow W$ //is not necessary if W can be overwritten

for $k \leftarrow 1$ ***to*** n ***do***
for $i \leftarrow 1$ ***to*** n ***do***
for $j \leftarrow 1$ ***to*** n ***do***

$$D[i, j] \leftarrow \min\{D[i, j], D[i, k] + D[k, j]\}$$

for i = 1 to n do
  if D[i, i] < 0 then return('graph contains a negative cycle')
return D

Transitive Closure



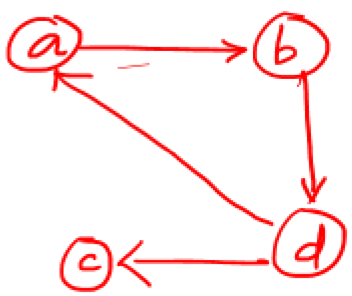
ALGORITHM *Warshall*$(A[1..n, 1..n])$

//Implements Warshall's algorithm for computing the transitive closure
//Input: The adjacency matrix A of a digraph with n vertices
//Output: The transitive closure of the digraph

$R^{(0)} \leftarrow A$
for $k \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** n **do**
    for $j \leftarrow 1$ **to** n **do**
      $R^{(k)}[i, j] \leftarrow R^{(k-1)}[i, j]$ **or** $(R^{(k-1)}[i, k]$ **and** $R^{(k-1)}[k, j])$
return $R^{(n)}$

$$\begin{matrix} 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{matrix} \longrightarrow \begin{matrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{matrix}$$

Example



	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

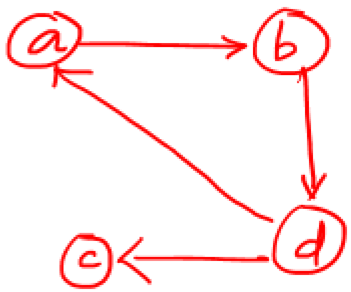
$\Rightarrow$

$D^{(a)}$

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

$(d,a) = 1 \quad (a,b) = 1 \quad \therefore (d,b) = 1$

Example



	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	1	1	0

$\Rightarrow$

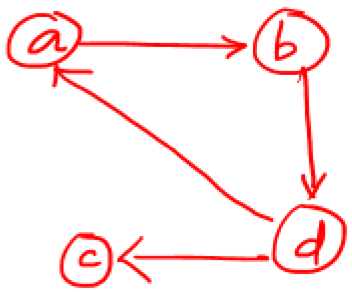
$D^{(b)}$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$(a,b)=1 \quad (b,d)=1 \quad \therefore (a,d)=1$

$(d,b)=1 \quad (b,d)=1 \quad \therefore (d,d)=1$

Example



	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

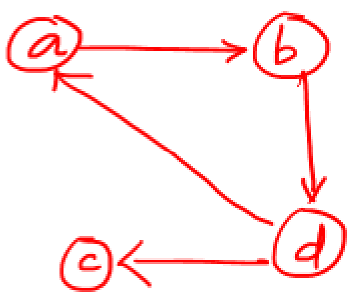
	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$\Rightarrow$

$D^{(c)}$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

Example



	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$D^{(d)}$

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	1	1	1

$\Rightarrow$

	a	b	c	d
a	1	1	1	1
b	1	1	1	1
c	0	0	0	0
d	1	1	1	1

THANK YOU!

BITS Pilani
Hyderabad Campus