SNS College of Engineering
Coimbatore - 641107

Floyd’s and warshall’s algorithm

Warshall's algorithm

- Transitive closure of directed graph, with n vertices.
- It can be defined as n-by-n Boolean matrix

$T=\{t_{ij}\}$ elements in i^{th} row & j^{th} column

$(1<=i<=n)$ $(1<=j<=n)$

$$R^0 \ldots R^{(k-1)}, R^{(K)} \ldots R^{(n)}$$

$$r_{ij}^{(k)} = r_{ij}^{(k-1)}$$

CONDITIONS:

if an element $.r_{ij}$ is 1 in $R^{(k-1)}$, it remains 1 in R^k , it remains 1 in R^k

if an element $.r_{ij}$ is 1 in $R^{(k-1)}$, it remains 0 in R^k , it remains 1 in R^k

if & only if element in i^{th} row & K column

if & only if element in j^{th} column & K row

Algorithm

// implementing warshall and floyd for computing transitive closure

// I/P : Adjacency matrix A

// O/P : Transitive closure

$R^{(0)}$ <- A

for k<-1 to n do

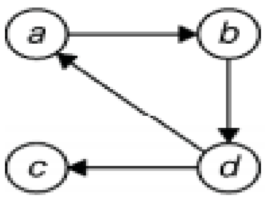
for i<- 1 to n do

for j<-1 to n do

$R^{(k)}$[I,j]<- $R^{(k-1)}$ [I,j]

Return $R^{(n)}$

Problem



$R^{(0)} =$

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

Ones reflect the existence of paths with no intermediate vertices ($R^{(0)}$ is just the adjacency matrix); boxed row and column are used for getting $R^{(1)}$.

$R^{(1)} =$

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

Ones reflect the existence of paths with intermediate vertices numbered not higher than 1, i.e., just vertex *a* (note a new path from *d* to *b*); boxed row and column are used for getting $R^{(2)}$.

$R^{(2)} =$

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

Ones reflect the existence of paths with intermediate vertices numbered not higher than 2, i.e., *a* and *b* (note two new paths); boxed row and column are used for getting $R^{(3)}$.

$R^{(3)} =$

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

Ones reflect the existence of paths with intermediate vertices numbered not higher than 3, i.e., *a*, *b*, and *c* (no new paths); boxed row and column are used for getting $R^{(4)}$.

$R^{(4)} =$

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

Ones reflect the existence of paths with intermediate vertices numbered not higher than 4, i.e., *a*, *b*, *c*, and *d* (note five new 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

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

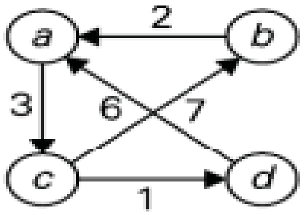
Algorithm

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

$D \leftarrow W$ //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**
            $D[i, j] \leftarrow \min\{D[i, j], D[i, k] + D[k, j]\}$
return D

Problem



$D^{(0)} =$

	a	b	c	d
a	0	∞	3	∞
b	2	0	∞	∞
c	∞	7	0	1
d	6	∞	∞	0

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

$D^{(1)} =$

	a	b	c	d
a	0	∞	3	∞
b	2	0	**5**	∞
c	∞	7	0	1
d	6	∞	**9**	0

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e. just *a* (note two new shortest paths from *b* to *c* and from *d* to *c*).

$D^{(2)} =$

	a	b	c	d
a	0	∞	3	∞
b	2	0	5	∞
c	**9**	7	0	1
d	6	∞	9	0

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e. a and *b* (note a new shortest path from *c* to *a*).

$D^{(3)} =$

	a	b	c	d
a	0	**10**	3	**4**
b	2	0	5	**6**
c	9	7	0	1
d	6	**16**	9	0

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e. a, *b*, and *c* (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

$D^{(4)} =$

	a	b	c	d
a	0	10	3	4
b	2	0	5	6
c	**7**	7	0	1
d	6	16	9	0

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e. a, *b*, *c*, and *d* (note a new shortest path from *c* to *a*).

Analysis

$A(n, k)$ = sum for upper triangle + sum for the lower rectangle

$= \sum_{i=1}^{n} \sum_{j=1}^{n} + \sum_{k=1}^{n} i$

$= \Theta(n^3)$

Activity

Time complexity $O(n^3)$ then c(n) is ?