Transitive closure of graphs

and

all-pairs shortest paths

Transitive closure (accessibility)

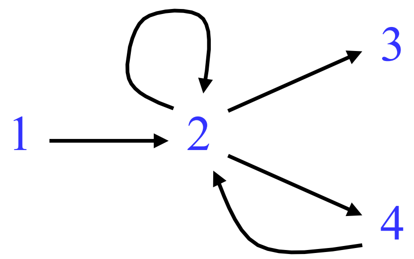
Problem:

$G = (S, A)$ directed graph

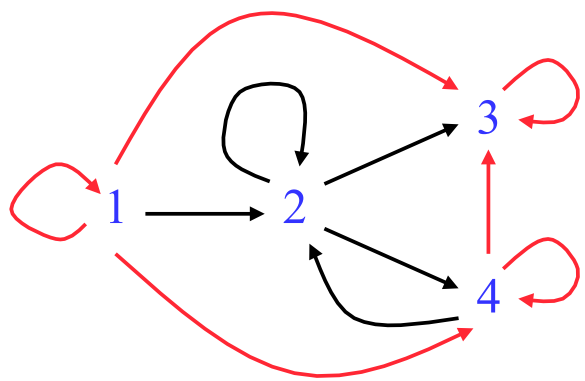
Compute $H = (S, B)$ where B is the reflexive and transitive closure of A.

Remark: $(s,t) \in B$ iff there exists a path from s to t in G

graph *G:*



graph *H:*



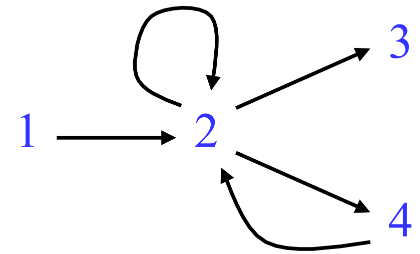
Matrix representation

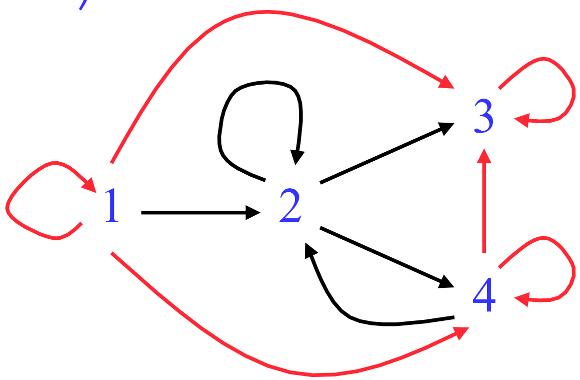
Matrix $n \times n$ where $n = |S|$

A adjacency matrix of G
= matrix of paths of length 1

B adjacency matrix of H
= matrix of paths of H

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \qquad B = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$





Closure by matrix multiplicaiton

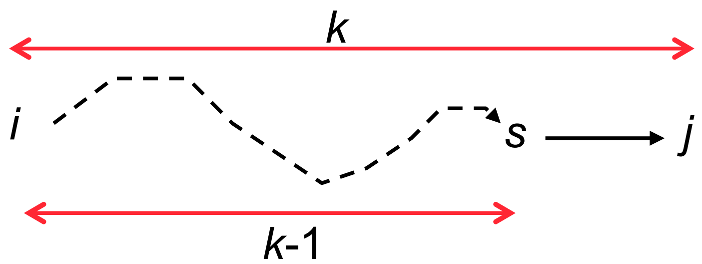
Notation

A_k = matrix of paths of length k in G

A_0 = I (identity matrix)
A_1 = A (matrix of paths of length 1)

Lemma

For all $k \geq 0$, $A_k = A^k$
(boolean matrix multiplication)



Proof:

$A_k[i,j] = 1$ iff $\exists\, s \in S\ A_{k-1}[i,s] = 1$ and $A[s,j] = 1$

let $A_k[i,j] = \Sigma_s\, A_{k-1}[i,s] \cdot A[s,j]$ where Σ boolean sum (OR).

that is, $A_k = A_{k-1} \cdot A$ and $A_0 = I$

then $A_k = A^k$

Closure by matrix multiplicaiton (cont)

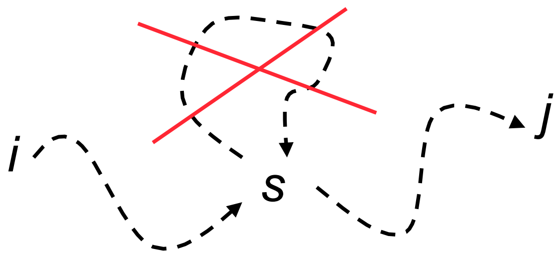
Simple path:

path with all vertices distinct (=not containing a cycle)

Lemma

∃ path from i to j in G iff

∃ simple path from i to j in G



$B[i,j]=1$ iff ∃ path from i to j in G

iff ∃ simple path from i to j in G

iff $\exists\, k,\ 0 \le k \le |S|-1,\ A_k[i,j]=1$

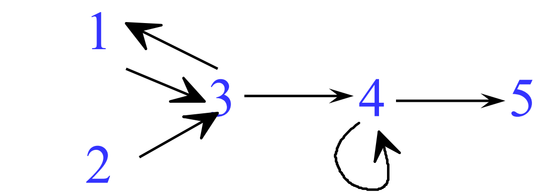
iff $\exists\, k,\ 0 \le k \le |S|-1\ A^k[i,j]=1$

therefore $B = I + A + A^2 + \ldots + A^{|S|-1}$

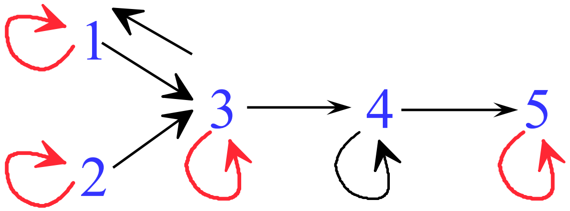
Computation of *B* using Horner's rule:

$A_0 = I$

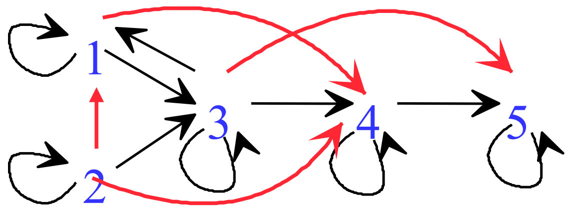
$A_i = I + A_{i-1} A$ for $i=1..|S|-1$



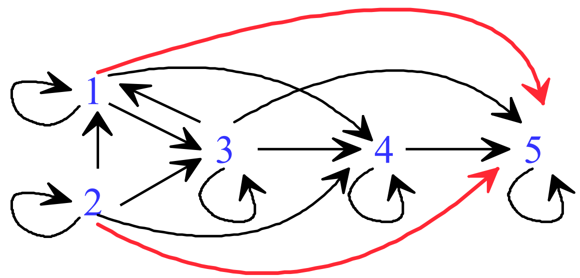
$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$



$$I + A = \begin{pmatrix} 1&0&1&0&0 \\ 0&1&1&0&0 \\ 1&0&1&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$



$$I + A + A^2$$
$$= I + (I{+}A).A = \begin{pmatrix} 1&0&1&1&0 \\ 1&1&1&1&0 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$



$$I + A + A^2 + A^3$$
$$= I + (I{+}A{+}A^2).A = \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

3 matrix products

$$= I + A + A^2 + A^3 + A^4$$
$$= I + (I{+}A{+}A^2 {+}A^3).A = B$$

Time complexity

$n=|S|$

$n-1$ additions and $n-1$ products of boolean matrices $n \times n$
=> $O(n \cdot M(n))$

each product is done in $O(n^3)$ operations => $O(n^4)$

there exist matrix multiplication algorithms running in time $o(n^3)$:
Strassen 1969: $O(n^{2.37})$ (now improved to $O(n^{2.37})$)
Four russians (Арлазаров, Диниц, Кронрод, Фарадзев) 1970: $O(n^3/log^2(n))$

Time complexity

n=$|S|$

n-1 additions and n-1 products of boolean matrices $n \times n$
=> $O(n \cdot M(n))$

each product is done in $O(n^3)$ operations => $O(n^4)$

there exist matrix multiplication algorithms running in time $o(n^3)$:
Strassen 1969: $O(n^{2.37})$ (now improved to $O(n^{2.37})$)
Four russians (Арлазаров, Диниц, Кронрод, Фарадзев) 1970: $O(n^3/\log^2(n))$

$O(n^4)$ **is too much! can be done better with Dijkstra:**

Define v(i,j)=1 for each (i,j)∈A
For each node *i*, run Dijkstra's algorithm with source node *i*
B[i,j]=1 ssi δ(i,j)<∞
Running time $O(n \cdot n^2)=O(n^3)$

Speeding up

Notation

B_k = matrix of paths of length $\leq k$ in G

B_0 = I (identity matrix)

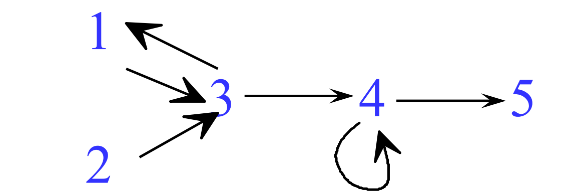
B_1 = matrix of paths of length ≤ 1 = $I + A$

B_{n-1} = matrix of simple paths = B

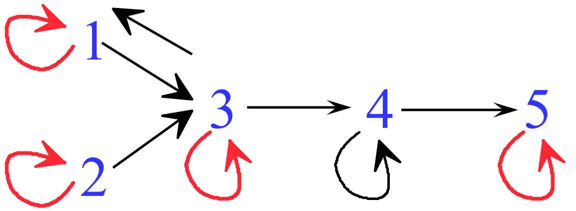
Lemma: $B_k = B_{k-1} \cdot (I+A)$

$\Rightarrow$ For all $k \geq 1$, $B_k = (I+A)^k$ and then $B_{2k} = B_k \,.\, B_k$

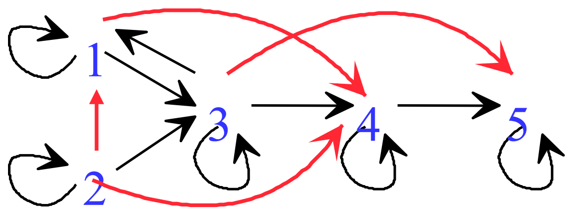
Compute B as an n-1 power in time $O(\log(n) \cdot M(n)) = O(\log(n) \cdot n^3)$



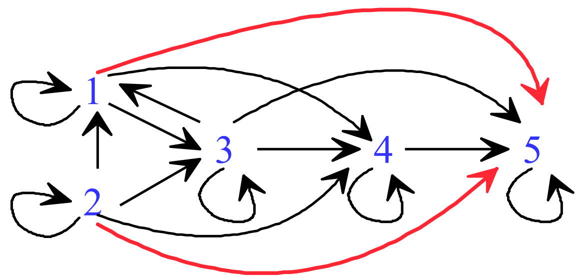
$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$



$$B_1 = \begin{pmatrix} 1&0&1&0&0 \\ 0&1&1&0&0 \\ 1&0&1&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$



$$B_2 = \begin{pmatrix} 1&0&1&1&0 \\ 1&1&1&1&0 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$



$$B = B_4 = \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

2 matrix products

Warshall's algorithm (Roy-Warshall)

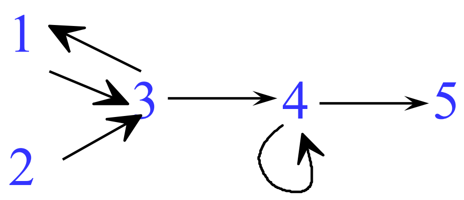
$G = (S, A)$ with $S = \{1, 2, ..., n\}$

Paths in G : $i \rightarrow s_1 \rightarrow s_2$ .... $s_m \rightarrow j$

Intermediate nodes : $s_1, s_2, \ldots, s_m$

Notation:

C_k	= matrix of paths in G with intermediate nodes $\leq k$
C_0	$= I + A$
C_n	= matrix of paths in G $= B$

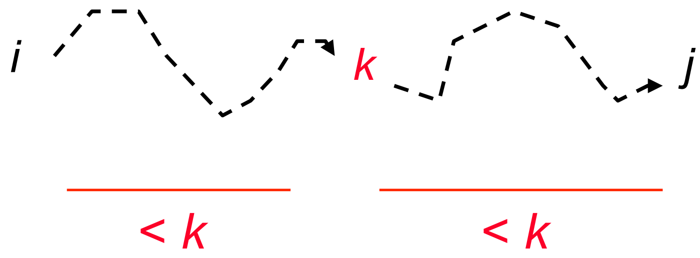


Paths from 2 to 4: (2,3), (3,1), (1,3), (3,4), (4,4)

intermediate nodes: 1, 3, 4

Recurrence

Simple path



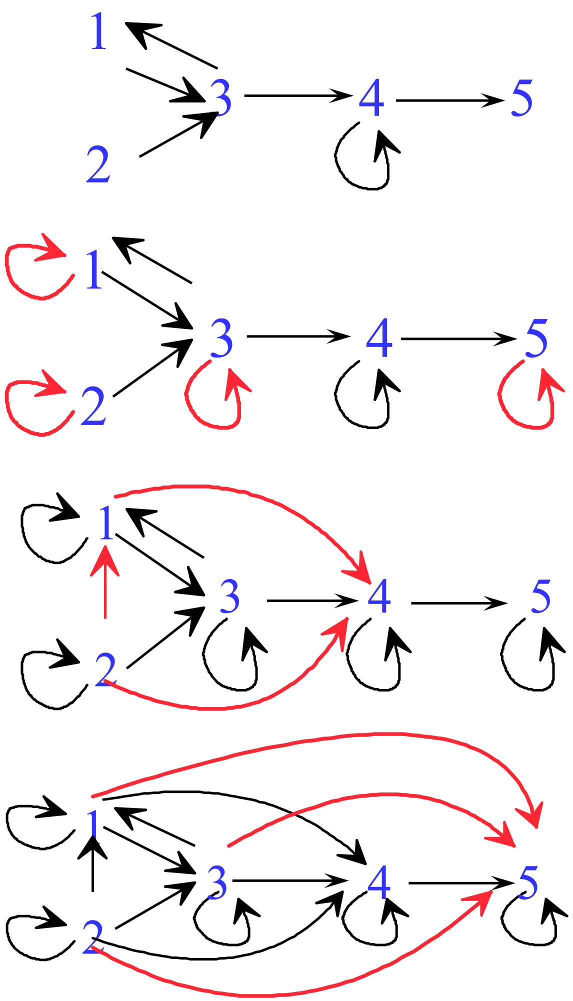
Lemma For all $k \geq 1$,

$$C_k[i,j] = 1 \text{ iff } C_{k-1}[i,j] = 1 \text{ or } (\, C_{k-1}[i,k] = 1 \text{ and } C_{k-1}[k,j] = 1 \,)$$

Computation

of C_k from C_{k-1} in time $O(n^2)$

of $B = C_n$ in time $O(n^3)$



$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$

$$C_0 = C_1 = C_2 = \begin{pmatrix} 1&0&1&0&0 \\ 0&1&1&0&0 \\ 1&0&1&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

$$C_3 = \begin{pmatrix} 1&0&1&1&0 \\ 1&1&1&1&0 \\ 1&0&1&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

$$B = C_4 = C_5 = \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

~1 matrix product

```
function closure (graph G = (S, A)) : matrix ;
begin
  n ← |S| ;
  for i ← 1 to n do
      for j ← 1 to n do
          if i = j or A[i,j] = 1 then
                B[i,j] ← 1 ;
          else
                B[i,j] ← 0 ;
  for k ← 1 to n do
      for i ← 1 to n do
          for j ← 1 to n do
                B[i,j] ← B[i,j] + B[i,k] · B[k,j] ;
  return B ;
end
```

+ is the boolean sum ; running time $O(n^3)$

what we have so far

Three algorithms to compute the transitive closure:

- matrix polynomial: $O(n \cdot M(n)) = O(n^4)$
- matrix power: $O(\log n \cdot M(n)) = O(\log n \cdot n^3)$
- Roy-Warshall algorithm : $O(n^3)$

We now generalize these ideas to compute all-pairs shortest paths in a weighted graph

Distances

$G = (S, A, v)$ weighted graph $S = \{1,2,\dots,n\}$ $v : A \rightarrow \mathbf{R}$.

We assume that there is no negative-cost cycle, but negative-cost edges may be present.

Weight matrix: $W = (w\,[i,j])$ with

$$w\,[i,j] = \begin{cases} 0 & \text{if } i = j \\ v((i,j)) & \text{if } (i, j) \in A \\ \infty & \text{else} \end{cases}$$

Weight of a sequence $c = (\,(s_0,s_1), (s_1,s_2), \dots, (s_{k-1},s_k)\,)$ where $s_i \in S$

$$w(c) = \Sigma\, w\,[\,s_{i-1},s_i\,]$$

Distance from s to t

$$d(s, t) = \min\{\, w(c) \mid c \text{ sequence from } s \text{ to } t \,\}$$

Shortest path from s to t :

path c, if it exists, such that $w(c) = d(s, t)$

First method: matrix product

Let $d^{(m)}(i,j)$ be the minimal value of a path from i to j provided that this path contains at most m edges

$d(i,j)=d^{(n)}(i,j)$

Idea : proceed by induction on m

$$d^{(0)}(i,j) = \begin{cases} 0 \text{ if } i=j \\ \infty \quad \text{else} \end{cases}$$

For m≥1,

$$d^{(m)}(i,j) = \min\left(d^{(m-1)}(i,j),\ \min\{d^{(m-1)}(i,t)+w_{tj} \mid 1\le t\le n\}\right) =$$

$$\min\{d^{(m-1)}(i,t)+w_{tj} \mid 1\le t\le n\}$$

In terms of matrices, we have $D^{(\mathrm{m})}=D^{(\mathrm{m}-1)}\cdot W$ where
min plays the role of + and
+ plays the role of ·

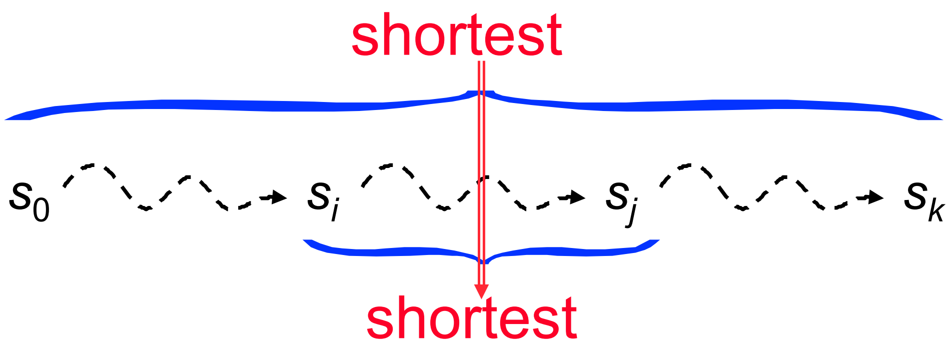
Computing $D=W^n$ by repeated squaring leads to the time complexity $O(n^3 \cdot \log n)$

Algorithm based on intermediate nodes

Basic Lemma (reminder)

$((s_0,s_1), \ldots, (s_i,s_{i+1}), \ldots, (s_{j-1}, s_j), \ldots, (s_{k-1},s_k))$
a shortest path from s_0 to s_k in G

$\Rightarrow ((s_i,s_{i+1}), \ldots, (s_{j-1},s_j))$ a shortest path from s_i to s_j in G



Floyd(-Warshall) algorithm

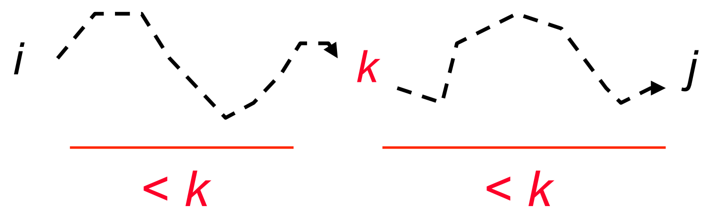
Notation

$D_k \quad = (D_k[i, j] \mid 1 \leq i, j \leq n)$ with

$D_k[i, j] = \min\{ w(c) \mid c$ path from i to j with all intermediate nodes $\leq k \}$

$D_0 \quad = W$

$D_n \quad =$ distance matrix of $G \quad = D$



Lemma For all $k \geq 1$,

$$D_k[i,j] = \min\{ D_{k-1}[i,j] , D_{k-1}[i,k] + D_{k-1}[k,j] \}$$

Computation

of D_k from D_{k-1} in time $O(n^2)$

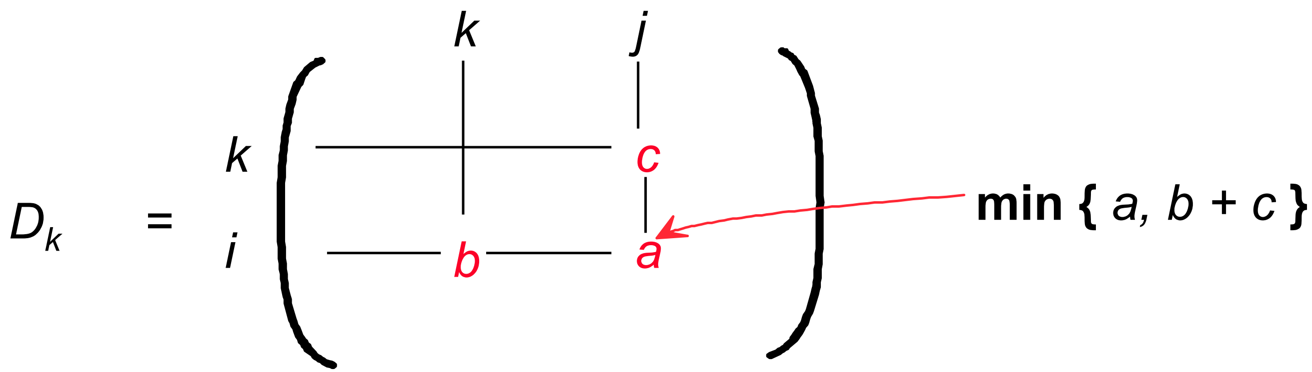
of $D = D_n$ in time $O(n^3)$

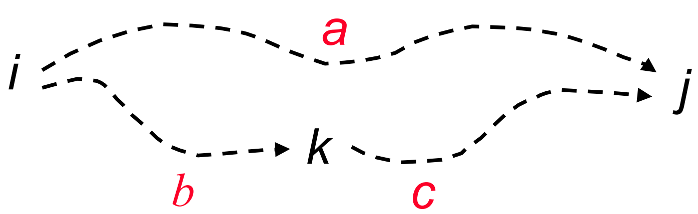
for $k \leftarrow 1$ **to** n **do**

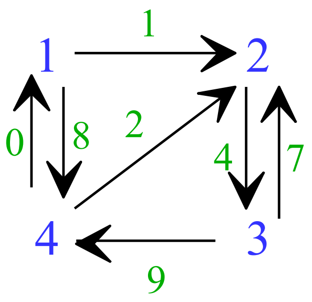
  for $i \leftarrow 1$ **to** n **do**

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

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







$$D_0 = W = \begin{pmatrix} 0 & 1 & \infty & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 2 & \infty & 0 \end{pmatrix}$$

$$D_1 = \begin{pmatrix} 0 & 1 & \infty & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & \infty & 0 \end{pmatrix}$$

$$D_2 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & 13 \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix}$$

$$D_4 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ 13 & 0 & 4 & 13 \\ 9 & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix}$$

Representing shortest paths

Explicitely storing shortest paths from i to j, $1 \le i, j \le n$

n^2 paths of maximal length n-1: space $O(n^3)$

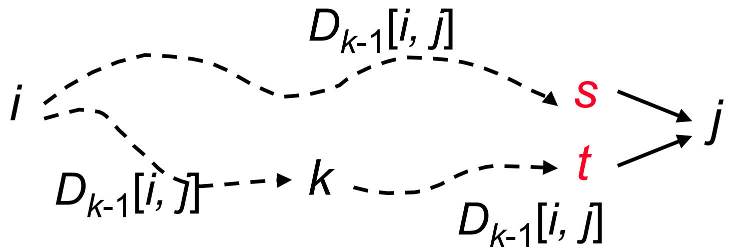
Predecessor matrix: space $\Theta(n^2)$

$\pi_k = (\pi_k[i, j] \mid 1 \le i, j \le n)$ where

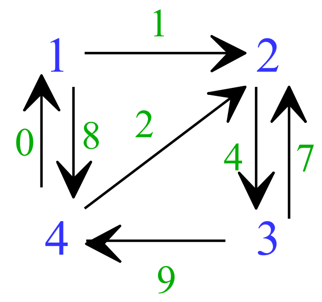
$\pi_k[i, j]$ = predecessor of j on some shortest path from i to j with all intermediate nodes $\le k$

Recurrence

$$\pi_0[i, j] = \begin{cases} i & \text{if } i \ne j \text{ and } (i, j) \in A \\ \text{nil} & \text{else} \end{cases}$$



$$\pi_k[i, j] = \begin{cases} \pi_{k-1}[i, j] & \text{if } D_{k-1}[i, j] \le D_{k-1}[i, k] + D_{k-1}[k, j] \\ \pi_{k-1}[k, j] & \text{else} \end{cases}$$



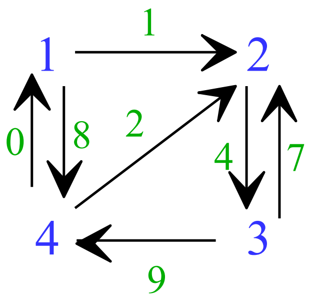
$$D_0 = W = \begin{pmatrix} 0 & 1 & \infty & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 2 & \infty & 0 \end{pmatrix} \qquad P_0 = \begin{pmatrix} - & 1 & - & 1 \\ - & - & 2 & - \\ - & 3 & - & 3 \\ 4 & 4 & - & - \end{pmatrix}$$

$$D_1 = \begin{pmatrix} 0 & 1 & \infty & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & \infty & 0 \end{pmatrix} \qquad P_1 = \begin{pmatrix} - & 1 & - & 1 \\ - & - & 2 & - \\ - & 3 & - & 3 \\ 4 & 1 & - & - \end{pmatrix}$$

$$D_2 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad P_2 = \begin{pmatrix} - & 1 & 2 & 1 \\ - & - & 2 & - \\ - & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & 13 \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad P_3 = \begin{pmatrix} - & 1 & 2 & 1 \\ - & - & 2 & 3 \\ - & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$

$$D_4 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ 13 & 0 & 4 & 13 \\ 9 & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad P_4 = \begin{pmatrix} - & 1 & 2 & 1 \\ 4 & - & 2 & 3 \\ 4 & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$



$$D_4 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ 13 & 0 & 4 & 13 \\ 9 & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad P_4 = \begin{pmatrix} - & 1 & 2 & 1 \\ 4 & - & 2 & 3 \\ 4 & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$

Example of a path

distance from 2 to 1 = $D_4[2,1] = 13$

$P_4[2,1] = 4$; $P_4[2,4] = 3$; $P_4[2,3] = 2$;

$$2 \xrightarrow{4} 3 \xrightarrow{9} 4 \xrightarrow{0} 1$$