Transitive closure of graphs and all-pairs shortest paths

Transitive closure (accessibility)

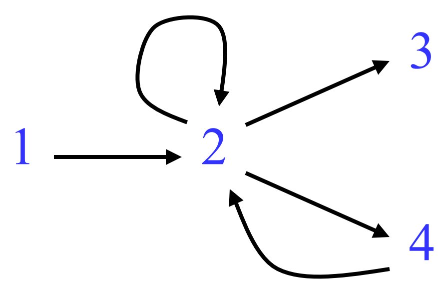
Problem:

$G = (V, E)$ (unweighted) directed graph

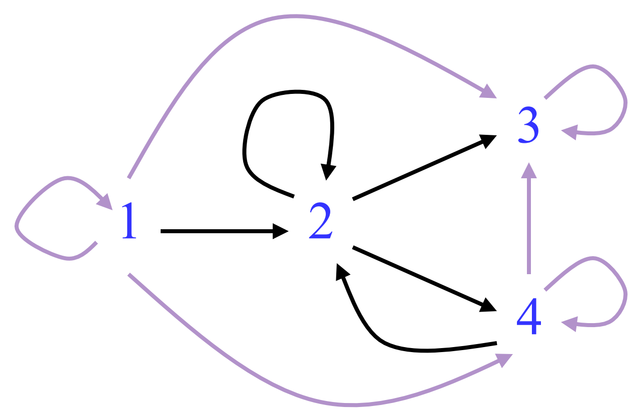
Compute $H = (V, B)$ where B is the reflexive and transitive closure of E

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

graph *G:*



graph *H:*



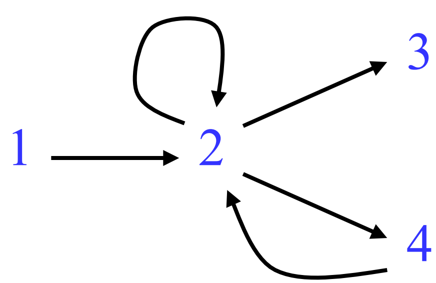
Matrix representation

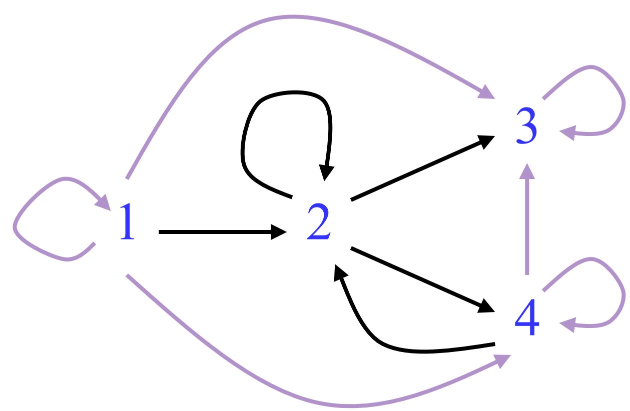
Matrix $n{\times}n$ where $n = |V|$

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

B adjacency matrix of H (= matrix of paths of H)

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





Boolean matrix multiplication

- $A = \left(a_{ij}\right) \in \{0,1\}^{n\times k},\ B = \left(b_{ij}\right) \in \{0,1\}^{k\times m}$
- $AB = C,\ \ C = \{0,1\}^{n\times m}$
- $c_{ij} = (a_{i1} \wedge b_{1j}) \vee \cdots \vee (a_{ik} \wedge b_{kj}) = \bigvee_{l=1}^{k} (a_{il} \wedge b_{lj})$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}.$$

$$\text{AB} = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Closure by matrix multiplication

Notation

A_k = matrix of paths of length k in G

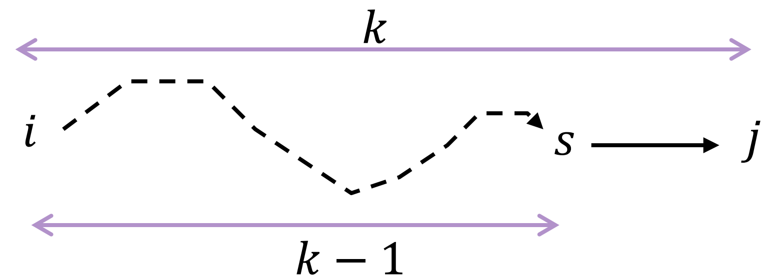
A_0 = I (identity matrix)

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 there exists $s \in V$: $A_{k-1}[i,s] = 1$ and $A[s,j] = 1$

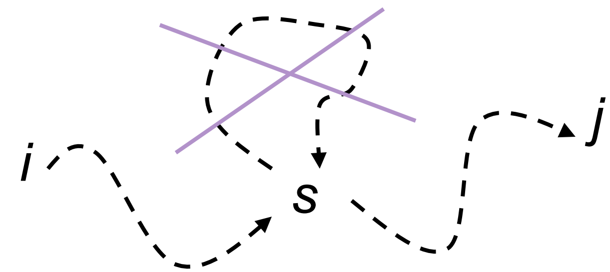
that is, $A_k[i,j] = \bigvee_s (A_{k-1}[i,s] \wedge A[s,j])$

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

then $A_k = A^k$

Closure by matrix multiplication

there exists path from i to j in G ⇔
there exists a path from i to j without cycle (*simple path*) ⇔
there exists a path from i to j of length $\leq n-1$

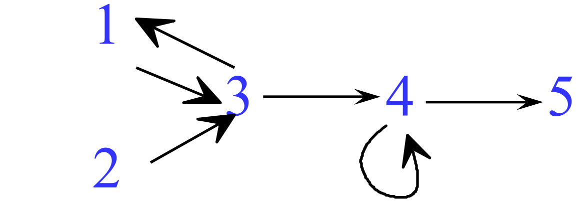


$B[i,j] = 1$ iff $\exists k,\ 0 \leq k \leq n-1,\ A^k[i,j] = 1$

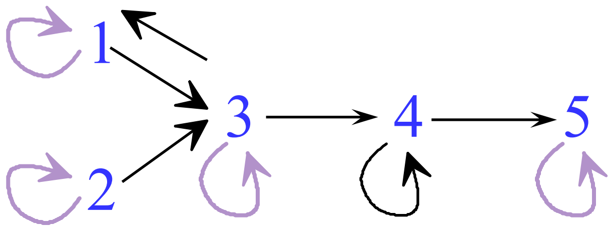
therefore $B = I + A + A^2 + \cdots + A^{n-1}$ where + is ∨

Computation of B using Horner's rule:

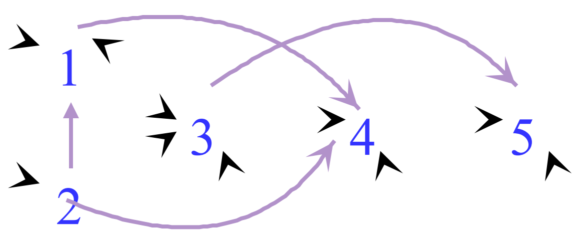
$B_0 = I$,
$B_i = I + B_{i-1}A$ for $i = 1..n-1$. Then $B = B_{n-1}$



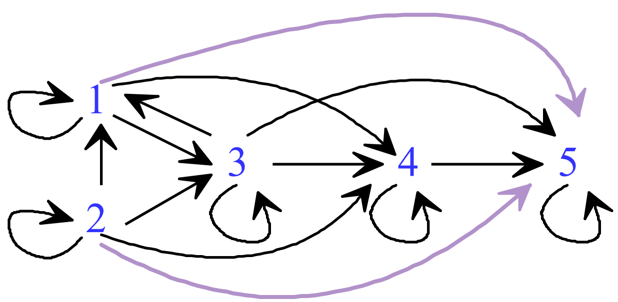
$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$



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



$$\begin{aligned} B_2 &= I + A + A^2 \\ &= I + B_1 \cdot A = \end{aligned} \begin{pmatrix} 1&0&1&1&0 \\ 1&1&1&1&0 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$



$$\begin{aligned} B_3 &= I + A + A^2 + A^3 \\ &= I + B_2 \cdot A = \end{aligned} \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

$$\begin{aligned} B_4 &= I + A + A^2 + A^3 + A^4 \\ &= I + B_3 \cdot A = B \end{aligned}$$

3 matrix products overall

Time complexity

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

each product is done in $O(n^3)$ operations $\Longrightarrow O(n^4)$

there exist matrix multiplication algorithms running in time $o(n^3)$*:*
Strassen 1969: $O(n^{2.8})$ (now improved to $O(n^{2.37})$)

Four russians (Арлазаров, Диниц, Кронрод, Фарадзев) 1970:
$O(n^3/\log^2 n)$ (now improved to $O(n^3/\log^4 n)$)

Time complexity

$n-1$ additions and $n-1$ products of boolean matrices $n{\times}n$
$\Longrightarrow O(n \cdot M(n))$

each product is done in $O(n^3)$ operations $\Longrightarrow O(n^4)$

there exist matrix multiplication algorithms running in time $o(n^3)$*:*
Strassen 1969: $O(n^{2.8})$ (now improved to $O(n^{2.37})$)

Four russians (Арлазаров, Диниц, Кронрод, Фарадзев) 1970:
$O(n^3/\log^2 n)$ (now improved to $O(n^3/\log^4 n)$)

$O(n^4)$ **is too much! can be done better with BFS/DFS:**
For each node i, run BFS with source node i
$B[i,j]=1$ iff j is reachable from i
Running time $O(n \cdot (n+m)) = O(n^3)$

Speeding up

Notation

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

$(B_k = A_0 + \cdots + A_k)$

$B_0 \;=\; I$ (identity matrix)

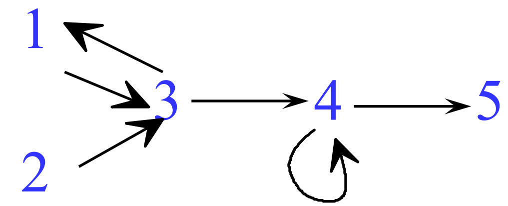
B_1 = matrix of paths of length ≤ 1 $\;=\; I + A$

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

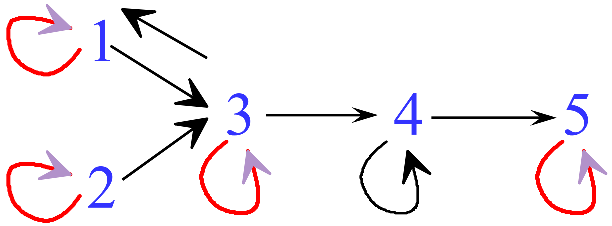
Lemma: $B_k = B_{k-1} \cdot (I + A)$

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

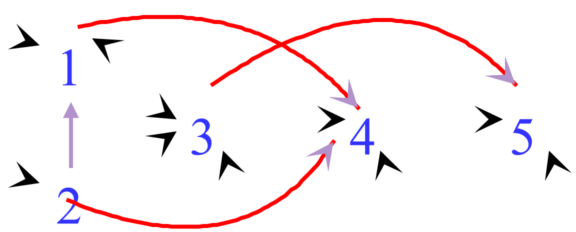
Compute B as an $n - 1$ power in time $O(\log(n) \cdot M(n)) = O(\log(n) \cdot n^3)$



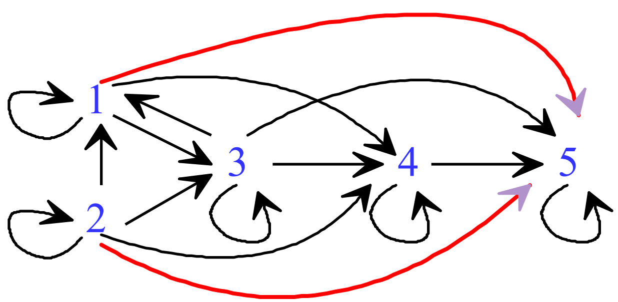
$$A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



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



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



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

2 matrix products

Warshall's (Roy-Warshall) algorithm (~1962)

$G = (V, E)$ with $V = \{1, 2, \ldots, n\}$

Paths in G: $i \to s_1 \to s_2 \cdots \to s_l \to j$

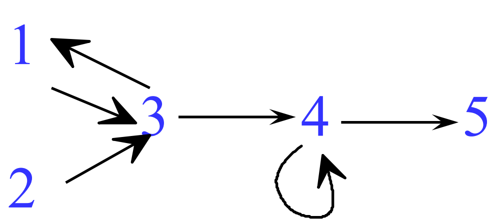
Intermediate nodes: $s_1, s_2, \ldots, s_l$

Notation:

C_k = matrix of paths in G with intermediate nodes $\leq k$

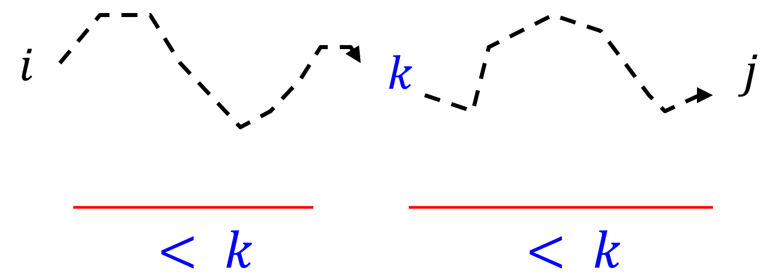
$C_0 = I + A$

C_n = matrix of paths in G $= B$



Recurrence

Simple path



Lemma For all $k \geq 1$,

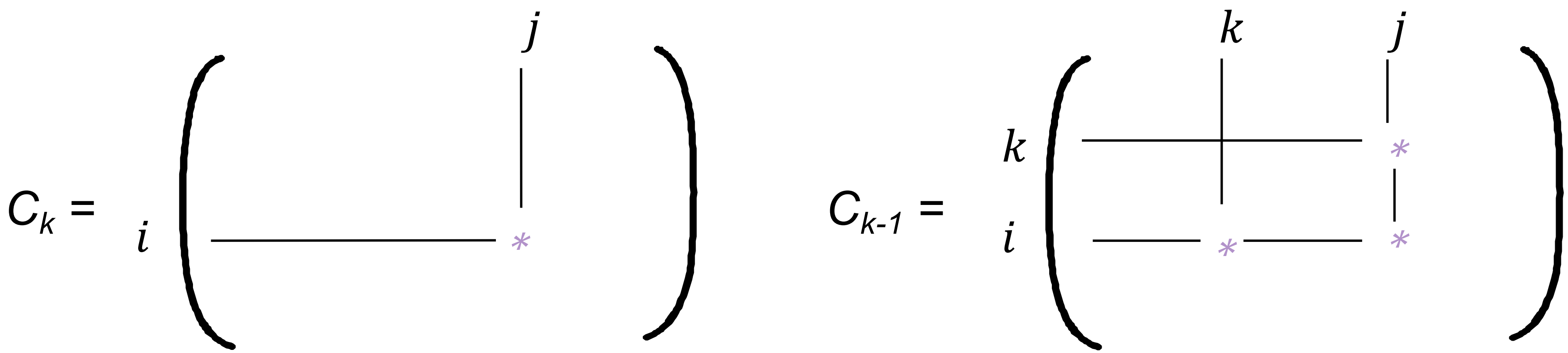
$C_k[i,j] = 1$ iff $C_{k-1}[i,j] = 1$ or $(C_{k-1}[i,k] = 1$ and $C_{k-1}[k,j] = 1)$

Computation

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

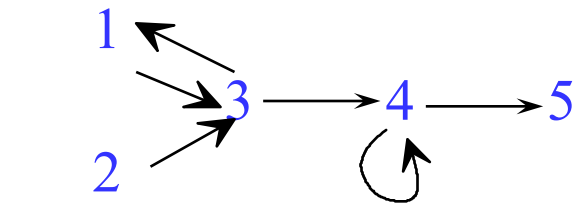
of $B = C_n$ in time $O(n^3)$

Computing C_k from C_{k-1}

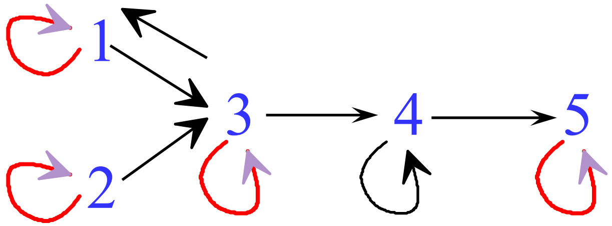


$$C_k[i,j] = C_{k-1}[i,j] \vee (C_{k-1}[i,k] \wedge C_{k-1}[k,j])$$

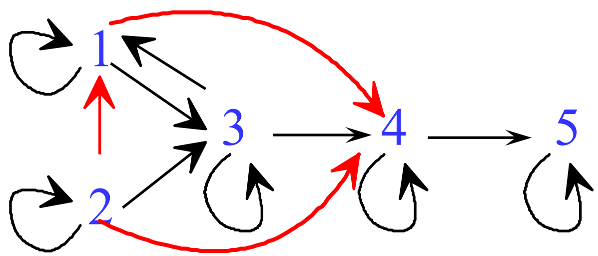
$$C_0 = I + A, \qquad C_k[i,j] = 1 \text{ iff } C_{k-1}[i,j] = 1 \text{ or } (C_{k-1}[i,k] = 1 \text{ and } C_{k-1}[k,j] = 1)$$



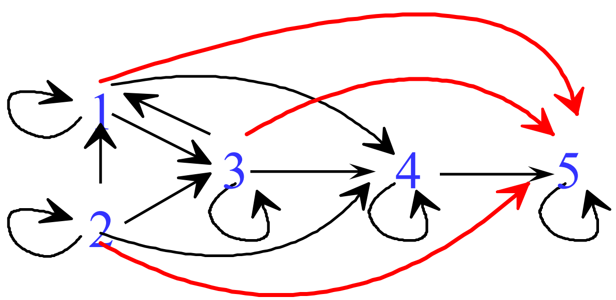
$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$



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

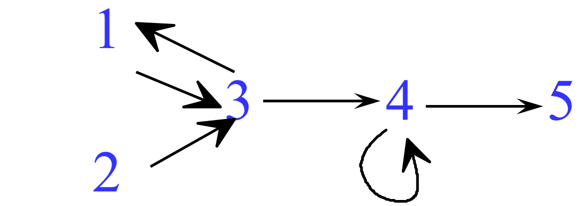


$$C_3 = \begin{pmatrix} 1&0&1&1&0 \\ 1&1&1&1&0 \\ 1&0&1&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

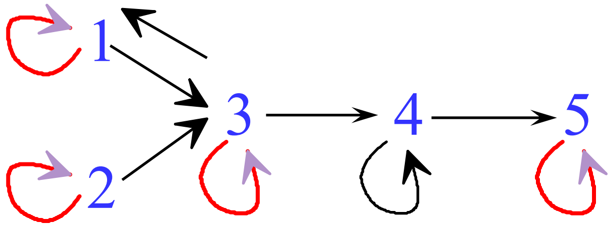


$$B = C_4 = C_5 = \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

$$C_0 = I + A, \qquad C_k[i,j] = 1 \text{ iff } C_{k-1}[i,j] = 1 \text{ or } (C_{k-1}[i,k] = 1 \text{ and } C_{k-1}[k,j] = 1)$$

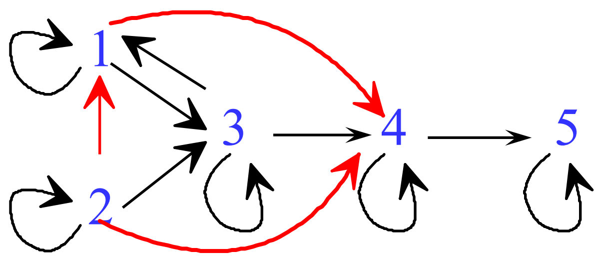


$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$

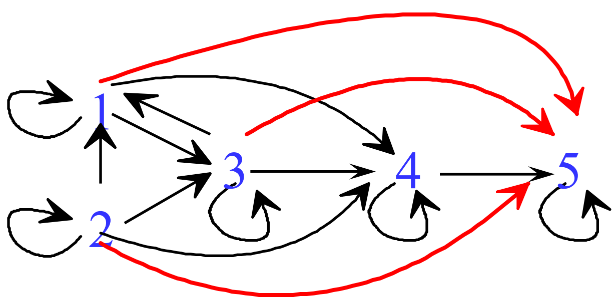


$$C_0 = C_1 = C_2 = \begin{pmatrix} 1&0&1&0&0 \\ 0&1&1&0&0 \\ 1&0&1&1&0 \\ 0&0&\mathbf{0}&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

computing $C_1[4,3]$

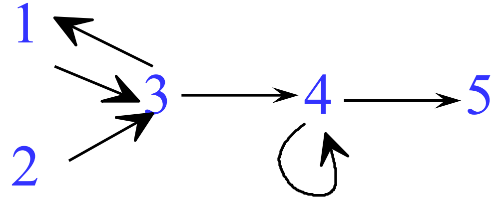


$$C_3 = \begin{pmatrix} 1&0&1&1&0 \\ 1&1&1&1&0 \\ 1&0&1&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

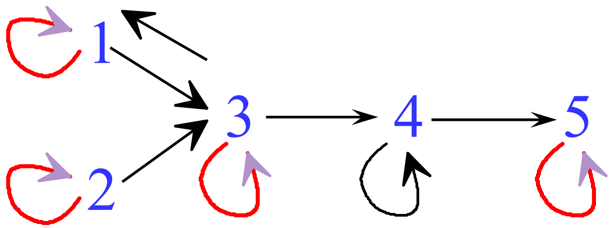


$$B = C_4 = C_5 = \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

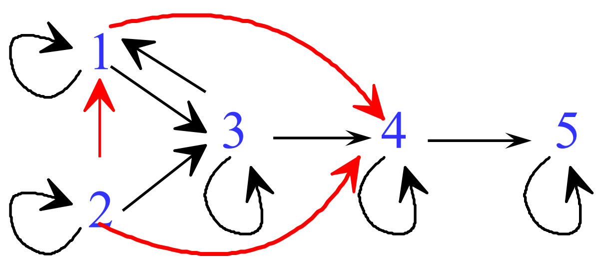
$$C_0 = I + A, \qquad C_k[i,j] = 1 \text{ iff } C_{k-1}[i,j] = 1 \text{ or } (C_{k-1}[i,k] = 1 \text{ and } C_{k-1}[k,j] = 1)$$



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

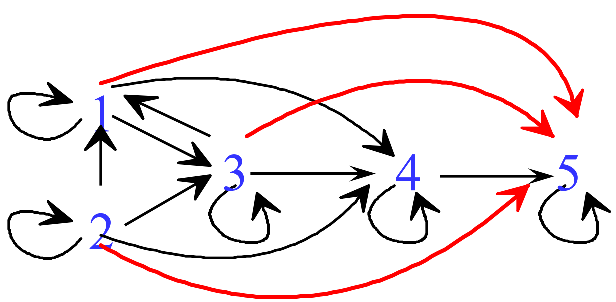


$$C_0 = C_1 = C_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$



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

computing $C_3[2,4]$



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

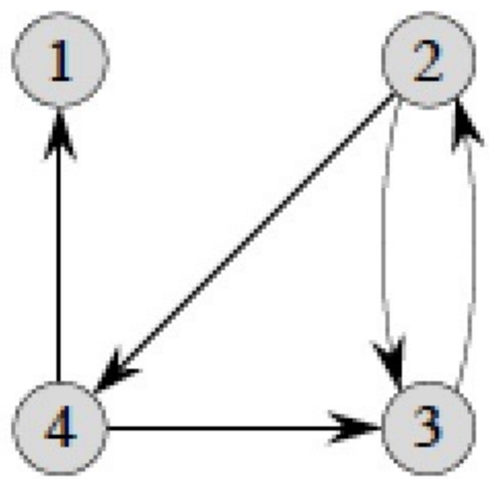
Warshall's algorithm: code

WARSHALL($G = (V, E)$)
$n = |V|$
for $i = 1$ **to** n **do**
  for $j = 1$ **to** n **do**
    if $i = j$ **or** $A[i, j] = 1$ **then**
      $C_0[i, j] = 1$
    else
      $C_0[i, j] = 0$
for $k = 1$ **to** n **do**
  for $i = 1$ **to** n **do**
    for $j = 1$ **to** n **do**
      $C_k[i, j] = C_{k-1}[i, j] \vee (C_{k-1}[i, k] \wedge C_{k-1}[k, j])$
return C_n

running time $O(n^3)$

Quiz 3.1.2

- Compute the transitive closure of the following graph using Warshall's algorithm



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

What about weighted graphs?

$G = (V, E, w)$ weighted graph $V = \{1,2, \dots, n\}$, $w: E \to \mathbb{R}$

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

Weight matrix W defined by

$$W[i,j] = \begin{cases} 0 & \text{if } i = j \\ w(i,j) & \text{if } (i,j) \in E \\ \infty & \text{otherwise} \end{cases}$$

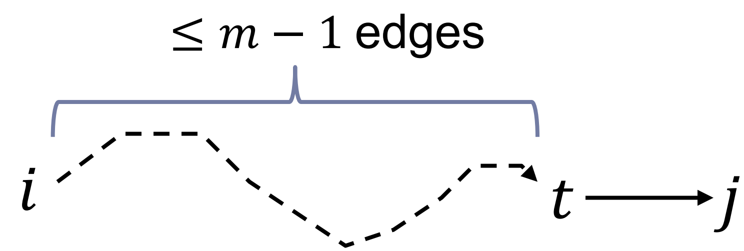
First method: matrix product

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

We have to compute $d[i, j] = d^{(n-1)}[i, j]$

Idea : proceed by induction on m

$$d^{(0)}[i,j] = \begin{cases} 0 & \text{if } i = j \\ \infty & \text{otherwise} \end{cases}$$



For $m \geq 1$,

$$d^{(m)}[i,j] = \min\left(d^{(m-1)}[i,j], \min_{1\leq t\leq n}\{d^{(m-1)}[i,t] + W[t,j]\}\right) = \min_{1\leq t\leq n}\{d^{(m-1)}[i,t] + W[t,j]\}$$

In terms of matrices, we have $D^{(m)} = D^{(m-1)} \cdot W$, where

min plays the role of addition and

$+$ plays the role of multiplication

Computing $D = W^{n-1}$ by repeated squaring leads to the time complexity $O(n^3 \cdot \log n)$

Algorithm based on intermediate nodes: Floyd(-Warshall) algorithm

Notation

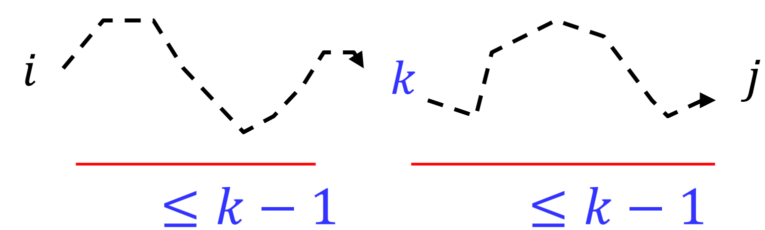
$D_k \;=\; (D_k\,[i,j] \mid 1 \leq i,j \leq\; n\,)$ with

$D_k\,[i,j] \;=$ min{ $w(c) \mid c$ path from i to j with

all intermediate nodes $\leq k$ }

$D_0 \;=\; W$

$D_n \;=$ 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 $O(n^3)$

FLOYD(G,w)

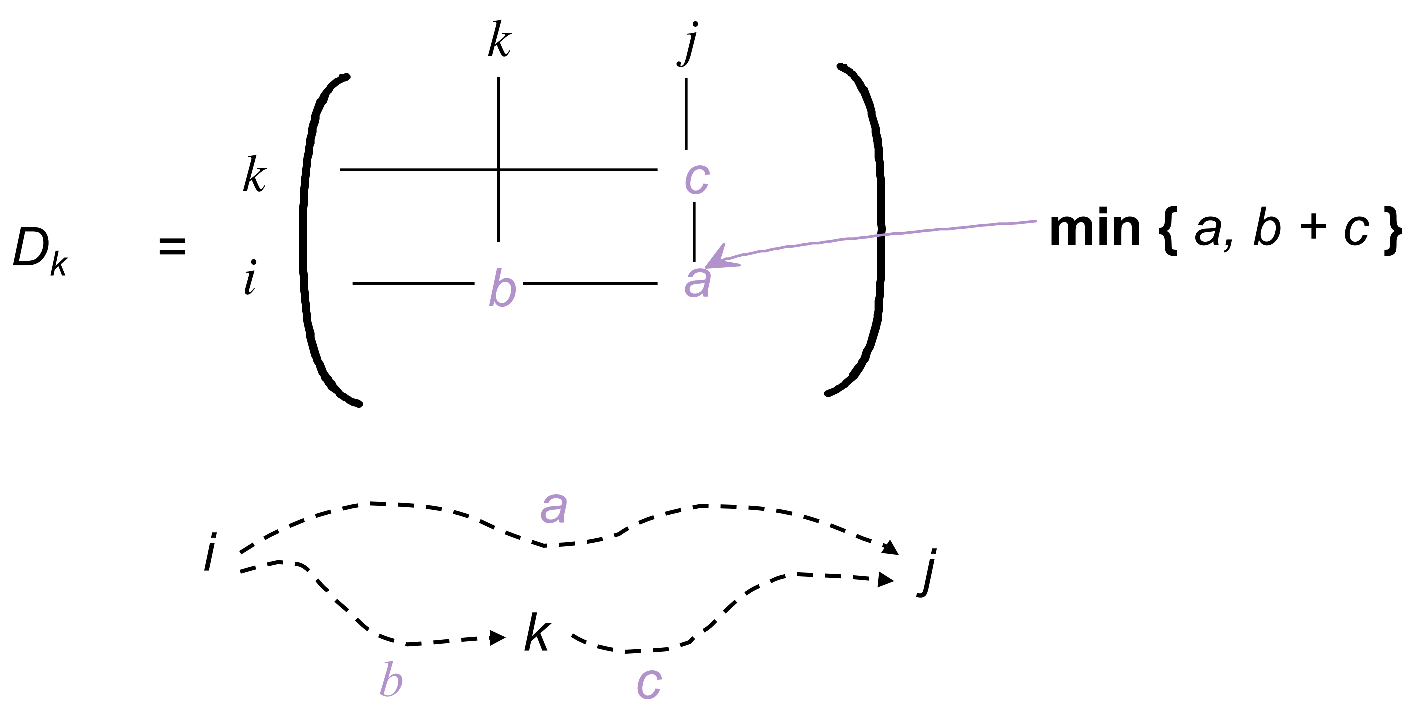
$D_0 = W$

for $k = 1$ **to** n **do**

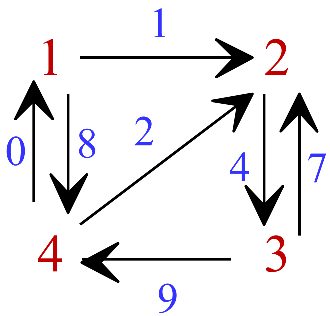
    for $i = 1$ **to** n **do**

        for $j = 1$ **to** n **do**

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



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



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

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

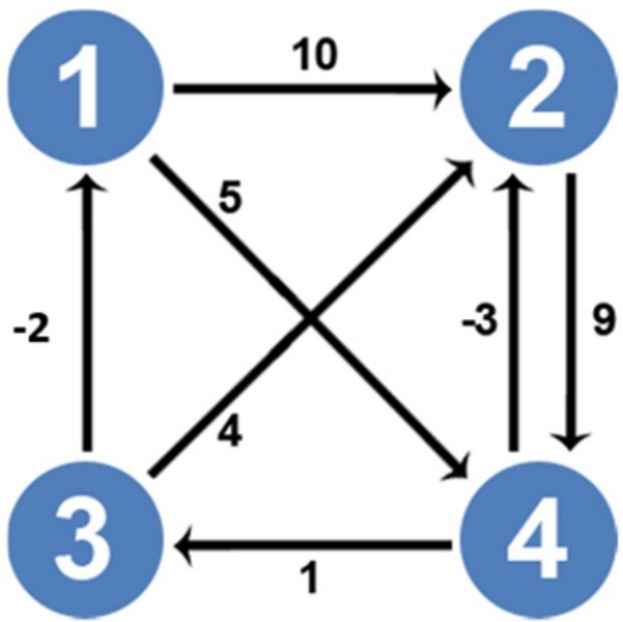
$$D_2 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix}$$

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

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

Quiz 3.1.3

Run Floyd's algorithm to compute all-pairs shortest distances. Output the sum of the shortest distances between all pairs of vertices.



Representing shortest paths

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

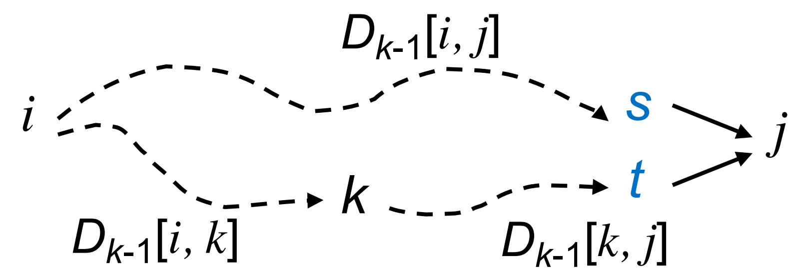
n^2 paths of maximum length $n-1$: space $O(n^3)$

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

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

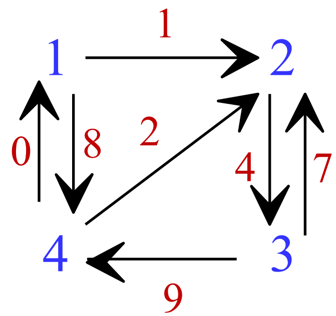
$\pi_k[i,j] =$ predecessor of j on some shortest path from i to j with all intermediate nodes $\leq k$

Recurrence



$$\pi_0[i,j] = \begin{cases} i, \text{ if } i \neq j \text{ and } (i,j) \in E \\ nil \qquad\qquad \text{otherwise} \end{cases}$$

$$\pi_k[i,j] = \begin{cases} \pi_{k-1}[i,j], \text{if } D_{k-1}[i,j] \leq D_{k-1}[i,k] + D_{k-1}[k,j] \\ \pi_{k-1}[k,j] \qquad\qquad \text{otherwise} \end{cases}$$



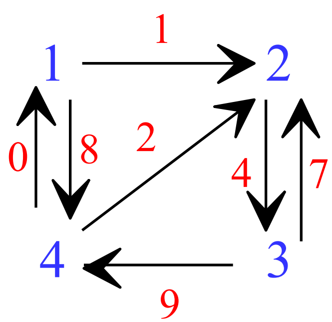
$$D_0 = W = \begin{pmatrix} 0 & 1 & \infty & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 2 & \infty & 0 \end{pmatrix} \qquad \pi_0 = \begin{pmatrix} - & 1 & - & 1 \\ - & - & 2 & - \\ - & 3 & - & 3 \\ 4 & 4 & - & - \end{pmatrix}$$

$$D_1 = \begin{pmatrix} 0 & 1 & \infty & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & \infty & 0 \end{pmatrix} \qquad \pi_1 = \begin{pmatrix} - & 1 & - & 1 \\ - & - & 2 & - \\ - & 3 & - & 3 \\ 4 & 1 & - & - \end{pmatrix}$$

$$D_2 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & \infty \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad \pi_2 = \begin{pmatrix} - & 1 & 2 & 1 \\ - & - & 2 & - \\ - & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$

$$D_3 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ \infty & 0 & 4 & 13 \\ \infty & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad \pi_3 = \begin{pmatrix} - & 1 & 2 & 1 \\ - & - & 2 & 3 \\ - & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$

$$D_4 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ 13 & 0 & 4 & 13 \\ 9 & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad \pi_4 = \begin{pmatrix} - & 1 & 2 & 1 \\ 4 & - & 2 & 3 \\ 4 & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$



$$D_4 = \begin{pmatrix} 0 & 1 & 5 & 8 \\ 13 & 0 & 4 & 13 \\ 9 & 7 & 0 & 9 \\ 0 & 1 & 5 & 0 \end{pmatrix} \qquad \pi_4 = \begin{pmatrix} - & 1 & 2 & 1 \\ 4 & - & 2 & 3 \\ 4 & 3 & - & 3 \\ 4 & 1 & 2 & - \end{pmatrix}$$

Example of a path

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

$\pi_4[2,1] = 4$; $\pi_4[2,4] = 3$; $\pi_4[2,3] = 2$;

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

Remarks

- For sparse graphs represented by adjacency lists there exists Johnson's algorithm that works in time $O(n^2 \cdot \log n + nm)$.

- Warshall's and Floyd-Warshall algorithms are examples of the ***dynamic programming*** technique that we will study later in more details

Shortest paths: summary

Unweighted single-source shortest paths

Breadth-first search $O(|V| + |E|)$

Weighted single-source shortest paths

depending on assumptions:

Dijkstra's algorithm $O(|V|^2)$

or $O(|V| + |E| \cdot \log |V|)$

Bellman-Ford algorithm $O(|E| \cdot |V|)$

All-pairs shortest paths

Floyd-Warshall algorithm $O(|V|^3)$