Transitive closure of graphs and all-pairs shortest paths

Transitive closure (accessibility)

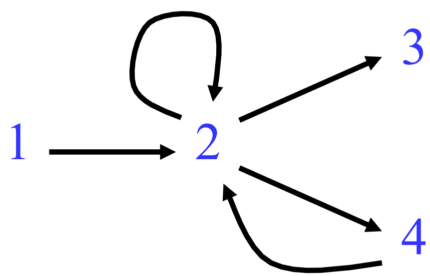
Problem:

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

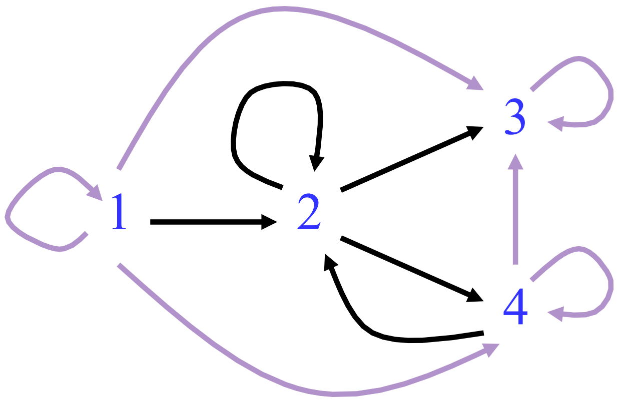
Compute $H = (V, B)$ where B is the reflexive and transitive closure of E.

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

graph *G:*



graph *H:*



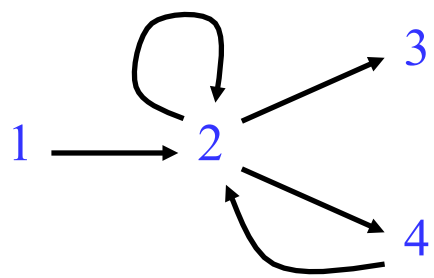
Matrix representation

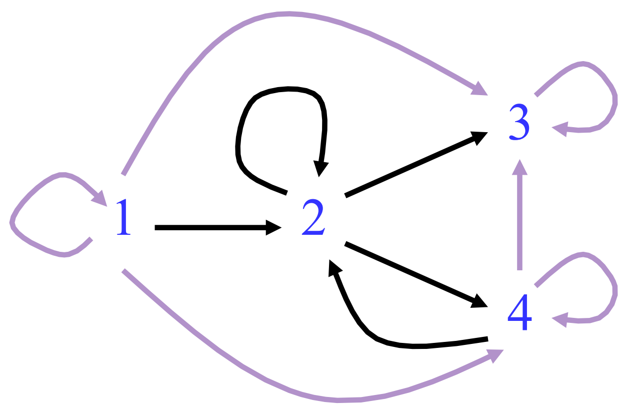
Matrix $n \times n$ where $n = |V|$

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

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

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





Closure by matrix multiplication

Notation

A_k = matrix of paths of length k in G

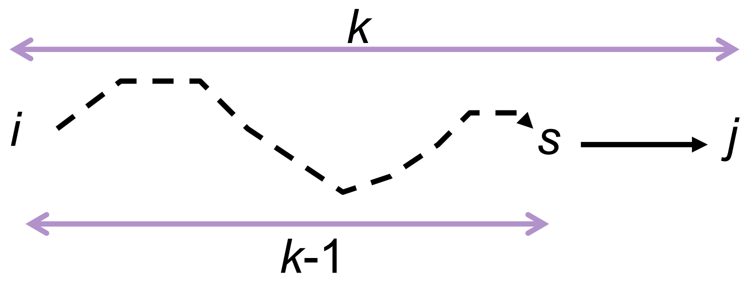
A_0 = I (identity matrix)

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 $A[s,j] = 1$

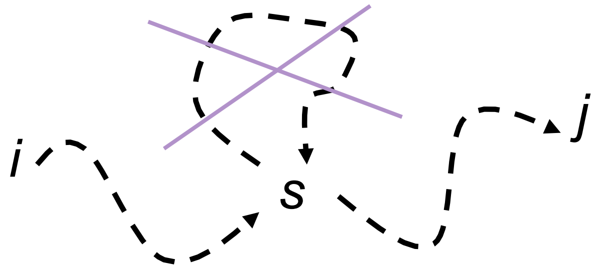
let $A_k[i,j] = \bigvee_s A_{k-1}[i,s] \cdot A[s,j]$ where $\bigvee$ boolean sum (OR).

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 ≤ n-1



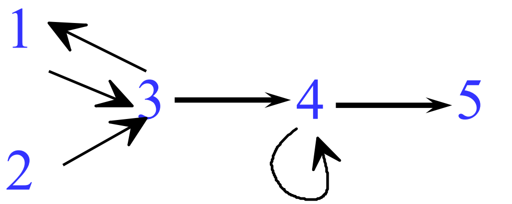
$B[i,j] = 1$ iff $\exists\, k,\ 0 \le k \le n-1\ \ A^k[i,j] = 1$

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

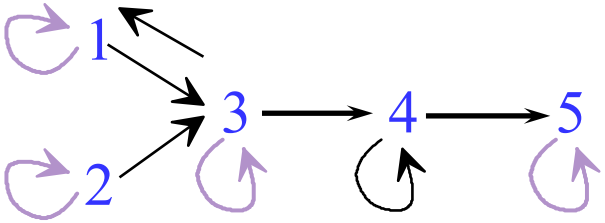
Computation of B using Horner's rule:

$B_0 = I,$

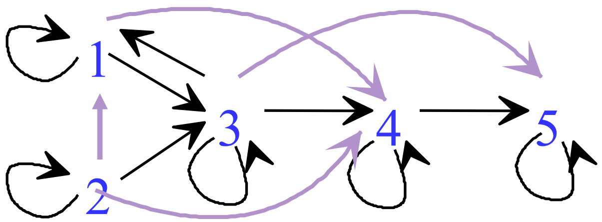
$B_i = I + B_{i-1} A$ for $i=1..n-1$. Then $B= B_{n-1}$



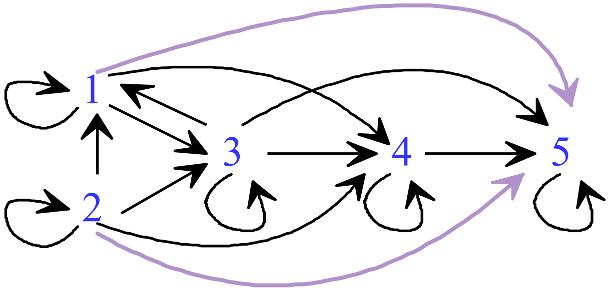
$$A = \begin{pmatrix} 0&0&1&0&0 \\ 0&0&1&0&0 \\ 1&0&0&1&0 \\ 0&0&0&1&1 \\ 0&0&0&0&0 \end{pmatrix}$$



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



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



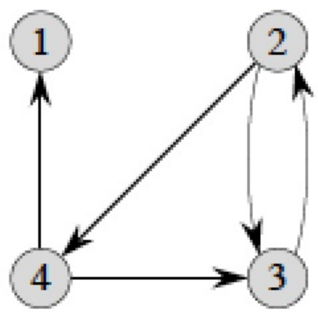
$$B_3 = I + A + A^2 + A^3$$
$$= I + B_2 \cdot A = \begin{pmatrix} 1&0&1&1&1 \\ 1&1&1&1&1 \\ 1&0&1&1&1 \\ 0&0&0&1&1 \\ 0&0&0&0&1 \end{pmatrix}$$

$$B_4 = I + A + A^2 + A^3 + A^4$$
$$= I + B_3 \cdot A = B$$

3 matrix products overall

Exercise

- Compute the transitive closure for the following graph



Time complexity

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: $O(n^{2.8})$ (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

$n\text{-}1$ additions and $n\text{-}1$ 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: $O(n^{2.8})$ (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:**
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

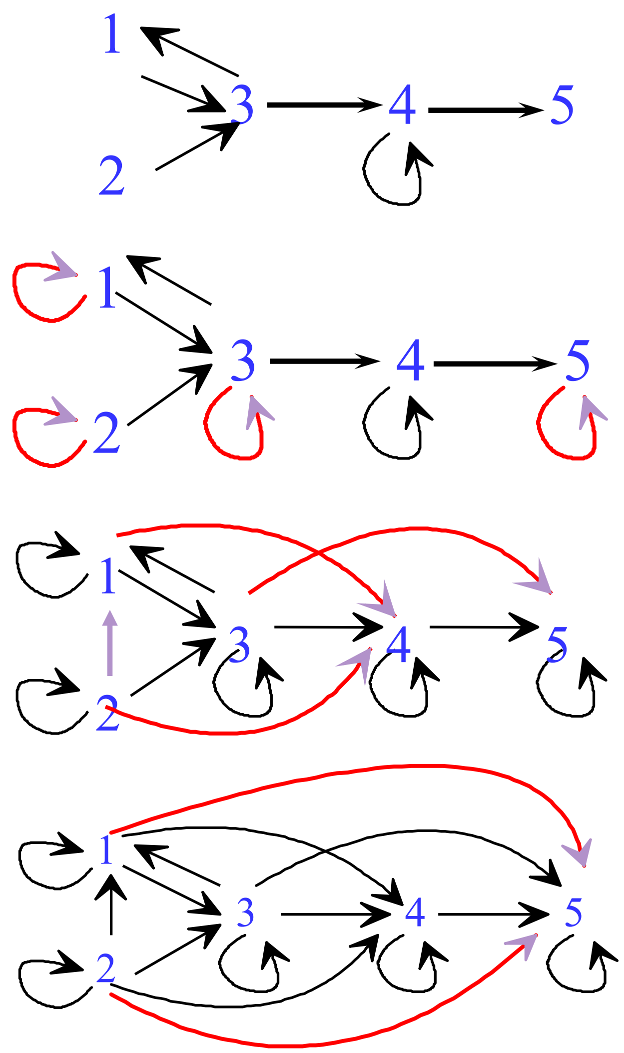
B_k = matrix of paths of length $\leq k$ in G

B_0 = I (identity matrix)
B_1 = matrix of paths of length ≤ 1 = $I + A$
$B_{n\text{-}1}$ = 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 $O(\log(n) \cdot M(n)) = O(\log(n) \cdot n^3)$



2 matrix products

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

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

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

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

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

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

Paths in G : $i \rightarrow s_1 \rightarrow s_2 \ \ldots \rightarrow s_l \rightarrow j$

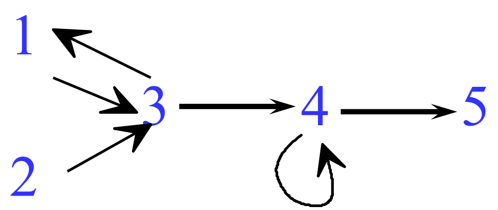
Intermediate nodes : $s_1, s_2, \ldots, s_l$

Notation:

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

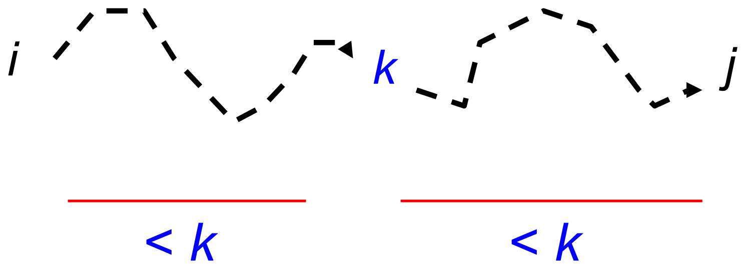
$C_0 = I + A$

C_n = matrix of paths in G = B



Recurrence

Simple path



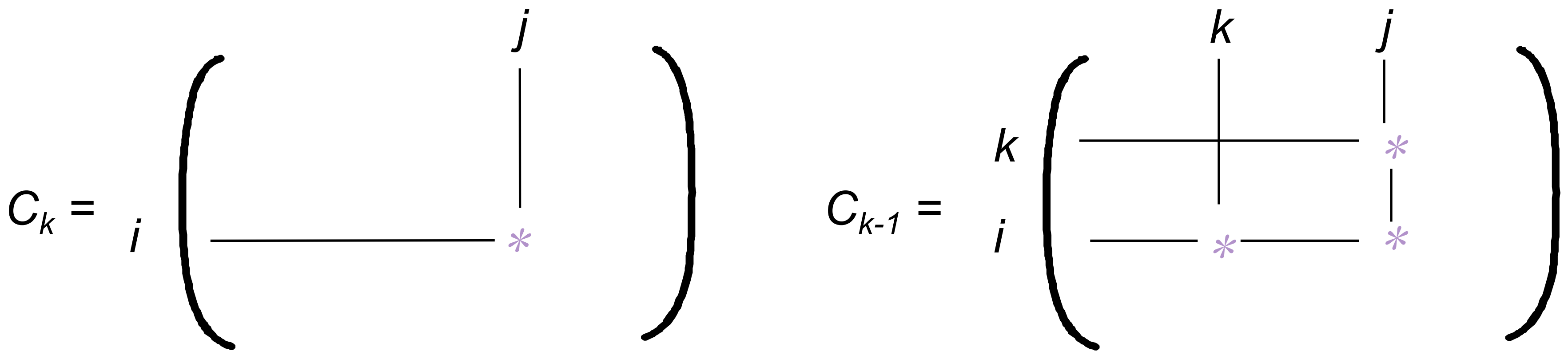
Lemma For all $k \geq 1$,

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

Computation

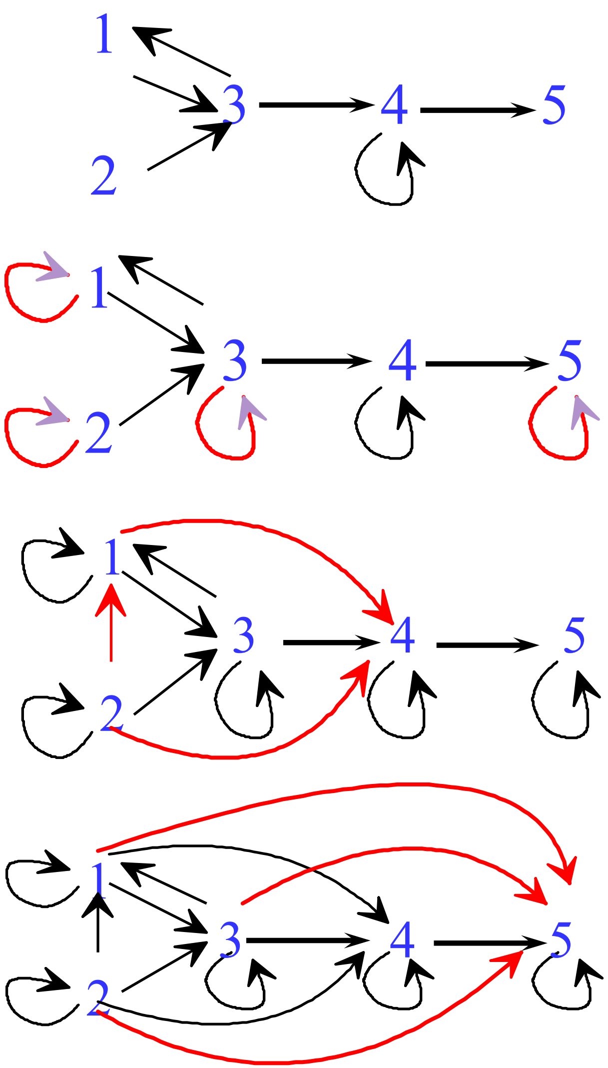
of C_k from $C_{k\text{-}1}$ in time $O(n^2)$
of $B = C_n$ in time $O(n^3)$

Computing C_k from C_{k-1}



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

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



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

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

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

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

~1 matrix product

```
function closure (graph G = (V, E)) : matrix ;
begin
    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] + C_{k-1}[i,k] · C_{k-1}[k,j] ;
    return C_n ;
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