Algorithms
Chapter 25
All-Pairs Shortest Paths

在有向图上算任意2点的最短距

Associate Professor: Ching-Chi Lin
林清池 副教授
chingchi.lin@gmail.com

Department of Computer Science and Engineering
National Taiwan Ocean University

Outline

- **Shortest paths and matrix multiplication**
- The Floyd-Warshall algorithm
- Johnson's algorithm for sparse graphs

用矩陣乘法算任意2点的最短距 ⇒ $O(n^3 \lg n)$

Floyd-Warshall 演算法 ⇒ $O(n^3)$

} 二者都是 dynamic-programing

Overview$_{1/2}$

- **Input:** A weighted directed graph $G = (V, E)$.
- **Output:** An $n \times n$ matrix of shortest-path distances $\delta(u,v)$.
- Could run BELLMAN-FORD once from each vertex:
  - $O(n^2 m)$ — which is $O(n^4)$ if the graph is **dense** ($E = \Theta(n^2)$).

⇒ 用Bellman-Ford
$n \cdot O(nm) = O(n^2 m)$

- If no negative-weight edges, could run Dijkstra's algorithm algorithm once from each vertex:
  - $O(nm\lg n)$ with binary heap — $O(n^3 \lg n)$ if dense.
  - $O(n^2 \lg n + nm)$ with Fibonacci heap — $O(n^3)$ if dense.
- We'll see how to do in $O(n^3)$ in all cases, with no fancy data structure.

不用複雜的資結，直接
有 $O(n^3)$ 方法

用Dijkstra
$n \cdot (m \lg n) = O(nm \lg n)$ - binary
$n \cdot (m + n \lg n) = O(nm + n^2 \lg n)$ - Fibonacci

Overview$_{2/2}$

- **Input:** The adjacency matrix W of a weighted directed graph $G = (V, E)$, where

$$w_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \text{the weight of edge } (i, j) & \text{if } i \neq j \text{ and } (i, j) \in E, \\ \infty & \text{if } i \neq j \text{ and } (i, j) \notin E. \end{cases}$$

  - Negative weights – "allow". Negative-weight cycles – "no".

negative - weight edge : 可

- **Output:** A matrix $D = (d_{ij})$, where $d_{ij} = \delta(i, j)$.

D : 目前最短距

假設台北到高雄最短距離經过台中

台北到高雄最短距 = 台北到台中最短距 + 台中到高雄最短距

Shortest paths and matrix multiplication

- A dynamic programming approach
- **Optimal substructure:** subpaths of shortest paths are shortest paths. 問題的最佳解包含子問題的最佳解
- **Recursive solution:** Let $l_{ij}^{(m)}$ = weight of shortest path from *i* to *j* that contains at most *m* edges.

$l_{ij}^{(m)}$: i 到 j 最多可用 m 边

  - *m* = 0, there is a shortest path from *i* to *j* with no edges if and only if *i* = *j* .

$$l_{ij}^{(0)} = \begin{cases} 0 & \text{if } i = j, \\ \infty & \text{if } i \neq j. \end{cases}$$

$$l_{ij}^{m} = \min \begin{cases} \text{case 1: } i \xrightarrow{m-1} j \\ \text{case 2: } i \xrightarrow{m-1} k \longrightarrow j \end{cases}$$

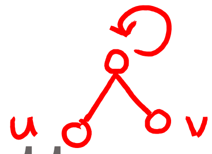
  - $m \geq 1$, $l_{ij}^{(m)} = \min\{l_{ij}^{(m-1)}, \min_{1\leq k\leq n}\{l_{ik}^{(m-1)} + w_{kj}\}\}$

∵ $w_{jj}=0$, case 1 為 case 2 特例

$$= \min_{1\leq k\leq n}\{l_{ik}^{(m-1)} + w_{kj}\} \quad (\text{since } w_{jj} = 0).$$

  - *m* = 1, we have $l_{ij}^{(1)} = w_{ij}$. 只能用一個边, l_{ij}^{1} = 原图

Compute a solution bottom-up$_{1/2}$



path 超过 n-1 边，定会重覆，删除更小
⇒ 最短 path 最多有 n-1 边

- $d_{ij} = \delta(i, j) = l_{ij}^{(n-1)} = l_{ij}^{(n)} = l_{ij}^{(n+1)} = \ldots$

All simple shortest paths contain at most $n - 1$ edges.

- Compute $L^{(1)}, L^{(2)}, \ldots, L^{(n-1)}$, where $L^{(1)} = W$.

先算最多可用1個边，再算最多2個边 …

- Go from $L^{(m-1)}$ to $L^{(m)}$.

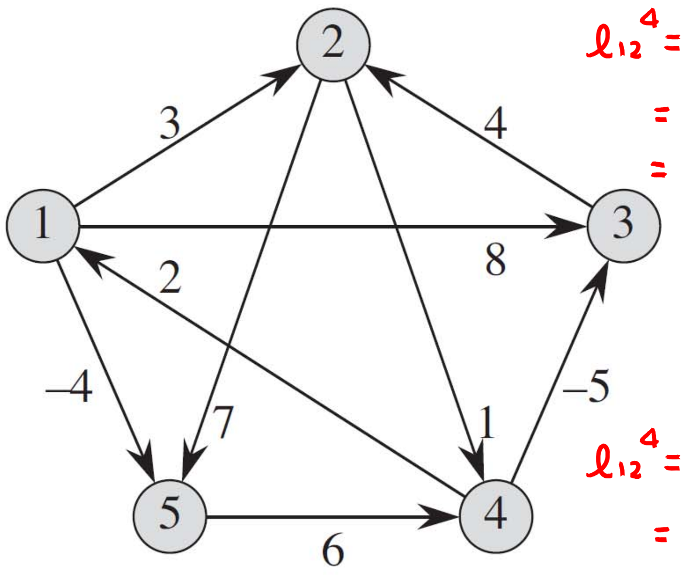
EXTEND-SHORTEST-PATHS(L, W) 用 l_{ij}^{m-1} 算 l_{ij}^{m}

```
n ← rows[L]
let L' = (l'_ij) be an n × n matrix
for i ← 1 to n
    for j ← 1 to n
        l'_ij ← ∞
        for k ← 1 to n
            l'_ij ← min{l'_ij, l_ik + w_kj}
return L'
```

有 n×n 個元素要算，
每一個元素有 n 种可能性
⇒ $O(n^3)$

$i \xrightarrow{m-1} 1 \longrightarrow j$
$i \xrightarrow{m-1} 2 \longrightarrow j$
$\vdots$
$i \xrightarrow{m-1} n \longrightarrow j$

- L for $L^{(m-1)}$ and L' for $L^{(m)}$.
- Time: $\Theta(n^3)$.



$$l_{12}^{4} = \min\{l_{11}^{2} + l_{12}^{2},\ l_{12}^{2} + l_{22}^{2},\ l_{13}^{2} + l_{32}^{2},\ l_{14}^{2} + l_{42}^{2},\ l_{15}^{2} + l_{52}^{2}\}$$
$$= \min\{0+3,\ 3+0,\ 8+4,\ 2-1,\ -4+\infty\}$$
$$= \min\{3,\ 3,\ 12,\ 1,\ \infty\}$$

$$l_{12}^{4} = \min\{l_{11}^{3} + w_{12},\ l_{12}^{3} + w_{22},\ l_{13}^{3} + w_{32},\ l_{14}^{3} + w_{42},\ l_{15}^{3} + w_{52}\}$$
$$= \min\{0+3,\ 3+0,\ -3+4,\ 2+\infty,\ -4+\infty\}$$
$$= \min\{3,\ 3,\ 1,\ \infty,\ \infty\}$$

$$L^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad L^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 2 & -4 \\ 3 & 0 & -4 & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & \infty & 1 & 6 & 0 \end{pmatrix}$$

$$L^{(3)} = \begin{pmatrix} 0 & 3 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad L^{(4)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix}$$

Compute a solution bottom-up$_{2/2}$

- **Observation:** Extend is like matrix multiplication: $C = A \cdot B$.
  - $L \to A$, $W \to B$, $L' \to C$ 和矩陣乘法很像
  - $\min \to +$, $+ \to *$, $\infty \to 0$

Square-Matrix-Multiply(A, B)

```
n ← rows[A]
let C be an n × n matrix
for i ← 1 to n
    for j ← 1 to n
        c_ij ← 0
        for k ← 1 to n
            c_ij ← c_ij + a_ik * b_kj
return C
```

$\boxed{c_{ij} \leftarrow 0}$, $\boxed{c_{ij} \leftarrow c_{ij} + a_{ik} * b_{kj}}$

Extend-Shortest-Paths(L, W)

```
n ← rows[L]
let L' = (l'_ij) be an n × n matrix
for i ← 1 to n
    for j ← 1 to n
        l'_ij ← ∞
        for k ← 1 to n
            l'_ij ← min{l'_ij, l_ik + w_kj}
return L'
```

- $L^{(1)} = L^{(0)} \cdot W = W$, $L^{(2)} = L^{(1)} \cdot W = W^{(2)}$, $L^{(3)} = L^{(2)} \cdot W = W^{(3)}$
  $L^{(n-1)} = L^{(n-2)} \cdot W = W^{(n-1)}$.
- Call Extend $n-1$ times, $D = W^{(n-1)}$ can be computed in $\Theta(n^4)$ time.

1次 w^2

2次 $w^4 = w^2 \cdot w^2$

3次 $w^8 = w^4 \cdot w^4$

$\lceil \lg(n-1) \rceil$次 $= w^{2^{\lceil \lg(n-1) \rceil}} \geq w^{n-1}$

Improving the running time

最多算 $\lceil \lg(n-1) \rceil$次

直接算 $w^2, w^4, w^8, \ldots w^{n-1}$

- **Goal:** to compute $W^{(n-1)}$.
  - Don't need to compute **all** the intermediate $W^{(1)}, W^{(2)}, \ldots, W^{(n-2)}$.
  - Could compute $W^2 = W \cdot W$, $\quad W^{(4)} = W^{(2)} \cdot W^{(2)}$,
    $W^{(8)} = W^{(4)} \cdot W^{(4)}, \ldots, W^{2^{\lceil \lg(n-1) \rceil}} = W^{2^{\lceil \lg(n-1) \rceil - 1}} \cdot W^{2^{\lceil \lg(n-1) \rceil - 1}}$
  - Only $\lceil \lg(n-1) \rceil$ matrix products is computed.
  - Since $2^{\lceil \lg(n-1) \rceil} \geq n-1$, the final product is equal to $W^{(n-1)}$.

FASTER-ALL-PAIRS-SHORTEST-PATHS(W) **Time :** $\Theta(n^3 \lg n)$

```
n ← rows[W]
L^(1) ← W
m ← 1
while m < n − 1
      do L^(2m) = EXTEND-SHORTEST-PATHS(L^(m), L^(m))
            m ← 2m
return L^(m)
```

Lines 1–3: $O(1)$

Lines 4–6: $\lceil \lg(n-1) \rceil \cdot \Theta(n^3)$

一次所需時間

Outline

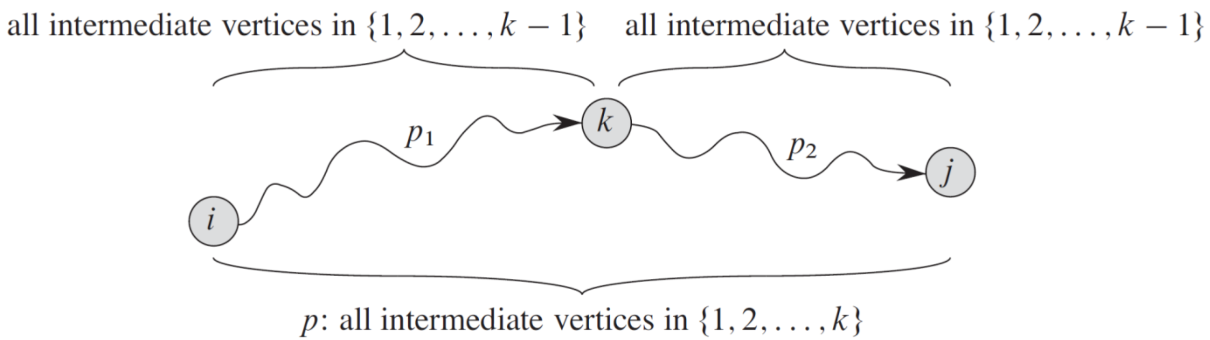
- Shortest paths and matrix multiplication } 二者都是 dynamic-programing
- **The Floyd-Warshall algorithm**
- Johnson's algorithm for sparse graphs

$d_{ij}^{k} = \min \begin{cases} \text{case 1: 不經过 } k \Rightarrow d_{ij}^{k-1} \\ \text{case 2: 經过 } k \Rightarrow i \longrightarrow k \longrightarrow j = d_{ik}^{k-1} + d_{kj}^{k-1} \end{cases}$

Floyd-Warshall algorithm

- A different dynamic-programming approach
- Let $d_{ij}^{(k)}$ be the weight of a shortest path from i to j with all intermediate vertices in $\{1, 2, . . . , k\}$. d_{ij}^{k}：i到j只能經过編号為 1~k 的点
- Consider a shortest path p from i to j with all intermediate vertices in $\{1, 2, . . . , k\}$: π_{ij}^{k}：i到j只能經过編号為 1~k 的点，j 的前一個点
  - If k is not an intermediate vertex, then all intermediate vertices of p are in $\{1, 2, . . . , k-1\}$.
  - If k is an intermediate vertex, then

$\pi_{ij}^{k} = \begin{cases} \text{case 1: 不經过 } k \Rightarrow \pi_{ij}^{k-1} \\ \text{case 2: 經过 } k \Rightarrow \pi_{kj}^{k-1} \end{cases}$



$d_{ij}^{k} = \min \begin{cases} \text{case 1: 不經过 } k \Rightarrow d_{ij}^{k-1} \\ \text{case 2: 經过 } k \Rightarrow (i) \rightarrow (k) \rightarrow (j) = d_{ik}^{k-1} + d_{kj}^{k-1} \end{cases}$

Recursive formulation

- A recursive solution:

$$d_{ij}^{(k)} = \begin{cases} w_{ij} & \text{if } k = 0, \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}) & \text{if } k \geq 1. \end{cases}$$

- $d_{ij}^{(0)} = w_{ij}$, because have no intermediate vertices.
  - Such a path has ≤ 1 edge. $k=0$，不能經过其他点
    $d_{ij}^{(0)} = w_{ij}$
- Goal: $D^{(n)} = (d_{ij}^{(n)})$
  - Because for any path, all intermediate vertices are in the set {1, 2,..., *n*}. 目標：算 d_{ij}^{n}

Compute bottom up

- Compute the values $d_{ij}^{(k)}$ in order of increasing values of k.

FLOYD-WARSHALL(W)

$n \leftarrow rows[W]$
$D^{(0)} \leftarrow W$
$O(1)$

for $k \leftarrow 1$ **to** n
  for $i \leftarrow 1$ **to** n
    for $j \leftarrow 1$ **to** n
      $d_{ij}^{(k)} = \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)})$
$\Theta(n^3)$

三個 for loop $1 \le i, j, k \le n$
$\Rightarrow O(n^3)$

return $D(n)$

- Time: $\Theta(n^3)$.

先算只能經过編号為{1}的点
再算 〃 {1, 2}的点
再算 〃 {1, 2, 3}的点
⋮ ⋮ ⋮

π_{ij}^{k}：i到j只能經过編号為1~k的点，j的前一個点

Constructing a shortest path

- $\pi_{ij}^{(k)}$ is the predecessor of vertex j on a shortest path from vertex j with all intermediate vertices in the set $\{1,2,\cdots,k\}$.

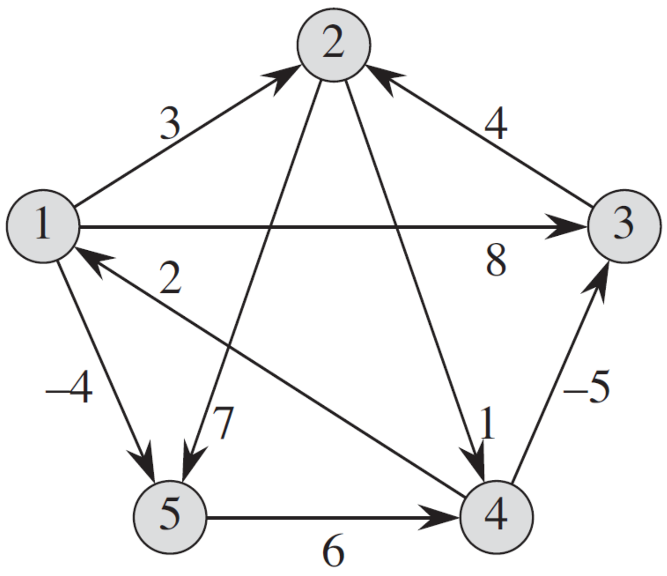
$$\pi_{ij}^{(0)} = \begin{cases} \text{NIL} & \text{if } i = j, \text{or } w_{ij} = \infty, \\ i & \text{if } i \neq j, \text{and } w_{ij} < \infty. \end{cases}$$

i和j沒有边
i和j有边

- For $k \geq 1$, we have

$$\pi_{ij}^{(k)} = \begin{cases} \pi_{ij}^{(k-1)} & \text{if } d_{ij}^{(k-1)} \leq d_{ik}^{(k-1)} + d_{kj}^{(k-1)}, \\ \pi_{kj}^{(k-1)} & \text{if } d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}. \end{cases}$$

$\pi_{ij}^{k} = \begin{cases} \text{case 1: 不經过 } k \Rightarrow \pi_{ij}^{k-1} \\ \text{case 2: 經过 } k \Rightarrow \pi_{kj}^{k-1} \end{cases}$



$d_{42}^{1} = \min\{$經过 1 , 不經过 1$\}$

$= \min\{d_{41}^{0} + d_{12}^{0}, d_{42}^{0}\}$

$= \min\{2 + 3, \infty\}$

$= 5$

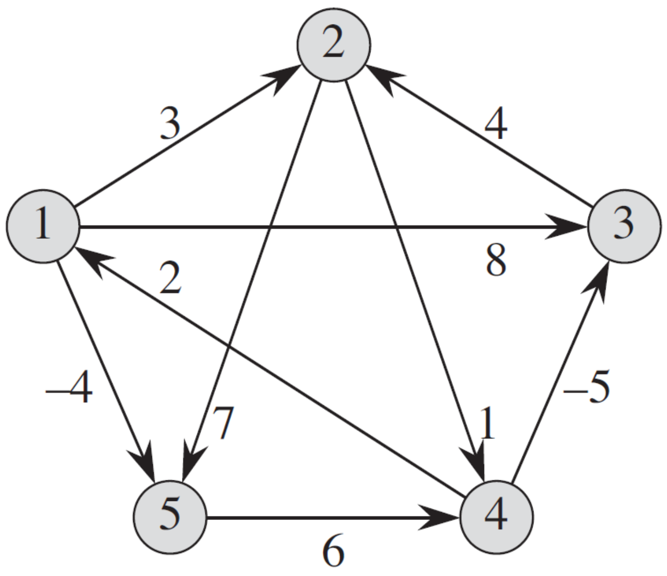
$\pi_{42}^{1} =$ 經过 1 $\Rightarrow \pi_{12}^{0} \Rightarrow 1$

不經过 1 $\Rightarrow \pi_{42}^{0} \Rightarrow$ NIL

有經过 $\Rightarrow \pi_{42}^{1} = 1$

$$D^{(0)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & \infty & -5 & 0 & \infty \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(0)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & \text{NIL} & 4 & \text{NIL} & \text{NIL} \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(1)} = \begin{pmatrix} 0 & 3 & 8 & \infty & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & \infty & \infty \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(1)} = \begin{pmatrix} \text{NIL} & 1 & 1 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & \text{NIL} & \text{NIL} \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$



$$d_{42}^{3} = \min\{\text{經过}3,\ \text{不經过}3\}$$
$$= \min\{d_{43}^{2} + d_{32}^{2},\ d_{42}^{2}\}$$
$$= \min\{-5+4,\ 5\}$$
$$= -1$$

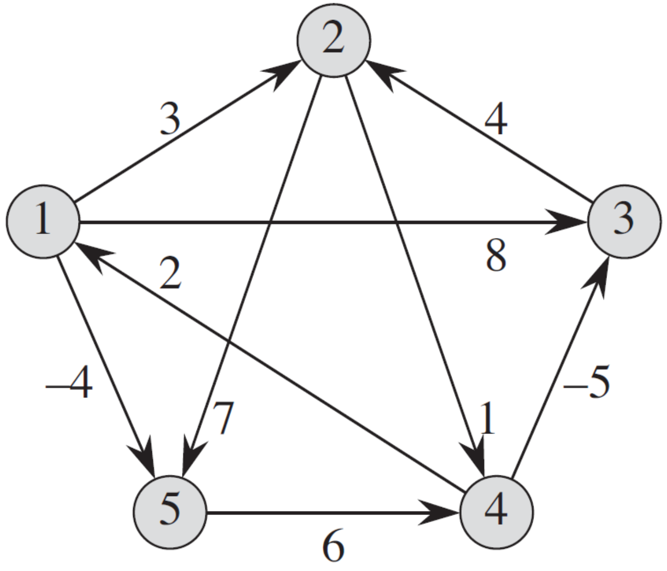
π_{42}^{3} = 經过 3 ⇒ π_{32}^{2} ⇒ 3

不經过3 ⇒ π_{42}^{2} ⇒ 1

有經过 ⇒ $\pi_{42}^{3} = 3$

$$D^{(2)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & 5 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(2)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 1 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(3)} = \begin{pmatrix} 0 & 3 & 8 & 4 & -4 \\ \infty & 0 & \infty & 1 & 7 \\ \infty & 4 & 0 & 5 & 11 \\ 2 & -1 & -5 & 0 & -2 \\ \infty & \infty & \infty & 6 & 0 \end{pmatrix} \qquad \Pi^{(3)} = \begin{pmatrix} \text{NIL} & 1 & 1 & 2 & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 2 & 2 \\ \text{NIL} & 3 & \text{NIL} & 2 & 2 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ \text{NIL} & \text{NIL} & \text{NIL} & 5 & \text{NIL} \end{pmatrix}$$



$$D^{(4)} = \begin{pmatrix} 0 & 3 & -1 & 4 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(4)} = \begin{pmatrix} \text{NIL} & 1 & 4 & 2 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

$$D^{(5)} = \begin{pmatrix} 0 & 1 & -3 & 2 & -4 \\ 3 & 0 & -4 & 1 & -1 \\ 7 & 4 & 0 & 5 & 3 \\ 2 & -1 & -5 & 0 & -2 \\ 8 & 5 & 1 & 6 & 0 \end{pmatrix} \qquad \Pi^{(5)} = \begin{pmatrix} \text{NIL} & 3 & 4 & 5 & 1 \\ 4 & \text{NIL} & 4 & 2 & 1 \\ 4 & 3 & \text{NIL} & 2 & 1 \\ 4 & 3 & 4 & \text{NIL} & 1 \\ 4 & 3 & 4 & 5 & \text{NIL} \end{pmatrix}$$

遞移性閉包：表示可達性

：① Floyd-Warshall 演算法

② 使用性質來節省時間和空間 ⇒ $O(n^3)$

Transitive closure$_{1/2}$

- The transitive closure of G is defined as the graph $G^* = (V, E^*)$, where (i,j)有邊 ⇔ 有 path 可到 j

  $E^* = \{(i, j)$: there is a path from vertex i to vertex j in $G\}$.

- We give two methods to compute the transitive closure of a graph in the following, both in $\Theta(n^3)$ time.

- **Method 1:**
  - Assign a weight of 1 to each edge, then run FLOYD-WARSHALL.
  - If there is a path from vertex i to vertex j, we get $d_{ij} < n$.
  - Otherwise, we get $d_{ij} = \infty$.

法 1：① 將 edge 上的 weight 設為 1

② i 有 path 到 j ⇔ $d_{ij} < n$

③ $d_{ij} < n$ ⇔ $t_{ij} = 1$

Transitive closure$_{2/2}$

只要知道有沒有 path，因此
法 2：使用性質來節省時間和空間

- **Method 2:**
  - Save time and space in practice. t_{ij} 只要存 0 或 1，較節省空間
  - Substitute other values and operators in FLOYD-WARSHALL.
    - min $\rightarrow$ $\vee$ (OR)
    - $+$ $\rightarrow$ $\wedge$ (AND)

    $\vee$ 和 $\wedge$，較 min 和 + 來的快

- $$t_{ij}^{(k)} = \begin{cases} 1 & \text{if there exists a path from } i \text{ to } j \text{ with all intermediate} \\ & \text{vertices in } \{1, 2, \ldots, k\}, \\ 0 & \text{otherwise.} \end{cases}$$

- $$t_{ji}^{(0)} = \begin{cases} 0 & \text{if } i \neq j \text{ and } (i, j) \notin E, \\ 1 & \text{if } i = j \text{ or } (i, j) \in E. \end{cases}$$

- $$t_{ij}^{(k)} = t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{ki}^{(k-1)}).$$

  $i \xrightarrow{1\sim k} j$ = $i \xrightarrow{1\sim k-1} j$ 或 ($i \xrightarrow{1\sim k-1} k$ 且 $k \xrightarrow{1\sim k-1} j$)

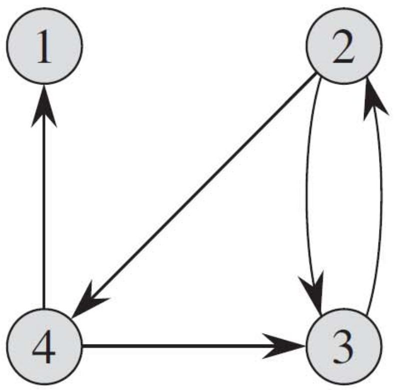
Compute bottom up

- Compute the values $t_{ij}^{(k)}$ in order of increasing values of k.

TRANSITIVE-CLOSURE(G)

$n \leftarrow |V[G]|$ — $O(1)$
for $i \leftarrow 1$ **to** n
    for $j \leftarrow 1$ **to** n
        if $i = j$ or $(i, j) \in E[G]$
            $t_{ij}^{(0)} \leftarrow 1$
        else $t_{ij}^{(0)} \leftarrow 0$ — $\Theta(n^2)$ 初始化
for $k \leftarrow 1$ **to** n
    for $i \leftarrow 1$ **to** n
        for $j \leftarrow 1$ **to** n
            $t_{ij}^{(k)} \leftarrow t_{ij}^{(k-1)} \vee (t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)})$ — $\Theta(n^3)$ 算可以經过前k個
return $T(n)$

- Time: $\Theta(n^3)$.
- Only 1 bit is required for each $t_{ij}^{(k)}$.
- G^* can be used to determine the strongly connected components of G.



$$t_{42}^{3} = t_{42}^{2} \vee (t_{43}^{2} \wedge t_{32}^{2})$$
$$= 0 \vee (1 \wedge 1)$$
$$= 1$$

$$T^{(0)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(1)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad T^{(2)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix}$$

$$T^{(3)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad T^{(4)} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$