------------------------------- 25 ------------------------------

All-Pairs Shortest Paths

Input: the adjacent matrix *W* of a weighted directed graph *G*=(*V*, *E*), where

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

(Negative weights can present. But, *G* contains no negative-weight cycles.)

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

A *predecessor matrix* $\Pi=(\pi_{ij})$, where π_{ij} is the predecessor of *j* on some shortest path from *i*.
(Subgraph induced by row *i* of Π is a shortest-paths tree with root *i*.)

25.1 Shortest paths and multiplication

(A dynamic-programming approach)

Optimal structure: all subpaths of a shortest path are shortest paths.

A recursive solution:

Let $d_{ij}^{(m)}$ be the minimum weight of any path from *i* to *j* that contains at most *m* edges.

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

$$d_{ij}^{(m)} = \min_{1 \le k \le n} \{d_{ik}^{(m-1)} + \omega_{kj}\}$$

Since *G* contains no negative-weight cycles,

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

* $D^{(1)} = W$

* $D = D^{(n-1)} = D^{(n)} = D^{(n+1)} = \ldots$

Computing $D^{(m)}$ from $D^{(m-1)}$

EXTEND-SHORTEST-PATHS(D, W)

$n \leftarrow rows[D]$
let $D' = (d'_{ij})$ be an $n \times n$ matrix
for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $d'_{ij} \leftarrow \infty$
            for $k \leftarrow 1$ **to** n
                do $d'_{ij} \leftarrow \min(d'_{ij}, d_{ik} + w_{kj})$
return D'

* D for $D^{(m-1)}$ and D' for $D^{(m)}$
* Time: $\Theta(n^3)$
* Similar to matrix multiplication $C=A\times B$:

$d_{ij}^{(m-1)}$ --> a_{ij} ω_{ij} --> b_{ij} $d_{ij}^{(m)}$ --> c_{ij}

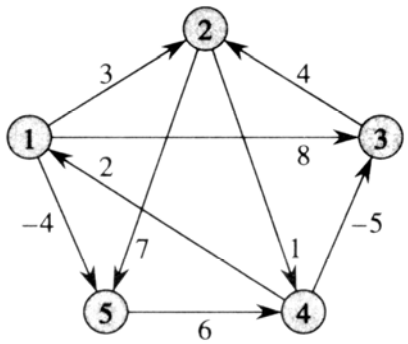
min --> +   + --> *

MATRIX-MULTIPLY(A, B)

$n \leftarrow rows[A]$
let C be an $n \times n$ matrix
for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
        do $c_{ij} \leftarrow 0$
            for $k \leftarrow 1$ **to** n
                do $c_{ij} \leftarrow c_{ij} + a_{ik} \cdot b_{kj}$
return C

* $D^{(1)} = D^{(0)}W = W$ $D^{(2)} = D^{(1)}W = W^2$

$D^{(3)} = D^{(2)}W = W^3$ ...



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

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

SLOW-ALL-PAIRS-SHORTEST-PATHS(W)

$n \leftarrow rows[W]$
$D^{(1)} \leftarrow W$
for $m \leftarrow 2$ **to** $n-1$
    do $D^{(m)} \leftarrow$ EXTEND-SHORTEST-PATHS$(D^{(m-1)}, W)$
return $D^{(n-1)}$

* Time: n-2 $\times$ $O(n^3)$ = O(n^4).

* Space: $O(n^2)$
(Note that only two matrix is really required.)

Improving the running time by repeated squaring

$W^2 = W \times W \qquad W^4 = W^2 \times W^2$

$W^8 = W^4 \times W^4 \qquad \ldots$

$$W^{2^{\lceil \lg(n-1) \rceil}} = W^{2^{\lceil \lg(n-1) \rceil - 1}} \times W^{2^{\lceil \lg(n-1) \rceil - 1}} = D$$

(Note that $D = W^{n-1} = W^n = W^{n+1} = \ldots$)

FASTER-ALL-PAIRS-SHORTEST-PATHS(W)

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

Time: $\Theta(n^3 \lg n)$

25.2 The Floyd-Warshall algorithm
(A dynamic-programming approach)

A recursive solution:

Let $d_{ij}^{(k)}$ be the weight of a shortest path from i to j with all ***intermediate*** vertices in {1, 2, ..., k}.

$$d_{ij}^{(0)} = \omega_{ij}$$

$$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\} \quad \text{for } k \geq 1$$

(since G contains no negative-weight cycles)

* $d_{ij} = \delta(i, j) = d_{ij}^{(n)}$.

FLOYD-WARSHALL(W)

```
n ← rows[W]
D^(0) ← W
for k ← 1 to n
     do for i ← 1 to n
           do for j ← 1 to n
                 do d_ij^(k) ← min(d_ij^(k−1), d_ik^(k−1) + d_kj^(k−1))
return D^(n)
```

Time: $\Theta(n^3)$

Constructing a shortest path: Refer to textbook

Transitive closure of a directed graph G

$G^* = (V, E^*)$,

where $E^* = \{(i, j) \mid \text{if there is a path from } i \text{ to } j \text{ in } G\}$.

Method 1: assign a weight 1 to each edge of G and then perform Floyd-Warshall algorithm. We have (i, j) in E^* iff $d_{ij} < n$.

Method 2: (Save time and space in practice)
Define $t_{ij}^{(k)}=1$ if there is a path from i to j with all ***intermediate*** vertices in $\{1, 2, ..., k\}$.

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

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

TRANSITIVE-CLOSURE(G)

$n \leftarrow |V[G]|$
for $i \leftarrow 1$ **to** n
  do for $j \leftarrow 1$ **to** n
    do if $i = j$ or $(i, j) \in E[G]$
      then $t_{ij}^{(0)} \leftarrow 1$
      else $t_{ij}^{(0)} \leftarrow 0$
for $k \leftarrow 1$ **to** n
  do for $i \leftarrow 1$ **to** n
    do for $j \leftarrow 1$ **to** n
      do $t_{ij}^{(k)} \leftarrow t_{ij}^{(k-1)} \vee \left(t_{ik}^{(k-1)} \wedge t_{kj}^{(k-1)}\right)$
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.

Homework: Ex. 25.1-5, 25.1-6, 25.1-10, 25.2-3, 25.2-4, 25.2-8, Pro. 25-1.