All Pairs Shortest Paths

Given a directed, connected weighted graph $G(V, E)$, for each edge $\langle u, v\rangle \in E$, a weight $w(u, v)$ is associated with the edge. The **all pairs of shortest paths problem** (APSP) is to find a shortest path from u to v for every pair of vertices u and v in V.

1 The representation of G

The input is an $n \times n$ matrix $W = (w_{ij})$.

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

2 Algorithms for the APSP problem

- Matrix Multiplication / Repeated Squaring
- The Floyd-Warshall Algorithm
- Transitive Closure of a Graph

3 Matrix Multiplication Algorithm

The algorithm is based on dynamic programming, in which each major loop will invoke an operation that is very similar to matrix multiplication. Following the DP strategy, the structure of this problem is, for any two vertices u and v,

1. if $u = v$, then the shortest path p from u to v is 0.

2. otherwise, decompose p into $u \to x \to v$, where p' is a path from u to x and contains at most k edges and it is the shortest path from u to x.

A recursive solution for the APSP problem is defined. Let $d_{ij}^{(k)}$ be the minimum weight of any path from i to j that contains at most k edges.

1. If $k = 0$, then

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

2. Otherwise, for $k \geq 1$, $d_{ij}^{(k)}$ can be computed from $d_{ij}^{(k-1)}$ and the adjacency matrix w.
$d_{ij}^{(k)} = \min\{d_{ij}^{(k-1)}, \min_{1 \leq l \leq n}\{d_{il}^{(k-1)} + w_{lj}\}\} = \min_{1 \leq l \leq n}\{d_{il}^{(k-1)} + w_{lj}\}$

```
SPECIAL-MATRIX-MULTIPLY(A, B)
1   n ← rows[A]
2   C ← new n × n matrix
3   for i ← 1 to n
4       do for j ← 1 to n
5              do c_ij ← ∞
6                 for k ← 1 to n
7                     do c_ij ← min(c_ij, a_ik + b_kj)
.                        /* Here's where this algorithm */
.                        /* differs from MATRIX-MULTIPLY. */
8   return C
```

The optimal solution can be computed by calling SPECIAL-MATRIX-MULTIPLY$(D^{(k)}, W)$ for $1 \leq k \leq n-2$. We only need to run to $n-2$ because that will give us $D^{(n-1)}$ giving us all the shortest path lengths of at most $n-1$ edges (you only need $n-1$ edges to connect n vertices). Since SPECIAL-MATRIX-MULTIPLY is called $n-2$ times, the total running time is $O(n^4)$.

3.1 The repeated squaring method

Since each $D^{(k)}$ matrix contains the shortest paths of at most k edges, and W really is $D^{(1)}$, all we were doing in the earlier solution was going: "Given the shortests paths of at most length k, and the shortests paths of at most length 1, what is the shortests paths of at most length $k+1$?"

This situation is ripe for improvement. The repeated squaring method rephrases the question to: "Given the shortests paths of at most length k, what is the shortests paths of at most length $k+k$?" The correctness of this approach lies in the observation that the shortests paths of at most m edges is the same as the shortest paths of at most $n-1$ edges for all $m > n-1$. Thus:

$$\begin{aligned} &D^{(1)} = W \\ &D^{(2)} = W^2 = W.W \\ &D^{(4)} = W^4 = W^2.W^2 \\ &\vdots \\ &D^{(2^{\lceil \log(n-1) \rceil})} = W^{(2^{\lceil \log(n-1) \rceil})} = W^{(2^{\lceil \log(n-1) \rceil}-1)}.W^{(2^{\lceil \log(n-1) \rceil}-1)} \end{aligned}$$

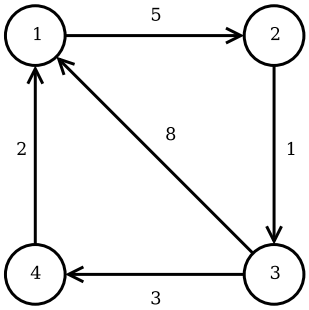

Figure 1: Example graph for the Repeated Squaring method.

Using repeated squaring, we only need to run SPECIAL-MATRIX-MULTIPLY $\lceil \log(n-1) \rceil$ times. Hence the running time of the improved matrix multiplication is $O(n^3 \log n)$.

```
ALL-PAIRS-SHORTEST-PATHS(W)
1  n ← rows[W]
2  D^(1) ← W
3  m ← 1
4  while m < n − 1
5        do D^(2m) ← SPECIAL-MATRIX-MULTIPLY(D^(m), D^(m))
6           m ← 2m
7  return D^(n)
```

3.2 Repeat Squaring Example

Take the graph in Figure 1 for example. The weights for this graph in matrix form...

$$W = \begin{pmatrix} 0 & 5 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 8 & \infty & 0 & 3 \\ 2 & \infty & \infty & 0 \end{pmatrix} = D^{(1)}$$

Repeatedly squaring this gives us...

$$D^{(1)}.D^{(1)} = \begin{pmatrix} 0 & 5 & 6 & \infty \\ 9 & 0 & 1 & 4 \\ 5 & 13 & 0 & 3 \\ 2 & 7 & \infty & 0 \end{pmatrix} = D^{(2)}$$

$$D^{(2)}.D^{(2)} = \begin{pmatrix} 0 & 5 & 6 & 9 \\ 6 & 0 & 1 & 4 \\ 5 & 10 & 0 & 3 \\ 2 & 7 & 8 & 0 \end{pmatrix} = D^{(4)}$$

$D^{(4)}$ contains the all-pairs shortest paths. It is interesting to note that at $D^{(2)}$, the shortest path from 2 to 1 is 9 using the path $\langle 2, 3, 1 \rangle$. Since the final solution ($D^{(4)}$) allows for up to 4 edges to be used, a shorter path $\langle 2, 3, 4, 1 \rangle$ was found with a weight of 6.

4 The Floyd-Warshall Algorithm

Floyd-Warshall's algorithm is based upon the observation that a path linking any two vertices u and v may have zero or more intermediate vertices. The algorithm begins by disallowing all intermediate vertices. In this case, the partial solution is simply the initial weights of the graph or infinity if there is no edge.

The algorithm proceeds by allowing an additional intermediate vertex at each step. For each introduction of a new intermediate vertex x, the shortest path between any pair of vertices u and v, $x, u, v \in V$, is the minimum of the previous best estimate of $\delta(u, v)$, or the combination of the paths from $u \to x$ and $x \to v$.

$$\delta(u, v) \leftarrow \min(\delta(u, v), \delta(u, x) + \delta(x, v))$$

Let the directed graph be represented by a weighted matrix W.

```
FLOYD-WARSHALL(W)
1  n ← rows[W]
2  D^(0) ← W
3  for k ← 1 to n
4      do for i ← 1 to n
5             do for j ← 1 to n
6                    do d_ij^(k) ← MIN(d_ij^(k-1), d_ik^(k-1) + d_kj^(k-1))
7  return D^(n)
```

The time complexity of the algorithm above is $O(n^3)$.

4.1 Floyd-Warshall Example

Let us now work through the steps of the FLOYD-WARSHALL algorithm when it is applyed on the graph depicted in Figure 2.

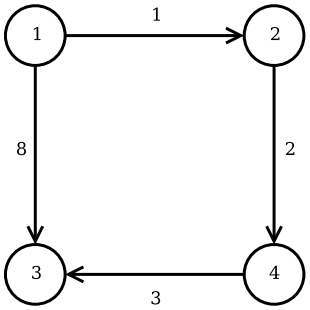

Figure 2: Example graph for Floyd-Warshall.

$$W = \begin{pmatrix} 0 & 1 & 8 & \infty \\ \infty & 0 & \infty & 2 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 3 & 0 \end{pmatrix}$$

1. Allowable intermediate vertices $\{\}$: $D^{(0)} = W$
2. Allowable intermediate vertices $\{1\}$: $D^{(1)}$ no change
3. Allowable intermediate vertices $\{1, 2\}$:

$$D^{(2)} = \begin{pmatrix} 0 & 1 & 8 & \mathbf{3} \\ \infty & 0 & \infty & 2 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 3 & 0 \end{pmatrix}$$

   Where $3 = \min\{\infty, 1 + 2\}$, using paths $1 \to 2$ and $2 \to 4$.
4. Allowable intermediate vertices $\{1, 2, 3\}$: $D^{(3)}$ no change
5. Allowable intermediate vertices $\{1, 2, 3, 4\}$:

$$D^{(4)} = \begin{pmatrix} 0 & 1 & \mathbf{6} & 3 \\ \infty & 0 & \mathbf{5} & 2 \\ \infty & \infty & 0 & \infty \\ \infty & \infty & 3 & 0 \end{pmatrix}$$

   Where $6 = \min\{8, 3 + 3\}$, using paths $1 \to 4$ and $4 \to 3$.
   Where $5 = \min\{\infty, 2 + 3\}$, using paths $2 \to 4$ and $4 \to 3$.

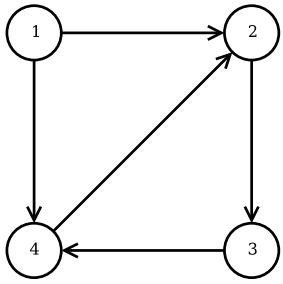

Figure 3: Example graph for Transitive Closure.

5 The transitive closure of a directed graph

The **transitive closure** of a directed graph $G = (V, E)$ tells us if there is a path from i to j, for each pair of vertices $i, j \in V$. It is itself another graph $G^* = (V, E^*)$ where $E^* = \{\langle i, j\rangle : \text{there is a path from } i \text{ to } j \text{ in } G\}$. The transitive closure of a directed graph G can be determined by the following algorithm.

```
TRANSITIVE-CLOSURE(G)
1   n ← |V|
2   for i ← 1 to n
3       do for j ← 1 to n
4             do if i = j or (i, j) ∈ E
5                   then t_ij^(0) ← 1
6                   else t_ij^(0) ← 0
7   for k ← 1 to n
8       do for i ← 1 to n
9             do for j ← 1 to n
10                  do t_ij^(k) ← t_ij^(k−1) ∨ (t_ik^(k−1) ∧ t_kj^(k−1))
11   return T^(n)
```

5.1 Transitive Closure Example

The matrices below illustrate the state of the TRANSITIVE-CLOSURE algorithm as it operates on the graph depicted in Figure 3.

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

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