Chapter 26: All-Pairs Shortest Path

A trivial solution is to use SSSP algorithms for APSP

With Dijkstra’s algorithm (no negative weights!) the running time would become
\(O(V(V \log V + E)) = O(V^2 \log V + V^2 E)\)

With the Bellman-Ford algorithm the running time would become \(O(V(E)) = O(V^2 E)\)

Three approaches for improvement:

<table>
<thead>
<tr>
<th>algorithm</th>
<th>cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>matrix multiplication</td>
<td>(\Theta(V^3 \log V))</td>
</tr>
<tr>
<td>Floyd-Warshall</td>
<td>(\Theta(V^3))</td>
</tr>
<tr>
<td>Johnson</td>
<td>(O(V^2 \log V + V^E))</td>
</tr>
</tbody>
</table>

**Matrix Multiplication**

Define the \(V \times V\) matrix \(D^{(m)} = (d^{(m)}_{ij})\) by:

\[d^{(m)}_{ij} = \begin{cases} 
0 & \text{if } i = j, \\
\infty & \text{otherwise}, \\
w_{ij} & \text{if } i \neq j, (i,j) \in E,
\end{cases}\]

and for all \(i, j, p, q\),

\[d^{(p+q)}_{ij} = \min_{1 \leq k \leq n} (d^{(p)}_{ik} + d^{(q)}_{kj}).\]

\(D(V-1)\) is the matrix \((\delta(i,j))\).

**Computing \(D^{(p+q)}\) from \(D^{(p)}\) and \(D^{(q)}\) using matrix multiplication**

\[D^{(p+q)} = D^{(p)} \cdot D^{(q)}\]

where \((\min, +)\) is used as the computational basis instead of \((+, \times)\)

\[
\begin{pmatrix}
10 & 9 & 4 \\
8 & 6 & 12
\end{pmatrix}
\begin{pmatrix}
\vdots
\end{pmatrix}
= \begin{pmatrix}
\min(10+8,9+6,4+12) \\
\vdots
\end{pmatrix}
\begin{pmatrix}
\vdots
\end{pmatrix}
\]

The complexity is \(O(V^3 \log V)\)

**Q1. How can you check the existence of negative weight cycles?**
**Method 2: Floyd-Warshall**

Define the $V \times V$ matrix $W^{(m)} = (f_{ij}^{(m)})$ by:

$$f_{ij}^{(m)}$$ is the shortest path length from $i$ to $j$ with only nodes $\leq m$ in between

Define $f_{ij}^{(0)} = w_{ij}$. Then for every $i, j$ and every $k \geq 1$,

$$f_{ij}^{(k)} = \min(f_{ij}^{(k-1)}, f_{ik}^{(k-1)} + f_{kj}^{(k-1)}),$$

$W(V)$ is the matrix $(\delta(i,j))$.

Compute $W^k$ from $W^{k-1}$ for $k = 1, \ldots, V$

Q2. How many steps are needed for computing an entry?

Q3. How many entries are evaluated in total?

Q4. So, what is the total cost?
**Johnson’s Algorithm**

Define a new weight function \( \hat{w} \) so that

- the shortest paths are preserved and
- \( \hat{w}(u, v) \geq 0 \) for all \( u, v \)

Then use Dijkstra’s algorithm to compute the shortest path

---

**Theorem A** Let \( h \) be any mapping of \( V \) to \( \mathbb{R} \). Define \( \hat{w}(u, v) = w(u, v) + h(u) - h(v) \) and \( \hat{d}(u, v) \) the shortest path with respect to \( \hat{w} \). If \( \hat{d}(u, v) \) is defined for all \( u, v \), then

\[
\hat{d}(u, v) = \hat{d}(u, v) + h(v) - h(u);
\]

i.e., the new weight function preserves the shortest paths.

**Proof** For any path \( p = [v_1, \ldots, v_k] \) the path length of \( p \) under \( \hat{w} \) is

\[
\sum_{i=1}^{k-1} \left( w(v_{i+1}, v_i) + h(v_i) - h(v_{i+1}) \right).
\]

This is equal to

\[
\left( \sum_{i=1}^{k-1} \right) + \sum_{i=1}^{k-1} \left( h(v_i) - h(v_{i+1}) \right).
\]

The right hand-side is \( h(v_1) - h(v_k) \). So for every \( u \) and \( v \), \( \hat{d}(u, v) = \hat{d}(u, v) + h(u) - h(v) \).
After Dijkstra

Back to Original Weights