All Pair Shortest Paths

Alison Liu

Given a directed weighted graph, the *all-pair shortest paths* problem aims to find the shortest path between any pair of vertices u and v.

A directed idea for solving the all-pair shortest paths problems is to run $|V|$ single-source shortest paths algorithms, each starts from one vertex as a source. Therefore, the time complexity for answering the all-pair shortest paths is at most $O(|V|^2 \log |V| + |V||E|)$ on graphs without negative weights (by Dijkstra's algorithm) and at most $O(|V|^2|E|)$ for general graphs without negative cycle (by Bellman-Ford algorithm).

1 Dynamic programming by number of edges

The first algorithm is a dynamic programming algorithm which restricts the number of edges on the shortest path. Let $\text{dist}_{uv}^{(\ell)}$ be the distance of the shortest path from u to v using at most ℓ edges. We get the following algorithm:

Algorithm 1 Dynamic Programming Approach 1

for all u, v **do**
  $\text{dist}_{uv}^{(0)} \leftarrow \infty$, $\text{dist}_{uu}^{(0)} \leftarrow 0$
for $\ell = 1$ to $|V| - 1$ **do**
  for all vertices u **do**
    for all vertices v **do**
      $\text{dist}_{uv}^{(\ell)} \leftarrow \text{dist}_{uv}^{(\ell-1)}$
      for all predecessors k of v **do**
        $\text{dist}_{uv}^{(\ell)} \leftarrow \min\{\text{dist}_{uv}^{(\ell)}, \text{dist}_{uk}^{(\ell-1)} + \text{dist}_{kv}^{(\ell-1)}\}$

The time complexity of the algorithm is $O(|V|^4)$.

By slightly rewriting the algorithm, the algorithm is equivalent to Algorithm 2. And the time complexity is $O(|V|^2|E|)$.

For further acceleration, one can double the number of involved edges in each round (instead of increasing the number one by one). See Algorithm 3. The time complexity is $O(|V|^3 \log |V|)$.

Algorithm 2 Dynamic Programming Approach 1, re-written

for all u, v **do**
  $\mathrm{dist}_{uv}^{(0)} \leftarrow \infty$, $\mathrm{dist}_{uu}^{(0)} \leftarrow 0$
for $\ell = 1$ to $|V| - 1$ **do**
  for all vertices u **do**
    for each edge (k, v) **do**
      $\mathrm{dist}_{uv}^{(\ell)} \leftarrow \min\{\mathrm{dist}_{uv}^{(\ell)}, \mathrm{dist}_{uk}^{(\ell-1)} + \mathrm{dist}_{kv}^{(\ell-1)}\}$

Algorithm 3 Dynamic Programming Approach 1.5

for all u, v **do**
  $\mathrm{dist}_{uv}^{(0)} \leftarrow \infty$, $\mathrm{dist}_{uu}^{(0)} \leftarrow 0$
for $\ell = 1$ to $\lceil \log |V| \rceil$ **do**
  for all vertices u **do**
    for all vertices v **do**
      $\mathrm{dist}_{uv}^{2^i} \leftarrow \infty$
      for all vertices k **do**
        $\mathrm{dist}_{uv}^{(2^i)} \leftarrow \min\{\mathrm{dist}_{uv}^{(2^i)}, \mathrm{dist}_{uk}^{(2^{i-1})} + \mathrm{dist}_{kv}^{(2^{i-1})}\}$

2 `Floyd-Warshall`: Dynamic programming by vertices

There is another approach of dynamic programming, where the restriction is on the subset of vertices the shortest paths are using. The algorithm first gives labels $1, 2, \cdots, n$ to vertices. Let $\mathrm{dist}_{uv}^{(k)}$ denote the shortest distance from vertex u to vertex v that only goes through the vertices $1, 2, \cdots, k$. Initially, all $\mathrm{dist}_{uv}^{(0)}$ is set to be 0 if there is an edge (u, v) and ∞ otherwise. The shortest distance from u to v in the graph is $\mathrm{dist}_{uv}^{(n)}$. For any $\mathrm{dist}_{u,v}^{(k)}$, it is the minimum value between the shortest distance of paths going through k and the shortest distance of paths not going through k. This approach only takes constant comparison in the inner for-loop. (See Algorithm 4.) Therefore, the time complexity is $O(|V|^3)$.

Algorithm 4 Dynamic Programming Approach 2: `Floyd-Warshall`

for all u, v **do**
  $\mathrm{dist}_{uv}^{(0)} \leftarrow w(u, v)$
for $k = 1$ to $|V|$ **do**
  for all vertices u **do**
    for all vertices v **do**
      $\mathrm{dist}_{uv}^{(k)} \leftarrow \min\{\mathrm{dist}_{uk}^{(k-1)} + \mathrm{dist}_{kv}^{(k-1)}, \mathrm{dist}_{uv}^{(k-1)}\}$

3 Johnson: Reweighing the edges

Another different approach is to reweight the edges such that there is no negative weight in the graph, and run $|V|$ Dijkstra's algorithms on the reweighted graph. The edge reweighting should satisfy the following two properties: 1) For all edge (u, v), the new weight $w'(u, v) \geq 0$, and 2) The path P is the shortest path from s to t in the original graph if and only if it is the shortest path from s to t in the reweighted graph.

Johnson's algorithm is an algorithm that first reweights the edges such that there is no negative weight edge and runs $|V|$ Dijkstra's algorithms on the new graph. Formally, the algorithm first assigns h-value for each vertex, and then reweight the edge weight $w(u, v)$ by $w'(u, v) = w(u, v) + h(u) - h(v)$. The h-values of the vertices are generated as follows. First, the algorithm adds a dummy vertex δ to the original graph. Then, for any vertex v in the original graph, there is a new edge (σ, v) with weight 0. The h-value of the vertex v is the shortest distance from σ to v, which can be found in $O(|V||E|)$ time by the Bellman-Ford algorithm. Then, the algorithm runs $|V|$ Dijkstra's algorithms on the reweighted graph, each start from a vertex in V. The total time complexity of the Johnson's algorithm is $O(|V|^2 \log |V| + |V||E|$, which is dominated by running $|V|$ Dijkstra's algorithms.

The new weights are non-negative. For any edge (u, v), the new weight $w'(u, v) = w(u, v) - h(u) + h(v) = w(u, v) - \delta(\sigma, u) + \delta(\sigma, v) \geq 0$, where the last inequality is from $\delta(\sigma, v) \leq \delta(\sigma, u) + w(u, v)$.

Shortest paths preservation. The reweighting through h-values of the vertices preserves the shortest paths as follows. Consider any path P from s to t, the new weight of the path $w'(P) = w(P) - h(s) + h(t)$. Therefore, for any shortest path P from s to t such that $w(P) \leq w(Q)$ for any path Q from s to t, $w'(P) \leq w'(Q)$.

Algorithm 5 Johnson's algorithm

```
FIND-H(G)
For all (u,v) ∈ E, w'(u,v) ← h(u) + w(u,v) − h(v)
For each u ∈ V, Dijkstra(G', u)
For each v ∈ V, δ(u,v) ← δ(u,v) − h(u) + h(v)
Return all δ(u,v)
```

Algorithm 6 Find-H (G)

$V' \leftarrow V \cup \{\sigma\}$
$E' \leftarrow \cup_{v \in V} \{(\sigma, v)\}$
For all $v \in V$, $w(\sigma, v) \leftarrow 0$
$G' \leftarrow (V', E')$
`Bellman-Ford`(G', σ)
For each $v \in V$, $h(v) \leftarrow \delta(\sigma, v)$