All-Pairs Shortest Paths

Yufei Tao

Department of Computer Science and Engineering
Chinese University of Hong Kong

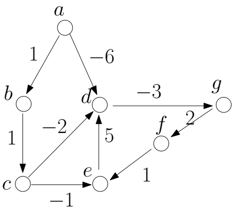
In this lecture, we will study a problem called **all-pairs shortest paths** which is closely related to the SSSP (single-source shortest path) problem discussed in the previous lectures. We will learn two algorithms: **the Floyd-Warshall algorithm** and **Johnson's algorithm**.

All-Pairs Shortest Paths (APSP)

Input: Let $G = (V, E)$ be a simple directed graph. Let w be a function that maps each edge in E to an integer, **which can be positive, 0, or negative**. It is guaranteed that G has **no negative cycles**.

Output: We want to find a shortest path (SP) from s to t, for all $s, t \in V$. More specifically, the output should be $|V|$ shortest-path trees, each rooted at a distinct vertex in V.

Example



Shortest path distances:
$spdist(a, a) = 0$, $spdist(a, b) = 1$, ..., $spdist(a, g) = -9$
$spdist(b, a) = \infty$, $spdist(b, b) = 0$, ..., $spdist(b, g) = -4$
...
$spdist(g, a) = \infty$, $spdist(g, b) = \infty$, ..., $spdist(g, g) = 0$

We omit the shortest paths in this example.

If all the weights are non-negative, we can run Dijkstra's algorithm $|V|$ times. The total time is $O(|V|(|V| + |E|) \log |V|)$.

For the general APSP problem (arbitrary weights), we can run Bellman-Ford's algorithm $|V|$ times. The total time is $O(|V|^2|E|)$.

We will solve the (general) APSP problem in time

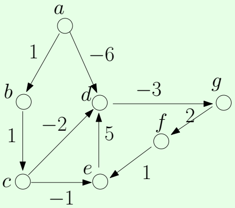
$$O\left(\min\{|V|^3, |V|(|V| + |E|) \log |V|\}\right).$$

Note that the complexity strictly improves that in the second box.

The Floyd-Warshall Algorithm

Set $n = |V|$.
Assign each vertex in V a distinct id from 1 to n.

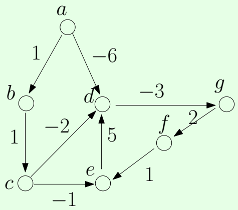
Example:



Let us assign to 1 vertex a, 2 to vertex b, ..., 7 to vertex g.

Define *spdist*$(i, j \mid\leq k)$ as the smallest length of all paths from the vertex **with id** i to the vertex **with id** j that **pass only intermediate vertices with ids** $\leq k$.

Example:



Vertex ids: 1 for a, 2 for b, ..., 7 for g.

$spdist(1, 5 \mid 0) = \infty$, $spdist(1, 5 \mid 1) = \infty$, $spdist(1, 5 \mid 2) = \infty$,
$spdist(1, 5 \mid 3) = -1$,
$spdist(1, 5 \mid 4) = -1$, $spdist(1, 5 \mid 5) = -1$, $spdist(1, 5 \mid 6) = -1$,
$spdist(1, 5 \mid 7) = -6$

Obviously, $spdist(i, j \mid\leq 0)$ equals $w(i, j)$ if E has an edge (i, j), or ∞, otherwise.

Lemma: It holds for all $i, j, k \in [1, n]$ that

$$spdist(i, j \mid\leq k) = \min \begin{cases} spdist(i, j \mid\leq k-1) \\ spdist(i, k \mid\leq k-1) + spdist(k, j \mid\leq k-1) \end{cases}$$

The proof is left as a regular exercise.

Observe that $spdist(i, j \mid\leq n) = spdist(i, j)$.
Our goal is therefore to compute $spdist(i, j \mid\leq n)$ for all $i, j \in [1, n]$.

This clearly points to a dynamic programming algorithm that finishes in $O(|V|^3)$ time.

Extending the algorithm to report paths is easy and left to you.

Johnson's Algorithm

Recall:

> If all the weights are non-negative, we can run Dijkstra's algorithm $|V|$ times. The total running time is $O(|V|(|V| + |E|) \log |V|)$.

We cannot apply Dijkstra's because our graph may have negative-weight edges. Can we convert all the weights into non-negative values **while preserving all shortest paths**?

Interestingly, the answer is yes.

Re-weighting

Introduce an arbitrary function $h : V \to \mathbb{Z}$, where $\mathbb{Z}$ represents the set of integer values.

For each edge (u, v) in E, redefine its weight as:

$$w'(u, v) = w(u, v) + h(u) - h(v).$$

Denote by G' the graph where

- the set V of vertices and the set E of edges are the same as G;
- the edges are weighted using function w'.

Re-weighting

Lemma: Consider any path $v_1 \to v_2 \to ... \to v_x$ in G where $x \geq 1$. If the path has length ℓ in G, then it has length $\ell + h(v_1) - h(v_x)$ in G'.

Proof: The length of the path in G' is

$$\begin{aligned} & \sum_{i=1}^{x-1} w'(v_i, v_{i+1}) \\ = & \sum_{i=1}^{x-1} (w(v_i, v_{i+1}) + h(v_i) - h(v_{i+1})) \\ = & \left(\sum_{i=1}^{x-1} w(v_i, v_{i+1}) \right) + h(v_1) - h(v_x). \end{aligned}$$

□

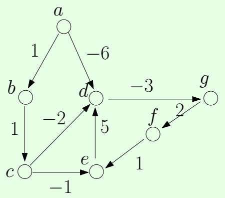
Re-weighting

Corollary: Let π be a shortest path from vertex u to vertex v in G, it is also a shortest path from u to v in G'.

Proof: Let π' be any other path from u to v in G'. Denote by ℓ and ℓ' the length of π and π' in G, respectively. It holds that $\ell \leq \ell'$. By the lemma of the previous slide, we know that π and π' have length $\ell + h(u) - h(v)$ and $\ell' + h(u) - h(v)$ in G', respectively. $\square$

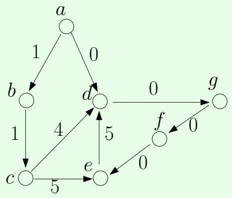
Example

Example:



$h(a) = 0$
$h(b) = 0$
$h(c) = 0$
$h(d) = -6$
$h(e) = -6$
$h(f) = -7$
$h(g) = -9$

After re-weighting:



We want to make sure

$$w'(u, v) \geq 0$$

for all edges (u, v) in E. Not every function $h(.)$ fulfills the purpose.

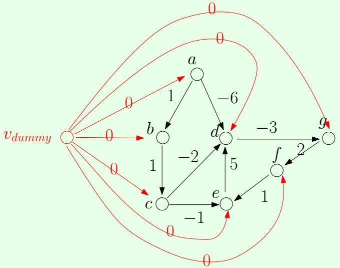
Next, we will introduce a **dummy-vertex trick** to find a good $h(.)$.

A "Dummy-Vertex" Trick

From $G = (V, E)$, construct a graph $G^{\Delta} = (V^{\Delta}, E^{\Delta})$ where:

- $V^{\Delta} = V \cup \{v_{dummy}\}$;
- E^{Δ} includes all the edges in E, and additionally, a new edge from v_{dummy} to every other vertex in V;
- Each edge inherited from E carries the same weight as in E. Every newly added edge carries the weight 0.

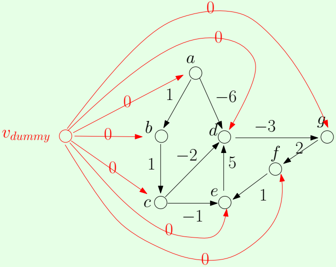
Example:



A "Dummy-Vertex" Trick

In $G^{\Delta} = (V^{\Delta}, E^{\Delta})$, find the shortest path distance from v_{dummy} to every other vertex. This is an SSSP problem which can be solved by Bellman-Ford's algorithm in $O(|V||E|)$ time.

Example:



$spdist(v_{dummy}, a) = 0$
$spdist(v_{dummy}, b) = 0$
$spdist(v_{dummy}, c) = 0$
$spdist(v_{dummy}, d) = -6$
$spdist(v_{dummy}, e) = -6$
$spdist(v_{dummy}, f) = -7$
$spdist(v_{dummy}, g) = -9$

A "Dummy-Vertex" Trick

Recall that we are looking for a good function $h(.)$ to re-weight the edges of G. We now design the function as follows:

$$h(u) = spdist(v_{dummy}, u)$$

for every $u \in V$.

Lemma: After re-weighting the edges of G with the above $h(.)$, all edge weights in G' (i.e., the graph after re-weighting) are non-negative.

The proof is left as an exercise.

We can now apply Dijkstra's algorithm to solve the APSP problem in time $O(|V|(|V| + |E|) \log |V|)$.