Contents

1 Graph Algorithms

1.1 Lecture 1: Single-Source Shortest Paths

We approach the SSSP problem with possibly negative edge weights. As input, we get the directed graph $G = (V, E, w)$ with weight function $w : E \to \mathbb{R}$ and start vertex $s \in V$. We also take for a set of edges S, $w(S) = \sum_{s \in S} w(s)$. The algorithm should output the lengths of the shortest paths from s to any other $v \in V$ (or the shortest path tree). We know a few algorithms for this.

- Djikstra (with a Fibonacci Heap): $O(m + n \log n)$, which only works if $w \geq 0$.
- Bellman-Ford: $O(mn)$.
- Bernstein-Nanongkai-Wulff-Nilsen: $\tilde{O}(m \log W)$, where $|w| \leq W$.

The $\tilde{O}(f)$ means $f \cdot \text{Polylog}(f)$ We discuss the third one today.

We define a price function as a function $\phi : V \to \mathbb{R}$ and the associated price-reduced weight function as:

$$w_\phi((u, v)) = w((u, v)) + \phi(u) - \phi(v)$$

Note the following observations. For any path $P = (v_0, \ldots, v_r)$, $w_\phi(P) = w(P) + \phi(v_0) - \phi(v_r)$. As a corollary, for all cycles C, $w(C) = w_\phi(C)$. This implies the shortest path in $G_\phi = (V, E, w_\phi)$ is the same as in G. Our goal is thus to find a ϕ such that for all edges, $w_\phi \geq 0$, then reduce to Djikstra.

Theorem 1.1
A ϕ satisfying $w_\phi \geq 0$ exists if and only if there exist no negative cycles in the original graph G.

Proof
Clearly if such a ϕ exists, $w_\phi(C) \geq 0$ for any cycle which is the same as $w(C)$ by our observations above.
For the other direction, let $\phi(v) = d(s, v)$, where $d(s, v)$ is the length of the shortest path from s to v. This means taking some neighbor u of v,

$$\phi(v) = d(s, v) \leq d(s, u) + w(u, v)$$

This means that $w_\phi(u, v) = w(u, v) + \phi(u) - \phi(v) \geq w(u, v) + d(s, u) - d(s, u) - w(u, v) = 0$.

The first person to use this technique was [G ’75]. He showed the following result.

Theorem 1.2
If you have an algorithm that finds a good ϕ in time $T(m)$ for special case when $w(e) \geq -1$, then you can solve the general case of $|w| \leq W$ in time $O(T(m) \log W)$.

Proof
Round W up to the nearest power of 2, $W = 2^k$. We proceed by induction on k. If $k = 0$, all the edge weights are already bigger than -1, so we’re finished.
For an inductive case, we use the following algorithm, using solve as as our subroutine:

function A($G = (V, E, w), k$)
  if $k = 0$ **then**
    return Solve(G)
  else
    $\hat{w}(e) \leftarrow \left\lceil \frac{w(e)}{2} \right\rceil$
    $\hat{\phi} \leftarrow A(V, E, \hat{w}, k - 1)$
    $\phi \leftarrow 2\hat{\phi}$
    return Solve(G_ϕ)

To prove correctness, we need to show that G_ϕ is solvable. Take some edge $e = (u, v)$.

$$\begin{aligned}
w(e) &\geq 2\left\lceil \frac{w(e)}{2} \right\rceil - 1 \\
&= 2\hat{w}(e) - 1 \\
w_\phi(e) &= w(e) + \phi(u) - \phi(v) \\
&\geq 2\hat{w}(e) - 1 + 2\hat{\phi}(u) - 2\hat{\phi}(v) \\
&= 2\hat{w}_{\hat{\phi}}(e) - 1 \\
&\geq -1
\end{aligned}$$

Since the weights are nonnegative with the good $\hat{\phi}$.

Now, this is the novel portion. We will focus on G_s which is G with a dummy vertex s which has weight-0 to everyone else. We will focus on finding a ϕ for G_s (which we will label as G for brevity).

The algorithm has the following ingredients:

1. The subroutine Low Diameter Decomposition. $LDD(G, D)$ takes in a graph G where $w \geq 0$, and outputs $E^{rem} \subseteq E$ such that every strongly connected component of $G \setminus E^{rem}$ has weak diameter at most D. The weak diameter is $\max_{u,v \in \text{same SCC}} d_G(u, v)$. Furthermore, $\mathbb{P}\left[e \in E^{rem}\right] \leq O\left(\frac{w(e)\log^2 n}{D} + n^{-10}\right)$. This is a fast algorithm, with running time $\tilde{O}(m)$.
2. The subroutine Fix DAG Edges. This finds ϕ that makes $w_\phi \geq 0$ when G is a DAG. Using an SCC graph, this will mean $w_\phi(e) \geq 0$ for e going across SCCs. To implement this in linear time, just find the distance from a source with dynamic programming and just set the price function to be that.
3. The subroutine Elim Neg. This finds a ϕ in time $O(\log n \cdot \sum_v (1 + \eta_G(v)))$ where $\eta_G(v)$ is the number of negative edges on the shortest path to v. This algorithm assumes all in-degrees and out-degrees are $O(1)$.
4. The subroutine Scale Down. It takes in two numbers Δ and B. Assumes that $\eta(G) = \max_{v \in V} \eta_G(v) \leq \Delta$ and assumes all edges $w(e) \geq -2B$. It outputs ϕ such that $w_\phi \geq -B$.

Furthermore, in the original graph we can assume WLOG, all degrees are $O(1)$, so we can actually use the third condition. We do this with graph blow-up on each vertex v. Suppose v has x in-degree and y out-degree. We can make a $x + y$-cycle with all edge weights 0. Each vertex has one of v's original edges, either an incoming or outgoing one. This does not change shortest paths. All this does is blow up the number of vertices to $O(m)$ at most, which doesn't affect the runtime given.

Let's put them all together.

Algorithm 1.1

function $\textsc{Main}(G = (V, E, w))$
 $B \leftarrow 2n$ (rounded up to nearest power of 2)
 $\bar{w} \leftarrow Bw$
 $\phi_0 \leftarrow 0$
 for $i = 1$ to $\log_2 B$ **do**
  $\psi_i \leftarrow \text{ScaleDown}(G_{\phi_{i-1}}, \Delta = n, \frac{B}{2^i})$
  $\phi_i \leftarrow \phi_{i-1} + \psi_i$
 $w^* \leftarrow \bar{w}_{\phi_{\log B}} + 1$