1 Running Time of Dijkstra’s Algorithm

- Each vertex is removed from queue at most once
  - when $v$ is removed $d[v] = d(s,v)$, so $d[v]$ can never go down again. So $v$ can never go back into queue.
  - $\leq |V|$ extract Min() calls $|V| \log |V|$
  - $\leq |E|$ update() calls $|E| \log |V|$

- Running time is $O((|V| + |E|) \log |V|)$ (using Fibonacci $O((|V| + |E|) \log |V|)$)

Note: only works (efficiently) when no negative weight edges

2 Negative Weight Edges

Definition 1. Negative cycle is a cycle whose sum weight of edge is a negative value, then walks of arbitrary low weight can be constructed by repeatedly following the cycle, so there may be no shortest path.

3 Bellman-Ford Algorithm (single-source shortest path)

Basic idea: Bellman-Ford is based on dynamic programming approach. In its basic structure it is similar to Dijkstra’s Algorithm, but instead of greedily selecting the minimum-weight node not yet processed to relax, it simply relaxes all the edges, and does this $|V| - 1$ times.

The algorithm can be summarized as:

$$(V, E) = G$$

for $v \in V$

$d[v] = \infty$, $\pi[v] = \bot$$
\[ d[s] = 0 \]
for \( i = 1 \) to \(|V| - 1\)
for each \((u, v) \in E\)
relax \((u, v)\)
end
end
for \((u, v) \in E\)
if relax \((u, v)\)
return NEG CYCLE
return \((d, \pi)\)
end

the above algorithm is \(O(|E| \cdot |V|)\) and the relax algorithm is

\[
\begin{align*}
\text{relax} (u, v) \\
\text{if } d[u] + w(u, v) < d[u] \\
d[v] &= d[u] + w(u, v) \\
\pi [v] &= u
\end{align*}
\]

**Definition 2.** \(d_i(u, v) = \text{length of shortest path crossing } \leq i \text{ edges}\)

**Proposition 3.** Suppose no negative cycle, then \(\forall v \text{ there is a shortest path cross } \leq |V| - 1 \text{ edges.}\)

Proof: otherwise
If no negative weight cycle, then \(d(u, v) = d_{n-1}(u, v)\)

**Theorem 4.** After \(i\)th interation, \(d[v] = d_i(s, v)\). Well \(d_0(s, v) = \begin{cases} 0 & \text{if } v = s \\ \infty & \text{otherwise} \end{cases}\)

Proof: (By induction) Suppose true after \(i\)th iteration. Let \(s \to v_1 \to \cdots \to v_i \to v_{i+1}\) be a shortest path to \(v_{i+1}\) crossing \(\leq i + 1\) edges. At beginning of \((i + 1)\)th iteration \(d[v_i] = d_i(s, v_i)\). After \((i + 1)\)the iteration, \(d[v_{i+1}] \leq d[v_i] + w(v_i, v_{i+1}) = d_i(s, v_i) + w(v_i, v_{i+1}) = d_{i+1}(s, v_{i+1})\)

**Corollary 5.** If no negative weight cycle, then algorithm computes \(d[v] = d(s, v)\).

**Proposition 6.** Final loop correctly detects negative weighted.

Proof: Suppose there is no negative weighted cycle \(\iff \forall v, d_n(s, v) = d_{n-1}(s, v)\)
\(\iff \forall v, d_{n-1}(s, v) = d_n(s, v)\)
\(\iff\) Algorithm does not return NEG CYCLE
Suppose \(d_i(s, v) = d_{i-1}(s, v)\), then \(d[\cdot]\) after \(i\)th iteration. So \(d[\cdot]\) will not change in \((i + 1)\)th iteration, i.e. \(d_{i+1}(s, v) = d_i(s, v)\).