Notes, 6(b)

ECE 606

Shortest Paths, Revisited

Recall the single-source shortest distances/paths problem:

- Inputs:
  1. Weighted directed or undirected graph, $G = \langle V, E, w \rangle$, and,
  2. A source-vertex, $s \in V$.
- Output:
  - Shortest-path weights from s to every $u \in V$.
    * Auxiliary output: a shortest paths tree rooted at s.

Turns out, a sufficient condition for problem to possess a greedy choice: non-negative edge weights only, i.e., $w : E \to \mathbb{R}_0^+$.

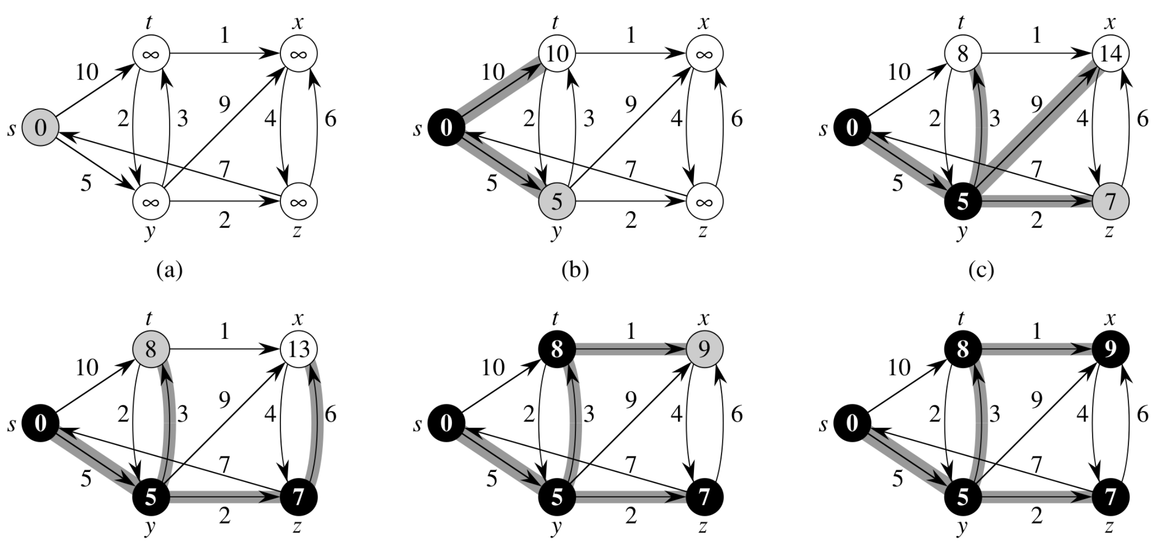
DIJKSTRA$(G = \langle V, E, w \rangle, s)$

INITIALIZE-SINGLE-SOURCE(G, s)
Done $\leftarrow \emptyset$
$Q \leftarrow$ priority queue of vertices by smallest d value
while $Q \neq \emptyset$ **do**
    $u \leftarrow$ extract vertex with minimum d value from Q
    Done $\leftarrow$ *Done* $\cup \{u\}$
    foreach $v \in Adj[u] \setminus$ *Done* **do**
        RELAX(u, v, w) and change v's position in Q as appropriate

Recall helper subroutines:

INITIALIZE-SINGLE-SOURCE(G, s)
1 **for** each vertex $v \in V[G]$
2   **do** $d[v] \leftarrow \infty$
3         $\pi[v] \leftarrow$ NIL
4 $d[s] \leftarrow 0$

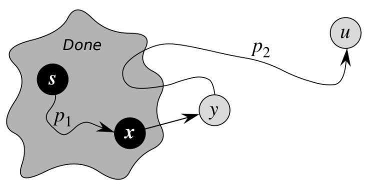
RELAX(u, v, w)
1 **if** $d[v] > d[u] + w(u, v)$
2   **then** $d[v] \leftarrow d[u] + w(u, v)$
3         $\pi[v] \leftarrow u$



(a) (b) (c) (d) (e) (f)

Claim 1. *Suppose u is a vertex to which a path exists from s. Then, at the moment a vertex u is added to Done in Line (6), $d[u] = \delta(s, u)$.*

Proof. Suppose u, to which a path exists from s, is the first vertex to be added to *Done* such that $d[u] \neq \delta(s, u)$. Then, $u \neq s$. Therefore, a shortest-path $s \rightsquigarrow u$ can be decomposed into: $s \overset{p_1}{\rightsquigarrow} x \to y \overset{p_2}{\rightsquigarrow} u$, with the possibility that $s = x$ and/or $y = u$.



As the path $s \rightsquigarrow u$ is a shortest-path, so is the subpath $s \overset{p_1}{\rightsquigarrow} x \to y$. Also, when x was added to *Done*, we did Relax(x, y, w), and since then, $d[y] = \delta(s, y)$ — see the "Convergence property" under "Properties of shortest paths and relaxation" in Lecture 4 of the textbook.

Also, because $s \overset{p_1}{\rightsquigarrow} x \to y$ is a subpath of $s \overset{p_1}{\rightsquigarrow} x \to y \overset{p_2}{\rightsquigarrow} u$, and all edge-weights are non-negative, we know that $\delta(s, y) \leq \delta(s, u)$. Also, $\delta(s, u) \leq d[u]$ — see the "Upper-bound property" under "Properties of shortest paths and relaxation" in Lecture 4 of the textbook.

So we have, at the moment u is chosen in Line (6):

- $d[y] = \delta(s, y) \leq \delta(s, u) \leq d[u]$, and,
- $d[u] \leq d[y]$, because we pick the vertex of smallest d value from $V \setminus Done$ in Dijkstra.

Therefore, $d[y] = \delta(s, y) = \delta(s, u) = d[u]$ at the moment we add u to *Done* in Dijkstra. $\square$

The above is kind of like a "cut and paste" proof like we did for interval scheduling because an optimal solution is to add y to *Done*. But we replace that choice with u, which our algoritm, Dijkstra, happens to choose.

Time-efficiency of DIJKSTRA:

Our baseline is Bellman-Ford: runs in $\Theta(|V||E|) = \Theta\big(|V|^3\big)$ in the worst-case.

Time-efficiency of DIJKSTRA depends on how we realize priority queue. Two options:

- Keep an array of vertices with their current d values.
  - Line (5) requires scan of entire array, i.e., $\Theta(|V|)$ time. We do this once for every vertex in the graph.
  - Line (8) requires a possible decrease to d value: $\Theta(1)$ time. We do this once for every edge in the graph.
  - So total time: $\Theta(|V|^2 + |E|) = \Theta(|V|^2)$ in the worst-case.
- Realize priority queue as a binary heap (see textbook).
  - Each of Line (5) and (8) takes $\Theta(\lg|V|)$.
  - So total: $\Theta((|V| + |E|)\lg|V|)$.
  - This is better than $\Theta(|V|^2)$ if $|E| = o\big(|V|^2/\lg|V|\big)$.