Intro to Algorithms, COMP-160, Homework #12

Benjamin Tanen, 04/29/2016

3. **Let $G = \{V, E\}$ be a weighted directed graph with no negative cycles. However, G contains precisely one negative edge e, from vertex x to vertex y. Do not assume that x is the source.**

   (a) **Show that we can still solve SSSP in $O(E \log V)$ time.**

   Say we are attempting to solve SSSP for a starting node s and an ending node t. We can start by first removing our negative edge e from the graph G to get G'. We can then run Dijkstra's algorithm on G' to return the shortest paths between the nodes of G'.

   From here, we can consider three paths returned from Dijkstra's algorithm: p_1 (the minimized path from s to t in G'), p_2 (the minimized path from s to x in G'), and p_3 (the minimized path from y to t in G'). However, p_3 was never actually computed by our original run of Dijkstra's algorithm, so we must run it again once more (starting from y) to get p_3.

   Since merging p_2, e, and p_3 forms a path from s to t through our negative edge, we can compare this new combined path to our original optimized path p_1. If $|p_1| \leq |p_2| + w(e) + |p_3|$, then we know that our non-negative optimized path is the best one (even if we consider e), so our SSSP from s to t is p_1. (Note: consider $|p|$ to be equal to the sum of the weights for all the edges in a path p).

   If, however, $|p_1| > |p_2| + w(e) + |p_3|$, we can see that the negative edge optimizes the path from s to t. Thus, our SSSP from s to t would be the merged path of p_2, e, and p_3.

   (b) **How fast can we solve SSSP if we have more negative edges (say, $k > 1$)?**

   In order to solve SSSP for a graph G with k negative edges, we can use a similar method to that used for part (a). However, in part (a), we considered two different paths: one with our negative edge e and one without it. We can see this as all combinations of using our negative edge e, otherwise expressed as $\binom{1}{0} + \binom{1}{1} = 2$ runs of Dijkstra's.

   If we expand this out to include k negative edges, we must consider all combinations of including these k negative edges. Since we can form any combination of our edges and choose not to include them, we can express the total combinations of paths to consider as $\binom{k}{0} + \binom{k}{1} + \ldots + \binom{k}{k} = \sum_{i=0}^{k} \binom{k}{i} = O(2^k)$.

   Since we must first run Dijkstra's algorithm to build our table of minimized paths, we encounter $O(E \log V)$ work originally. We then must do comparisons between all of our 2^k paths, where each time we must run Dijkstra's from the end point of our removed negative edge since this is a new source. This, therefore, adds $O(2^k)O(E \log V)$, bringing our total work to $O(2^k) \cdot O(E \log V) = O(2^k E \log V)$.

   Note: if we notice that if k is large enough such that $O(2^k E \log V) + > O(EV)$, we should simply use Bellman-Ford to improve our search.