Illinois Institute of Technology
Department of Computer Science

Homework 7 Solutions

CS 430 Introduction to Algorithms
Fall Semester, 2010

1. **Problem 23.2-3 on page 637**.

   Solution: The runtime of Prim's algorithm is $O(V \lg V + E \lg V)$ with a binary heap implementation and $O(V \lg V + E)$ with a Fibonacci heap implementation. If $|E| = \Theta(V)$, then both implementations are $O(V \lg V)$. However, if $|E| = \Theta(V^2)$, then the binary heap implementation is $O(V^2 \lg V)$ whereas the Fibonacci heap implementation is $O(V^2)$. Hence the Fibonacci heap implementation is faster for dense graphs. If $|E| = \omega(V)$ (which implies that $|V| = o(E)$), then the Fibonacci heap implementation is $o(E \lg V)$ which is faster than the binary heap implementation which is $O(E \lg V)$.

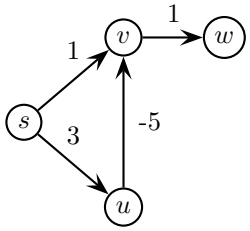
2. **Problem 23-3(a) on page 640**.

   Solution: We shall prove this by contradiction. Suppose that a bottleneck spanning tree (BST) is not a minimum spanning tree (MST). For a given undirected graph G, let T be a MST and T' be a BST where $e \in T$ and $e' \in T'$ are the edges with the maximum weight in their respective trees. Since T' is a BST and T is not a BST, $w(e') < w(e)$. This implies that $e \notin T'$, otherwise the max weight of T' would be the same as T. If we add e to T', this creates a cycle. Since $w(e) > w(e')$, this makes e the edge with the largest weight in the cycle. Therefore T is not a MST since substituting another edge in the cycle for e would result in a tree with less weight. Hence, we have a contradiction, so our assumption that a BST is not a MST is incorrect.

3. **Problem 24.3-2 on page 663**.

   Solution: Consider the following graph. If we use Dijkstra's algorithm to find the shortest paths from vertex s, it will relax edges (s, u) and (s, v). At this point $v.d = 1 < u.d = 3$, so the next vertex to be extracted from the heap will be v. Relaxing edge (v, w) results in $w.d = 2$. Next u is extracted from the heap and edge (u, v) is relaxed resulting in $v.d = -2$ and $v.\pi = u$. However, since v has already been extracted from the heap, the edge (v, w) is not relaxed again. Thus $w.d$ remains 2 when the cost of the shortest path from s to w is really -1.

   The proof of Theorem 24.6 does not work with negative weight edges since when a vertex u is added to the set S, $u.d$ may not be $\delta(s, u)$. A negative edge from a vertex in $V - S$ may result in a path with weight $\delta(s, u)$.



4. **Problem 25.2-6 on page 700**.

   Solution: There are at least two ways to detect negative weight cycles.

   (a) Check the entries along the main diagonal of the matrix $D^{(n)}$. If $d_{ii}^{(n)} < 0$ for some vertex i, then there is a negative weight cycle.

   (b) Run the FLOYD-WARSHALL algorithm for another iteration and check if any of the values in the matrix D change. If there is a negative weight cycle, then some shortest path cost will decrease with the extra iteration. If there are no negative weight cycles then none of the shortest path costs will decrease with an extra iteration, since they are already optimal.

5. **Problem 25.3-4 on page 705**.

   Solution: It changes the shortest paths. Consider the following graph. The min weight is -10 so we add 10 to every weight to make $\hat{w}$. With w, the shortest path from s to y is $s \to x \to y$ with weight 4. With $\hat{w}$, the shortest path from s to y is just $s \to y$ with weight 15 (the path $s \to x \to y$ now has weight 24). By adding the same constant amount to each edge, paths with more edges are penalized even if their weights are relatively low.