CS180 Homework 4

Due: 8:00pm, 2/7/2019

1. Given n files and an array $f[1..n]$, where $f[i]$ stores the access frequency count of file i. You are asked to build a binary tree and put n files as n *leaves* of the tree. The cost of accessing each file i is the depth of the leaf where file i is placed at. The goal is to minimize the total cost of all accesses $\sum_{i=1}^{n} f[i] \cdot \text{depth}(i)$. In class, we have seen an $O(n \log n)$ time algorithm builds the optimal binary tree using the heap. A student suggested in class that if the array f is *sorted* in increasing order, the optimal binary tree can be constructed in $O(n)$ time. Give an $O(n)$ algorithm assume f is sorted and prove your algorithm works.

2. Given a list of n items such that the order of any two items can be determined by one comparison. Design an $O(n)$ algorithm to construct a heap contains all n items.

3. Given a directed weighted graph $G = (V, E)$ with every edge e has weight $w(e) > 1$. Assume G contains a node s that can reach all other nodes.

   (a) Prove that there exists a directed tree T in G with root s as a subgraph in G, with the property that the path from s to any other node v in T is the shortest path from s to v in G.

   (b) Suppose we square the weights of the edges in G. In other words, we have the same graph G except every edge e has a squared weight $w_1(e) = w(e)^2$. The shortest path tree now might change compared to what you have before. If we change the tree by squaring the weights again, until two successive shortest path trees are identical. Argue that further squaring will not change the shortest path tree. Thus, these shortest paths now can be defined with no reference to the actual weights. i.e., given two paths instead of evaluating whether path p_1 is shorter than path p_2 can be determined just by comparing weights, without having the operation of addition of weights. Describe this characterization of the way of comparing the length of paths.

   Example: Suppose when we started there are just two paths, each of two edges, from s to some v, One path has two edges of weights 5 and 5 and the other 7 and 2. At the beginning, the 7,1 path is the shortest because its length is 9 and the other is 10, but after one squaring we will get 25, 25, and 49, 4, respectively. Now the shortest path is the previous 5,5! Why this happened? The most weighty edge between the two paths is 7, and squaring enough will get it to dominate. If the two paths were tied with respect to the most weighty, then the second most weighty will break the tie, etc.

   (c) Given a directed weighted graph $G' = (V, E)$ such that every edge is bi-directional, i.e., an edge (u, v) in G implies another edge (v, u) and they have same weights. Fix an arbitrary node v in G, and like what we have done in part(c), we can find a stable tree T'_k for G'_k. Prove that T'_k is the minimum spanning tree of G' if ignoring the directions on G'.

   (d) Use the way we compared directed paths before to extend it to compare between two spanning tree which one is "better." Argue that the MST defined as the tree that has the minimum sum of weights of its edges, is the same tree that is smaller than all other trees if we just determine which is smaller among 2 trees by comparison rather than addition.

4. Given a set $\Omega = \{e_1, e_2, ..., e_n\}$ and a collection of sets $S_1, S_2, .., S_n$. Each element e_i has a corresponding set $S_i = \{e_i\}$. You are now given a sequence of commands of type **Union**(e_x, e_y)

interspersed with commands **Query**(e_x, e_y). The union operation takes the two sets one containing e_x, and the other containing e_y (they might be the same set, but not at the beginning) and union it to a single set containing the elements of both. As we go creating sets we query command asks whether the two elements in its argument NOW (at the time you reached it in the sequence) belong to the same set or a different set.

(a) Given a list of $O(n)$ operations of Union and Query. Design an algorithm that "processes" the sequence of commands in $O(n \log n)$ time. To process the sequence means to answer all the queries in the sequence. (Hint: keep a set as a rooted tree with depth that is $O(\log n)$. The Crux is how to union two trees and keep this property.)

(b) Given an undirected weighted graph $G = (V, E)$. Show an implementation of Kruskal's Minimum Spanning Tree(MST) algorithm by using the Union and Query commands. Your implementation should have $O(|E| \log |V|)$ time complexity. (Hint: In Kruskal's algorithm, it builds the MST by processing every edge in G and asks whether two endpoints of an edge belong to two disjoint trees. If so, the algorithm adds the edge making two disjoint trees into a single tree.)

★ Express your algorithm in a well-structured manner. Use pseudo code and the textbook has good examples to follow. Avoid using a long continuous piece of text to describe your algorithm. Start each problem on a NEW page. Unless specified, you should justify the time complexity of your algorithm and why it works. For grading, we will take into account both the correctness and the clarity. Your answers are supposed to be in a simple and understandable manner and sloppy answers are expected to receiver fewer points.

★ Homework assignments are due on Gradescope. Email attachments or paper submissions are NOT acceptable.

★ Raw photo is NOT acceptable. Upload your homework as a PDF scan by using a scanner or mobile scanner app. Match each problem with your answer on Gradescope. Use dark pen or pencil and your handwriting should be clear and legible.

★ We recommend using LaTeX, LyX or other word processing software for writing the homework. This is **NOT** a requirement but it helps us to grade and give feedback.