Fundamental Algorithms, Assignment 9

Due April 13/14 in Recitation

For every complex problem there is a simple solution. And it's always wrong.
H.L. Mencken, 1880-1956, American satirist.

1. (*) Suppose that the Huffman Code for $\{v, w, x, y, z\}$ has 0 or 1 as the code word for z. *Prove* that the frequency for z cannot be less than $\frac{1}{3}$. *Give an example* where the frequency for z is 0.36 and z does get code word 0 or 1.

2. (a) What is an optimal Huffman code for the following code when the frequencies are the first eight Fibonacci number?

   $$a : 1, b : 1, c : 2, d : 3, e : 5, f : 8, g : 13, h : 21$$

   (b) The Fibonacci sequence is defined by initial values $0, 1$ with each further term the sum of the previous two terms. Generalize the previous answer to find the optimal code when there are n letters with frequencies the first n (excluding the 0) Fibronacci numbers.

3. Suppose that in implementing the Huffman code we weren't so clever as to use Min-Heaps. Rather, at each step we found the two letters of minimal frequency and replaced them by a new letter with frequency their sum. How long would that algorithm take, in Thetaland, as a function of the initial number of letters n.

4. **DO NOT SUBMIT** Consider the undirected graph with vertices $1, 2, 3, 4, 5$ and adjacency lists (arrows omitted) $1 : 25$, $2 : 1534$, $3 : 24$, $4 : 253$, $5 : 412$. Show the d and π values that result from running BFS, using 3 as a source. Nice picture, please!

5. Show the d and π values that result from running BFS on the undirected graph of Figure A, using vertex u as the source.

6. We are given a set V of boxers. Between any two pairs of boxers there may or may not be a rivalry. Assume the rivalries form a graph G which is given by an adjacency list representation, that is, $Adj[v]$ is a list of the rivals of v. Let n be the number of boxers (or nodes) and r the number of rivalries (or edges). Give a $O(n + r)$ time algorithm that determines whether it is possible to designate some of boxers as

GOOD and the others as BAD such that each rivalry is between a GOOD boxers and a BAD boxer. If it is possible to perform such a designation your algorithm should produce it.

Here is the approach: Create a new field TYPE[v] with the values GOOD and BAD. Assume that the boxers are in a list L so that you can program: For all $v \in L$. The idea will be to apply BFS[v] – when you hit a new vertex its value will be determined. A cautionary note: BFS[v] might not hit all the vertices so, just like we had DFS and DFS-VISIT you should have an overall BFS-MASTER (that will run through the list L) and, when appropriate, call BFS[v].

Note: The cognescenti will recognize that we are determining if a graph is bipartite!

7. **DO NOT SUBMIT**Show how DFS works on Figure B. All lists are alphabetical except we put R before Q so it is the first letter. Show the discovery and finishing time for each vertex.

8. Show the ordering of the vertices produced by TOP-SORT when it is run on Figure C, with all lists alphabetical.

9. **DO NOT SUBMIT** Let G be a DAG with a specific designated vertex v. Uno and Dos (Spanish for One and Two) play the following game. A token is placed on v. The players alternate moves, Uno playing first. On each turn if the token is on w the player moves the token to some vertex u with (w, u) an edge of the DAG. When a player has no move, he or she loses. Except for the first part below, we assume Uno and Dos play perfectly.

   (a) Argue that the game *must* end. Indeed, argue that if G has n vertices then the game *cannot* take more than $n-1$ moves. (Key: Its a DAG!)

   (b) Define VALUE[z] to be the winner of the game (either Uno or Dos) where the token is initially placed at vertex z and Uno plays first. (That is, VALUE[z] being Uno means that the player who has the move will win, VALUE[z] being Dos means that the player who has the move will lose.) When z is a leaf node and Uno plays first, Uno has no move and so loses and therefore VALUE[z] is Dos. But what if z is *not* a leaf node. Suppose the VALUE[w] are known for all $w \in Adj[z]$. How do those values determine VALUE[z]? (To give part of the answer: Suppose there is some $w \in Adj[z]$ with

`VALUE[w]` equal Dos. From z Uno's winning strategy is to move to w.)

(c) Using the above idea modify `DFS-VIST[v]` to find who wins the original game. In your modified algorithm there will be an extra function `VALUE[w]` which is originally set to `NIL` for all vertices w, representing that the winner of the game starting at w has not yet been determined. When the unmodified `DFS-VISIT[w]` would be finished add a couple of lines of pseudocode to give `VALUE[w]`. Give an upper bound on the time of your algorithm.

I cannot live without people. – Pope Francis