Homework 4
Fundamental Algorithms, Fall 2004, Professor Yap

Due: Thu Nov 18, in class

INSTRUCTIONS:

- Please read carefully.
- General remark: we do encourage students to work in groups for discussion. However, when you finally write up the solutions, they *must* represent your individual work.

---

1. (30 Points) DFS
   Note that the graph in part (a) is a bigraph, but parts (b) and (c) it is a digraph.
   (a) Suppose T is the DFS Tree for a connected bigraph G. Let $u{-}v$ be a non-tree edge. Prove that either u is an ancestor of v or vice-versa.
   (b) Let $G = (V, E)$ be a digraph. Suppose we call $DFS((V, E; s_0))$ in which INIT$((V, E; s_0))$ colors s_0 as `seen`, but every node as `unseen`. Prove every vertex that is reachable from s_0 will be seen. NOTE: vertex u is **reachable** from s_0 iff there is a path from s_0 to u.
   (c) Let $G = (V, E; s_0)$ be a digraph. Fill in the shell subroutines of the DFS Algorithm so that it correctly classifies every edge of G into tree, back, forward, cross and unseen edges. You must briefly argue why your algorithm is correct.

2. (25 Points) Radius and Diameter
   (a) Let $G = (V, E)$ be a connected bigraph. For any vertex $v \in V$ define

   $$\text{radius}(v, G) := \max_{u \in V} \delta(v, u)$$

   where $\delta(v, u)$ is the link distance from u to v. The *center* of G is the vertex v_0 such that radius(v_0, G) is minimized. We call radius(v_0, G) the *radius* of G and denote it by radius(G). Define the *diameter* diameter(G) of G to be the maximum value of $\delta(u, v)$ where $u, v \in V$. Prove that

   $$\text{diameter}(G) \geq \cdot\text{radius}(G) \geq \lceil \text{diameter}(G)/2 \rceil .$$

   (b) Show that for every natural number n, there are graphs G_n and H_n such that $n = \text{radius}(G_n) = \text{diameter}(G_n)$ and $n = \text{radius}(H_n) = \lceil \text{diameter}(H_n)/2 \rceil$.
   (c) Give an efficient algorithm to compute the diameter of a undirected tree (i.e., connected acyclic undirected graph). Please use the "shell" subroutines for BFS or DFS. What is the complexity of your algorithm?

3. (20 Points) Greedy
   (a) Give an algorithm that computes the optimal bin packing for an input (M, w) where $w = (w_1, \ldots, w_n)$ are positive weights. You must reduce this to linear bin packing, by considering all $n!$ permutations of $w_1, \ldots, w_n$.
   (b) What is the complexity of your algorithm?
   (c) Improve your complexity by algorithm by considering only $(n-1)!$ permutations. What is the new complexity of your algorithm?
   (d) Further improve upon (c) by using $2(n-2)!$ permutations only.

4. (25 Points) Huffman
   In our Huffman tree algorithm, we represented the Huffman tree as a binary tree. Now consider a more compact representation of a Huffman tree T by exploiting its Sibling property: suppose T has $k \geq 1$ leaves. Each of its $2k-1$ nodes is identified by its rank, *i.e.*, a number from 0 to $2k-2$. Hence node i has rank i. We use two arrays

   $$W[0..2k-2], \qquad L[0..2k-2]$$

---

of length $2k-1$ where $W[i]$ is the weight of node i, and $L[i]$ is the left child of node i. So $L[i]+1$ is the right child of node i (by the Sibling Property). In case node i is a leaf, we let $L[i]$ denote the **canonical code** for the letter that it represents in Σ. I.e., view Σ as a subset of a universal set U where $U \subseteq \{0,1\}^N$. In reality, U might be the set of ASCII characters and $N = 7$. The transmitter and receiver both know this global parameter N and the set U.

(a) Please implement the RESTORE(u) subroutine in detail, using the above representation. In order to ensure sufficient details, we ask you to use either `C, C++` or `Java`.
(b) Illustrate your algorithm above by showing how to transmit our familiar string `hello world!`.

5. (0 Points)
The problems below carry no credit. Please do not hand such problems; they are for your own practice. We strongly recommend that you do them.

6. (0 Points) Greedy
Do Exercises 1.1, 1.5.

7. (0 Points) Huffman
Do Exercises 2.1, 2.2 and 2.3 in Lecture V.