Homework 4
Fundamental Algorithms, Spring 2003, Professor Yap

Due: Mon April 14, in class

INSTRUCTIONS:

- Please read the questions carefully. When in doubt, please ask.

---

1. (20 Points)
In Lecture V.2, we define a code tree $T = T_C$ corresponding to any prefix code C. Let f be the frequency function that assigns to each leaf u of T a frequency $f(u) \geq 0$. The COST of this tree with frequency f is defined to be $COST(T, f) = \sum_u f(u)d(u)$ where u ranges over the leaves of T and $d(u)$ is the depth of u in T. We noted (without proof) an alternative way to compute the $COST(T, f)$: first extend f so that it assigns a frequency to every node v in T: if v is a leaf, then $f(v)$ is already defined. Otherwise, if the two children v_L, v_R of v have been assigned frequencies, define $f(v) = f(v_L) + f(v_R)$. Alternatively, $f(v)$ is just the sum of all the frequencies of the leaves below v. When v is the root of T, we also write $f(T)$ instead of $f(v)$. Let $C(T, f) = \sum_v f(v)$ where v ranges over all the nodes of T. You are to prove that $COST(T, f) = C(T, f)$.

2. (20 Points)
Consider the insertion of the following sequence of keys into an initially empty tree:

$$1, -1, 2, -2, 3, -3, \ldots, n, -n.$$

Let T_n be the final splay tree. In Lecture VI.2, we gave an insertion algorithm in which splay the tree before inserting the key. Call this Method 1. We noted in class an alternative, Method 2, where you first insert *as in an ordinary binary tree*, and then splay the newly inserted node.
(i) Show T_n for $n = 1, 2, 3, 4$ under Method 1.
NOTE: Please be careful – if T_i is wrong, then T_{i+1} is assumed wrong!
(ii) Show T_n for $n = 1, 2, 3, 4$ under Method 2.
(iii) Formulate a conjecture about the shape of T_n under Method 2. Prove it.

3. (15 Points)
In many applications of binary search trees, we need to maintain more information than just the search key. Thus, in the convex hull application (Lecture VI.4), we assumed that each node u has a reference to its successor $u.\texttt{pred}$ and predecessor $u.\texttt{succ}$. Assume that $u.\texttt{pred} = null$ if u contains the minimum node (i.e., the one with smallest key in the tree); similarly $u.\texttt{succ} = null$ if u is the maximum node. Consider maintaining such pointers in splay trees. What additional steps must be carried out to maintain these pointers when you perform the following splay tree operations?
(i) Rotation. (Careful – this could be considered a trick question.)
(ii) Insertion.
(ii) Deletion.

4. (20 Points)
In the problem of maintaining an upper hull H, we represent H as a splay tree T. Lecture VI.4 gives an outline of the procedure called $\texttt{FindLeftTangentPoint}(p, T)$ where we already know that p (the query point) lies outside the upper hull. You are to flesh out the details of this procedure, reducing all numerical detail computation to just the **left turn predicate**. This predicate is defined[1] for any three points $p, q, r \in \mathbb{R}^2$:

$$LeftTurn(p, q, r) = \begin{cases} +1 & \text{if the path } (p, q, r) \text{ represents a left turn} \\ -1 & \text{if the path } p, q, r) \text{ represents a right turn} \\ 0 & \text{if } p, q, r \text{ are collinear} \end{cases}$$

[1]Note that in computational geometry, "predicates" are often 3-valued as in this instance. Contrast this to the 2-valued (true/false) logic that is more common elsewhere. The LeftTurn predicate is only one example of the class of "orientation predicates". See class notes and Exercise 4.1 for how to implement the LeftTurn predicate as the sign of a determinant. But this information is irrelevant for the present problem.

Note that `FindLeftTangent` could return the special point $v_S = (0, -\infty)$ (South Pole).

ASSUMPTIONS: For simplicity, you may assume that the input points are **non-degenerate** in the sense that $LeftTurn(p, q, r)$ on any *input* points p, q, r will never return a 0, and that the x-coordinate of input points and query point are all distinct. However, your solution must take care of what happens when a node u has no successor or no predecessor.

5. (0 Points)
THIS PROBLEM IS NOT REQUIRED, so do not submit for grading. It is for your practice, and we will publish solution.

Let T be a splay tree on n nodes, and let T' be the result of inserting a new key into T using the standard insertion algorithm. So, the new key appears as a leaf u in T' but in all other respects T and T' are identical. The following will step you through a proof of

$$\Phi(T') - \Phi(T) = O(\lg n). \tag{1}$$

Consider the path $\pi = (u_0, u_1, \ldots, u_k)$ from the root u_0 of T' to $u = u_k$.
(i) Let v_i be the sibling of u_i for $i = 1, 2, \ldots, k$, and let s_i be the size of the subtree rooted at s_i. If u_i has no sibling, then v_i is undefined and s_i is 0. Let Δ_i be the increase in potential of u_i in going from T to T'. Express Δ_i in terms of the s_j's ($j = i, i+1, \ldots, k$).
(ii) Express $\Phi(T') - \Phi(T)$ in terms of the Δ_i's.
(iii) Try to bound the expression from (ii) as a telescoping sum, and deduce (1),