CS 473U: Undergraduate Algorithms, Fall 2006
Homework 1

Due Tuesday, September 12, 2006 in 3229 Siebel Center

Starting with this homework, groups of up to three students can submit or present a single joint solution. If your group is submitting a written solution, please remember to **print the names, NetIDs, and aliases of *every* group member on *every* page**. Please remember to submit **separate, individually stapled** solutions to each of the problems.

---

1. Recall from lecture that a *subsequence* of a sequence A consists of a (not necessarily contiguous) collection of elements of A, arranged in the same order as they appear in A. If B is a subsequence of A, then A is a *supersequence* of B.

   (a) Describe and analyze a **simple** recursive algorithm to compute, given two sequences A and B, the length of the *longest common subsequence* of A and B. For example, given the strings `ALGORITHM` and `ALTRUISTIC`, your algorithm would return 5, the length of the longest common subsequence `ALRIT`.

   (b) Describe and analyze a **simple** recursive algorithm to compute, given two sequences A and B, the length of a *shortest common supersequence* of A and B. For example, given the strings `ALGORITHM` and `ALTRUISTIC`, your algorithm would return 14, the length of the shortest common supersequence `ALGTORUISTHIMC`.

   (c) Let $|A|$ denote the length of sequence A. For any two sequences A and B, let $\text{lcs}(A, B)$ denote the length of the longest common subsequence of A and B, and let $\text{scs}(A, B)$ denote the length of the shortest common supersequence of A and B.
   Prove that $|A| + |B| = \text{lcs}(A, B) + \text{scs}(A, B)$ for all sequences A and B. *[Hint: There is a simple non-inductive proof.]*

   In parts (a) and (b), we are *not* looking for the most efficient algorithms, but for algorithms with simple and correct recursive structure.

2. You are a contestant on a game show, and it is your turn to compete in the following game. You are presented with an $m \times n$ grid of boxes, each containing a unique number. It costs \$100 to open a box. Your goal is to find a box whose number is larger than its neighbors in the grid (above, below, left, and right). If you spend less money than your opponents, you win a week-long trip for two to Las Vegas and a year's supply of Rice-A-Roni™, to which you are hopelessly addicted.

   (a) Suppose $m = 1$. Describe an algorithm that finds a number that is bigger than any of its neighbors. How many boxes does your algorithm open in the worst case?

   (b) Suppose $m = n$. Describe an algorithm that finds a number that is bigger than any of its neighbors. How many boxes does your algorithm open in the worst case?

   *(c) **[Extra credit]**[1] Prove that your solution to part (b) is asymptotically optimal.

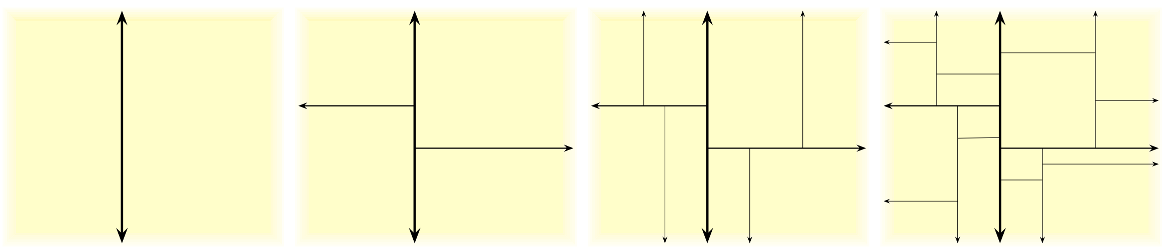
---

[1]The "I don't know" rule does not apply to extra credit problems. There is no such thing as "partial extra credit".

3. A kd-tree is a rooted binary tree with three types of nodes: horizontal, vertical, and leaf. Each vertical node has a *left* child and a *right* child; each horizontal node has a *high* child and a *low* child. The non-leaf node types alternate: non-leaf children of vertical nodes are horizontal and vice versa. Each non-leaf node v stores a real number p_v called its *pivot value*. Each node v has an associated *region* $R(v)$, defined recursively as follows:

   - $R(\mathit{root})$ is the entire plane.
   - If v is is a horizontal node, the horizontal line $y = p_v$ partitions $R(v)$ into $R(\mathit{high}(v))$ and $R(\mathit{low}(v))$ in the obvious way.
   - If v is is a vertical node, the vertical line $x = p_v$ partitions $R(v)$ into $R(\mathit{left}(v))$ and $R(\mathit{right}(v))$ in the obvious way.

   Thus, each region $R(v)$ is an axis-aligned rectangle, possibly with one or more sides at infinity. If v is a leaf, we call $R(v)$ a *leaf box*.

   The first four levels of a typical kd-tree.

   Suppose T is a perfectly balanced kd-tree with n leaves (and thus with depth exactly $\lg n$).

   (a) Consider the horizontal line $y = t$, where $t \neq p_v$ for all nodes v in T. *Exactly* how many leaf boxes of T does this line intersect? *[Hint: The parity of the root node matters.]* Prove your answer is correct. A correct $\Theta(\cdot)$ bound is worth significant partial credit.

   (b) Describe and analyze an efficient algorithm to compute, given T and an arbitrary horizontal line ℓ, the number of leaf boxes of T that lie *entirely above* ℓ.