Cs545 — Homework #2
Amortize analysis, dynamic tables and splay trees
Due 10/11/06

1. The question deals with variants of dynamic tables studied in class. Let $\beta > 1$ be a fixed constant. Consider a table with m cells, containing n keys. The operation of *table doubling* takes place each time that the table is full (that is, $n = m$), and is performed by allocating a new table with βm cells, and copying all the keys from the old table into the new table. The time complexity for this operation is proportional to the size of the new table, $\Theta(\beta m)$.

   (a) Consider a sequence of n insertion operations into a table whose original size is 2. Show that the total time complexity for this sequence is $\leq cn$, where c is a constat. What is the relationship between c and β.

   (b) Let $\gamma < 1/2$ be a constant fixed parameter. The operation *table splitting* (of a table of m cells) is performed by allocating a new table of $\lceil m/2 \rceil$ cells, and copying the keys from the old table into the new table. The time complexity for this operation is $\Theta(m)$. Assume that we perform table-splitting once the number n of keys in the old table is $\leq \gamma m$. Show then any sequence of n operations, each is either insertion or deletion, takes time $O(n)$. It is recommended to use the aggregation/account method of allocating dollars.

2. Let $0 < \alpha < 1/3$ be a fixed parameter. Let T be a binary search tree T. For a node $v \in T$ let n_v denote the number of nodes in the subtree whose root is v. We say that T is a $BB[\alpha]$-tree (also sometimes called as a *weight-balanced tree*) if for every node v, which is not the root, it holds that $n_v \leq (1 - \alpha) \cdot n_{parent(v)}$. In the questions below, assume that T is a $BB[\alpha]$ tree containing n nodes.

   (a) Show that the height of T is $O(\log n)$.

   (b) Show an $O(n)$ time algorithm for constructing a $BB[\alpha]$ tree from a sorted set of n nodes. Hint — consider first the case $\alpha = 1/2$.

(c) Show how to support the operations `insert`(x) and `delete`(x) that insert/delete keys into/from T, such that each operation takes **amortized** time $O(\log n)$, and in addition, T remains $BB[\alpha]$ after each operation.

(d) Show an example of a balanced search tree containing n keys (for arbitrary large n), which is not $BB[\ 0.1\ ]$. The tree must be a red-black tree, an AVL tree or a 2-3 trees (one of the three). If you are not familiar with any of these trees, please contact me.

3. Consider a data structure D for a set $S = \{p_1 \ldots p_n\}$ of points in three-dimensional, such that constructing D takes $\Theta(n^2)$ time, and performing a *nearest-neighbor* query with a query point q takes $O(\log n)$ time. The answer to such a query reports for what is the nearest point of S to q.

   Describe a semi-dynamic version of D, such that performing a nearest-neighbor query takes $O(\log^2 n)$, and adding a new point to D takes amortized time $O(n)$. (So a sequence of n insertions takes $O(n^2)$ time.)

4. Suggest a version of the semi-dynamic data structure studied in class for the semi-dynamic Voronoi-diagram, but based on base-b structure, rather than base-2. So for for every integer i, the data structure might include at most $b - 1$ structures, each constructed for a set of exactly b^i points.

   Under these terms. What would be the query time of the new variant, and what is the amortized time per each insertion, when $b > 2$? express your answer in terms of b and n.

5. Play with the Splay tree simulation in the course webpage.

6. Prove using a potential function that a sequence of n `inc` operations on a binary counter of m bits takes time $O(n + m)$.