UNIVERSITY OF CALIFORNIA
Department of Electrical Engineering
and Computer Sciences
Computer Science Division

CS61B **M. Brudno**
Summer 2000

Homework #5

Due: Tuesday, July 25, 2000

Create a directory to hold your answers to this homework set. To submit your homework, use the command '`submit hw5`' from within the directory containing the file called `hw5.txt`, in which you will place all of your solutions.

1. For each of the following determine whether it is true or provide a counter-example.

a. If the average number of items in the bins of a hashtable is $\Theta(1)$, then the lookup time is $\Theta(1)$.

b. If the largest bin in a hash table has K items in it, then if the items are all moved to a table that has twice as many bins, the largest bin will have $O(k)$ items in it.

c. If we traverse 2 levels in a trie to find a key K, then after inserting one new key into the trie it is possible that we will need to traverse 8 levels to find K.

d. The height of a trie is $\Theta(L)$ where L is the length of the longest string in it.

e. The height of a trie is $\Omega(L)$ where L is the length of the shortest string in it.

f. It takes time $\Theta(n)$ to find the smallest element in a max-heap.

2. Determine whether the following facts are true, provide an explanation for each.

a. A binary tree of height h contains $O(2^(2h))$ nodes.

b. A binary tree containing N nodes has height $\Omega(logn)$

c. A complete binary tree of height h contains $\Theta(2^{h-3})$ nodes.

d. If I add K items to the beginning of a list of N items, and if those K new items are all smaller than the remaining N, then the number of inversions in the list increases by $O(K^2)$.

e. A $\Theta(N)$ algorithm is always preferable to a $\Theta(N^2)$ one.

3. You are given a rectangular n by m array (matrix) of ints, where all rows and columns are sorted: $a_{i0} < a_{i1} < ... < a_{in}$ and $a_{0j} < a_{1j} < ... < a_{mj}$ for all i and j. Your task is to find out whether a certain int is present in the array. The trivial solution is $O(n * m)$, however significantly faster algorithms exist.
Describe (in pseudo-code or English) an algorithm for solving this problem.

4. You are given an array with k elements and are asked to determine whether a certain element within that array constitutes a majority, that is whether $> k/2$ elements in the array are equal to it. The elements are not comparable (that means you can't sort them). The trivial solution is $O(n^2)$, however significantly faster algorithms exist.
Describe (in pseudo-code or English) an algorithm for solving this problem. `Hint:` Think Stack.

5. Goodrich & Tamassia p. 242 C-6.4, C-6.5

6. Design a hash function which would hash matrices (square arrays) of size mxn. Each entry of the matrix can be one a 0, 1 or 2. Your hash function should be robust: collisions should be minimized, especially if two matrices closely resemble each other.