Midterm 1

CSC 282

20 October 2005

*Write your **NAME** legibly on the bluebook. Work all problems. No book, one page (letter size, both sides) of notes OK. Please turn in notes with test.*

1. Adversarial Thinking: 15 Min.

My `Quicksort` (A, p, q) quicksorts the elements between p and q in array A. It computes a pivot as the median of the first, last, and middle elements (i.e. the elements at p, q, and $\lfloor (q-p)/2 \rfloor$.)

Can an adversary force $O(n^2)$ performance? If so say how and if not say why not.

2. Sorting: 10 Min.

A. Design and describe an algorithm to sort n integers in the range $[0, n^2 - 1]$ in $O(n)$ time. (Prose is fine!).

B. Is the performance of your sorting algorithm, on real computers, still $O(n)$ with keys the range $[0, n^4 - 1]$, with $n = 100$?

C. Still using the same unit of computation time (say a single arithmetic instruction), is the performance of your sorting algorithm, on real computers, still $O(n)$ with keys in the range $[0, n^{100} - 1]$, with $n = 100$? If not, what goes wrong and what is the new $O(n)$ time?

3. Knapsacks: 15 Min.

A. Give a $O(nW)$ dynamic programming solution to the 0-1 knapsack problem, where you are given a set of n items with values $v_1 ... v_n$ and integer weight $w_1 ... w_n$, and you want to find the subset of items with the highest total value and with total weight less than an integer W, the maximum weight you can hold in your knapsack.

B. Suppose the maximum weight for any item is k. Give an $O(kn \log n)$ time algorithm for this version of the problem.

4. Binary Tree Induction: 10 Min.

In a binary tree, individual nodes may have two children, one child, or no children. Show by induction that for any configuration of a binary tree, the number of nodes with two children is one less than the number of nodes with no children.

5. Recurrence: 5 Min.

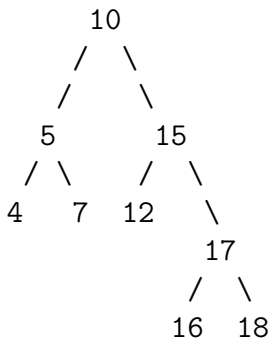
Give an exact solution for the recurrence $T(n) = T(n/2) + n$, $T(1) = 1$.

6. R-B Tree: 10 Min.

A. Consider the problem of testing, given an RB-tree T and three keys a, b, and c, whether there is a downward path in T on which all the three keys appear. Design and describe an $O(\log n)$-step algorithm for this problem, where n is the size of T (prose is fine, pseudocode, whatever...). The output of your algorithm should be "yes" if there is such a path and "no" otherwise.
B. Can you maintain $O(\log n)$ performance as the number of keys rises to its maximum value? If so, convince me.

7. BST 10 Min.

The following diagram represents a binary search tree, showing each node's key:



Draw the tree after *BstDelete* is called on the node with key 15.