CS2223
HW#3

DUE: Monday, November 15

1. (9 points) One possible data structure to implement the abstract data type *priority queue* is a *binary search tree* (described in Chapter 12 of our text). If a node η in a search tree contains *x* and a node in the left subtree of η contains *y* then $y \le x$, and if a node in the right subtree of η contains *y* then $y \ge x$.

.***a*** What is the smallest possible height of a search tree of *n* nodes?

b What is the largest height possible for a search tree of *n* nodes? Describe an input permutation which yields this worst-case height.

c In order to estimate the average height (assuming all input permutations occur with equal probability) of a search tree of *n* nodes, program and test routines to MAKE a search tree, to INSERT a node into a search tree, and to compute the height of a search tree.

- Grow random search trees of 2^3, 2^4, 2^5,… nodes.
- Compute the heights of your trees.
- See if you can find a constant *c* such that the heights are approximately $c\lg n$.

2. (6 points) Assume array *A* is *partially sorted* in that it consists of two increasing runs. That is, there exists a *r*, $1 \le r \le n$, such that $A[i] \le A[i+1]$ if $1 \le i < r$ or if $r < i < n$. **However**, you **don't know** the value of *r*. Prove that if you know that *A* is partially sorted, then you can sort it in worst-case time in $O(n)$.

CS2223
HW#3 SOLUTIONS

1. ***a*** $\lfloor \lg n \rfloor$

b If the input is sorted, in increasing or decreasing order, the height is n-1. Note that these are not the only worst-case instances.

2. The idea is to find r in linear time, and then MERGE $A[1..r]$ with $A[r+1..n]$, again in linear time.

$r \leftarrow 1$

while ($A[r] \le A[r+1]$ **and** $r<n$) **do** $r \leftarrow r+1$ $\quad O(n)$

MERGE ($A[1..r]$, $A[r+1..n]$) $\quad \Theta(n)$