CS2223
HW#2

DUE: Tuesday, November 11

1. (6 points) Order the following functions according to their rates of growth (from fastest growth to slowest). If list includes functions *f* and *g* such that $f(n) \in \Theta(g(n))$, then group them in a set, like $\{f, g\}$.

$$.0001n^4 + 3n^2 + 25$$
$$1/n$$
$$3^{2n}$$
$$\lg n$$
$$n!$$
$$10\log_{10} n$$
$$42$$

2. (8 points) Recall that for function $g: \mathbb{N} \rightarrow \mathbb{R}^+$, the set $O(g)$ was defined as

$$\{ f \mid (\exists c)(\exists n_0)(\forall n \geq n_0)\ f(n) \leq cg(n) \}$$

Prove or give a counterexample to the following conjecture which simplifies the definition.

<u>CONJECTURE</u>: For any $g: \mathbb{N} \rightarrow \mathbb{R}^+$, define $O^*(g) = \{ f \mid (\exists c)(\forall n \in \mathbb{N})\ f(n) \leq cg(n) \}$.

$$O(g) = O^*(g).$$

If the CONJECTURE is true, then you need to show that for any $g: \mathbb{N} \rightarrow \mathbb{R}^+$:

- $O(g) \subseteq O^*(g)$, and,
- $O^*(g) \subseteq O(g)$.

If the CONJECTURE is false, then you need to show a $g: \mathbb{N} \rightarrow \mathbb{R}^+$ which belongs to exactly one of $O(g)$ and $O^*(g)$.

3. (7 points) ***a*** Show that testing if array $A[1..n]$ is a heap has a time complexity in $O(n)$.
b Show that testing if array $A[1..n]$ is a heap has a time complexity in $\Omega(n)$.

4. (10 points) Binary Search Trees, described in Chapter 12 of our text, are an important data structure. Note the definition of the *height* of a tree on pg. 1088 of our text.
a What is the maximum height of all binary search trees with *n* nodes? Give an example of an TREE-INSERTion sequence which yields this height.
b Write and execute a program to estimate the average height of all binary search trees with *n* nodes, assuming that all input INSERTion sequences are equally likely.

CS2223
HW#2 SOLUTIONS

1. $n!$, 3^{2n}, $.0001n^4 + 3n^2 + 25$, { lg$n$, 10log$_{10}n$}, 42, 1/n

2. Clearly if $g \in \{f \mid (\exists c)(\forall n \in \mathbb{N})\, f(n) \le cg(n)\}$ then choosing $n_0 = 0$ satisfies $g \in \{f \mid (\exists c)(\exists n_0)(\forall n \ge n_0)\, f(n) \le cg(n)\}$.

Assume that $(\exists c)(\exists n_0)(\forall n \ge n_0)\, f(n) \le cg(n)$, and let $c^* = \max\left(c, \frac{f(0)}{g(0)}, \ldots, \frac{f(n_0)}{g(n_0)}\right)$.

There are two cases:

- $n \le n_0$ Since $c^* \ge \frac{f(n)}{g(n)}$, then $cg(n) \ge \frac{f(n)}{g(n)} g(n) = f(n)$.
- $n > n_0$ Since $c^* \ge c$, then $c^* g(n) \ge cg(n) \ge f(n)$.

In either case, it follows that $(\forall n \in \mathbb{N})\, c^* g(n) \ge f(n)$.

3. ***a*** Assume we're testing if A is a (max)-heap. A is a (max)-heap if and only if every element is less than or equal to its parent.

> *IsHeap* $\leftarrow$ true
> **for** $i \leftarrow 2$ **to** n **do**
>   **if** $A[i] > A[i/2]$ **then** *IsHeap* $\leftarrow$ false
> **return** *IsHeap*

b Assume that there's an algorithm to test if array $A[1..n]$ is a heap, and the algorithm's time complexity is not in $\Omega(n)$. That is, the algorithm's time complexity is less than linear. Consider the array $A[1..n]$ such that for all $1 \le i \le n$, $A[i]=i$. When the algorithm processes A and returns true, there must be some i^*, $1 \le i^* \le n$, such that the algorithm never examined $A[i^*]$. Consider array A^* such that $A^*[i] = \begin{cases} i, \text{if } i \ne i^* \\ \infty, \text{if } i = i^* \end{cases}$. The algorithm does the same comparisons on A^* that it did on A, and it receives the same answers (since $A^*[i^*]$ was never examined), so it returns true. However, A^* is not a heap, so the algorithm is wrong, which yields a contradiction.