Written Assignment 2 (due by 15:40PM, Oct 31)

1. (8+12%) We say $f(n) = O(h(n))$ if there exist constants $n_0 > 0$ and $c > 0$ so that $0 \leq f(n) \leq ch(n)$ for every $n \geq n_0$. Based on this definition, prove that:

   (a) If $f(n) = O(h(n))$ and $g(n) = O(k(n))$, then

   $$f(n) \cdot g(n) = O\left(h(n) \cdot k(n)\right).$$

   (b) If $f(n) = O(n)$, then $f(n) + \log^8 n = O(n)$.

2. (20%) Let A be an n by n matrix where each entry $A[i][j]$ is a positive integer. We say an entry $A[i][j]$ is a z-entry if $A[i][j]$ is a **multiple** of z. Give an $O(n^2)$-time algorithm to decide whether there exists a *monotonic path* from $A[1][1]$ to $A[n][n]$ (a path that goes only upward and rightward) so that the path visits more 3-entries than 5-entries. Give the pseudocode of your algorithm and explain why it runs in $O(n^2)$ time.

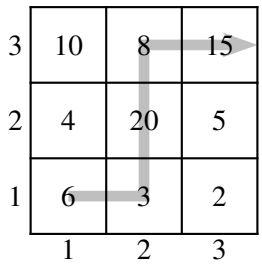

Figure 1: A monotonic path that visits three 3-entries and two 5-entries.

3. (20%) We say a subsequence S of an array A is *bitonic* if there exists an index k so that $S[i] < S[j]$ for every $i < j$, $j \leq k$ and $S[i] > S[j]$ for every $i < j$, $i \geq k$. Devise an algorithm to compute a longest bitonic subsequence of an array A of length n, and analyze its running time. If your algorithm runs in $O(n \log n)$ time, you get full credit. Partial credit will be given for algorithms that have suboptimal running time, but no credit will be given for algorithms that have severe errors.

4. (20%) Given an integer n, the *integer partition problem* asks to output the count of sets whose elements are all positive integers and sum up to n. For $n = 4$, these sets are

   $$\{4\}, \{1, 3\}, \{1, 1, 2\}, \{2, 2\}, \{1, 1, 1, 1\},$$

   so a correct algorithms needs to output 5. Devise an algorithm to solve the integer partition problem, and analyze its running time. If your algorithm runs in $O(n^2)$ time, you get full credit. Partial credit will be given for algorithms that have suboptimal running time, but no credit will be given for algorithms that have severe errors.

5. (20%) Given n stones where each stone s_i has weight $w_i \in \{1, 3\}$ and value $v_i \geq 0$, devise an algorithm to find a subset T of the n stones so that

   $$\sum_{i \in T} w_i \leq m \text{ and } \sum_{i \in T} v_i \text{ is maximized,}$$

   where m is a given parameter, and analyze the running time of your algorithm. If your algorithm runs in $O(n)$ time, you get full credit. Partial credit will be given for algorithms that have suboptimal running time, but no credit will be given for algorithms that have severe errors.

6. (8+17%) It is known that the *element uniqueness problem* has a lower bound of $\Omega(n \log n)$ in the comparison-based model. That is, given n integers $a_1, a_2, \ldots, a_n$, any algorithm in the comparison-based model requires $\Omega(n \log n)$ time to decide whether there exist $a_i = a_j$ for some $i \neq j$. Algorithm $\mathcal{A}$ can output the **k-th mode** of $a_1, a_2, \ldots a_n$, i.e. the value that appears **k-th** most frequently. If there are two values that have the same frequency, then $\mathcal{A}$ can break the tie arbitrarily. If the k-th mode does not exist, then $\mathcal{A}$ outputs "not found." Take $(a_1, a_2, a_3, a_4, a_5, a_6) = (4, 4, 3, 3, 4, 1, 3)$ for example. The mode may be 3, the second mode may be 4, the third mode is 1, and the fourth mode does not exist. We assume that our model is the comparison-based model.

   (a) Show that if $k = 3$, any $\mathcal{A}$ requires $\Omega(n \log n)$ time.

   (b) Show that if $k = \lfloor \log n \rfloor$, any $\mathcal{A}$ requires $\Omega(n \log n)$ time.