CSC 225: Assignment 4

Due at the beginning of class on Thursday Nov. 16

Question	1	2	3	4	5	6	7	8
Marks								

1. We have claimed without proof that if we solve for time complexity for an algorithm like mergeSort with the assumption that $n = 2^k$, that the resulting formula gives the correct asymptotic time complexity for n such that $2^k < n < 2^{k+1}$.

   (a) [4] Prove that $k\, 2^k \in \Omega((k+1)\, 2^{k+1})$.

   (b) [4] Prove that $(k+1)\, 2^{k+1} \in O(k\, 2^k)$.

   (c) [2] What does this tell you about the function $n \log_2 n$ for n such that $2^k < n < 2^{k+1}$?

2. Consider these three recurrence relations that are defined for $n = 2^k$ for some $k \geq 0$:

   $L(n) = n + 2\, L(n/2),\ \ L(1) = 1.$
   $T(n) = 5n + 3 * \log_2(n) + 6 + 2\, T(n/2),\ \ T(1) = 4.$
   $U(n) = 14n + 2\, U(n/2),\ \ U(1) = 14.$

   For parts (a), (b), (c), and (d), answer these questions directly (without solving for a solution to the recurrence relations).

   (a) [5] Prove by induction that $U(n) = 14 * L(n)$.

   (b) [5] Prove by induction that $L(n) < T(n) < U(n)$.

   (c) [3] Prove that $T(n) \in \Theta(L(n))$.

   (d) [2] If the actual number of operations is given by $T(n)$, does it make sense to solve for $L(n)$ instead if the asymptotic time complexity is desired? Justify your answer.

3. You are given a set $\{s_1, s_2, \cdots, s_n\}$ of strings. Each string s_i is over the alphabet $\{a, b, c, \cdots, z\}$. Let $k_i = |\, s_i\, |$ denote the length of s_i, and let $m = \sum_{i=1}^{n} k_i$.

   (a) [10] Give detailed pseudocode for an iterative algorithm (a modification of Radix sort, do not use a recursive approach) that runs in $O(m)$ time to lexicographically sort all the strings. Note that if your algorithm runs in $\Theta(n * Max\{k_1, k_2, \cdots, k_n\})$, this is not a correct solution to this problem and you will not get any marks for this question. As an example for the requirement of sorting lexicographically: the string ‘‘rock’’ is smaller than ‘‘rocket’’ which is smaller than ‘‘rot’’. You can assume that the *jth* symbol of a string can be accessed in $O(1)$ time.

   (b) [5] Prove that your algorithm takes at most $O(m)$ time.

Questions 4-7 refer to the following problem.

Watches are designed so that they remain water-tight under water up to a certain depth, but if a scuba diver wearing the watch goes deeper, the increase in water pressure will cause the watch to leak. Define the *water-resistance of a watch* to be the maximum depth such that the watch does not leak when used by a scuba diver at that depth for one hour. The watch manufacturer wants to determine an integer value for the water-resistance that falls into a range 0 to n. The water-resistance is 0 if the watch leaks when submersed one meter under water. If the watch does not leak even at the depth of n meters, the water-resistance is defined to be n.

To test the water resistance at a depth k, a scuba diver does a dive at k meters and stays under water at that depth for one hour. If the watch leaks, it is discarded. If it does not leak, it can be reused in another experiment.

4. If the number of watches is unlimited, the strategy that minimizes the number of experiments is a binary search.

   (a) [5] How many watches leak in the worst case (worst case here means the maximum number of damaged watches)? Describe the worst case situation and then give a recurrence relation for the number of damaged watches.

   Assume that the midpoint is chosen for the sub-problem for row *lower* to *upper* by the formula $mid = \left\lfloor \frac{lower + upper}{2} \right\rfloor$. The first subproblem has lower = 1 and upper = n. Note: do not make assumptions about n in stating the recurrence- your recurrence should work for any value of n (use floor or ceiling in your formula).

   (b) [5] Use repeated substitution to solve (give a closed formula for) your recurrence from (a) assuming that $n = 2^k - 1$ for some integer k.

5. Given w watches, we propose the following strategy for this problem. We do binary search as in Question 4(a) until only one watch remains. Then a linear search is used.

   (a) [10] Draw the decision tree that represents the experiments done by this algorithm when $n = 10$, and $w = 2$. The left branch should represent the case where a watch leaks and the right branch when it does not leak. Label the nodes with the depth used in the trial.

   (b) [5] What is the worst case complexity of this algorithm in terms of the **number of experiments completed** when $n = 10$ and $w = 2$?

   (c) [5] What is the worst case complexity of this algorithm (number of experiments) for arbitrary n when $w = 2$? Justify your answer by describing the worst case scenario.

6. [10] A student who has not taken this course argues that the strategy suggested for Question 5 must be optimal (minimum number of experiments) in the worst case since linear search is required when there is only one watch and binary search is the optimal strategy when there is no shortage of watches. Prove that this argument is incorrect by providing a decision tree for $n = 10$ and $w = 2$ which corresponds to an algorithm with better worst case complexity. (You might want to look back at your answers to 5(a) and 5(b) when doing this question).

7. Consider the problem from Question #5 for $w = 2$ watches. You may assume that $n = \sum_{i=1}^{p} i$, for some integer $p \geq 1$. The goal is to minimize the number of experiments.

   (a) [5] Describe the optimal algorithm. Hint: Try the case with $n = 15$ if you have not seen the pattern yet.

   (b) [5] What does the decision tree for this algorithm look like?

   (c) [5] What is the worst case complexity of this algorithm (use Θ notation)? (Hint: refer to your answer to Question #8).

8. A function $g(n)$ is a *lower bound* for $f(n)$ if for all n in the range considered (for this problem, it is $n \geq 1$), $g(n) \leq f(n)$. A function $g(n)$ is a *upper bound* for $f(n)$ if for all n in the range considered, $f(n) \leq g(n)$. Parts (a) and (b), concern lower bounds and upper bounds (and are not Big Oh types of questions).

   Let $n = \sum_{i=1}^{k} i$, for some integer $k \geq 1$.

   (a) [5] Prove that $k/2$ is a lower bound for $\sqrt{n}$.

   (b) [5] Prove that $k + 1$ is an upper bound for $\sqrt{n}$.

   (c) [5] Use the definition of Θ given in class and parts (a) and (b) to prove that $k \in \Theta(\sqrt{n})$.