Practice Midterm

Instructions

- **Answer each question on a separate page.**
- **This exam is roughly representative of the difficulty and length of the actual midterm, for which you will have 75 minutes.**
- **You may use any algorithm or theorem we saw in class without proof, as long as you state it correctly. For all questions asking you to give algorithms, you do *not* have to give detailed pseudo-code, or even any pseudo-code. It is enough to give a clear description of your algorithm.**
- **This exam is only for practice. You need not submit your solutions.**

Question 1: (30 points)

Answer true or false for the following. If true, provide a brief justification, if false provide a counterexample.

1. If $f = \Omega(g)$ and $g = \Omega(h)$ then $f = \Omega(h)$.
2. Assume that $f(n) = \Theta(\log_2(n!))$ and $g(n) = \Theta(n \log_2 n)$. Then $f = \Theta(g)$. (Hint: compare $n!$ to $(\frac{n}{2})^{\frac{n}{2}}$ and n^n.)
3. Assume $T(1) = T(2) = 1$ and that for $n > 2$ we have the recurrence
$$T(n) = T\left(\left\lceil \frac{n}{3} \right\rceil\right) + T\left(\left\lceil \frac{2n}{3} \right\rceil\right) + n\ .$$
Then $T(n) = \Theta(n \log_2 n)$. (You can ignore the ceiling function.)

Question 2: (35 points)

Let $A[1, \ldots, n]$ be an array of n *strictly increasing* integers, starting with the number 1. Consider the set of natural numbers denoted by $\mathbb{N} = \{1, 2, 3, 4, \ldots, n, \ldots, \}$. If any of these numbers are not in A, we say it is a *missing number*. In this problem, you will be working to identify the missing numbers based on some constraints.

1. Design a recursive $O(\log n)$-time algorithm that determines the smallest missing number. For example, if $A = [1, 2, 4, 5, 6, 8]$ then this would be 3 whereas if $A = [1, 2, 3, 4, 5, 6]$ then the smallest missing number is 7.
2. In addition to A and n, you are also provided a positive integer $k \geq 1$ and the task is to determine the k-th missing number (so the case $k = 1$ is as before). For example, if $A = [1, 2, 4, 5, 6, 8, 9, 10]$ and $k = 1$, the output should be 3, if $k = 2$ the output should be 7, and if $k = 4$ the output should be 12. Design a recursive $O(\log n)$-time algorithm that returns the k-th missing number given an array A of n integers and the input k.

Question 3: (35 points)

Let $M_k(n)$ be the number of possible strings of length n where the 1's must be no closer than k apart (i.e., there must be at least k zeros separating any two 1s). For example, for $n = 6$ and $k = 2$, "100100" is a valid string, but "100101" is not.

1. For the special case where $k = 1$ justify why $M_1(n) = M_1(n-1) + M_1(n-2)$.
2. Give the recurrence relation that the $M_k(n)$ satisfy. Namely, express $M_k(n)$ as a function of the values $M_k(i)$ for $i < n$ and provide the base case.
3. Design a dynamic programming algorithm that takes as input two natural numbers n and k, and outputs $M_k(n)$. Give its running time, with a brief justification.

Extra practice questions

The following questions are more challenging than the level of the exam, but may be useful for further practice:

Question 4:

Consider a set S of $n \geq 2$ distinct numbers. Call a pair of distinct numbers $x, y \in S$ *close* in S if

$$|x - y| \leq \frac{1}{n-1}(\max_{z \in S} z - \min_{z \in S} z).$$

That is, if the distance between x and y is at most the average distance between consecutive numbers in the sorted order. For example if $S = \{1, 2, 7\}$ then the average distance is $\frac{7-1}{2} = 3$ and so in this case $(1, 2)$ are close.

1. Prove that every set S of $n \geq 2$ distinct numbers contains a pair of close elements.
2. Show that for any real numbers $a < p < b$, and natural numbers $k_1, k_2 > 1$,

   $$\min\left(\frac{p-a}{k_1 - 1}, \frac{b-p}{k_2 - 1}\right) \leq \frac{b-a}{k_1 + k_2 - 2}.$$

   (Hint: You can try to first prove this for the special case: $k_1 = k_2$. You will get half the points for this part for proving it only for the special case.)
   (Hint: Check that $\frac{b-a}{k_1+k_2-2} = \frac{k_1-1}{k_1+k_2-2} \cdot \frac{p-a}{k_1-1} + \frac{k_2-1}{k_1+k_2-2} \cdot \frac{b-p}{k_2-1}$. Now what happens if the inequality does not hold?).
3. Suppose that we partition around a pivot element $p \in S$ organizing the result into two subsets of S: $S_1 = \{x \in S \mid x \leq p\}$ and $S_2 = \{x \in S \mid x \geq p\}$. Prove that for some $k \in \{1, 2\}$ every close pair in S_k is also a close pair in S. (Hint: use part 2. Show that part 2 implies that the average distance between elements in some S_k is at most the average distance between elements in S).
4. Describe an $O(n)$ time algorithm to find a close pair of numbers in S. Justify the correctness and run time bound of your algorithm. (Hint: We would like to keep choosing p in the middle of the set we are considering and then have to consider only one of the sets, S_1 and S_2).