CS 170 Algorithms
Spring 2015 Christos Papadimitriou and Prasad Raghavendra **HW 2**

Due February 6, 5:00pm

Instructions: You are welcome to form small groups (up to four people) to work through the homework, but you **must** write up all your solutions strictly by yourself, and you must acknowledge any ideas you got from others (including from books, papers, web pages, etc.) Please read the collaboration policy on the course web page. List your study partners for homework on the first page, or "none" if you had no partners.

Each problem should begin on a new page. Each page should be clearly labeled with the problem number and the page number of the problem (e.g. Problem 5, page 2 out of 3). The pages of your homework submissions must be in order (all pages of problem 1 in order followed by all pages of problem 2 in order, etc...). You risk receiving no credit for any homework that does not adhere to these guidelines.

On homeworks, we insist that you provide a clear explanation of your algorithms. It is not acceptable to provide a long pseudocode listing with no explanation. In our experience, in such cases the algorithm usually turns out to be incorrect. Even if your algorithm turns out to be correct, we reserve the right to deduct points if it is not clearly explained. We will not grade messy or unreadable submissions. If a problem can be interpreted in more than one way, clearly state the assumptions under which you solve the problem. In writing up your homework you are allowed to consult any book, paper, or published material. If you do so, you are required to cite your source(s). Model solutions will be made available after the due date.

No late homeworks will be accepted. No exceptions. Please don't ask for extensions. We don't mean to be harsh, but we prefer to make model solutions available shortly after the due date, which makes it impossible to accept late homeworks. Out of a total of approximately 12 homework assignments, the lowest two scores will be dropped.

This homework is due Friday, February 6, at 5:00pm electronically. You need to submit it via Gradescope. Please ask on Piazza for details on Gradescope and format.

1. (15 pts.) Recurrences

Solve the following recurrence relations and give a bound for each of them. Give a short justification for each of your answers.

(a) $T(n) = 7T(n/2) + n^2 + \log n$

(b) $T(n) = T(n-1) + 2/n$

(c) $T(n) = T(\sqrt{n}) + 10$

2. (10 pts.) k-ary Search

In binary search, in order to determine the index of an element in a sorted array, we compared to the element to the median of the array, and thus were able to recursively search only half the array.

We can generalize this approach for any constant $k \geq 2$, by checking $k-1$ indices in the sorted array, and similarly recursing on $1/k$ of the array.

Show that such a generalized k-ary search, for any k, would still have the same asymptotic runtime as binary search.

3. (15 pts.) Peak Elements

Given an array with n distinct elements, define a peak element to be one that is strictly larger than both of its neighbors (or larger than its only neighbor, if it is on the edge of the array). For example, in the array $[2,7,1,8]$, there are two peak elements: the 7 and the 8.

Devise an efficient algorithm to find the index of a peak element in an array. If an array has multiple peak elements, you may return any of them.

4. (20 pts.) Overlapping Intervals

You are given a list of n intervals $[x_i, y_i]$, where x_i, y_i are integers with $x_i \leq y_i$. The interval $[x_i, y_i]$ represents the set of integers between x_i and y_i. For instance, the interval $[3,6]$ represents the set $\{3,4,5,6\}$. Define the *overlap* of two intervals I, I' to be $|I \cap I'|$, i.e. the number of integers that are members of both intervals.

Devise a divide-and-conquer algorithm that, when given n intervals, finds and outputs the pair of intervals with highest overlap (you may resolve ties arbitrarily). A trivial $\Theta(n^2)$ algorithm can be achieved by comparing all pairs of intervals; look for something better.

Hint: Try splitting the list using the left endpoints of the intervals.

5. (20 pts.) Find the Missing Integer

An unsorted array A of length n contains all the integers from 0 to n except one. In this problem, we cannot access an entire integer in A with a single operation. The elements of A are represented in binary, and the only operation we can use to access them is "fetch the jth bit of $A[i]$". Using only this operation to access A, give an algorithm that determines the missing integer by looking at only $O(n)$ bits.

Note that there are $O(n \log n)$ bits total in A, so we can't even look at all the bits. This means, for example, that we cannot add up all the numbers in A, which requires looking at all the bits in A. Your overall algorithm, however, may take up to $O(n \log n)$ time and $O(n \log n)$ space.

6. (20 pts.) Polynomials and Complex Numbers

In the following parts, let ω be the n-th root of unity, where n is a multiple of 4.

(a) Calculate $\omega^{2015n/4}$.

(b) Calculate $1 + \omega^3 + \omega^6 + ... + \omega^{3n}$.

(c) Calculate $\omega^{3n/2+1} + \omega$.

7. (1 pts.) Extra Credit: Dealing with Infinity

*Note: This is an **optional** bonus problem. We encourage you to solve it if you want an extra challenge; however, this is probably not the most time-efficient way to improve your grade in the course.*

(a) You are given a sorted array A of size n, but you *do not know* n. The data structure is implemented so that calls $A[i]$ return the error message "∞" if $i > n$. Give an $O(\log n)$ algorithm that on input x returns i such that $A[i] = x$ if such an i exists.

(b) Now suppose your computer will explode if it incurs more than $\log_k n$ "∞" errors, where $k \geq 2$ is given. Given a $O(\log_\alpha n)$ algorithm for the same task under this new constraint (i.e. your computer does not explode), where $\alpha = k/(k-1)$.