CS 170 Efficient Algorithms and Intractable Problems
Spring 2016 Alessandro Chiesa and Umesh Vazirani **HW 2**

Due February 1, 11:59pm

1. (15 pts.) Complex numbers review *Briefly justify your answers to parts (ii) and (iii).*

(i) Write each of the following numbers in the form $\rho(\cos\theta + i\sin\theta)$ (for real ρ and θ):

(a) $-\sqrt{3}+i$

(b) The three 3-rd roots of unity

(c) The sum of your answers to the previous item

(ii) Let $sqrt(x)$ represent one of the complex square roots of x, so that $sqrt(x) = \pm\sqrt{x}$. What are the possible values of $sqrt(sqrt(-1))$?

You can use any notation for complex numbers, e.g., rectangular, polar, or complex exponential notation.

(iii) Let $A(x) = ax^2+bx+c$ and $B(x) = dx^2+ex+f$. Define $C(x) = A(x)B(x)$. If $A(3) = 7$ and $B(3) = -2$, do you have enough information to determine $C(3)$? If so, what is the value of $C(3)$? If not, explain why not.

2. (15 pts.) Two sorted arrays

(a) **(15 pts.)** You are given two sorted arrays, each of size n. Give as efficient an algorithm as possible to find the k-th smallest element in the union of the two arrays. What is the running time of your algorithm as a function of k and n?

(b) **(5 pts. of optional extra credit)** Prove your algorithm is optimal in the comparisons model.

3. (15 pts.) Peak element

Prof. B. Inary just moved to Berkeley and would like to buy a house with a view on Euclid Ave. Needless to say, there are n houses on Euclid Ave, they are arranged in a single row, and have distinct heights. A house "has a view" if it is taller than its neighbors. For example, if the houses heights were $[2,7,1,8]$, then 7 and 8 would "have a view".

Devise an efficient algorithm to help Prof. Inary find a house with a view on Euclid Ave. (If there are multiple such houses, you may return any of them.)

4. (20 pts.) Majority Elements

An array $A[1\ldots n]$ is said to have a *majority element* if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be **no** comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as GIF files, say.) However you *can* answer questions of the form: "is $A[i] = A[j]$?" in constant time.

(a) Show how to solve this problem in $O(n\log n)$ time. (Hint: Split the array A into two arrays A_1 and A_2 of half the size. Does knowing the majority elements of A_1 and A_2 help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

(b) Can you give a linear-time algorithm? (Hint: Here's another divide-and-conquer approach:

- Pair up the elements of A arbitrarily, to get about $n/2$ pairs
- Look at each pair: if the two elements are different, discard both of them; if they are the same, keep just one of them
- If $|A|$ is odd, there will be one unpaired element. What should you do with this element?

Show that after this procedure there are at most $n/2$ elements left, and that they have a majority element if A does.)

5. (20 pts.) Squaring vs multiplying: matrices

The square of a matrix A is its product with itself, AA.

(a) Show that five multiplications are sufficient to compute the square of a 2×2 matrix.

(b) What is wrong with the following algorithm for computing the square of an $n \times n$ matrix?

"Use a divide-and-conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$ thanks to part (a). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in $\Theta(n^{\log_2 5})$ time."

(c) In fact, squaring matrices is no easier than multiplying them. Show that if $n \times n$ matrices can be squared in $\Theta(n^c)$ time, then any $n \times n$ matrices can be multiplied in $\Theta(n^c)$ time.

6. (15 pts.) The Hadamard matrices

The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

- H_0 is the 1×1 matrix $[1]$
- For $k > 0, H_k$ is the $2^k \times 2^k$ matrix

$$H_k = \left[\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array}\right]$$

Show that if v is a column vector of length $n = 2^k$, then the matrix-vector product $H_k v$ can be calculated using $O(n \log n)$ operations. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time.

7. (5 pts.) Optional bonus problem: More medians

(This is an *optional* challenge problem. It is not the most effective way to raise your grade in the course. Only solve it if you want an extra challenge.)

We saw in class a randomized algorithm for computing the median, where the expected running time was $O(n)$. Design a algorithm for computing the median where the *worst-case* running time is $O(n)$.

Hint: It's all in finding a good pivot. If you divide the array into small groups of c elements, can you use that to help you a good pivot?