CS 170 Homework 2

Due **2/3/2020, at 10:00pm**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write none.

2 Asymptotic Properties

Provide a proof of each of the following properties of Θ:

(a) For all fixed $c > 0$, $\sum_{i=1}^{n} i^c = \Theta(n^{c+1})$.

(b) Consider $\sum_{i=0}^{n} \alpha^i$, where α is a constant. Three different things may happen, depending on the value of α:

(i) Show that if $\alpha > 1$, $\sum_{i=0}^{n} \alpha^i = \Theta(\alpha^n)$.

(ii) Show that if $\alpha = 1$, $\sum_{i=0}^{n} \alpha^i = \Theta(n)$.

(iii) Show that if $0 < \alpha < 1$, $\sum_{i=0}^{n} \alpha^i = \Theta(1)$.

3 In Between Functions

Find a function $f(n) \geq 0$ such that:

- For all $c > 0$, $f = \Omega(n^c)$
- For all $\alpha > 1$, $f = \mathcal{O}(\alpha^n)$

Give a proof for why it satisfies both these properties.

4 Median of Medians

The QUICKSELECT(A, k) algorithm for finding the kth smallest element in an unsorted array A picks an arbitrary pivot, then partitions the array into three pieces: the elements less than the pivot, the elements equal to the pivot, and the elements that are greater than the pivot. It is then recursively called on the piece of the array that still contains the kth smallest element.

(a) Consider the array $A = [1, 2, \ldots, n]$ shuffled into some arbitrary order. What is the worst-case runtime of QUICKSELECT$(A, n/2)$ in terms of n? Construct a sequence of 'bad' pivot choices that achieves this worst-case runtime.

(b) The above 'worst case' has a chance of occurring even with randomly-chosen pivots, so the worst-case time for a random-pivot QUICKSELECT is $\mathcal{O}(n^2)$.

Let's define a new algorithm BETTER-QUICKSELECT that deterministically picks a better pivot. This pivot-selection strategy is called 'Median of Medians', so that the worst-case runtime of BETTER-QUICKSELECT(A, k) is $\mathcal{O}(n)$.

Median of Medians

1. Group the array into $\lfloor n/5 \rfloor$ groups of 5 elements each (ignore any leftover elements)
2. Find the median of each group of 5 elements (as each group has a constant 5 elements, finding each individual median is $\mathcal{O}(1)$)
3. Create a new array with only the $\lfloor n/5 \rfloor$ medians, and find the true median of this array using BETTER-QUICKSELECT.
4. Return this median as the chosen pivot

Let p be the chosen pivot. Show that for least $3n/10$ elements x we have that $p \geq x$, and that for at least $3n/10$ elements we have that $p \leq x$.

(c) Show that the worst-case runtime of BETTER-QUICKSELECT(A, k) using the 'Median of Medians' strategy is $\mathcal{O}(n)$.

Hint: Using the Master theorem will likely not work here. Find a recurrence relation for $T(n)$, and try to use induction to show that $T(n) \leq c \cdot n$ for some $c > 0$.

5 Two sorted arrays

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?

Hint: You can use techniques similar to those developed in the previous problem.

Give a 3-part solution.

6 Werewolves

You are playing a party game with n other friends, who play either as werewolves or citizens. You do not know who is a citizen and who is a werewolf, but all your friends do. There are always at least two more citizens than there are werewolves.

Your goal is to identify all the citizens exactly.

Your allowed 'query' operation is as follows: you pick two people. You ask each person if their partner is a citizen or werewolf. When you do this, a citizen must tell the truth about the identity of their partner, but a werewolf doesn't have to (they may lie or tell the truth about their partner).

Your algorithm should work regardless of the behavior of the werewolves.

(a) Show how to solve this problem in $O(n \log n)$ queries (where one query is asking a pair of people to identify the other person).

You cannot use a linear-time algorithm here, as we would like you to get practice with divide and conquer.

Give a 3-part solution.

(b) (**Extra Credit**) Can you give a linear-time algorithm?

Give a 3-part solution.

7 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

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

(a) Write down the Hadamard matrices H_0, H_1, and H_2.

(b) Compute the matrix-vector product $H_2 \cdot v$ where H_2 is the Hadamard matrix you found above, and

$$v = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

Note that since H_2 is a 4×4 matrix, and the vector has length 4, the result will be a vector of length 4.

(c) Now, we will compute another quantity. Take v_1 and v_2 to be the top and bottom halves of v respectively. Therefore, we have that

$$v_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Compute $u_1 = H_1(v_1 + v_2)$ and $u_2 = H_1(v_1 - v_2)$ to get two vectors of length 2. Stack u_1 above u_2 to get a vector u of length 4. What do you notice about u?

(d) Suppose that

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

is a column vector of length $n = 2^k$. v_1 and v_2 are the top and bottom half of the vector, respectively. Therefore, they are each vectors of length $\frac{n}{2} = 2^{k-1}$. Write the matrix-vector product $H_k v$ in terms of H_{k-1}, v_1, and v_2 (note that H_{k-1} is a matrix of dimension $\frac{n}{2} \times \frac{n}{2}$, or $2^{k-1} \times 2^{k-1}$). Since H_k is a $n \times n$ matrix, and v is a vector of length n, the result will be a vector of length n.

(e) Use your results from (c) to come up with a divide-and-conquer algorithm to calculate the matrix-vector product $H_k v$, and show that it 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. You do not need to prove correctness.

8 Warmup: FFT

The *n-th roots of unity* are the n complex numbers ω such that $\omega^n = 1$. They are:

$$e^{2\pi i k/n}, \qquad k = 0, 1, 2, \ldots, n-1$$

(a) Find an n-th root of unity ω such that for all n-th roots of unity z, $z = \omega^k$. In other words, find an ω such that all n-th roots of unity are powers of ω.

(b) Show that the squares of the $2n$-th roots of unity are the n-th roots of unity.

Hint: This requires showing two things: (1) the square of every $2n$-th root of unity is an n-th root of unity, and (2) every n-th root of unity is the square of some $2n$-th root of unity.

(c) For a given n-th root of unity $\omega = e^{2\pi i k/n}$, determine all of the $2n$-th roots of unity whose squares are ω.