CS 170 HW 1

Due on 2017-01-29, at 11:59 pm

1 (★) Study Group

List the names and SIDs of the members in your study group.

2 (★★★) Recurrence Relations

Solve the following recurrence relations and give a Θ bound for each of them.

(a) (i) $T(n) = 3T(n/4) + 4n$

(ii) $T(n) = 45T(n/3) + .1n^3$

(iii) $T(n) = T(n-1) + c^n$, where c is a constant.

(b) $T(n) = 2T(\sqrt{n}) + 3$, and $T(2) = 3$. (Hint: this means the recursion tree stops when the problem size is 2)

3 (★★★★) Majority Elements

An array $A[1 \dots n]$ 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. Four part solutions are required for each part below.

(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?

4 (★★★★) 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.

5 (★★★) 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 $H_2 v$, where H_2 is the Hadamard matrix you found above, and $v = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$ is a column vector. Note that since H_2 is a 4×4 matrix, and v is a vector of 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 $v_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$, and $v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$. 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.