CS 170 HW 2

Due on 2019-02-04, at 10:00 pm

1 Study Group

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

2 Three Part Solution

For each of the algorithm-design questions, please provide a three part solution which includes:

1. The main idea **or** pseudocode underlying your algorithm
2. A proof of correctness
3. A runtime analysis

3 Asymptotic Complexity Comparisons

(a) Order the following functions so that $f_i = O(f_j) \iff i \leq j$. Do not justify your answers.

(i) $f_1(n) = 3^n$
(ii) $f_2(n) = n^{\frac{1}{3}}$
(iii) $f_3(n) = 12$
(iv) $f_4(n) = 2^{\log_2 n}$
(v) $f_5(n) = \sqrt{n}$
(vi) $f_6(n) = 2^n$
(vii) $f_7(n) = \log_2 n$
(viii) $f_8(n) = 2^{\sqrt{n}}$
(ix) $f_9(n) = n^3$

(b) In each of the following, indicate whether $f = O(g)$, $f = \Omega(g)$, or both (in which case $f = \Theta(g)$). ***Briefly*** justify each of your answers. Recall that in terms of asymptotic growth rate, logarithmic < linear < polynomial < exponential.

	$f(n)$	$g(n)$
(i)	$\log_3 n$	$\log_4 n$
(ii)	$n \log(n^4)$	$n^2 \log(n^3)$
(iii)	$\sqrt{n}$	$(\log n)^3$
(iv)	2^n	2^{n+1}
(v)	n	$(\log n)^{\log \log n}$
(vi)	$n + \log n$	$n + (\log n)^2$
(vii)	$\log(n!)$	$n \log n$

4 In Between Functions

In this question, we will try to find functions whose rate of growth is strictly between polynomials and exponentials. Specifically, prove or disprove the following claim: If $f : \mathbb{N} \to \mathbb{N}$ is any positive-valued function, then either (1) there exists a constant $c > 0$ so that $f(n) \in O(n^c)$, or (2) there exists a constant $\alpha > 1$ so that $f(n) \in \Omega(\alpha^n)$.

5 Bit Counter

Consider an n-bit counter that counts from 0 to $2^n - 1$.

When $n = 5$, the counter has the following values:

Step	Value	# Bit-Flips
0	`00000`	–
1	`00001`	1
2	`00010`	2
3	`00011`	1
4	`00100`	3
⋮	⋮	
31	`11111`	1

For example, the last two bits flip when the counter goes from 1 to 2. Using $\Theta(\cdot)$ notation, find the growth of the *total* number of bit flips (the sum of all the numbers in the "# Bit-Flips" column) as a function of n.

6 Recurrence Relations

For each part, find the asymptotic order of growth of T; that is, find a function g such that $T(n) = \Theta(g(n))$.

(a) $T(n) = 4T(n/2) + 42n$

(b) $T(n) = 4T(n/3) + n^2$

(c) $T(n) = T(\sqrt{n}) + 1$ (You may assume that $T(2) = T(1) = 1$)

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 $H_2 v$, 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.

8 Fastest Winning Strategy

Suppose a group of n friends, G, are playing a Werewolf-like party game where the group is divided into two sets of people, the citizens and the werewolves. The goal of the citizens is to figure out who the werewolves are or equivalently who the other citizens are. In this game, each person can determine whether someone else is a citizen or a werewolf based on a brief conversation. However, while the citizens will truthfully report whether the person they are paired with is a citizen, the werewolves have no incentive to report the truth. Assuming that there are more than $n/2$ citizens in the group of n people, can you devise efficient algorithms to determine the identities of the citizens assuming each interaction between people takes $O(1)$ time (Two pairs of people cannot interact at the same time). Please provide a three part solution to both problems.

(a) Show how to solve this problem in $O(n \log n)$ time. (Hint: Split the group of people into two sub-groups of equal size. Can you devise an appropriate divide and conquer algorithm to solve the problem? This should give you an $O(n \log n)$ algorithm.)

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