CS 170 Homework 1

Due **9/6/2021, at 10:00 pm (grace period until 11:59pm)**

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.

In addition, we would like to share correct student solutions that are well-written with the class after each homework. Are you okay with your correct solutions being used for this purpose? Answer "Yes", "Yes but anonymously", or "No"

2 Recurrence Relations

For each part, find the asymptotic order of growth of T; that is, find a function g such that $T(n) = \Theta(g(n))$. In all subparts, you may ignore any issues arising from whether a number is an integer.

(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 $T(2) = T(1) = 1$)

3 Euclid GCD

Euclid's algorithm is an algorithm that computes the greatest common divisor between two integers $a, b \geq 0$:

```
function EUCLID-GCD(a, b)
    if a > b then
        return EUCLID-GCD(b, a)
    else if a = 0 then
        return b
    else
        return EUCLID-GCD(b mod a, a)
```

As an introduction to the types of things we'll be doing in CS 170, let's analyze this algorithm's runtime and correctness.

(a) Let $n = a + b$. We will try to write this algorithm's runtime as $\mathcal{O}(f(n))$ for some function f, counting every integer operation (including **mod**) as one operation taking constant time. Starting with EUCLID-GCD(a, b) where $a < b$, what will the arguments be after two recursive calls?

(b) Assume still that $a < b$, and let n' be the sum of a and b after two recursive calls. Show that after two recursive calls, $n' \leq \frac{1}{2}n$. What runtime do we end up with, in the form $\mathcal{O}(f(n))$?

Hint: Use the properties that: (1) b **mod** $a \leq a$, (2) If $b \geq a$, then b **mod** $a \leq b - a$.

```
function EUCLID-GCD(a, b)
    if a > b then
        return EUCLID-GCD(b, a)
    else if a = 0 then
        return b
    else
        return EUCLID-GCD(b mod a, a)
```

(c) Let's look at correctness. We can prove that this algorithm returns the correct answer using a proof by induction. Note that the result of the algorithm does not depend on if $a > b$ or $a \leq b$, so let's focus on the case that $a \leq b$, and the other case follows.

As our base case, let $a = 0$. Argue that EUCLID-GCD$(0, b)$ returns the correct answer for all b.

(d) Now for the inductive step: Assume that EUCLID-GCD(a, b) computes the right answer for all $a \leq b \leq k - 1$. Show that EUCLID-GCD(a, b) computes the right answer for all $a \leq b = k$.

Hint: Use the property that $d \mid a$ and $d \mid b \iff d \mid a$ and $d \mid (b$ **mod** $a)$.

4 Sequences

Suppose we have a sequence of integers A_n, where $A_0, \ldots, A_{k-1} < 50$ are given, and each subsequent term in the sequence is given by some integer linear combination of the k previous

terms: $A_i = A_{i-1}b_1 + A_{i-2}b_2 + \cdots + A_{i-k}b_k$. You are given as inputs A_0 through A_{k-1} and the coefficients b_1 through b_k.

(a) Devise an algorithm which computes A_n **mod** 50 in $\mathcal{O}(\log n)$ time (**Hint:** use the matrix multiplication technique from class). You should treat k as a constant, and you may assume that all arithmetic operations involving numbers of $\mathcal{O}(\log n)$ bits take constant time. [1]

Give a 3-part solution as described in the homework guidelines.

(b) Devise an even faster algorithm which doesn't use matrix multiplication at all. Once again, you should still treat k as a constant.

Hint: Exploit the fact that we only want the answer mod a constant (here 50).

Give a 3-part solution as described in the homework guidelines.

5 Decimal to Binary

Given the n-digit decimal representation of a number, converting it into binary in the natural way takes $O(n^2)$ steps. Give a divide and conquer algorithm to do the conversion and show that it does not take much more time than Karatsuba's algorithm for integer multiplication.

Just state the main idea behind your algorithm and its runtime analysis; no proof of correctness is needed as long as your main idea is clear.

[1] A similar assumption – that arithmetic operations involving numbers of $\mathcal{O}(\log N)$ bits take constant time, where N is the number of bits needed to describe the entire input – is known as the *transdichotomous word RAM model* and it is typically at least implicitly assumed in the study of algorithms. Indeed, in an input of size N it is standard to assume that we can index into the input in constant time (do we not typically assume that indexing into an input array takes constant time?!). Implicitly this is assuming that the register size on our computer is at least $\log N$ bits, which means it is natural to assume that we can do all standard machine operations on $\log N$ bits in constant time.