CS 170 | Efficient Algorithms and Intractable Problems
February 7, 2016 | Liam Leahy

HW 1

Due January 25, 11:59pm

Instructions: You are welcome to form small groups (up to 4 people total) to work through the homework, but you **must** write up all solutions by yourself. List your study partners for homework on the first page, or "none" if you had no partners.

Begin each problem on a new page. Clearly label where each problem and subproblem begin. The problems must be submitted in order (all of P1 must be before P2, etc).

For questions asking you to give an algorithm, you must respond in what we will refer to as the *four-part format* for algorithms: high-level description, pseudocode, running time analysis, and proof of correctness.

Read the **Homework FAQ** post on Piazza before doing the homework for more explanation on the four-part format and other clarifications for our homework expectations.

You risk receiving no credit for any homework that does not adhere to these guidelines.

No late homeworks will be accepted. **No exceptions.** Do not ask for extensions. This is not out of a desire to be harsh, but rather out of fairness to all students in this large course. Out of a total of approximately 13 homework assignments, the lowest two scores will be dropped.

This homework is due Monday, January 25, at 11:59pm via Gradescope. Please submit via PDF.

1. (15 pts.) Compare Growth Rates

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.

	$f(n)$	$g(n)$
(a)	$n^{1/2}$	$n^{2/3}$
(b)	$\log_3 n$	$\log_4 n$
(c)	$2^n + n^5$	$2^n + \log n$
(d)	$n \log(n^4)$	$n^2 \log(n^3)$
(e)	$n^{1.01}$	$n \log^2 n$
(f)	$n \log n$	$(\log n)^{\log n}$
(g)	$\sqrt{n}$	$(\log n)^3$
(h)	2^n	2^{n+1}
(i)	4^n	$n!$

(a) $f = O(g)$ as $g = n^{2/3} = n^{1/2 \cdot 4/3} = f^{\frac{4}{3}}$.

(b) $f = \Theta(g)$ as $f = \log_3 n = \frac{\log_4 n}{\log_4 3} = \frac{g}{\log_4 3}$.

(c) $f = \Theta(g)$ as 2^n is the dominating term in each expression.

(d) $f = O(g)$ as $\log n^k = k \cdot \log n \implies n \cdot \log n^4 = 4 \cdot n \cdot \log n$ and the same for g yields $3 \cdot n^2 \cdot \log n$. Which reduces the comparison to $n \cdot \log n$ vs. $n^2 \cdot \log n$.

(e) $f = \Omega(g)$ as if we take the $\log_2$ of both sides, we find: $\log_2 f = \log_2 \left(n^{1.01}\right) = \log_2 \left(n \cdot n^{0.01}\right) = \log_2 n + 0.01 \cdot \log_2 n$ and $\log_2 g = \log_2 \left(n \cdot \log_2^2 n\right) = \log_2 n + 2 \cdot \log_2 \left(\log_2 n\right)$. Comparing the two, we find that we are merely comparing $0.01 \cdot \log_2 n$ and $2 \cdot \log_2 \left(\log_2 n\right)$.

(f) $f = O(g)$ as we can compare them by taking the logarithm of each. For f, we find that $\log\left(n \cdot \log n\right) = \log n + \log\left(\log n\right)$. For g, we obtain $\log\left(\left(\log n\right)^{\log n}\right) = \log n \cdot \log\left(\log n\right) = O(f) \cdot \log\left(\log\left(n\right)\right)$.

(g) $f = \Omega(g)$ as we can take the logarithm of each and compare the results. For f, we find that $\log\left(n^{1/2}\right) = \frac{1}{2} \cdot \log\left(n\right)$. For g, we obtain $\log\left(\log^3\left(n\right)\right) = 3 \cdot \log\left(\log\left(n\right)\right)$. We are now comparing $\log\left(n\right)$ to $\log\left(\log\left(n\right)\right)$.

(h) $f = \Theta(g)$ as $f = 2 \cdot g$.

(i) $f = O(g)$ as we note that $f = 4 \cdot 4 \cdot 4 \cdots 4$ and $g = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdots n$. As n gets arbitrarily large, the products of the i-th and the $i+1$-th terms ($i \in \{1, \ldots, n-1\}$) should scale much more than those of f (always $4 \cdot 4$). Going by definition, multiplying g by 4^4 (some constant) yields something always greater than f.

2. (15 pts.) Geometric Growth

Prove that the following formula holds, given a positive real number c.

$$\sum_{i=0}^{k} c^i = \begin{cases} \Theta(c^k) & \text{if } c > 1 \\ \Theta(k) & \text{if } c = 1 \\ \Theta(1) & \text{if } c < 1 \end{cases}$$

The moral: in asymptotics, the sum of a geometric series is simply the first term if the series is strictly decreasing, the last term if the series is strictly increasing, or the number of terms if the series is unchanging. This idea underlies the Master theorem for solving recurrences.

Proof: We begin with $\sum_{i=0}^{k} c^i = 1 + c + c^2 + \ldots + c^k = s_k$ and $c \cdot s_k = c + c + 2 + \ldots + c^{k+1}$.

$$\implies c \cdot s_k - s_k = c^{k+1} - 1$$

$$\implies s_k(c-1) = c^{k+1} - 1$$

$$\implies s_k = \frac{c^{k+1}-1}{c-1} \qquad \text{(Assuming } c \neq 1.)$$

$$c > 1 \implies \frac{c^{k+1}-1}{c-1} \in \Theta(c^k).$$

$$c = 1 \implies \sum_{i=0}^{k} c^i = \sum_{i=0}^{k} 1 = (k+1) \in \Theta(k).$$

$$c < 1 \implies \lim_{k \to \infty} \frac{c^{k+1}-1}{c-1} = \frac{0-1}{c-1} = \frac{1}{1-c} \in \Theta(1).$$

□

3. (25 pts.) 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.

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

(b) (i) Consider the recurrence relation $T(n) = 2T(n/2) + n\log n$. We can't plug it directly into the Master theorem, so solve it by adding the size of each layer.
Hint: split up the $\log(n/(2^i))$ *terms into* $\log n - \log(2^i)$*, and use the formula for arithmetic series.*

(ii) A more general version of Master theorem, like the one on Wikipedia, incorporates this result. The case of the master theorem which applies to this problem is:
If $T(n) = aT(n/b) + f(n)$ *where* $a \geq 1$*,* $b > 1$*, and* $f(n) = \Theta(n^c \log^k n)$ *where* $c = \log_b a$*, then* $T(n) = \Theta(n^c \log^{k+1} n)$.
Use the general Master theorem to solve the following recurrence relation:
$T(n) = 9T(n/3) + n^2 \log^3 n$.

(a) (i) Using the Master Theorem, keeping in mind that $f(n) \in O(n^c)$, $1 > \log_4 3$, and $3 \cdot n \leq k \cdot 3 \cdot n$, $k \leq 1, n \gg 1$, then the answer becomes $\Theta(n)$.

(ii) We know that $3 < \log_3(45)$, which means that we can use the Master Theorem's first case as enumerated in Wikipedia. This implies that the answer is $T(n) \in \Theta(n^{\log_3(45)})$.

(iii) This is much simpler than the preceding questions. We can rewrite it as $\sum_{i=0}^{n-1} c^n$. The solution depends on what value c takes on and the results are enumerated and proven in question 2 already.

(iv) We can write this as some summation from $i = 0$ to some unknown value. It must stop when the square root function has been applied to n enough times to reduce it 2 (as $T(2) = 3$). This implies that $n^{1/2^k} = 2 \implies \log_n(2) = \frac{1}{2^k} \implies \log_2(n) = 2^k \implies k = \log(\log(n))$. Therefore, the summation must be from $i = 0$ to k. Thus, we write the recurrence relation complexity as $\sum_{i=0}^{\log(\log(n))} 3 \cdot 2^i \in \Theta(2^{\log(\log(n))}) = \Theta(\log(n))$.

(b) (i) The first few terms are: $n\log n + 2\left(\frac{n}{2}\right) \cdot \log\frac{n}{2} + 4\left(\frac{n}{4}\right) \cdot \log\frac{n}{4} + \cdots$. Which is equal to

$$\begin{aligned}
&= n \cdot \log n + n \cdot \sum_{i=1}^{\log_2 n} \log\frac{n}{2^i} \\
&= n \cdot \log n + n \cdot \sum_{i=1}^{\log n} \log n - \log 2^i \\
&= n \cdot \log n + n \cdot \log^2 n - n \cdot \sum_{i=1}^{\log n} i \\
&= n \cdot \log n + n \cdot \log^2 n - n \cdot \frac{\log n}{2}(\log n + 1) \\
&\in \Theta(n \cdot \log^2 n).
\end{aligned}$$

(ii) Plugging directly in to the Master Theorem, we obtain $\Theta(n^2 \cdot \log^4 n)$.

4. (25 pts.) Integer Multiplication In class, we covered the integer multiplication technique used by Al Khwarizmi and some European countries. With this method, we multiply and divide numbers by 2 at each iteration. However, there's nothing special about the number 2 here – we could perform this technique by multiplying and dividing another number instead. In particular, let's modify this technique to multiply and divide the numbers by 3 at each iteration. Following the format described in the book, we can create two columns, one for each number being multiplied. For each row, we divide the number in the left column by 3, ignoring the remainder, and multiply the number in the right column by 3. To get the final result, we need to add rows of the right column up. Which rows should be added together to get the final result?

Hint: you may have to multiply some rows by a constant.

(a) Show how to multiply the numbers 11 and 13 using your modified multiplication scheme.

(b) Formally describe and prove the correctness of your modified algorithm. Use the four-part format for algorithms (High-level description, pseudocode, running time analysis, and proof of correctness). Your proof of correctness may be brief.

(a)

a	b	remainder
11	13	2
3	39	0
1	117	-

$\implies a \cdot b = 117 + 13 \cdot 2 = 143.$

(b) **Main Idea:** To multiply two integers, a and b, we continually divide a by 3 and multiply b by 3, keeping track of remainders until a is equal to 1. Upon completion, we take the final value of b and add each previous value of b times its respective remainder to find the solution.

Pseudocode:

```
1.  mul($a$, $b$):
2.      if $a$ or $b$ is equal to 0:
3.          return 0
4.      if only one of $a$ and $b$ is negative, remember this and regardless make them both positive
5.      $c$ = mul($\lfloor \frac{a}{3} \rfloor$, $b$)
6.      if $a$ modulus 3 is equal to 1:
7.          $c = c + a$
8.      else if $a$ modulus 3 is equal to 2:
9.          $c = c + 2 \cdot a$
10.     if only one of $a$ and $b$ was negative:
11.         $c = -c$
12.     return c
```

Proof of Correctness: We use the fact that

$$a \cdot b = \begin{cases} 3\left(b \cdot \lfloor \frac{a}{3} \rfloor\right) & \text{if } a \text{ is divisible by 3,} \\ b + 3\left(b \cdot \lfloor \frac{a}{3} \rfloor\right) & \text{if } a \text{ \% 3 is equal to 1,} \\ 2b + 3\left(b \cdot \lfloor \frac{a}{3} \rfloor\right) & \text{if } a \text{ \% 3 is equal to 2.} \end{cases}$$

From this, we can easily see that the algorithm follows this logic exactly when processing. Each respective term for b is multiplied by the remainder and is added to the final sum.

Running Time: $O(n^2)$.

Justification: Let n be the number of ternary digits in a. Every division by 3 is simply moving one ternary digit to the right. There must be $O(n)$ such divisions before we finish the algorithm. There is a possibility of addition each iteration. Therefore, the time taken is $O(n^2)$.

5. (20 pts.) More Integer Multiplication The following algebraic identities can be used to design a divide-and-conquer algorithm for multiplying n-bit numbers. Assume we are multiplying two numbers a and b. Let a_2, a_1, and a_0 be the top $n/3$, middle $n/3$, and bottom $n/3$ bits of a respectively. Define b_2, b_1, and b_0 in the same way. We can write the product $a \cdot b$ as

$$\begin{aligned} a \cdot b &= \left(2^{2n/3}a_2 + 2^{n/3}a_1 + a_0\right)\left(2^{2n/3}b_2 + 2^{n/3}b_1 + b_0\right) \\ &= 2^{4n/3} \cdot a_2b_2 + 2^n \cdot (a_1b_2 + a_2b_1) + 2^{2n/3} \cdot ((a_2b_0 + a_0b_2) + a_1b_1) + 2^{n/3} \cdot (a_1b_0 + a_0b_1) + a_0b_0 \end{aligned}$$

Furthermore, we know the terms in the above expression can be written as follows:

$$\begin{aligned} (a_1b_2 + a_2b_1) &= (a_1 + a_2) \cdot (b_1 + b_2) - a_1b_1 - a_2b_2 \\ (a_0b_2 + a_2b_0) &= (a_0 + a_2) \cdot (b_0 + b_2) - a_0b_0 - a_2b_2 \\ (a_0b_1 + a_1b_0) &= (a_0 + a_1) \cdot (b_0 + b_1) - a_0b_0 - a_1b_1 \end{aligned}$$

Using this, we can derive a recursive multiplication algorithm.

(a) Rewrite the product $a \cdot b$ in terms of as few distinct recursive multiplications as possible (i.e. one such recursive multiplication might be the product $a_0 \cdot b_0$.

(b) Write the recurrence relation for running time $T(n)$ if we perform multiplications in this way.

(c) What is the running time of the algorithm?

(a) Simply plugging in the values from the three given equations enumerated above, we obtain $a \cdot b = 2^{4n/3} \cdot a_2 \cdot b_2 + 2^n((a_1 + a_2)(b_1 + b_2) - a_1 \cdot b_1 - a_2 \cdot b_2) +$ $2^{2n/3}(((a_0 + a_2)(b_0 + b_2) - a_0 \cdot b_0 - a_2 \cdot b_2) + a_1 \cdot b_1) +$ $2^{n/3}((a_0 + a_1)(b_0 + b_1) - a_0 \cdot b_0 - a_1 \cdot b_1) + a_0 \cdot b_0$. Which is a total of 6 multiplications: $(a_0 \cdot b_0, a_1 \cdot b_1, a_2 \cdot b_2, (a_1 + a_2)(b_1 + b_2), (a_0 + a_2)(b_0 + b_2), (a_0 + a_1)(b_0 + b_1))$.

(b) $T(n) = 6T(\frac{n}{3}) + O(n)$.

(c) Using the Master Theorem, we find that $c = 1 < \log_3 6$ which means that we can apply the first case of the theorem. We find that $T(n) \in \Theta(n^{\log_3 6})$.