

Homework 1, CS420, Fall 2011

Due 8/30 at recitation

Modified 8/27 11:05

We are going to postpone problem 9 to Homework 2

Due Tuesday, August 30, 5:00, at the recitation. If you can't make the recitation, put it under my door by then, or give it to me on Monday. I'll go over solutions during the recitation.

1. In mathematics, we only try to prove or refute things that are unambiguously true or unambiguously false.

Definition: In mathematics, a *statement* is a claim that must be unambiguously true or unambiguously false.

Henceforth, when we use the term *statement*, we mean a statement that is either true or false, not “neither” and not “maybe.” Determine which of the following are statements:

- (a) Whenever x is a real number, $x^2 > 0$.
 - (b) This sentence is false.
 - (c) $x^2 - 5x + 6 = 0$
 - (d) For every real number x , $x^2 + 5x + 6 = 0$.
 - (e) There exists a real number x such that $x^2 + 5x + 6 = 0$.
 - (f) In every set of positive real numbers, there is a smallest number.
 - (g) For every integer $n > 1$, $n^2 + n + 17$ is a prime number.
 - (h) The following statement in formal logic: “(Eighty people signed up for CS420 for Fall '11) implies (The instructor has an orange Mohawk.)”
 - (i) Disks, points, and the empty set are the only convex sets in the x-y plane that have the same height no matter how they are rotated. ¹
 - (j) Every even number greater than two is the sum of two primes.
 - (k) Whenever two perfect players of tic-tac-toe play, the game always ends in a draw.²
 - (l) Whenever two perfect chess players play, White always wins.
2. For each of the items in the previous exercise that is a statement, try to determine whether it is true or false. Only give an answer if you are certain of it; recognizing whether you can be certain of your answer is part of the exercise. You get half credit for a blank answer.
 3. Use the technique of “hurting your case” to get a big- Θ bound for $\sum_{i=1}^n i^2$.

¹A set of points is *convex* if every line segment joining two points of the set consists of points in the set. In other words, it can't have holes or indentations.

²A perfect player is one who analyzes all possible sequences of moves, starting from the current configuration, and selects a move that leads to the best outcome that he can force with certainty. In other words, he selects a move that leads to a subset of sequences where the best outcome for his opponent is minimized.

4. Compare the following pairs, and tell whether $f(n) = o(g(n))$, $\Theta(g(n))$, or $\omega(g(n))$. Justify your answer.

- (a) $f(n) = n$ vs. $g(n) = 4^{\lg n}$ (Consider rewriting the 4 as 2^2 and apply identities from pages 55 and 56.)
- (b) $\lg n$ vs. $\log_{10} n$
- (c) n^2 vs $4^{\lg n}$
- (d) 2^n vs. 2^{n+1}
- (e) 2^n vs 2^{2n}
- (f) $n!$ vs. $(n+1)!$
- (g) n^{100} vs 2^n
- (h) 2^n vs e^n
- (i) $\lg n!$ vs n
- (j) $\lg n$ vs $\lg \lg n$
- (k) $(\lg n)^{(\lg n)}$ vs n^3 . The first comparison, above, gives you a hint.

5. For each of the following, use iterative expansion to show that the master theorem gives the right answer. You only need to show what the righthand summation looks like when you have expanded it down to the base case, then justify your closed-form bound. Assume that n is a power of 3.

- (a) $T(n) = 3T(n/3) + n^2$
- (b) $T(n) = 3T(n/3) + n$.
- (c) $T(n) = 9T(n/3) + n$.

6. Get a big- Θ bound. Proceed as in the previous problem, but assume that n is a power of two. *Hint:* $\sum_{i=1}^n 1/i = \Theta(\log n)$.

- $T(n) = 2T(n/2) + n/\lg n$

7. Refresh your memory of binary heaps (Chapter 6.) Study the **Max-Heapify** algorithm, and the book's justification for recurrence at the bottom of page 155, $T(n) = T(2n/3) + \Theta(1)$.

Here's another way to get the bound: use the technique of "hurting your case."

- (a) Make the simplifying assumption that the last level of the tree is full. (This happens whenever the number n of elements is one less than a power of two.) What recurrence do you get in this case?
- (b) Briefly argue that for arbitrary n , the worst-case running time of **Max-Heapify** on n elements is no worse than it is if we boost the number of elements to the next higher integer n' that is one less than a power of two. Therefore, boosting the number of elements to n' can only hurt your case. A skeptic will not object that you are cheating.
- (c) Argue that $n' < 2n$.

- (d) Use this to show that the big-O bound for n is $O(\log n)$.
8. Look at the way the book gets the $O(n)$ bound for **Build-Max-Heap** on page 159. Here's an easier way. To implement **Build-Max-Heap**, if the heap has one element, return. Otherwise, recursively call it on the left subtree, then recursively call it on the right subtree. Then call **Max-Heapify** on the root.
- (a) Prove by induction on n that this recursive implementation is correct. Your proof needs to talk about the precondition of **Max-Heapify** if you claim that it does what it is supposed to.
- (b) As in the previous problem, we can boost n to the next higher integer that is one less than a power of two; this can only make our running time worse, appeasing the skeptic. Now both subtrees have the same size. Get a recurrence and use the master theorem.
9. **The multiplication algorithm you learned in grade school is not optimal.**
(Postponed to Homework 2)

The number 3276 is just the value of the polynomial $f(n) = 3x^3 + 2x^2 + 7x + 6$, evaluated at 10. If we want to find $3276 * 2421$, we can multiply the polynomials $f(x) = 3x^3 + 2x^2 + 7x + 6$ and $g(x) = 2x^3 + 4x^2 + 2x + 1$ to get $h(x) = 6x^6 + 10x^5 + 28x^4 + 47x^3 + 40x^2 + 19x + 6$, and evaluate $h(10)$ to get our answer.

Given two integers with a total of n digits that we want to multiply, we can pad the two integers with leading zeros so that our polynomials $f(x)$ and $g(x)$ have a number n' of terms that is the smallest power of two that is as big as the number of digits in the larger integer. We can then find $h()$, then evaluate it at 10. The padding just hurts our case, so a skeptic will not object if we get a better time bound for these larger polynomials.

- Convince yourself that the grade-school method is $\Theta(n^2)$.
- Convince yourself that n' is less than $4n$.
- We can then multiply the two polynomials to get $h()$. Convince yourself that the number of terms in $h()$ is $O(n')$?
- Convince yourself that it takes $O(n')$ time, hence $O(n)$ time, to evaluate $h(10)$.

Work parts *a* and *b* of 30-1 on page 920. I've shown part *c* above, assuming you can show *a* and *b*.

For part *b*, rather than explaining the algorithm, show me that you know how it works by illustrating it on the example of $f(n) = 3x^3 + 2x^2 + 7x + 6$ and $g(n) = 2x^3 + 4x^2 + 2x + 1$. They are implying that you should divide $f(n)$ into $a(n) = 3x + 2$ and $b(n) = 7x + 6$, and $g(n)$ into $c(n) = 2x + 4$ and $d(n) = 2x + 1$, and apply your strategy from part *a*, where *a*, *b*, *c*, and *d* are themselves these smaller polynomials. To illustrate it, do only the following:

- (a) Tell what polynomials result from your three recursive calls.
- (b) Show how to add, subtract, and shift these to get the coefficients of $h(n)$.

Then give a recurrence for the general algorithm and use the master theorem to get a better bound than $\Theta(n^2)$.