Take a moment before reading to recall the following notation: $O(f(n)), o(f(n)), \Omega(f(n)), \Theta(f(n))$. You can find definitions in any good algorithms text (such as *Algorithm Design* by Kleinberg and Tardos) or pithily defined on the very useful Theory Cheat Sheet

1 Introduction: Lower Bounds for *Problems*, not only for *Algorithms*

By now you've seen many arguments that provide upper bounds on the running times (or space usage) of algorithms. But how do you know whether an algorithm is optimal: that is, how do you know whether or not there might be some other algorithm that performs asymptotically better? To answer such a question, it would seem that we need to be able to say something about the *inherent difficulty of the problem* X that the algorithm solves. Something of the form: "*Any* algorithm that solves problem X takes (in the worst case—or perhaps on average) at least $\Omega(f(n))$ time". That sounds hard. [Spoiler alert: It is.]
The good news is that in certain cases it can be done and the methods used—as well as the results themselves—are worth learning something about. That's the goal for this week.

2 How Can We Reason About *All* Algorithms That Solve X?!

Ok, we probably can't. But we *can* reason about families of algorithms that solve X. Said differently, we can specify a *model of computation* and then argue about how long any algorithm consistent with that model might take to solve problem X. We'll look at the following models

Comparison-based: The only operations on data are comparisons.

Decision Tree: We model the algorithm base on its conditional statements. Of particular note are

- Linear Decision Tree
- Algebraic Decision Tree

Algebraic Computation Tree

As we explore these models, we'll use certain kinds of arguments, including

- Adversary (oracle) arguments
- Topological arguments
- Problem reduction arguments

3 Game Plan

Unlike most of the weekly assignments, this one has readings that are not too demanding (unless you get to the optional final reading by Lubiw). The bulk of the work will be in solving the problems. So for this week, I'd like you and your partner to share in the presentation of problem solutions.

1. Read Blum's notes (Lecture 5) on comparison-based sorting
2. Read Erickson's notes (Lecture 28) on lower bounds
3. Problem: Find a lower bound for number of compares required to merge two sorted lists.
4. Problem: Find a lower bound for number of compares to determine whether y is in array $x[]$.

   Note 1. *Be more specific: Are items comparable?*

5. Problem: Do Problem 2 from Erickson's Lecture 28

   Note 2. *Add hint for part (a)? k^{th} smallest element can be found in $\theta(n)$ time.*

6. Read Blum's notes (Lecture 6) on concrete lower and upper bounds

7. Problem: Find a lower bound for number of compares needed to determine the median of a list of n integers.

   Note 3. *Aim for $n-1$. Challenge: $(3n-3)/2$.*

8. Problem: Find a lower bound for number of compares to find both the max and min elements.

   Note 4. *Aim for $(3n-4)/2$.*

9. Problem You are given 12 balls and a two-pan balance scale (it you're too young to know what this is, Google it!). Of the 12 balls,11 have same weight, one has a different weight. Figure out which ball has the different weight with as few weighings as possible and whether its weight is lower or higher than that of the other 11 balls.

10. Read Erickson's notes (Lecture 29) on adversary arguments and evasive graph properties.[1]

11. Problem: Do Problem 2 from Erickson's Lecture 29

12. Problem: Do Problem 4 from Erickson's Lecture 29

13. Often one can prove a lower bound for a particular problem via *problem reduction*, the same technique that one uses to show that problems are NP-complete. The next problem asks you to use this technique.

14. Problem: Given a set X of n points in the plane, the *convex hull of X* is defined to be the smallest polygon containing X. The *Convex Hull Problem* is that of, given X, producing a description of this polygon as a list of vertices in counter-clockwise order. Prove that any comparison-based [2] algorithm for computing the convex hull of n points must take time $\Omega(n \log n)$ by showing that the problem of sorting n integers can be reduced to the Convex Hull Problem. For bonus points [3], design an $O(n \log n)$ algorithm for computing the convex hull of X.

15. Because the first week is short, I'm not requiring this last item but if you have time you really should at least scan Lubiw's paper: *A Lower Bound for the Integer Element Distinctness Problem*. If you do get a chance, I'd love to hear what you thought of it.

4 What to Present

This week, student A should present the most interesting[4] aspects of the problems assigned. Student B will give an overview of the key[5] ideas from the assigned readings.

[1] Pages 5-6 are intentionally missing!

[2] Here "comparison-based" means that the only operations allowed on the points are those that ask whether a point is to the left of, on, or to the right of, a line determined by two of the other points.

[3] I'm not sure what that even means.

[4] Here "interesting" can refer to both aspects of the problems you successfully solved as well as obstacles to any that you did not.

[5] Similarly, "key" may refer to important points that you mastered as well as ones that you did not.