Note: Your TA may not get to all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. The discussion worksheet is also a resource you can use to practice, reinforce, and build upon concepts discussed in lecture, readings, and the homework.

In this class, we care a lot about the runtime of algorithms. However, we don't care too much about concrete performance on small input sizes (most algorithms do well on small inputs). Instead we want to compare the *long-term (asymptotic)* growth of the runtimes.

Asymptotic Notation: The following are definitions for $\mathcal{O}(\cdot), \Theta(\cdot)$, and $\Omega(\cdot)$:

- $f(n) = \mathcal{O}(g(n))$ if there exists a $c > 0$ where after large enough n, $f(n) \leq c \cdot g(n)$. *(Asymptotically, f grows at most as much as g)*
- $f(n) = \Omega(g(n))$ if $g(n) = \mathcal{O}(f(n))$. *(Asymptotically, f grows at least as much as g)*
- $f(n) = \Theta(g(n))$ if $f(n) = \mathcal{O}(g(n))$ and $g(n) = \mathcal{O}(f(n))$. *(Asymptotically, f and g grow the same)*

If we compare these definitions to the order on the numbers, $\mathcal{O}$ is a lot like $\leq$, Ω is a lot like $\geq$, and Θ is a lot like $=$ (except all are with regard to asymptotic behavior).

1 Asymptotics and Limits

If we would like to prove asymptotic relations instead of just using them, we can use limits.

Asymptotic Limit Rules: If $f(n), g(n) \geq 0$:

- If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$, then $f(n) = \mathcal{O}(g(n))$.
- If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = c$, for some $c > 0$, then $f(n) = \Theta(g(n))$.
- If $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \infty$, then $f(n) = \Omega(g(n))$.

Note that these are all sufficient conditions involving limits, and are not true definitions of $\mathcal{O}$, Θ, and Ω. (you should check on your own that these statements are correct!)

(a) Prove that $n^3 = \mathcal{O}(n^4)$.

(b) Find an $f(n), g(n) \geq 0$ such that $f(n) = \mathcal{O}(g(n))$, yet $\lim_{n\to\infty} \frac{f(n)}{g(n)} \neq 0$.

(c) Prove that for any $c > 0$, we have $\log n = \mathcal{O}(n^c)$.

Hint: Use L'Hôpital's rule: If $\lim_{n\to\infty} f(n) = \lim_{n\to\infty} g(n) = \infty$, then $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$ (if the RHS exists)

(d) Find an $f(n), g(n) \geq 0$ such that $f(n) = \mathcal{O}(g(n))$, yet $\lim_{n\to\infty} \frac{f(n)}{g(n)}$ does not exist. In this case, you would be unable to use limits to prove $f(n) = \mathcal{O}(g(n))$.

2 Asymptotic Notation Practice

(a) For each pair of functions $f(n)$ and $g(n)$, state whether $f(n) = \mathcal{O}(g(n))$, $f(n) = \Omega(g(n))$, or $f(n) = \Theta(g(n))$. For example, for $f(n) = n^2$ and $g(n) = 2n^2 - n + 3$, write $f(n) = \Theta(g(n))$.

(i) $f(n) = n$ and $g(n) = n^2 - n$
(ii) $f(n) = n^2$ and $g(n) = n^2 + n$
(iii) $f(n) = 8n$ and $g(n) = n \log n$
(iv) $f(n) = 2^n$ and $g(n) = n^2$
(v) $f(n) = 3^n$ and $g(n) = 2^{2n}$

(b) For each of the following, state the order of growth using Θ notation, e.g. $f(n) = \Theta(n)$.

(i) $f(n) = 50$
(ii) $f(n) = n^2 - 2n + 3$
(iii) $f(n) = n + \cdots + 3 + 2 + 1$
(iv) $f(n) = n^{100} + 1.01^n$
(v) $f(n) = n^{1.1} + n \log n$
(vi) $f(n) = (g(n))^2$ where $g(n) = \sqrt{n} + 5$

3 Find the valley

You are given an array A of integers of length N. A has the following property: it is decreasing until element j, at which point it is increasing. In other words, there is some j such that if $i < j$ we have $A[i] > A[i+1]$ and if $i \geq j$ we have $A[i] < A[i+1]$. Assuming you do not already know j, give an algorithm to find j.

For simplicity, you may assume that N is a power of 2.

4 Runtime and Correctness of Mergesort

In general, this class is about design, correctness, and performance of algorithms. Consider the following algorithm called *Mergesort*, which you should hopefully recognize from 61B:

```
function MERGE(A[1,...,n], B[1,...,m])
    i, j ← 1
    C ← empty array of length n + m
    while i ≤ n or j ≤ m do
        if i ≤ n and (j > m or A[i] < B[j]) then
            C[i + j − 1] ← A[i]
            i ← i + 1
        else
            C[i + j − 1] ← B[j]
            j ← j + 1
    return C
function MERGESORT(A[1,...,n])
    if n ≤ 1 then return A
    mid ← ⌊n/2⌋
    L ← MERGESORT(A[1,...,mid])
    R ← MERGESORT(A[mid + 1,...,n])
    return MERGE(L, R)
```

Recall that MERGESORT takes an arbitrary array and returns a sorted copy of that array. It turns out that MERGESORT is asymptotically optimal at performing this task (however, other sorts like Quicksort are often used in practice).

(a) Let $T(n)$ represent the number of operations MERGESORT performs given a array of length n. Find a base case and recurrence for $T(n)$, use asymptotic notation.

(b) Solve this recurrence. What asymptotic runtime do you get?

(c) Consider the correctness of MERGE. What is the desired property of C once MERGE completes? What is required of the arguments to MERGE for this to happen?

(d) Using the property you found for MERGE, show that MERGESORT is correct.

5 Median of Medians

The QUICKSELECT(A, k) algorithm for finding the kth smallest element in an unsorted array A picks an arbitrary pivot, then partitions the array into three pieces: the elements less than the pivot, the elements equal to the pivot, and the elements that are greater than the pivot. It is then recursively called on the piece of the array that still contains the kth smallest element.

(a) Consider the array $A = [1, 2, \ldots, n]$ shuffled into some arbitrary order. What is the worst-case runtime of QUICKSELECT(A, $n/2$) in terms of n? Construct a sequence of 'bad' pivot choices that achieves this worst-case runtime.

(b) The above 'worst case' has a chance of occurring even with randomly-chosen pivots, so the worst-case time for a random-pivot QUICKSELECT is $\mathcal{O}(n^2)$.

Let's define a new algorithm BETTER-QUICKSELECT that deterministically picks a better pivot. This pivot-selection strategy is called 'Median of Medians', so that the worst-case runtime of BETTER-QUICKSELECT(A, k) is $\mathcal{O}(n)$.

Median of Medians

1. Group the array into $\lfloor n/5 \rfloor$ groups of 5 elements each (ignore any leftover elements)

2. Find the median of each group of 5 elements (as each group has a constant 5 elements, finding each individual median is $\mathcal{O}(1)$)
3. Create a new array with only the $\lfloor n/5 \rfloor$ medians, and find the true median of this array using Better-Quickselect.
4. Return this median as the chosen pivot

Let p be the chosen pivot. Show that for least $3n/10$ elements x we have that $p \geq x$, and that for at least $3n/10$ elements we have that $p \leq x$.

(c) Show that the worst-case runtime of Better-Quickselect(A, k) using the 'Median of Medians' strategy is $\mathcal{O}(n)$.

Hint: Using the Master theorem will likely not work here. Find a recurrence relation for $T(n)$, and try to use induction to show that $T(n) \leq c \cdot n$ for some $c > 0$.