Note: Your TA probably will not cover all the problems. This is totally fine, the discussion worksheets are not designed to be finished in an hour. They are deliberately made long so they can serve as 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 $\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty$, 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 $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$, 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 Complexity Comparisons

(a) Order the following functions so that for all i, j, if f_i comes before f_j in the order then $f_i = O(f_j)$. Do not justify your answers.

- $f_1(n) = 3^n$
- $f_2(n) = n^{\frac{1}{3}}$
- $f_3(n) = 12$
- $f_4(n) = 2^{\log_2 n}$
- $f_5(n) = \sqrt{n}$
- $f_6(n) = 2^n$
- $f_7(n) = \log_2 n$
- $f_8(n) = 2^{\sqrt{n}}$
- $f_9(n) = n^3$

As an answer you may just write the functions as a list, e.g. $f_8, f_9, f_1, \ldots$

(b) 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. Recall that in terms of asymptotic growth rate, logarithmic $<$ polynomial $<$ exponential.

	$f(n)$	$g(n)$
(i)	$\log_3 n$	$\log_4(n)$
(ii)	$n \log(n^4)$	$n^2 \log(n^3)$
(iii)	$\sqrt{n}$	$(\log n)^3$
(iv)	$n + \log n$	$n + (\log n)^2$

3 Hadamard matrices

The Hadamard matrices $H_0, H_1, H_2, \ldots$ are defined as follows:

- H_0 is the 1×1 matrix [1]
- For $k > 0, H_k$ is the $2^k \times 2^k$ matrix

$$H_k = \left[\begin{array}{c|c} H_{k-1} & H_{k-1} \\ \hline H_{k-1} & -H_{k-1} \end{array}\right]$$

(a) Write down the Hadamard matrices H_0, H_1, and H_2.

(b) Compute the matrix-vector product $H_2 \cdot v$ where H_2 is the Hadamard matrix you found above, and

$$v = \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix}$$

Note that since H_2 is a 4×4 matrix, and the vector has length 4, the result will be a vector of length 4.

(c) Now, we will compute another quantity. Take v_1 and v_2 to be the top and bottom halves of v respectively. Therefore, we have that

$$v_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

Compute $u_1 = H_1(v_1 + v_2)$ and $u_2 = H_1(v_1 - v_2)$ to get two vectors of length 2. Stack u_1 above u_2 to get a vector u of length 4. What do you notice about u?

(d) Suppose that

$$v = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

is a column vector of length $n = 2^k$. v_1 and v_2 are the top and bottom half of the vector, respectively. Therefore, they are each vectors of length $\frac{n}{2} = 2^{k-1}$. Write the matrix-vector product $H_k v$ in terms of H_{k-1}, v_1, and v_2 (note that H_{k-1} is a matrix of dimension $\frac{n}{2} \times \frac{n}{2}$, or $2^{k-1} \times 2^{k-1}$). Since H_k is a $n \times n$ matrix, and v is a vector of length n, the result will be a vector of length n.

(e) Use your results from (c) to come up with a divide-and-conquer algorithm to calculate the matrix-vector product $H_k v$, and show that it can be calculated using $O(n \log n)$ operations. Assume that all the numbers involved are small enough that basic arithmetic operations like addition and multiplication take unit time. You do not need to prove correctness.

4 Monotone matrices

A m-by-n matrix A is *monotone* if $n \geq m$, each row of A has no duplicate entries, and it has the following property: if the minimum of row i is located at column j_i, then $j_1 < j_2 < j_3 \ldots j_m$. For example, the following matrix is monotone (the minimum of each row is bolded):

$$\begin{bmatrix} \mathbf{1} & 3 & 4 & 6 & 5 & 2 \\ 7 & 3 & \mathbf{2} & 5 & 6 & 4 \\ 7 & 9 & 6 & 3 & 10 & \mathbf{0} \end{bmatrix}$$

Give an efficient (i.e., better than $O(mn)$-time) algorithm that finds the minimum in each row of an m-by-n monotone matrix A.

Give a 3-part solution. You do not need to write a formal recurrence relation in your runtime analysis; an informal summary of the runtime analysis such as "proof by picture" is fine.

5 Complex numbers review

A *complex number* is a number that can be written in the rectangular form $a + bi$ (i is the imaginary unit, with $i^2 = -1$). The following famous equation (*Euler's formula*) relates the polar form of complex numbers to the rectangular form:

$$re^{i\theta} = r(\cos\theta + i\sin\theta)$$

In polar form, $r \geq 0$ represents the distance of the complex number from 0, and θ represents its angle. Note that since $\sin(\theta) = \sin(\theta + 2\pi), \cos(\theta) = \cos(\theta + 2\pi)$, we have $re^{i\theta} = re^{i(\theta+2\pi)}$ for any r, θ.

The n-th *roots of unity* are the n complex numbers satisfying $\omega^n = 1$. They are given by

$$\omega_k = e^{2\pi ik/n}, \qquad k = 0, 1, 2, \ldots, n-1$$

(a) Let $x = e^{2\pi i 3/10}, y = e^{2\pi i 5/10}$ which are two 10-th roots of unity. Compute the product $x \cdot y$. Is this an n-th root of unity for some n? Is it a 10-th root of unity?

What happens if $x = e^{2\pi i 6/10}, y = e^{2\pi i 7/10}$?

(b) Show that for any n-th root of unity $\omega \neq 1$, $\sum_{k=0}^{n-1} \omega^k = 0$, when $n > 1$.

Hint: Use the formula for the sum of a geometric series $\sum_{k=0}^{n} \alpha^k = \frac{\alpha^{n+1}-1}{\alpha-1}$. It works for complex numbers too!

(c) (i) Find all ω such that $\omega^2 = -1$.

(ii) Find all ω such that $\omega^4 = -1$.

6 Extra Divide and Conquer Practice: Quantiles

Let A be an array of length n. The boundaries for the k quantiles of A are $\{a^{(n/k)}, a^{(2n/k)}, \ldots, a^{((k-1)n/k)}\}$ where $a^{(\ell)}$ is the ℓ-th smallest element in A.

Devise an algorithm to compute the boundaries of the k quantiles in time $\mathcal{O}(n \log k)$. For convenience, you may assume that k is a power of 2.

Hint: Recall that QUICKSELECT(A, ℓ) gives $a^{(\ell)}$ in $\mathcal{O}(n)$ time.