182 Week 1 Discussion Notes

Colin Ni

April 8, 2025

This week we will talk about Big-O notation and discuss some variants on the stable marriage problem.

Big-O notation

Warmup. Recall that given functions $f, g\colon \mathbb{R} \to \mathbb{R}$, we say that $f(x) \in O(g(x))$ if there exist a scalar $M > 0$ and a point $x_0 \in \mathbb{R}$ such that $|f(x)| \leq M|g(x)|$ for all $x \geq x_0$. Given the picture on the blackboard, discuss whether $f(x) \in O(g(x))$ and $g(x) \in O(f(x))$.

Intuition. Big-O notation captures the idea of one function being asymptotically bigger the other: roughly speaking, $f(x) \in O(g(x))$ means that $g(x)$ is grows at least as fast as $f(x)$ for large x.

In most cases, we can use calculus to express this idea:

Proposition. If the limit $\lim_{x\to\infty} f(x)/g(x)$ exists, then it is finite if and only if $f(x) \in O(g(x))$.

Exercise. Use calculus (or intuition) to order the following functions in such a way that each is O of the next:

$$1, \quad \log n, \quad n, \quad n \log n, \quad \sqrt{n}, \quad n^2 \log n, \quad n^2, \quad n!, \quad n^n, \quad e^n.$$

To get used to the definition of Big-O, let us prove some basic properties.

Basic Properties.

(i) If $f, g \in O(h)$, then $f + g \in O(h)$.

(ii) If $f \in O(h)$ and $g \in O(j)$, then $f \cdot g \in O(h \cdot j)$.

Proof. For (i), let $M_f > 0$ and $x_f \in \mathbb{R}$ be such that $|f(x)| \leq M_f|h(x)|$ whenever $x \geq x_f$, and let $M_g > 0$ and $x_g \in \mathbb{R}$ be such that $|g(x)| \leq M_g|h(x)|$ whenever $x \geq x_g$. Then

$$|(f+g)(x)| \leq |f(x)| + |g(x)| \leq M_f|h(x)| + M_g|h(x)| \leq \max\{M_f, M_g\}|h(x)|$$

whenever $x \geq \max\{x_f, x_g\}$.

For (ii), choose M_f, x_f, M_g, x_g as usual, and observe that

$$|(f \cdot g)(x)| = |f(x)||g(x)| \leq M_f|h(x)|M_g|j(x)| = M_f M_g|(h \cdot j)(x)|$$

whenever $x \geq \max\{x_f, x_g\}$. $\square$

Let us sketch some nontrivial examples, which will be helpful for Problem 2 on HW 1.

Example. If $f, g\colon \mathbb{R}_{>0} \to \mathbb{R}$ and $f \in O(g)$, then $f^a \in O(g^a)$ for any $a > 0$. Indeed, suppose $f \in O(g)$, and let $M > 0$ and $x_0 \in \mathbb{R}_{>0}$ be such that $|f(x)| \leq M|g(x)|$ whenever $x \geq x_0$. Since x^a is defined as $e^{a \ln x}$, it is straightforward to see that

$$|(f^a)(x)| = |f(x)|^a \leq M^a|g(x)|^a$$

whenever $x \geq x_0$.

Example. Let us show that $\log(n!) \in O(n \log n)$ and $n \log n \in O(\log(n!))$ or, in other words, that $\log(n!) \in \Theta(n \log n)$. Write

$$\log(n!) = \sum_{i=1}^{n} \log(i).$$

It is easy to bound this from above by $n \log n$, which shows that $\log(n!) \in O(n \log n)$, so let us sketch out why $n \log n \in O(\log(n!))$. Consider what happens when we throw out the first half of the sum:

$$\sum_{i=1}^{n} \log(i) \geq \sum_{i=\lfloor \frac{n}{2} \rfloor}^{n} \log(i) \geq \frac{n}{2} \log\left(\frac{n}{2}\right).$$

This shows $(n/2)\log(n/2) \in O(\log(n!))$, and using basic properties we can deduce our desired result.

Example. Here is an example of proving the negation. We will show that $x^x \notin O(x^k)$ for any positive integer k. Let $M > 0$ be a scalar and $x_0 \in \mathbb{R}$ be a point. Pick an integer $n \geq x_0$ such that $n > M$ and $n > k + 1$. Then

$$n^n > n^{k+1} = n \cdot n^k > Mn^k.$$

Thus $x^x \notin O(x^k)$.

Stable marriage

Recall. Recall that given an equal number of men and women and their preference list of the people of the opposite gender, the stable marriage problem seeks a stable perfect matching. A perfect matching means that every person is married to exactly one person of the opposite gender, and stable means that there does not exist two couples (m, w) and (m', w') such that m and w' would prefer to be married to each other (*i.e.* m prefers w' over w and w' prefers m over m'). The Gale-Shapley algorithm finds such a stable perfect matching in $O(n^2)$ time.

Let us discuss the Peripatetic Shipping Lines problem from homework.

Problem. There are an equal number of ships and ports, and on each day a ship is scheduled to either be at a port or at sea. The ships visit each port exactly once, and no two ships visit a port on the same day. Now, the company needs each ship to dock at a port indefinitely for repairs. Can this be done?

Example. For example, the schedules may look like this:

Day:	1	2	3	4	5	6	7
S_1:	P_1			P_3		P_2	
S_2:			P_1		P_2		P_3
S_3:	P_2	P_1	P_3				

In this case, we can dock S_1, S_2, S_3 at P_3, P_1, P_2, respectively.

Here are some naive $O(n)$ (or $O(n \log n)$) algorithms that do not work. Observe that from the point of view of each port, there is a unique final ship that visits that port.

- Dock the ship at that port. This fails because S_2 is the final ship for both ports P_1 and P_3.
- Of the ships that have not yet been assigned to dock at a port, dock the final one at that port. This fails because P_3 could be assigned S_2 and then P_2 could be assigned S_1, but now P_1 would be assigned S_3, which blocks S_2 from reaching P_1.
- Each day, if a ship is at a port and it is the last ship of that port, dock the ship there. This fails because then P_3 would be assigned S_3, and this would block S_1 from reaching P_3.

Idea. To solve the problem, we rephrase this in terms of a stable marriage. A match corresponds to a docking, and a perfect match means that the ships and docks are paired up exactly. There are some obvious possible choices of preference lists: the preference list for each ship can consist of the ports it visits in chronological or reverse-chronological order, and the preference list for each port can consist of the ships that visit it in chronological or reverse-chronological order. To decide which order they should be in, note that the only issue is the potential of a blocked port: we want to avoid the scenario of two dockings (S, P) and (S', P') where S needs to visit P' before docking at P but where P' was assigned to dock S' before S was scheduled to visit P'. You can now carefully decide how the preference lists should be constructed, and we will do this during discussion section.