CS 161 (Stanford, Winter 2025) Section 1

Asymptotic analysis

Review of definitions

Let f, g be functions from the positive integers to the non-negative reals.

Definition 1: (Big-Oh notation)

$f(n) = O(g(n))$ if there exist constants $c > 0$ and n_0 such that for all $n \geq n_0$,

$$f(n) \leq c \cdot g(n).$$

Definition 2: (Big-Omega notation)

$f(n) = \Omega(g(n))$ if there exist constants $c > 0$ and n_0 such that for all $n \geq n_0$,

$$f(n) \geq c \cdot g(n).$$

Definition 3: (Big-Theta notation)

$f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$.

Some additional definitions

You will use "Big-Oh notation", "Big-Omega notation", and "Big-Theta notation" A LOT in class. Additionally, you may occasionally run into "little-oh notation" and "little-omega notation":

Definition 4: (Little-oh notation)

$f(n) = o(g(n))$ if **for every constant** $c > 0$ there exist a constant n_0 such that for all $n \geq n_0$,

$$f(n) < c \cdot g(n).$$

Definition 5: (Little-omega notation)

$f(n) = \omega(g(n))$ if **for every constant** $c > 0$ there exist a constant n_0 such that for all $n \geq n_0$,

$$f(n) > c \cdot g(n).$$

1 Asymptotic analysis questions

1.1 Limit definitions of asymptotic relationships

If you've taken a calculus course, the first thing you'll learn is the definition of a limit. For our purposes, we'll only use limits as n approaches ∞:

Definition 6: (Limit at infinity)

$\lim_{n\to\infty} f(n) = c$ if for any $\epsilon > 0$, there exists n_0 such that $|f(n) - c| < \epsilon$ for all $n \geq n_0$.

$\lim_{n\to\infty} f(n) = \infty$ if for any $m > 0$, there exists n_0 such that $f(n) > m$ for all $n \geq n_0$.

(a) Show that if $\lim_{n\to\infty} f(n)/g(n) = c > 0$, then $f(n) = \Theta(g(n))$.

(b) Show that if $\lim_{n\to\infty} f(n)/g(n) = 0$, then $f(n) = o(g(n))$.

(c) Assuming $\lim_{n\to\infty} f(n)/g(n)$ exists, give conditions in terms of this limit that correspond to $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$, and $f(n) = \omega(g(n))$. (No need to prove that your conditions are correct.)

1.2 Properties of asymptotic relationships

Are the following statements true or false? $f(n)$, $g(n)$, and $h(n)$ are all non-negative functions of the integers $0, 1, 2, \ldots$

(a) If $f(n) = O(g(n))$, then $g(n) = \Theta(g(n) + f(n))$.

(b) If $f(n) = O(g(n))$, then $f(n) \cdot h(n) = O(g(n) \cdot h(n))$.

(c) If $f(n) = O(g(n))$, then either $f(n) = o(g(n))$ or $f(n) = \Theta(g(n))$.

(d) If $f(n) = O(g(n))$ and $h(n)$ is an integer-valued function of n that increases to ∞, then $h(f(n)) = O(h(g(n)))$.

(e) If $f(n) = O(g(n))$ and $h(n)$ is an integer-valued function of n that increases to ∞, then $f(h(n)) = O(g(h(n)))$.

(f) If Algorithm 1 runs in time $f(n)$ on inputs of size n, and Algorithm 2 runs in time $g(n)$ on inputs of size n, where $f(n) = o(g(n))$, then we should always prefer Algorithm 1 over Algorithm 2.

2 Induction

You're a chef at an internationally-renowned pancake restaurant, and have just made a big stack of n pancakes. However, you're serving a picky customer who wants the pancakes to be sorted by size, with the largest on the bottom of the stack and the smallest at the top. You can sort the stack by inserting your spatula somewhere in the stack and flipping the pancakes

above that point, reversing their order. Give an algorithm for sorting the stack of pancakes that requires $O(n)$ flip operations.

Hint: use induction to show that a stack of n pancakes can be sorted using $2n - 3$ flips.

Bonus question: Can we use this approach to beat all known sorting algorithms and sort a list in $O(n)$ time? Why or why not?

3 Bad induction

In this class, you will prove a lot of claims, many of them by induction. You might also prove some wrong claims, and catching those mistakes will be an important skill!

The following are examples of a false proof where an untrue claim has been 'proven' using induction (with some errors or missing details, of course). Your task is to investigate the 'proofs' and identify the mistakes made.

3.1

Fake Claim 1: All numbers are equal.

Inductive hypothesis: Any set of k numbers are all equal to each other.

Base Case: For $k = 1$, any one number is equal to itself.

Inductive Step: Suppose the inductive hypothesis holds for k; we will show that it is also true for $k + 1$. Given a set of $k + 1$ numbers in an arbitrary order, the first k of them are all equal by our inductive hypothesis. Similarly, the last k are all equal. Therefore all $k + 1$ must be equal to each other (see diagram below).

$$\underbrace{x_1, \overbrace{x_2, \ \ldots, x_k,}^{k \text{ numbers}}}_{k \text{ numbers}} \ x_{k+1}$$

Conclusion: By induction, the claim holds for all $k \geq 1$.

3.2

Fake Claim 2: For every non-negative integer n, $2^n = 1$.

Inductive Hypothesis: For all integers n such that $0 \leq n \leq k$, $2^n = 1$.

Base Case: For $n = 0$, $2^0 = 1$.

Inductive Step: Suppose the inductive hypothesis holds for k; we will show that it is also true for $k+1$, i.e. $2^{k+1} = 1$. We have

$$\begin{aligned} 2^{k+1} &= \frac{2^{2k}}{2^{k-1}} \\ &= \frac{2^k \cdot 2^k}{2^{k-1}} \\ &= \frac{1 \cdot 1}{1} && \text{(by strong induction hypothesis)} \\ &= 1 \end{aligned}$$

Conclusion: By strong induction, the claim follows.

3.3

Fake Claim 3:

$$\underbrace{\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \ldots}_{n \text{ terms}} = \frac{3}{2} - \frac{1}{n}. \tag{1}$$

Inductive Hypothesis: (1) holds for $n = k$

Base Case: For $n = 1$,

$$\frac{1}{1 \cdot 2} = 1/2 = \frac{3}{2} - \frac{1}{1}.$$

Inductive Step: Suppose the inductive hypothesis holds for $n = k$; we will show that it is also true for $n = k+1$. We have

$$\begin{aligned} \left(\frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \ldots \frac{1}{(k-1) \cdot k}\right) + \frac{1}{k \cdot (k+1)} &= \frac{3}{2} - \frac{1}{k} + \frac{1}{k \cdot (k+1)} && \text{(by induction hypothesis)} \\ &= \frac{3}{2} - \frac{1}{k} + \frac{1}{k} - \frac{1}{k+1} \\ &= \frac{3}{2} - \frac{1}{k+1}. \end{aligned}$$

Conclusion: By weak induction, the claim follows.

4 Matrix multiplication

In lecture, you have seen how digit multiplication can be improved upon with divide and conquer. Let us see a more generalized example of Matrix multiplication. Assume that we

have matrices X and Y and we'd like to multiply them. Both matrices have n rows and n columns.

For this question, you can make the simplifying assumption that the product of any two entries from X and Y can be calculated in $O(1)$ time.

4.1

What is the naive solution and what is its runtime? Think about how you multiply matrices.

4.2

Now if we divide up X and Y into quarters like this:

$$XY = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} E & F \\ G & H \end{bmatrix} = \begin{bmatrix} AE + BG & AF + BH \\ CE + DG & CF + DH \end{bmatrix}$$

We now have a divide and conquer strategy! Find the recurrence relation of this strategy and the runtime of this algorithm. (You may assume that n is a power of 2.)

4.3

Can we do better? It turns out we can by calculating only 7 of the sub problems:

$$\begin{aligned} P_1 &= A(F - H) & P_5 &= (A + D)(E + H) \\ P_2 &= (A + B)H & P_6 &= (B - D)(G + H) \\ P_3 &= (C + D)E & P_7 &= (A - C)(E + F) \\ P_4 &= D(G - E) \end{aligned}$$

And we can solve XY by

$$XY = \begin{bmatrix} P_5 + P_4 - P_2 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 + P_5 - P_3 - P_7 \end{bmatrix}$$

We now have a more efficient divide and conquer strategy! What is the recurrence relation of this strategy and what is the runtime of this algorithm?