Asymptotic Analysis

Asymptotic Analysis Definitions

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

Definition 1: (Big-Oh notation)

$f = O(g)$ 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)

$f = \Omega(g)$ 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)

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

Note: 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". You are not responsible for knowing the following definitions in this class:

Definition 4:(Little-oh notation)

$f = o(g)$ 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)

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

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

Asymptotic Analysis Problems

1. Prove that if $f = \Omega(g)$ then f is not in $o(g)$.

SOLUTION: Assume **by contradiction** that $f = \Omega(g)$ and also $f = o(g)$. By definition of $\Omega()$, there exist constants c and n_Ω such that for all $n > n_\Omega$,

$$f(n) \geq c \cdot g(n). \tag{1}$$

On the other hand, by definition of $o()$, there exists n_o such that for all $n > n_o$ and all positive constants including c,

$$f(n) < c \cdot g(n). \tag{2}$$

Consider n which is greater than both n_Ω and n_o (e.g. their max +1). Then (1) and (2) should both hold, but this is a contradiction!

Problem Solving Notes:

(a) Read and Interpret: Notice that you are being required to prove a conditional statement which contains a negation.

(b) Type of problem: Proof of a claim! You probably want to use some kind of a formal method to do this.

(c) Work out an example: To start with, some simpler examples to convince yourself that this statement is true can be very helpful. So picking something like $f(n) = n^3$ and $g(n) = n$ where $f(n)$ is clearly larger and also easily comparable to $g(n)$ is one good idea.

(d) Information Needed: Based on the question alone, one immediate hint is that you will likely need to use the definitions of Ω and o to proceed.

(e) Solve and Iterate: Try to write an outline using contradiction, and then fill in the blanks!

2. For each of the following functions, prove whether $f = O(g)$, $f = \Omega(g)$, or $f = \Theta(g)$. For example, by specifying some explicit constants n_0 and $c > 0$ such that the definition of Big-Oh, Big-Omega, or Big-Theta is satisfied.

(a)	$f(n) = n\log(n^3)$	$g(n) = n\log n$
(b)	$f(n) = 2^{2n}$	$g(n) = 3^n$
(c)	$f(n) = \sum_{i=1}^{n} \log i$	$g(n) = n\log n$

SOLUTION:

(a) $f(n) \in \Theta(g(n))$.
Since $f(n) = n\log(n^3) = 3n\log n$. To prove big O, choose any c above 3 (for example $c = 4$), then $f(n) = 3n\log n \leq 4n\log n = cg(n) \quad \forall n \geq n_0$ of any n_0 of your choice.

To prove big Omega, chose any c below 3 (for example $c = 2$, then $f(n) = 3n \log n \geq 2n \log n = cg(n) \quad \forall n \geq n_0$ of any n_0 of your choice.

(b) $f(n) \in \Omega(g(n))$.
Since $f(n) = 2^{2n} = 4^n$, to prove big Omega, choose $c = 1, n_0 = 1$, then $f(n) = 4^n \geq 1 \times 3^n = cg(n) \quad \forall n \geq n_0$.
To disprove Big O, use contradiction.

$$4^n \leq c3^n$$
$$n \log 4 \leq \log c + n \log 3$$
$$n \leq \frac{\log c}{\log 4 - \log 3}$$

So pick any c and $n > \frac{\log c}{\log 4 - \log 3}$.
We have a contradiction.

(c) $f(n) \in \Theta(g(n))$.
TO prove Big O, inspect summation:

$$\sum_{i=1}^{n} \log i = \log 1 + \log 2 + \log 3 + ... + \log n$$
$$\sum_{i=1}^{n} \log i \leq \log n + \log n + \log n + ... + \log n$$
$$\sum_{i=1}^{n} \log i \leq n \log n$$

So we can pick $c = 1$ and $n_0 = 2$.
In order to prove Big-Omega, inspect summation again

$$\sum_{i=1}^{n} \log i = \log 1 + \log 2 + \log 3 + ... + \log(\frac{n}{2}) + ... + \log n$$
$$\sum_{i=1}^{n} \log i \geq \log(\frac{n}{2}) + \log(\frac{n}{2}) + ... + \log(\frac{n}{2})$$
$$\sum_{i=1}^{n} \log i \geq \frac{n}{2} \log(\frac{n}{2}) = \frac{n}{2}(\log(n) - \log(2))$$

For Big-Omega to hold,

$$\frac{n}{2}(\log(n) - \log(2)) \geq cn \log n$$
$$(1 - 2c) \log n \geq 1$$

Now we can pick $c = \frac{1}{4}, n_0 = 4$

Problem Solving Notes:

(a) Read and Interpret: Before you can prove a statement, you must first decide which of the statements are true in each case. So this question will involve some amount of trial and error and convincing yourself that you've arrived at the right solution.

(b) Type of problem: A combination of analysis and proof of a claim! The proof part will either involve finding the right values of c and n_0, or proving a contradiction. It all depends on what we decide we want to prove.

(c) Work out examples: Try to use small values of n to build an intuition for the above functions.

(d) Similar Problems: We have seen examples of such problems in lecture and in CS103. These can be very helpful especially for asymptotics as there are generally a limited number of examples.

(e) Solution Plan: Since we are trying to compare values, one key insight for part (c) is that it helps to re-write the summation in a format which has $O(n)$ terms that each look like $\log(n)$, so that it can be compared more directly with $g(n)$.

3. Give an example of f, g such that f is not $O(g)$ and g is not $O(f)$.

SOLUTION: There are many such examples. Here is one:

$$f(n) = n.$$

$$g(n) = \begin{cases} 1 & \text{if } n \text{ is odd} \\ n^2 & \text{if } n \text{ is even} \end{cases}.$$

Problem Solving Notes:

(a) Read and Interpret: You are being asked to provide a single example that satisfies the requirements. So, we don't need any formal proofs or long write-ups!

(b) Work out examples: This is tricky - after all, every type of function we have seen so far does not meet the requirement. Try to think of more types of functions whose gradients have unusual behaviors!

(c) Information Needed: Look back at the definition of big-O. It relies on the non-decreasing property of these functions. Can you think of something which would not follow this pattern?

(d) Solution Plan: Any function which breaks the property above and does not have neat gradients is a good candidate. For example, a piece-wise function, or even trigonometric functions.

Divide & Conquer

4. 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 A and B 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 A and B can be calculated in O(1) time.

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

   SOLUTION: The naive solution is that we will multiply row by column to get each element of the new matrix. Each new element of the new matrix is a sum of a row multiplied by a column, which takes n time, and there are n^2 new element to compute, resulting in a runtime of $O(n^3)$.

   (b) Now if we divide up the problem 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.

   SOLUTION: The recurrence relation of this approach is $T(n) = 8T(\frac{n}{2}) + O(n^2)$ because you have 8 subproblems, and cutting subproblem size by 2, while doing n^2 additions to combine the subproblems. Using the recurrence, we know that at the last level of recursion we will have $8^{log(n)}$ subproblems of size 1.

$$8^{log(n)} = n^{log(8)} = n^3$$

   Thus, this approach is at least $O(n^3)$. Looks like we did not improve the running time at all!

   (c) 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?

SOLUTION: The recurrence relation of this algorithm is $T(n) = 7T(\frac{n}{2}) + O(n^2)$ because you have 7 subproblems, and cutting subproblem size by 2, while doing n^2 additions to combine the subproblems. Using similar calculation to above, we calculate the runtime of this method to be $\approx n^{2.81}$.

Problem Solving Notes:

i. Similar Problems: This exercise is almost identical to the process with which we introduced Karatsuba in Lecture! So it would definitely pay off to be familiar with lecture (and section) content.

How NOT to prove claims by induction

5. 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 is an example of a false proof where an obviously untrue claim has been 'proven' using induction (with some errors or missing details, of course). Your task is to investigate the 'proof' and identify the mistake made.

   Fake Claim 2:

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

   Inductive Hypothesis: (3) 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 $n = k + 1$. We have

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

   Conclusion: By weak induction, the claim follows.

SOLUTION: The first part of the long derivation of the inductive step is wrong — the summation in the parentheses only contains $k-1$ summands! It should contain k terms, so it is missing the last term.

Weak vs. Strong Induction

The difference between these two types of inductions appears in the inductive hypothesis. In weak induction, we only assume that our claim holds at the k-th step, whereas in strong induction we assume that it holds at all steps from the base case to the k-th step. In this section, let's examine how the two strategies compare.

6. Consider the following proof by weak induction.

   Claim: For any positive integer n, $6^n - 1$ is divisible by 5.

   Inductive Hypothesis: The claim holds for $n = k$. I.e., $6^k - 1$ is divisible by 5,

   Base Case: For $n = 1$,

   $$6^n - 1 = 5 = 5(1)$$

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

   $$\begin{aligned} 6^{k+1} - 1 &= 6(6^k) - 1 \\ &= 6(6^k - 1) - 1 + 6 \\ &= 6(6^k - 1) + 5 \end{aligned}$$

   By the weak inductive hypothesis, $6(6^k - 1)$ is divisible by 5, and the second term is also clearly divisible by 5. Therefore, $6^{k+1} - 1$ is divisible by 5.

   Conclusion: By weak induction, the claim follows.

   Would it be sufficient to use strong induction instead of weak induction for this proof?

SOLUTION: Yes.

7. Now consider the following proof by strong induction.

   First, we introduce a game called Nim, in which there are two players and two piles of matches. At each turn, a player removes some (non-zero) number of matches from one of the piles. The player who removes the last match wins.

   Consider the winning strategy for the second player: Suppose your opponent removes m matches from one pile. You remove m matches from the other pile. We prove by induction that this strategy is in fact a winning strategy by inducting on the number of matches in each pile.

Claim: If the two piles contain the same number of matches n at the start of the game, then by using the above strategy, the second player will always win.

Inductive Hypothesis: The claim holds for all values n where $1 \leq n \leq k$. I.e., if both piles have n matches at the beginning of the game, then the second player can always win.

Base Case: When there are exactly 1 matches in both piles, then the first player is forced to pick 1 match from one of the piles. Player 2 can win by picking 1 match from the other pile.

Inductive Step: Suppose the inductive hypothesis holds for $1 \leq n \leq k$, we will show that it also holds for $n = k + 1$.

If both piles contain $k + 1$ matches at the beginning of the game, any legal move by the first player involves removing j matches from one pile, where $0 \leq j \leq k+1$. The piles then contain $k + 1$ matches and $k + 1 - j$ matches.

The second player can now remove j matches from the other pile. This leaves us with two piles of $k + 1 - j$ matches. If $j = k + 1$, then the second player wins. If $j < k + 1$, we can use the inductive hypothesis to conclude that the second player can win from this new starting point in the game.

Conclusion: By strong induction, the claim follows.

Would it be sufficient to use weak induction instead of strong induction here?

SOLUTION: No.

As a general principle, any valid proof by induction which uses weak induction is still valid if we use strong induction instead. However, the vice-versa is not true.

More Induction

8. On a flat ice sheet, an *odd* number of penguins are standing such that their pairwise distances to each other are all different. At the strike of dawn, each penguin throws a snowball at another penguin that closest to them. Show that there is always some penguin that doesn't get hit by a snowball.

SOLUTION: Claim: For every non-negative number n, if there are $2n + 1$ penguins, then there exists a penguin that doesn't get hit by a snowball.

Inductive hypothesis: The above claim holds for all $n \leq k$.

Base Case: When $n = 1$, there are 3 penguins. Since the pairwise distances between these penguins are different, 2 of these penguins must form the closest pair, and will throw their snowballs at each other. Thus, the third penguin will not get hit with any snowball.

Inductive Step: Suppose the inductive hypothesis holds for $n \leq k$; we will show that it also holds for $n = k + 1$. There are $2(k + 1) + 1$ penguins. Choose any 2 penguins that are closest to each other and call them A and B. These two penguins will throw snowballs at each other.

Then, there are $2k + 1$ penguins remaining and by the inductive hypothesis, at least one of these penguins will not get hit by a snowball.

Conclusion: By induction, the claim follows.

Problem Solving Notes:

(a) Read and Interpret: You are being asked to write a formal proof by induction. Some assumptions are provided to you, such as: there are an odd number of penguins, and their pairwise distances to each other are all different. Think about why this is important.

(b) Work out examples: Try to visualize the smaller cases - start with 3 penguins. What will it look like when the snowballs are thrown?

(c) Information Needed: What information do you need in order to break this down into a smaller problem? How do you define a "smaller problem"? What can you induct on?

(d) Solution Plan: In this case, there are a limited number of iterable variables. The obvious, and perhaps the only option, is to induct on the number of penguins. Once you can reduce the problem into fewer penguins, you can use the inductive hypothesis to prove the claim.