CSCE 222 Discrete Structures for Computing – Fall 2022
Hyunyoung Lee

Problem Set 5

Due dates: Electronic submission of *yourLastName-yourFirstName-hw5.tex* and *yourLastName-yourFirstName-hw5.pdf* files of this homework is due on **Monday, 10/24/2022 before 11:59 p.m.** on `https://canvas.tamu.edu`. You will see two separate links to turn in the .tex file and the .pdf file separately. Please do not archive or compress the files. **If any of the two files are missing, you will receive zero points for this homework.**

Name: (Blake Dejohn) **UIN: (531002472)**

Resources. (All people, books, articles, web pages, etc. that have been consulted when producing your answers to this homework)

Resources Overall (used for the whole document)

1. LaTeX Typesetting System
2. How to denotes limits - `https://www.overleaf.com/learn/latex/Integrals%2C_sums_and_limits`

Problem 1

1. Section 11.1: Asymptotic Equality
2. L Hopital's Rule - `https://www.mathsisfun.com/calculus/l-hopitals-rule.html#:~:text=L'H%C3%B4pital%20is%20pronounced%20%22lopital,some%20special%20conditions%20shown%20later).`

Problem 2

1. Section 11.3: Asymptotically Tight Bounds

Problem 3

1. Section 11.4: Asymptotic Upper Bounds

Problem 4

1. Section 11.5: Asymptotic Lower Bounds

Problem 5

1. Section 11.6: Analysis of Algorithms
2. Problem Solving Exercise on Finding Run-time Complexity - `https://www.youtube.com/watch?v=Rif5XAF_YS0&ab_channel=HyunyoungLee`

Problem 6

1. Section 11.6: Analysis of Algorithms
2. Binary logarithm - `https://en.wikipedia.org/wiki/Binary_logarithm`

On my honor, as an Aggie, I have neither given nor received any unauthorized aid on any portion of the academic work included in this assignment. Furthermore, I have disclosed all resources (people, books, web sites, etc.) that have been used to answer this homework.

Electronic signature: (Blake Dejohn)

Total $100 + 10$ (bonus) points.

The intended formatting is that this first page is a cover page and each problem solved on a new page. You only need to fill in your solution between the `\begin{solution}` and `\end{solution}` environment. Please do not change this overall formatting.

Checklist:

✓ Did you type in your name and UIN?
✓ Did you disclose all resources that you have used?
(This includes all people, books, websites, etc. that you have consulted)
✓ Did you sign that you followed the Aggie Honor Code?
✓ Did you solve all problems?
✓ Did you submit both the .tex and .pdf files of your homework to each correct link on Canvas?

Problem 1. (10 points) Section 11.1, Exercise 11.3

Solution.

Prove whether or not the functions $f(n) = n^2+2n$ and $g(n) = n^2$ are asymptotically equal.

First off, two functions f and g are asymptotically equal if and only if the limit as n goes to infinty of the quotient of the first and second function is equal to 1. That is,

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1$$

So, to find if f and g are asymptotically equal, this limit must be evaluated.

$$\begin{aligned}
\lim_{n\to\infty} \frac{f(n)}{g(n)} &= \lim_{n\to\infty} \frac{n^2+2n}{n^2} \\
&= \lim_{n\to\infty} \frac{n(n+2)}{n(n)} \quad \text{by factoring out } n \\
&= \lim_{n\to\infty} \frac{n+2}{n} \quad \text{by canceling out } n \\
&= \lim_{n\to\infty} \frac{1}{1} \quad \text{by L'Hopital's Rule} \\
&= 1
\end{aligned}$$

From evaluating the limit, it can be seen that Ernie was the one that was correct. f and g are indeed asymptotically equal according to the definition.

Problem 2. (20 points) Section 11.3, Exercise 11.14. [Requirement: Study the definition of $\asymp$ involving the inequalities carefully and use the definition to answer the questions.]

Solution.

Supposing that f, g, and h are functions from the set of positive integers to the set of real numbers. Show that $\asymp$ is an equivalence relation.

Defintion of $\asymp$

f and g have the same order of growth, or $f \asymp g$, if and only if this equality holds for some values of c, C, and n_0:

$$c|g(n)| \leqslant |f(n)| \leqslant C|g(n)|$$

for all $n \geqslant n_0$.

(i) $f \asymp f$

By the defintion of $\asymp$, there are values of c, C, and n_0 that makes this inequality statement true.

$$c|f(n)| \leqslant |f(n)| \leqslant C|f(n)|$$

This is intuitively true for $c = \frac{1}{2}$, $C = 2$, and $n_0 = 1$. This is because any value that $f(n)$ outputs, the absolute value will make this value either zero or positive. This makes it so that, no matter what the value is, half of this value is always less than or equal to the original value and 2 times this value is always greater than or equal to the original value. This therefore proves that $\asymp$ is reflexive given the defintion of $\asymp$.

(ii) $f \asymp g$ if and only if $g \asymp f$

Formally, this means:

$$c|g(n)| \leqslant |f(n)| \leqslant C|g(n)| \longleftrightarrow c|f(n)| \leqslant |g(n)| \leqslant C|f(n)|$$

To prove this, it makes sense to make one side of this statement look like the other. In this way, when this side is true, then the other is bound to also be true since it looks the same/follows the same rules.

From the first part of the biconditonal:

$$c|g(n)| \leqslant |f(n)| \text{ means } |g(n)| \leqslant \frac{1}{c}|f(n)|$$
$$\text{and } |f(n)| \leqslant C|g(n)| \text{ means } \frac{1}{C}|f(n)| \leqslant |g(n)|$$

Putting these statements together, you get:

$$\frac{1}{C}|f(n)| \leqslant |g(n)| \leqslant \frac{1}{c}|f(n)|$$

From here, it can be seen that based on the first part of the biconditional, the second part of the biconditional is true (since the statement above looks very similar to the second part of the biconditional). Namely, there are values of c and C that makes $g \asymp f$ when $f \asymp g$. Specifically, these values include $c = 1/C$ where C is from the first part of the biconditional and $C = 1/c$ where c is also from the first part of the biconditional.

(iii) $f \asymp g$ **and** $g \asymp h$ **implies** $f \asymp h$

Similarily, in formal language:

$$(c|g(n)| \leqslant |f(n)| \leqslant C|g(n)| \quad \wedge \quad c|h(n)| \leqslant |g(n)| \leqslant C|h(n)|) \rightarrow c|h(n)| \leqslant |f(n)| \leqslant C|h(n)|$$

Since this statement would be vacuously true if the first part of the implication was false (i.e the hypothesis, which in this case was $f \asymp g \wedge g \asymp h$), it is sufficient to look at when the first part of the implication is true and reason out if the second part of the implication holds given this. Since logical and statements are only true when both parts are true, the two statements connected by the logical and have to be correct. Given this, we can solve inequalities having to do with the first part of the implication and assume they are true.

(from this point, since there will be lots of c's in the inequalities if denoted similarily as above, to avoid ambiguity, little c and big C pairs will be represented by alphabet pairs (e.g. a and b, c and d, etc.)

Since $f \asymp g$, $a|g(n)| \leqslant |f(n)| \leqslant b|g(n)|$ means $|g(n)| \leqslant \frac{1}{a}|f(n)|$ and $\frac{1}{b} \leqslant |g(n)|$.

Combining, you get $\frac{1}{b}|f(n)| \leqslant |g(n)| \leqslant \frac{1}{a}|f(n)|$.

Then, since $g \asymp h$, $c|h(n)| \leqslant |g(n)| \leqslant d|h(n)|$.

Pooling these statements together, you get

$$c|h(n)| \leqslant \frac{1}{b}|f(n)| \leqslant |g(n)| \leqslant \frac{1}{a}|f(n)| \leqslant d|h(n)|$$

Isolating important statements for the implication gives you, $c|h(n)| \leqslant |\frac{1}{b}f(n)|$ and $\frac{1}{a}|f(n)| \leqslant d|h(n)|$

Next, after simplifying, you get $bc|h(n)| \leqslant |f(n)|$ and $|f(n)| \leqslant ad|h(n)|$

Finally, after combining, you can see that $bc|h(n) \leqslant f(n) \leqslant ad|h(n)|$

This statement is very similar to the last part of the implication. This means that there does exist constants c and C for the $f \asymp h$ statement. Namely, the little c in the $f \asymp h$ statement is the big C of the $f \asymp g$ statment times the little c of the $g \asymp h$ statement while the big C in the $f \asymp h$ statement is the little c

of the $f \asymp g$ statement times the big C in the $g \asymp h$ statement. This therefore proves transitivity and overall that $\asymp$ is an equivalance relation.

Problem 3. (15 points) Prove that $3n^2 + 41 \in O(n^3)$ by giving a direct proof based on the definition of big-O involving the inequalities and absolute values, as given in the lecture notes Section 11.4.

To do so, first write out what $3n^2 + 41 \in O(n^3)$ means according to the definition. Then, you need to find a positive real constant C and a positive integer n_0 that satisfy the definition.

Solution.

Proof

According to the definition of big-O, $f \in O(g)$ if and only if there is a positive real constant C and a positive integer n_0 where $|f(n)| \leqslant C|g(n)|$ holds for all $n \geqslant n_0$.

Given this definition, it is proven that $f \in O(g)$ when $C = 100$ and $n_0 = 1$. This plugged into the inequality looks like $|3n^2 + 41| \leqslant 100|n^3|$ for all $n \geqslant n_0$ and is true for any n plugged in that is greater than n_0(which is one). As an example, when $n = 10$, this inequality evaluates to $|341| \leqslant 100|1000|$ which is obviously true. This proves that $f \in O(g)$.

Problem 4. (15 points) Prove that $\frac{1}{2}n^2 + 5 \in \Omega(n)$ by giving a direct proof based on the definition of big-Ω involving the inequalities and absolute values, as given in the lecture notes Section 11.5.

To do so, first write out what $\frac{1}{2}n^2 + 5 \in \Omega(n)$ means according to the definition. Then, you need to find a positive real constant c and a positive integer n_0 that satisfy the definition.

Solution.

Proof

According to the definition of big-Ω, $f \in \Omega(g)$ if and only if there is a positive real constant c and a positive integer n_0 where $c|g(n)| \leqslant |f(n)|$ holds for all $n \geqslant n_0$.

By this definition, this statement is true when $c = 1$ and $n_0 = 1$. When plugged into the inequality, it says $1|n| \leqslant |\frac{1}{2}n^2 + 5|$ for all $n \geqslant n_0$. This is true for any n value plugged that is greater than n_0. As an example, when $n = 10$, the inequality states, $|10| \leqslant |\frac{1}{2}100 + 5|$. This is a true statement. Therefore, it is proven that $f \in \Omega(g)$.

Problem 5. (10+10 = 20 points) Read Section 11.6 carefully before attempting this problem. Analyze the running time of the following algorithm using a step count analysis as shown in the Horner scheme (Example 11.40).

```
// search a key in an array a[1..n] of length n
search(a, n, key)        cost   times
  for k in (1..n) do      c1    [ n+1 ]
    if a[k]=key then      c2    [ n ]
       return k           c3    [ 1 ]
  endfor                  c4    [ 1 ]
  return false            c5    [ 1 ]
```

(a) Fill in the `[ ]`s in the above code each with a number or an expression involving `n` that expresses the step count for the line of code.

(b) Determine the worst-case complexity of this algorithm and give it in the Θ notation. Show your work and explain using the definition of Θ involving the inequalities.

Solution.

For part (b)

The worst-case complexity of this algorithm is given by

$$T(n) = c_1(n+1) + c_2(n) + c_4 + c_5 = \Theta(n)$$

This worst-case algorithm was gotten by multiplying the cost of a given statement times the amount of iterations this statement would run in the worst case. The for loop on line 2 would run $n+1$ times because it first goes from 1 to n which is a total of n iterations, but then in order to end the loop, the loop condition would be run one last time for a total of $n+1$. Continuing, the if statement inside the for loop would have n iterations. This is because it would not run when the loop condition is run for the final time but would run for all iterations when the for loop goes from 1 to n. Lastly, the ending statements all iterate only one time at the end of the program with the exception of the return k statement that does not run even once. This is because in the worst case scenario, the key is not found in the array which makes it so the algorithm checks the entire array, ends the for loop, then returns a false value to get the maximum amount of iterations within the program.

Next, in order for a function to be contained within Θ of another function($f \in \Theta(g)$), there must exist some constants c, C, and n_0 where

$$c|g(n)| \leqslant |f(n)| \leqslant C|g(n)|$$

holds for all $n \geqslant n_0$.

Given this definition, $T(n)$ is tightly bound by n since for constants $c = \frac{1}{2}$,

$C = 10$, and $n_0 = 1$, this inequality holds. This is because the highest degree that $T(n)$ holds for n is simply 1. This makes $T(n)$ have a somewhat linear growth rate. So, constants that decrease the linear growth rate of another function for the lower bound and increase it for the upper bound makes it so the definition of Θ of another function is true for these given constants.

Problem 6. (15+15 = 30 points) Read Section 11.6 carefully before attempting this problem. Analyze the running time of the following algorithm using a step count analysis as shown in the Horner scheme (Example 11.40).

```
// determine the number of digits of an integer n
binary_digits(n)            cost  times
  int cnt = 1                c1   [ 1 ]
  while (n > 1) do           c2   [ floor(log2(n))+1 ]
    cnt = cnt + 1            c3   [ floor(log2(n) ]
    n = floor( n/2.0 )       c4   [ floor(log2(n) ]
  endwhile                   c5   [ 1 ]
  return cnt                 c6   [ 1 ]
```

(a) Fill in the `[ ]`s in the above code each with a number or an expression involving `n` that expresses the step count for the line of code.

(b) Determine the worst-case complexity of this algorithm as a function of n and give it in the Θ notation. Show your work and explain using the definition of Θ involving the inequalities.

Solution.

For part (b)

The worst-case complexity of this algorithm is given by

$$T(n) = c_1 + c_2(\lfloor \log_2 n \rfloor + 1) + c_3(\lfloor \log_2 n \rfloor) + c_4(\lfloor \log_2 n \rfloor) + c_5 + c_6 = \Theta(\lfloor \log_2 n \rfloor)$$

This worst-case complexity was found slightly different than for problem 5. Overall, since this algorithm tries to convert numbers represented in base 10 into their base 2 counterparts, you can use the binary logithm equation for the iterations on the while loop in this program. This is because with this equation, when taking the floor, you can find the number of digits it takes to represent a base 10 number into base 2, which is exactly what this program aims to do. After this is completed, the program ends the while loop. Also, something that has to be kept in mind is that the while loop condition is checked in the beginning once, therefore this equation should have a plus 1 at the end of it. Next, with the exception of the first initial check of the while loop condition, the statements inside the while loop will run the exact number of times as the while loop (in other words, the iterations of the statements inside the while loop are the iterations of the while loop minus 1). The other statements run only 1 time since the last statements run at the end of the program and the first statement only runs in the beginning of the program.

Next, in order for a function to be contained within Θ of another function($f \in \Theta(g)$), there must exist some constants c, C, and n_0 where

$$c|g(n)| \leqslant |f(n)| \leqslant C|g(n)|$$

holds for all $n \geqslant n_0$.

This holds for constants $c = \frac{1}{2}$, $C = 10$, and $n_0 = 4$. Since $T(n)$ attributes most of its growth to $\lfloor \log_2 n \rfloor$, it can intuitively be seen that $\frac{1}{2}$ of a function with this same growth will always be lower after a given n and 10 times a function with this same growth will always be bigger after a given n. Therefore, $T(n)$ is tightly bounded by $\lfloor \log_2 n \rfloor$.