CSCE 222 Discrete Structures for Computing – Fall 2023
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, 10/23/2023 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: (Manas Navale) **UIN: (333006797)**

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

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: **(Manas Navale)**

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.

We have f(n) = $n^2 + 2n$ and g(n) = n^2.

Now, we want to see if the ratio of f(n) to g(n) approaches 1 as n goes to infinity. In other words, we're looking at $\lim_{n\to\infty} \frac{f(n)}{g(n)}$.

Let's simplify this ratio:

$$\lim_{n\to\infty} \frac{n^2 + 2n}{n^2}$$

Now, we can factor out n^2 from both the numerator and denominator:

$$\lim_{n\to\infty} \frac{n^2(1 + 2/n)}{n^2}$$

We can cancel out n^2 from the top and bottom:

$$\lim_{n\to\infty} \frac{1 + 2/n}{1}$$

Now, as n goes to infinity, the term $2/n$ approaches 0 because any number divided by infinity is essentially zero:

$$\lim_{n\to\infty} \frac{1 + 2/n}{1} = 1 + 2 \cdot 0 = 1$$

So, when n gets really large, the ratio of f(n) to g(n) is 1. This means that f(n) behaves very similarly to g(n) as n becomes large, and we can say that $f(n) \sim g(n)$.

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.

(i) $f \asymp f$: To show that a function is equivalent to itself, it is sufficient to demonstrate that the limit of the quotient of the two functions is equal to 1 as the input approaches infinity. Formally, we need to show that:

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

(ii) $f \asymp g$ if and only if $g \asymp f$: The equivalence of functions is symmetric, so if $f \asymp g$, then $g \asymp f$. To prove the "if and only if" part, we need to show that if $g \sim f$, then $f \sim g$.

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

(iii) $f \asymp g$ and $g \asymp h$ implies $f \asymp h$: To prove this, we can use the transitive property of equivalence. If $f \asymp g$ and $g \asymp h$, it means:

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

and

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

We want to show that $f \sim h$, which means:

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

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.

To prove that $3n^2 + 41 \in O(n^3)$, we need to show that there exist positive real constants C and n_0 such that for all $n \geq n_0$, the following inequality holds:

$$0 \leq 3n^2 + 41 \leq Cn^3$$

To find suitable values for C and n_0, we consider the inequality $0 \leq 3n^2 + 41 \leq Cn^3$. For $n \geq 1$, we have:

$$3n^2 + 41 \leq 3n^3 + 41n^2 \leq 44n^3$$

To make this inequality valid, we choose $C = 44$. Now, we need to find an n_0 such that:

$$3n^3 + 41n^2 \leq 44n^3 \quad \text{for all } n \geq n_0$$

Simplifying the inequality:

$$3 + \frac{41}{n} \leq 44$$

Solving for n in the middle term:

$$\frac{41}{n} \leq 44 - 3 = 41$$

This simplifies to:

$$1 \leq n$$

Thus, for $n_0 = 1$ and $C = 44$, the inequality $3n^2 + 41 \leq 44n^3$ holds for all $n \geq 1$. Therefore, $3n^2 + 41 \in O(n^3)$.

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.

We want to prove that $\frac{1}{2}n^2 + 5$ grows at least as fast as n as n becomes large.

According to the definition of big-Ω, we need to find positive constants c and n_0 such that the inequality $0 \leq c \cdot n \leq \frac{1}{2}n^2 + 5$ holds for all n greater than or equal to n_0.

For $n \geq 1$, we observe that $c \cdot n$ is always less than or equal to $\frac{1}{2}n^2$.

So, we set up the inequality as follows:

$$c \cdot n \leq \frac{1}{2}n^2 \leq \frac{1}{2}n^2 + 5$$

To make this inequality true, we can choose $c = \frac{1}{2}$. Now, we need to find an appropriate n_0.

Simplifying, we find that $n \leq 1$ ensures the inequality holds for all $n \geq n_0$.

Since n is always greater than or equal to 1, the inequality holds for all $n \geq 1$.

Therefore, we've shown that $\frac{1}{2}n^2 + 5$ is indeed in $\Omega(n)$ by choosing suitable values for c and n_0 based on the definition.

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]
    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 scenario occurs when the key is not found in the array, and the algorithm has to iterate through the entire array.

In this case, the for loop will run n times, where n is the length of the array. Therefore, the worst-case complexity of this algorithm is $O(n)$, as the number of steps is directly proportional to the length of the array. Final Answer: The worst-case complexity of this algorithm is $O(n)$.

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   [depends on n]
    cnt = cnt + 1            c3   [1]
    n = floor( n/2.0 )       c4   [1]
  endwhile                   c5   [depends on n]
  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))

(b) To determine the worst-case complexity of this algorithm, we need to understand how many times the while loop iterates in the worst case. The loop continues until the value of 'n' becomes less than or equal to 1. This can be found by dividing 'n' by 2 repeatedly until it reaches 1 or less. In other words, we want to find how many times we can divide 'n' by 2 until it's less than or equal to 1. Mathematically, we express this as:

$$n \leq 2^k$$

To find 'k,' we take the base-2 logarithm of both sides:

$$\log_2(n) \leq k$$

This means the number of iterations 'k' required for the while loop to terminate is bounded by the base-2 logarithm of 'n.' Therefore, the worst-case time complexity of this algorithm is approximately proportional to the logarithm of 'n,' which is often written as $\Theta(\log_2(n))$.

In simpler terms, the time it takes to find the number of binary digits in 'n' is closely related to the logarithm of 'n,' making the algorithm's complexity $\Theta(\log_2(n))$.