CS456:Algorithm Design and Analysis

Evans Kumi

January 22, 2024

Assignment 1

In doing this assignment, I consulted: Emmanuel Soumahoro

Problem 1

Part A

For Upper bound:

h(n) $\leq$ cg(n) , where c is a positive constant and n is non negative. **For Lower bound:**

cg(n) $\leq$ f(n), where c is a positive constant and n is non negative. **Using limits learned in calculus:**

Note that $\frac{\text{any smaller number}}{\infty} = 0$ ... eq(1)
I would be using these rules $10n^4 + 7n^2 + 3n$

image.png

Figure 1: Log rules

For Upper bound:

Step 1: Let $h(n) = 10n^4 + 7n^2 + 3n$ and $g(n) = n^4$.

Step 2: $10n^4 + 7n^2 + 3n \leq cn^4$ (Divide through with $g(n)$).

Step 3:

$$\frac{10n^4 + 7n^2 + 3n}{n^4} \leq \frac{c \cdot n^4}{n^4} \quad \Rightarrow \quad 10 + \frac{7}{n^2} + \frac{3}{n^3} \leq c$$

Step 4: Let $n = 1$, and following the rule in equation (1):

$$10 + 7 + 3 \leq c \quad \Rightarrow \quad 20 \leq c \quad \text{(Therefore, } c \text{ can be equal to 21 if } n = 1)$$

Hence, h(n) $\in$O(n^4), because when we take c=21 and n=1 any number from 1 and above the $10n^4 + 7n^2 + 3n \leq cn^4$ holds.

For Lower bound:

Step 1: Let $f(n) = 10n^4 + 7n^2 + 3n$ and $g(n) = n^4$.

Step 2: $cn^4 \leq 10n^4 + 7n^2 + 3n$ (Divide through with $g(n)$).

Step 3:

$$\frac{c \cdot n^4}{n^4} \leq \frac{10n^4 + 7n^2 + 3n}{n^4} \quad \Rightarrow \quad c \leq 10 + \frac{7}{n^2} + \frac{3}{n^3}$$

Step 4: Let $n = 1$, and following the rule in equation (1):

$$c \leq 10 + 7 + 3 \quad \Rightarrow \quad c \leq 20 \quad \text{(Therefore, } c \text{ can be equal to 19 if } n = 1)$$

Hence, f(n) $\notin$ Ω(n^4), because when we take c=19 and n to be any number from 2 and above, $cn^4 \leq 10n^4 + 7n^2 + 3n$. will not hold.

Using Limits:

Step 1: when $g(n) = n^4$

$$\lim_{n\to\infty}(10n^4 + 7n^2 + 3n) \quad \text{(Divide through with } g(n))$$

Step 2:

$$\lim_{n\to\infty} \frac{10n^4 + 7n^2 + 3n}{n^4} \Rightarrow \lim_{n\to\infty} 10 + \frac{7}{n^2} + \frac{3}{n^3}$$

Step 3:

Using the rule from equation (1):

$\lim_{n\to\infty} 10 + \frac{7}{n^2} + \frac{3}{n^3} = 10$

Hence, f(n) $\in \Theta(n^4)$. because $c > 0$.

b. $2n \log n + 10n$

For Upper bound:

Step 1: Let $h(n) = 2n \log n + 10n$ and $g(n) = n \log n$.

Step 2: $2n \log n + 10n \leq cn \log n$ (Divide through with $g(n)$).

Step 3:

$$\frac{2n \log n + 10n}{n \log n} \leq \frac{cn \log n}{n \log n} \quad \Rightarrow \quad 2 + \frac{10}{\log n} \leq c$$

Step 4: Let n=2:

$$2 + 10 \leq c \quad \Rightarrow \quad 12 \leq c \quad \text{(Therefore, } c \text{ can be equal to 13 if } n = 2)$$

Hence, h(n) $\notin$O(nlog n), because when we take $c = 13$ and $n =$ any number from 2 and above the $2n \log n + 10n \leq cn \log n$ doesn't hold.

For Lower bound:

Step 1: Let $f(n) = 2n \log n + 10n$ and $g(n) = n \log n$.

Step 2: $cn \log n \leq 2n \log n + 10n$ (Divide through with $g(n)$).

Step 3:

$$\frac{cn \log n}{n \log n} \leq \frac{2n \log n + 10n}{n \log n} \quad \Rightarrow \quad c \leq 2 + \frac{10}{\log n}$$

Step 4: Let n=2:

$$12 \leq c \quad \Rightarrow \quad 12 \leq c \quad \text{(Therefore, } c \text{ can be equal to 11 if } n = 2\text{)}$$

Hence, f(n) $\in \Omega(n \log n)$, because when we take c=11 and n= any number from 2 and above the $2n \log n + 10n \leq cn \log n$ holds.

Using Limits:

Step 1: when $g(n) = n \log n$

$$\lim_{n \to \infty} (2n \log n + 10n) \quad \text{(Divide through with } g(n)\text{)}$$

Step 2:

$$\lim_{n \to \infty} \frac{2n \log n + 10n}{n \log n} \Rightarrow \lim_{n \to \infty} 2 + \frac{10}{\log n}$$

Step 3:

$$\text{Using the rule from equation (1): } \lim_{n \to \infty} 2 + \frac{10}{\log n} = 2$$

Hence, f(n) $\in \Theta(n \log n)$. because $c > 0$.

c. $\sqrt{10n^4 + 7n^2 + 3n} \equiv \sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}$

For Upper bound:

Step 1: Let $h(n) = \sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}$ and g(n) = n^2

Step 2: $\sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n} \leq c\text{g(n)}(Dividethroughwith\text{g(n)}).\frac{\sqrt{10}n^2+\sqrt{7}n+\sqrt{3n}}{n^2} \leq \frac{cn^2}{n^2} \quad \Rightarrow \quad \sqrt{10} + \frac{\sqrt{7}}{n} + \frac{\sqrt{3n}}{n^2} \leq c$

Step 3: Let n =1, and following the rule in equation (1)

$$\sqrt{10} + \sqrt{7} + \sqrt{3} \leq c \quad \Rightarrow \quad \sqrt{10} + \sqrt{7} + \sqrt{3} \leq c \quad \text{(Therefore, } c \text{ can be equal to 8 if } n \to \infty\text{)}$$

Hence, h(n) $\notin$O(n$\log n$), as the $\sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n} \leq c\text{g(n)}doesn'tholdforc = 4andn = 1$.

For Lower bound:

Step 1: Let $f(n) = \sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}$ and $g(n) = n^2$.

Step 2: $cg(n) \leq \sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}$ (Divide through with $g(n)$).

Step 3:

$$\frac{cn^2}{n^2} \leq \frac{\sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}}{n^2} \quad \Rightarrow \quad c \leq \sqrt{10} + \frac{7}{n} + \frac{\sqrt{10}}{n^2}$$

Step 4: Let n $\to$, and following the rule in equation (1)

$$12 \leq c \quad \Rightarrow \quad \sqrt{10} \leq c \quad \text{(Therefore, } c \text{ can be equal to 2 if } n \to \infty\text{)}$$

Hence, f(n) $\in \Omega(n \log n)$, as the $\sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n} \leq c\text{g(n)}holdsforc = 4andn = 1$.

Using Limits:

Step 1: when $g(n) = n^2$

$$\lim_{n\to\infty}(\sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}) \quad \text{(Divide through with } g(n))$$

Step 2:

$$\lim_{n\to\infty}\frac{\sqrt{10}n^2 + \sqrt{7}n + \sqrt{3n}}{n^2} \Rightarrow \lim_{n\to\infty} sqrt10n^2 + \sqrt{7}n + \sqrt{3n}$$

Step 3:

$$\text{Using the rule from equation (1): } \lim_{n\to\infty}\sqrt{10} + \frac{\sqrt{7}}{n} + \frac{\sqrt{3n}}{n^2} = \sqrt{10}$$

This means that it's f(n) $\in \Theta(n \log n)$, as $c > 0$.

Hence, f(n) $\in O(n \log n)$, as $c > 0$.

d. $(n^3 + 1)^6$

For Upper bound:

Step 1: Let h(n) = $(n^3 + 1)^6$ and $g(n) = n^{18}$.

Step 2: $(n^3 + 1)^6 \leq$ cn^{18} (Divide through with $g(n)$).

Step 3:

$$\frac{(n^3+1)^6}{n^{18}} \leq \frac{cn^{18}}{n^{18}} \quad \Rightarrow \quad 1 + \frac{1}{n^3} \leq c$$

Step 4: Let $n = 1$, and following the rule in equation (1)

$$2 \leq c \quad \Rightarrow \quad 2 \leq c \quad \text{(Therefore, } c \text{ can be equal to 3 if } n = 1)$$

Hence, h(n) $\notin$O(n^{18}), as the when testing with $c = 3$ and $n = 1$ the expression would not hold.

For Lower bound:

Step 1: Let f(n) =$(n^3 + 1)^6$ and $g(n) = n^{18}$.

Step 2: c $n^2 \leq$($\mathrm{n}^3 + 1)^6$ (Divide through with $g(n)$).

Step 3:

$$\frac{cn^2}{n^{18}} \leq \frac{(n^3+1)^6}{n^{18}} \quad \Rightarrow \quad c \leq 1 + \frac{1}{n^3}$$

Step 4: Let $n = 1$, and following the rule in equation (1)

$$c \leq 2 \quad \Rightarrow \quad c \leq 2 \quad \text{(Therefore, } c \text{ can be equal to 1 if } n = 1)$$

Hence, f(n) $\in\Omega(n^{18})$, as we test the expression with $c = 1$ and n to be a positive number from 1 upwards.

Using Limits:

Step 1: when $g(n)$ =n^{18}

$$\lim_{n\to\infty}((n^3 + 1)^6) \quad \text{(Divide through with } g(n))$$

Step 2:

$$\lim_{n\to\infty}\frac{(n^3+1)^6}{n^{18}} \Rightarrow \lim_{n\to\infty} 1 + \frac{1}{n^3}$$

Step 3:

$$\text{Using the rule from equation (1): } \lim_{n\to\infty} 1 = 1$$

This means that it's O(Constant),as $c > 0$.

Hence, f(n) $\in \Theta(n^{18})$.

Problem 2

Part A

a. $\frac{2n(n-1)}{2} \in O(n^3)$

Using the Upper bound procedure:

Answer: True

In Big O notation, $O(f(n))$ represents an upper bound on the growth rate of a function $f(n)$. Therefore, when comparing $O(n^3)$ and $O(n^2)$, $O(n^2)$ is considered a "faster" growth rate than $O(n^3)$. This is because n^2 grows at a slower rate than n^3 as n becomes large.

The correct conclusion is based on comparing the growth rate of the function in question with the specified Big O class. Therefore, the conclusion is that $\frac{2n(n-1)}{2} \in O(n^3)$ because, as demonstrated earlier, $n(n-1)$ can be upper-bounded by $c \cdot n^3$ for suitable constants c and n_0.

b. $\frac{n(n-1)}{2} \in O(n^2)$

Answer: True

In Big O notation, $O(f(n))$ represents an upper bound on the growth rate of a function $f(n)$. In this case, when comparing $\frac{n(n-1)}{2}$ with n^2, the growth rate of $\frac{n(n-1)}{2}$ is considered to be within $O(n^2)$. This is because the expression $\frac{n(n-1)}{2}$ grows at a rate comparable to or less than n^2 as n becomes large.

c. $\frac{2n(n-1)}{2} \in \mathbf{\Theta}(n^3)$

Answer: True

In Big Θ notation, $\Theta(f(n))$ represents both an upper and a lower bound on the growth rate of a function $f(n)$. In this case, when comparing $\frac{2n(n-1)}{2}$ with n^3, the growth rate of $\frac{2n(n-1)}{2}$ is considered to be within $\Theta(n^3)$. This is because the expression $\frac{2n(n-1)}{2}$ grows at a rate comparable to n^3 as n becomes large.

Therefore, the correct conclusion is that $\frac{2n(n-1)}{2} \in \Theta(n^3)$.

d. $\frac{2n(n-1)}{2} \in \mathbf{\Omega}(n)$

Answer: True

In Big Ω notation, $\Omega(f(n))$ represents a lower bound on the growth rate of a function $f(n)$. In this case, when comparing $\frac{2n(n-1)}{2}$ with n, the growth rate of $\frac{2n(n-1)}{2}$ is considered to be within $\Omega(n)$. This is because the expression $\frac{2n(n-1)}{2}$ grows at a rate comparable to or greater than n as n becomes large.

Therefore, the correct conclusion is that $\frac{2n(n-1)}{2} \in \Omega(n)$.

Part B

a. $(n^2+1)^{10}$

Using Limits:

when $g(n)$ =n^{20}

$$\lim_{n\to\infty}((n^2+1)^{10}) \quad \text{(Divide through with } g(n))$$

$$\lim_{n\to\infty} \frac{(n^2+1)^{10}}{n^{2\cdot 10}} \Rightarrow \lim_{n\to\infty} 1 + \frac{1}{n^2}$$

$$\text{Using the rule from equation (1): } \lim_{n\to\infty} 1 = 1$$

This means that it's $(n^2+1)^{10} \in \Theta(\mathrm{n}^{20})$, as $c > 0$.

b. $\sqrt{10n^2+7n+3} \equiv \sqrt{10}n + \sqrt{7n} + \sqrt{3}$

Using Limits:

Let f(n) =$\sqrt{10}n + \sqrt{7n} + \sqrt{3}$ and $g(n) = n$

$$\frac{(\sqrt{10}n + \sqrt{7n} + \sqrt{3})}{n} \quad \text{(Divide through with } g(n)\text{)}$$

$$\lim_{n\to\infty} \frac{\sqrt{10}n + \sqrt{7n} + \sqrt{3}}{n} \leq \frac{c \cdot n}{n} \quad \Rightarrow \lim_{n\to\infty} \quad \sqrt{10} + \frac{\sqrt{7n}}{n} + \frac{\sqrt{3}}{n}$$

$$Using the L'hopitals rule \frac{\sqrt{7n}}{n} \quad \Rightarrow \quad \frac{0.5\sqrt{7}}{\sqrt{n}}$$

Using the rule from equation (1):

$$\lim_{n\to\infty} \sqrt{10} = \sqrt{10}$$

This means that it's $\sqrt{10}n + \sqrt{7n} + \sqrt{3} \in \Theta$(n), as $c > 0$.

c. $2n\log(n+2)^2 + (n+2)^2\log(\frac{n}{2})$

Using Limits:

Let f(n) =$2n\log(n+2)^2 + (n+2)^2\log(\frac{n}{2})$ and $g(n) =$(n+2)$^2 \log\left(\frac{n}{2}\right)$

$$\lim_{n\to\infty} f(n) \quad \text{(Divide through with } g(n)\text{)}$$

$$\lim_{n\to\infty} \frac{2n\log(n+2)^2 + (n+2)^2\log\left(\frac{n}{2}\right)}{(n+2)^2\log\left(\frac{n}{2}\right)} \quad \Rightarrow \lim_{n\to\infty} \frac{2n\log(n+2)^2}{(n+2)^2\log\left(\frac{n}{2}\right)} + 1(When simplified)$$

We can also deduce that:

$$\lim_{n\to\infty} \frac{2\log(n+2)^2}{(n+2)\log\left(\frac{n}{2}\right)} = 0$$

(As the denominator is heavier than the numerator)

$$\text{Therefore, } \lim_{n\to\infty} 1 = 1$$

Therefore, the correct conclusion is that 2nlog$(n+2)^2 + (n+2)^2\log(\frac{n}{2}) \in \Theta(n)$, because $c > 0$.

d. $2^{n+1} + 3^{n-1}$

Let $f(n) = 2^{n+1} + 3^{n-1}$ and $g(n) = 3^n$. Using the limit concept.

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

Divide through by g(n):

$$\lim_{n\to\infty} \frac{2^{n+1} + 3^{n-1}}{3^n} \quad \Rightarrow \quad \lim_{n\to\infty} \frac{2 \cdot \left(\frac{2}{3}\right)^n + \frac{1}{3}}{1}$$

So we would simplifying it:

$$\lim_{n\to\infty} \frac{2 \cdot \left(\frac{2}{3}\right)^n + \frac{1}{3}}{1} \Rightarrow \frac{1}{3}.$$

So, for $n = 1$, $\frac{f(1)}{g(1)}$ is $\frac{5}{3} \leq c$.

Therefore, the correct conclusion is that $2^{n+1} + 3^{n-1} \in \Theta(3^n)$, because $c > 0$.

e. $\log n$ Note: $\log 2$=1

For Limits:

Step 1: Let h(n) = $\log n$ and g(n) = $\log n$.

Step 2: $\lim_{n\to\infty} \log n)$ (Divide through with $g(n)$).

Step 3: (Lets' divide through by g(n)) $\lim_{n\to\infty} \frac{\log n}{\log n} \quad \Rightarrow \quad \lim_{n\to\infty} 1 = 1$
Hence, h(n) $\in \Theta(\log n)$, because $c > 2$.

Problem 3

i . **Input:**

- A : The input array to be sorted.
- B : The output array where the sorted elements will be placed.

Algorithm Steps:

Counting Sorting Array A

Step 1: After line 3

$$A = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]$$
$$C = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

Step 2: After line 5

$$C = [2, 2, 2, 2, 1, 0, 2, 0, 0, 0, 0]$$

Step 3: After line 8

$$C = [2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11]$$

Step 4: After line 12

$$B = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$
For $i = 11$ to 1
Let $j = A[i]$, update $B[C[j]] = A[i], C[j] = C[j] - 1$

Final Results

$$A = [6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2]$$
$$B = [0, 0, 1, 1, 2, 2, 3, 3, 4, 6, 6]$$
$$C = [2, 4, 6, 8, 9, 9, 11, 11, 11, 11, 11]$$

ii. **Algorithm Steps:**

1. **Initialize Count Array C:From line 1 to 3**

.... For $i = 0$ to k, $\quad C[i] = 0 \ldots 2k + 1$

2. **Count the occurrences of each element in A: From line 4 to 6**

$$\text{For } j = 1 \text{ to length of } A, \quad C[A[j]] = C[A[j]] + 1 \ldots 6n$$

3. **Update Count Array C to store the cumulative count: From line 7 to 9**

$$\text{For } i = 1 \text{ to } k, \quad C[i] = C[i] + C[i-1] \ldots 5k$$

4. **Place elements in sorted order: From line 10 to 12**

$$\text{For } j = \text{length of } A \text{ downto } 1, \quad B[C[A[j]]] = A[j], \quad C[A[j]] = C[A[j]] - 1 \ldots 6n$$

Final Result:

- $2k + 6n + 5k + 6n + 1 = O(n)$
- **Sorted Array (B):** The output array after sorting.
- **Count Array (C):** The cumulative count array after placing elements in their sorted positions.

iii.

Assumption Not True in Part (ii):

- **Implication:** Increased space complexity ($O(n + k)$) if the range k is not a constant.
- **Impact:** Counting Sort becomes less efficient for larger datasets or non-constant ranges.

Counting Sort Limitations:

- **Space Complexity:** Requires additional space for the count array.
- **Not Comparison-Based:** Limited to scenarios where a non-comparison approach is suitable.
- **Stability:** Maintains relative order of equal elements, but stability may not always be crucial.

Why Counting Sort is Not Generally Used:

- **Space Complexity:** Can be a limiting factor for larger datasets.
- **Comparison-Based Sorting Needs:** Inapplicable for tasks requiring comparison-based sorting.
- **Adaptability:** Specialized for specific scenarios; other algorithms (quicksort, mergesort) are more versatile.

Problem 4

Algorithm: Gaussian Elimination

Input:

- Matrix $A[1..n, 1..n]$ of a system's coefficients.
- Column-vector $b[1..n]$ of the system's right-hand side values.

Output: An equivalent upper-triangular matrix in place of A with the corresponding right-hand side values in the $(n + 1)$st column.

Procedure:

```
for i = 1 to n do.....n
    A[i, n + 1] = b[i]....3

for i = 1 to n - 1 do ...(n-1)
    for j = i + 1 to n do.....(n-1)
        for k = i to n + 1 ...(n+1)
            do A[j, k] = A[j, k] - A[i, k] * A[j, i] / A[i, i] ... (9) % Matrix Update
```

n(3) + 9(n-1)(n-1)(n+1)= O(n^3)
Therefore the run time would be O(n^3)

Problem 5

code.png

Figure 2: Linked list version

The linked list version of the insertion sort algorithm maintains a time complexity of (n^2), similar to the array version. This is due to the nested loops involved in both implementations, resulting in a quadratic runtime proportional to the square of the input size.