COMS 3110 Introduction to the Design and Analysis of Algorithms, Homework 1

Mobina Amrollahi

Due: September 20^{th}, 11:59pm

Late submission policy. An assignment that is submitted one day late will get a penalty of 10%. An assignment that is submitted two days late will get a penalty of 20%. Assignments submitted after two days will not be graded and will get no points. For example, if an assignment is due on Friday by midnight, a submission on Saturday will be penalized by 10%, and a submission on Sunday will be penalized by 20%. A submission on Monday will get no points.

Submission format. Homework solutions will have to be typed. You can use word, LaTeX, or any other type-setting tool to type your solution. Your submission file should be in pdf format. Do **NOT** submit a photocopy of handwritten homework except for diagrams that can be hand-drawn and scanned. We reserve the right **NOT** to grade homework that does not follow the formatting requirements. Name your submission file: `<Your-net-id>-3110-hw1.pdf`. For instance, if your netid is `asterix`, then your submission file will be named `asterix-3110-hw1.pdf`. Each student must hand in their own assignment. If you discussed the homework or solutions with others, a list of collaborators must be included with each submission. Each of the collaborators has to write the solutions in their own words (copies are not allowed).

General Requirements

- When proofs are required, do your best to make them both clear and rigorous. Even when proofs are not required, you should justify your answers and explain your work.
- When asked to present a construction, you should show the correctness of the construction.

Some Useful (in)equalities

- $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
- $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$
- $2^{\log_2 n} = n$, $a^{\log_b n} = n^{\log_b a}$, $n^{n/2} \leq n! \leq n^n$, $\log x^a = a \log x$
- $\log(ab) = \log a + \log b$, $\log(a/b) = \log a - \log b$
- $a + ar + ar^2 + ... + ar^{n-1} = \frac{a(r^n-1)}{r-1}$
- $1 + \frac{1}{2} + \frac{1}{2^2} + ... + \frac{1}{2^n} = 2(1 - \frac{1}{2^{n+1}})$
- $1 + 2 + 4 + ... + 2^n = 2^{n+1} - 1$

1. **(5 pts)** Prove $\left\lceil \frac{n(n+1)}{2} \right\rceil^2 - \frac{n^2(n^2+1)}{4} + 98 \in O(n^3)$.

Answer

$$
\begin{aligned}
&= \left\lceil \frac{n(n+1)}{2} \right\rceil^2 - \left\lceil \frac{n^2(n^2+1)}{4} \right\rceil + 98 \\
&= \left\lceil \frac{n^2+n}{2} \right\rceil^2 - \left\lceil \frac{n^4+n^2}{4} \right\rceil + 98 \\
&= \left\lceil \frac{n^4+2n^3+n^2}{4} \right\rceil - \left\lceil \frac{n^4+n^2}{4} \right\rceil + 98 \\
&= \left\lceil \frac{n^4+2n^3+n^2-n^4-n^2}{4} \right\rceil + 98 \\
&= \frac{2n^3}{4} + 98 \\
&= \frac{n^3}{2} + 98 \\
&\leq n^3 + 98 \\
&\leq n^3 + 98n^3 \\
&= 99n^3 \in O(n^3)
\end{aligned}
$$

Therefore for $c = 99, n_0 = 1$

$$\forall n \geq n_0 : \quad \left\lceil \frac{n(n+1)}{2} \right\rceil^2 - \frac{n^2(n^2+1)}{4} + 98 \in O(n^3)$$

2. **(5 pts)** Prove or disprove $2^{2^n} \in O(2^{2n})$.

Answer

Proving by Contradiction that $2^{2^n} \notin O(2^{2n})$.

We would like to prove $2^{2^n} \notin O(2^{2n})$, and to do so we will assume the opposite is true, which means $2^{2^n} \in O(2^{2n})$. Therefore, based on the definition of Big-O runtime, we have:

$$
\begin{aligned}
\exists c, n_0 > 0 \text{ s.t. } \forall n \geq n_0 : \quad & 2^{2^n} \leq c.2^{2n} \\
\iff \quad & \log_2 2^{2^n} \leq \log_2 c.2^{2n} \\
\iff \quad & \log_2 2^{2^n} \leq \log_2 c + \log_2 2^{2n} \\
\iff \quad & 2^n \log_2 2 \leq \log_2 c + 2n \log_2 2 \\
\iff \quad & 2^n \leq \log_2 c + 2n \\
\iff \quad & 2^n - 2n \leq \log_2 c \\
\iff \quad & 2.2^{n-1} - 2(n-1+1) \leq \log_2 c \\
\iff \quad & 2.2^{n-1} - 2(n-1) - 2 \leq \log_2 c \\
\iff \quad & 2.(2^{n-1} - (n-1)) - 2 \leq \log_2 c \\
\iff \quad & 2.(2^{n-1} - (n-1)) \leq \log_2 c + 2 \\
\iff \quad & 2.(2^{n-1} - (n-1)) \leq \log_2 c + \log_2 4 \\
\iff \quad & (2^{n-1} - (n-1)) \leq \frac{\log_2 c + \log_2 4}{2} \\
\iff \quad & (2^{n-1} - (n-1)) \leq \frac{\log_2 4.c}{2}
\end{aligned}
$$

Assume $\mathcal{N} = n - 1$ and $\mathcal{C} = \frac{\log_2 4.c}{2}$ be a constant. Therefore:

$$(2^{\mathcal{N}} - \mathcal{N}) \leq \mathcal{C}$$

The right-hand side is a fixed constant, however, on the left-hand side we have an exponential function. If we show that $2^{\mathcal{N}} - \mathcal{N}$ can grow larger than any constant, such as $\mathcal{C}$, then the above inequality can't happen, which leads to a contradiction.
First, we will show by induction that $2^n > 2n$ by using mathematical induction.

1 **Base Case:** $n = 3$

$$\begin{aligned} \text{LHS:} \quad & 2^n = 2^3 = 8 \\ \text{RHS:} \quad & 2n = 2 \times 3 = 6 \end{aligned}$$

Therefor for $n = 3 : 2^n > 2n$.

2 **Induction Assumption:**
Suppose for $n \geq 3$ we know that: $2^n > 2n$.

3 **Induction Step:**
We would like to prove for $2^{n+1} > 2(n+1)$.

$$\begin{aligned} \text{LHS:} \quad & 2^{n+1} = 2.2^n = 2^n + 2^n \\ \text{RHS:} \quad & 2(n+1) = 2n + 2 \end{aligned}$$

$$2^{n+1} = 2.2^n = 2^n + 2^n \geq \underbrace{2n + 2n}_{\text{Induction assumption}} \geq \underbrace{2n + 6}_{\text{since } n \geq 3} > \underbrace{2n + 2}_{\text{since } 6 > 2} = 2(n+1)$$

$$\Rightarrow 2^{n+1} > 2(n+1)$$

Hence, $\forall n \geq 3 : 2^n > 2n$. This result is equivalent to the following:

$$\forall n \geq 3 : 2^n - n > n$$

Now, if we consider the following:

$$\mathcal{M} := \max\{3, \mathcal{C}\} \text{ and take: } \mathcal{N} \geq \mathcal{M}$$

Then:

$$\underbrace{2^{\mathcal{N}} - \mathcal{N} > \mathcal{N}}_{\text{proved by induction}} \geq \mathcal{M} \geq \mathcal{C}$$

$$\Rightarrow 2^{\mathcal{N}} - \mathcal{N} > \mathcal{C}$$

Which is a contradiction with our initial assumption of $2^{2^n} \in O(2^{2n})$. Hence: $2^{2^n} \notin O(2^{2n})$.

3. **(5 pts)** Prove that any function that is in $O(\log_2 n)$ is also in $O(\log_3 n)$.

Answer
Suppose $f(n) \in O(\log_2 n)$. Then:

$$\exists c, n_0 \text{ s.t. } \forall n \geq n_0 : \quad f(n) \leq c.\log_2 n$$

If we multiply $c.\log_2 n$ by $\frac{\log_3 n}{\log_3 n}$ the inequality still hold since $\frac{\log_3 n}{\log_3 n} = 1$ and it doesn't change the value. Therefore:

$$\begin{aligned} & f(n) \leq c.\log_2 n \times \frac{\log_3 n}{\log_3 n} \\ \Leftrightarrow & f(n) \leq c.\log_3 n \times \frac{\log_2 n}{\log_3 n} \end{aligned}$$

Based on the logarithm relations we know $\frac{\log_2 n}{\log_3 n} = log_2 3$. Hence

$$f(n) \leq c.log_2 3. \log_3 n$$

Since $log_2 3 < 2$:

$$f(n) \leq c.log_2 3. \log_3 n < \underbrace{2c}_{C}. \log_3 n$$

If we consider $C = 2c$, then, for C and $\forall n \geq n_0$ we have the relationship above, which results in:

$$f(n) \in O(\log_3 n)$$

4. **(5 pts)** Prove that if $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$, then

$$\boldsymbol{f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))}$$

Answer
Since $f_1(n) \in O(g_1(n))$ then:

$$\exists c_1, n_1 > 0 \text{ s.t. } \forall n \geq n_1 : \quad f_1(n) \leq c_1.g_1(n)$$

And, since $f_2(n) \in O(g_2(n))$ then:

$$\exists c_2, n_2 > 0 \text{ s.t. } \forall n \geq n_2 : \quad f_2(n) \leq c_2.g_2(n)$$

Since c_1, c_2, n_1, n_2 are positive integers, we can take the following:

$$\mathcal{C} := \max\{c_1, c_2\} \text{ and } \mathcal{N} := \max\{n_1, n_2\}$$

Thus, $\mathcal{C}$ and $\mathcal{N}$ are also positive integers. Then:

$$\forall n \geq \mathcal{N} : \quad f_1(n) \leq c_1.g_1(n) \leq \mathcal{C}g_1(n)$$

$$\forall n \geq \mathcal{N} : \quad f_2(n) \leq c_2.g_2(n) \leq \mathcal{C}g_2(n)$$

If we add both sides of the inequalities, we have:

$$\forall n \geq \mathcal{N} : \quad f_1(n) + f_2(n) \leq \mathcal{C}g_1(n) + \mathcal{C}g_2(n) = \mathcal{C}(g_1(n) + g_2(n))$$

Therefore:

$$f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$$

5. **(10 pts)** Derive the runtime of the following loop structure as a function of n and determine its Big-O upper bound. You must show the derivation of the end result. Simply stating the final answer without any derivation steps will result in zero points. Assume atomic operations take unit time.

```
for (i = 1; i <= n; i++)
{
    for (j = n; j >= 1; j--)
    {
        for (k = 1; k <= i + j; k++)
        {
            // some constant number of atomic operations

        } // end k

    } // end j

} // end i
```

Answer

Inner Loop (Controlled by k): Assume the set-up time for the inner loop is c_1. In addition, since in the loop we do numbers of primitive operations, we can assume these operations take a constant time c_2. Therefore:

T_{inner} :

k	Number of Primitive Operations
1	c_2
2	c_2
$\vdots$	$\vdots$
$i+j$	c_2

$$\Rightarrow T_{inner} = \sum_{k=1}^{i+j} c_2 + c_1 = \underbrace{c_2 + c_2 + \cdots + c_2}_{i+j \text{ times}} + c_1 = (i+j)c_2 + c_1$$

Middle Loop (Controlled by j): Assume the set-up time for the middle loop is c_3. Therefore:

T_{middle} :

j	Number of Primitive Operations
n	$T_{inner} + c_4$
n-1	$T_{inner} + c_4$
$\vdots$	$\vdots$
1	$T_{inner} + c_4$

$$\begin{aligned}
\Rightarrow T_{middle} &= \sum_{j=1}^{n} (T_{inner} + c_4) + c_3 \\
&= \sum_{j=1}^{n} T_{inner} + \sum_{j=1}^{n} c_4 + c_3 \\
&= \sum_{j=1}^{n} ((i+j)c_2 + c_1) + \underbrace{c_4 + c_4 + \cdots + c_4}_{n \text{ times}} + c_3 \\
&= \sum_{j=1}^{n} (i+j)c_2 + \sum_{j=1}^{n} c_1 + nc_4 + c_3 \\
&= \sum_{j=1}^{n} jc_2 + \sum_{j=1}^{n} ic_2 + \underbrace{c_1 + c_1 + \cdots + c_1}_{n \text{ times}} + nc_4 + c_3 \\
&= \frac{n(n+1)}{2} c_2 + \underbrace{ic_2 + ic_2 + \cdots + ic_2}_{n \text{ times}} + nc_1 + nc_4 + c_3 \\
&= \frac{n^2}{2} c_2 + \frac{n}{2} c_2 + nic_2 + nc_1 + nc_4 + c_3
\end{aligned}$$

Outer Loop (Controlled by i): Assume the set-up time for the outer loop is c_5. Therefore:

i	Number of Primitive Operations
1	$T_{middle} + c_6$
2	$T_{middle} + c_6$
$\vdots$	$\vdots$
n	$T_{middle} + c_6$

$$
\begin{aligned}
\Rightarrow T_{outer} &= \sum_{i=1}^{n}(T_{middle} + c_6) + c_5 \\
&= \sum_{i=1}^{n} T_{middle} + \sum_{i=1}^{n} c_6 + c_5 \\
&= \sum_{i=1}^{n}(\frac{n^2}{2}c_2 + \frac{n}{2}c_2 + nic_2 + nc_1 + nc_4 + c_3) + \underbrace{c_6 + c_6 + \cdots + c_6}_{n \text{ times}} + c_5 \\
&= \sum_{i=1}^{n}(\frac{n^2}{2}c_2 + \frac{n}{2}c_2 + nic_2 + nc_1 + nc_4 + c_3) + nc_6 + c_5 \\
&= \sum_{i=1}^{n}\frac{n^2}{2}c_2 + \sum_{i=1}^{n}\frac{n}{2}c_2 + \sum_{i=1}^{n}nic_2 + \sum_{i=1}^{n}nc_1 + \sum_{i=1}^{n}nc_4 + \sum_{i=1}^{n}c_3 + nc_6 + c_5 \\
&= n\frac{n^2}{2}c_2 + n\frac{n}{2}c_2 + \frac{n(n+1)}{2}nc_2 + n.nc_1 + n.nc_4 + nc_3 + nc_6 + c_5 \\
&= \frac{n^3}{2}c_2 + \frac{n^2}{2}c_2 + \frac{n^3+n^2}{2}c_2 + n^2c_1 + n^2c_4 + nc_3 + nc_6 + c_5 \\
&= n^3c_2 + n^2(c_2 + c_1 + c_4) + n(c_3 + c_6) + c_5 \in O(n^3)
\end{aligned}
$$

6. **(10 pts)** Derive the runtime of the following loop structure as a function of n and determine its Big-O upper bound. You must show the derivation of the end result. Simply stating the final answer without any derivation steps will result in zero points. Assume atomic operations take unit time.

```
i = n;

while (i >= 2)
{
    for (j = 1; j <= i; j++)
    {
        // some constant number of atomic elementary operations

    } // end j

    i = i / 2;

} // end while
```

Answer

Inner Loop (Controlled by j): Assume the set-up time for the inner loop is c_1. In addition, since in the loop we do numbers of primitive operations, we can assume these operations take a constant time c_2. Therefore:

T_{inner} :

j	Number of Primitive Operations
1	c_2
2	c_2
$\vdots$	$\vdots$
i	c_2

$$\Rightarrow T_{inner} = \sum_{j=1}^{i} c_2 + c_1 = \underbrace{c_2 + c_2 + \cdots + c_2}_{i \text{ times}} + c_1 = ic_2 + c_1$$

Outer Loop (Controlled by i): Assume the set-up time for the middle loop is c_3. Therefore:

T_{outer} :

Round	i	Number of Primitive Operations
1	$\frac{n}{2^0}$	$T_{inner} + c_4 = ic_2 + c_1 + c_4 = \frac{n}{2^0}c_2 + c_1 + c_4$
2	$\frac{n}{2^1}$	$T_{inner} + c_4 = ic_2 + c_1 + c_4 = \frac{n}{2^1}c_2 + c_1 + c_4$
$\vdots$	$\vdots$	$\vdots$
k	$\frac{n}{2^{k-1}}$	$T_{inner} + c_4 = ic_2 + c_1 + c_4 = \frac{n}{2^{k-1}}c_2 + c_1 + c_4$

Since the loop runs k times, Therefore, $\frac{n}{2^k} = 1$ for the while loop to end. Thus,

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow \log_2 n = log_2 2^k \Rightarrow \log_2 n = k$$

$$\begin{aligned}
\Rightarrow T_{outer} &= (\frac{n}{2^0}c_2 + c_1 + c_4) + (\frac{n}{2^1}c_2 + c_1 + c_4) + \cdots + (\frac{n}{2^{k-1}}c_2 + c_1 + c_4) + c_3 \\
&= nc_2(\frac{1}{2^0} + \frac{1}{2^1} + \cdots + \frac{1}{2^{k-1}}) + \underbrace{c_1 + c_1 + \cdots + c_1}_{k = \log_2 n \text{ times}} + \underbrace{c_4 + c_4 + \cdots + c_4}_{k = \log_2 n \text{ times}} + c_3 \\
&= nc_2 \sum_{k=0}^{\log_2 n} \frac{1}{2^{k-1}} + (\log_2 n)c_1 + (\log_2 n)c_4 + c_3 \\
&= nc_2 2(1 - \frac{1}{2^{\log_2 n}}) + (\log_2 n)c_1 + (\log_2 n)c_4 + c_3 \\
&= 2nc_2(1 - \frac{1}{n}) + (\log_2 n)c_1 + (\log_2 n)c_4 + c_3 \\
&= 2nc_2 - \frac{2nc_2}{n} + (\log_2 n)c_1 + (\log_2 n)c_4 + c_3 \\
&= 2nc_2 + (\log_2 n)c_1 + (\log_2 n)c_4 + c_3 - 2c_2 \\
&\leq 2nc_2 + nc_1 + nc_4 + nc_3 - 2c_2 \quad (\text{Since } \log_2 n \leq n) \\
&\leq 2nc_2 + nc_1 + nc_4 + nc_3 \\
&\in O(n)
\end{aligned}$$

List of Collaborators

Ryan Holt and I discussed our solutions to questions 2 and 6.
Initially, to prove $2^{\mathcal{N}} - \mathcal{N} > \mathcal{C}$ in question 2, I just explained how this function is exponential and can outgrow any constant number. Ryan gave me the idea of using mathematical induction to prove this statement. The rest of the solution regarding the induction structure and conclusion is all mine and separate from him.
In question 6, Ryan and I had doubted whether the answer would belong to $O(\log_2 n)$ or $O(n)$. By using the table of primitive operations, Ryan calculated the run-time separately. I recalculated mine on my own, and we agreed that it belonged to $O(n)$ since, with the primitive table approach, our answers ended up being the same.