Rajat Jaiswal(2017CS50415) Kabir Tomer(2017CS50410) Ananye Agarwal(2017CS10326)

---

1. (10 points) Let $f(n)$ and $g(n)$ be functions from the nonnegative integers to the positive real numbers. Prove the following transitive property from the definition of big-$O$:

$$\text{If } f(n) \in O(g(n)) \text{ and } g(n) \in O(h(n)) \text{ then } f(n) \in O(h(n)).$$

**Solution:**

- $f(n) \in O(g(n)) \Leftrightarrow$ there exists constants $c_1 > 0, n_1 > 0$ such that for all $n \geq n_1, f(n) \leq c_1 \cdot g(n)$
- $g(n) \in O(h(n)) \Leftrightarrow$ there exists constants $c_2 > 0, n_2 > 0$ such that for all $n \geq n_2, g(n) \leq c_2 \cdot h(n)$

For all $n > max(n_1, n_2)$, $f(n) \leq c_1.g(n)$, and $g(n) \leq c_2.h(n)$, which together implies $f(n) \leq c_1 \cdot c_2 \cdot h(n)$. Since, $f(n) \leq c_1 \cdot c_2 \cdot h(n)$ for all $n > max(n_1, n_2)$, therefore following the definition of big-$O$, we get, $f(n) \in O(h(n))$. Hence Proved.

2. State True or False:

(a) (2 points) $2 \cdot (3^n) \in \Theta(3 \cdot (2^n))$
**False.** $f(n) \in \Theta(g(n)) \Rightarrow f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$. But in the given question, $f(n) \notin O(g(n))$. Because $f(n) = 2 \cdot (3^n)$ grows faster than $g(n) = 3 \cdot (2^n)$, as evident by $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2\cdot(3^n)}{3\cdot(2^n)} = \infty$

(b) (2 points) $(n^6 + 2n + 1)^2 \in \Omega((3n^3 + 4n^2)^4)$
**True.** $f(n) \in \Omega(g(n))$ means $f(n)$ grows at least as fast as $g(n)$. Here, both $f(n)$ and $g(n)$ are polynomials in degree 12 and so asymptotically they both have the same rate of growth.

(c) (2 points) $\log n \in \Omega((\log n) + n)$
**False.** $f(n) \in \Omega(g(n)) \Leftrightarrow g(n) \in O(f(n))$. Here, $g(n) = (\log n) + n$, has the largest term as $n$, whereas, $f(n) = \log n$ has the largest term as $\log n$. Therefore, $g(n) \notin O(f(n)) \Rightarrow f(n) \notin \Omega(g(n))$

(d) (2 points) $n \log n + n \in O(n \log n)$
**True.** $n \log n$ grows atleast as fast as $n$, so asymptotically we can drop the term term $n$ in $f(n) = n \log n + n$. Hence both $f(n)$ and $g(n) = n \log n$ have the same rate of growth asymptotically. Therefore, $f(n) \in O(g(n))$

(e) (2 points) $\log(n^{10}) \in \Theta(\log(n))$
**True.** $g(n) = \log n$, and $f(n) = \log n^{10} = 10 \cdot \log n = 10 \cdot g(n)$. We can drop the coefficient in $f(n)$. Both $f(n)$ and $g(n)$ grow at the same rate asymptotically. Therefore, $f(n) \in \Theta(g(n))$

(f) (2 points) $\sum_{i=1}^{n} i^k \in \Theta(n^{k+1})$
**True.** $i^k \leq n^k$ for all $1 \leq i \leq n$ and $k \geq 0$. So, we get $f(n) = \sum_{i=1}^{n} i^k \leq n \cdot n^k = n^{k+1} = g(n) \Rightarrow f(n) \in O(g(n))$.
To get the lower bound, the idea is to throw away the first half of the sum. $i^k \geq (\frac{n}{2})^k$ for all $\frac{n}{2} \leq i \leq n$ and $k \geq 0$. So, we get $f(n) = \sum_{i=1}^{n} i^k \geq \sum_{i=\frac{n}{2}}^{n} i^k \geq \frac{n}{2} \cdot (\frac{n}{2})^k = (\frac{n}{2})^{k+1} = (\frac{1}{2})^{k+1} \cdot g(n)$.
If we drop the constant term from RHS, we get that asymptotically, $f(n)$ grows atleast as fast as $g(n)$. Hence, $f(n) \in \Omega(g(n))$.
The above two conditions imply, $f(n) \in \Theta(g(n))$.

(g) (2 points) $(\log(n))^{\log(n)} \in O(n/\log(n))$
**False.** If $f(n) = (\log(n))^{\log(n)}$ grows no faster than $g(n) = n/\log(n)$, then $f(2^n) = n^n$ also grows no faster than $g(2^n) = (2^n)/n$, because exponential is strictly increasing function. But $f(2^n)$ grows much faster than $g(2^n)$, $f(2^n) \in \omega(g(2^n))$. Hence, contradiction.

(h) (2 points) $n! \in O(2^n)$
**False.** $\frac{i}{2} \geq 2$ for all $4 \leq i \leq n$. $f(n) = n!$ grows faster than $g(n) = 2^n$, as evident by $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{1\cdot 2\cdot 3\cdot 4\cdot 5 \cdots n}{\underbrace{2\cdot 2\cdot 2\cdot 2\cdots 2}_{n\text{-times}}} \geq \lim_{n\to\infty} \frac{3}{4}(2^{n-3}) = \infty$. Hence, $f(n) \notin O(g(n))$

3. Given two lists $A$ of size $m$ and $B$ of length $n$, our goal is to construct a list of all elements in list $A$ that are also in list $B$. Consider the following two algorithms to solve this problem.

**procedure** `Search1`(List $A$ of size $m$, List $B$ of size $n$)
1. Initialize an empty list $L$.
2. `SORT` list $B$
3. **for** each item $a \in A$,
4.     **if** (`BinarySearch`$(a, B) \neq 0$), **then**
5.         Append $a$ to list $L$.
6. **return** $L$

**Note**: Assume that the `SORT` algorithm used in `Search1` algorithm above takes time proportional to $k \log k$ on an input list of size $k$.

**procedure** `Search2`(List $A$ of size $m$, List $B$ of size $n$)
1. Initialize an empty list $L$
2. **for** each item $a \in A$,
3.     **if** (`LinearSearch`$(a, B) \neq 0$), **then**
4.         Append $a$ to list $L$.
5. **return** $L$

Answer the following questions:

(a) (2 points) Calculate the runtime of `Search1` in $\Theta$ notation, in terms of $m$ and $n$.
$\Theta(n \log n + m \log n)$**.** Sorting list $B$ of size $n$ takes $\Theta(n \log n)$ time. Binary search takes $O(\log n)$ time in worst-case. We are performing Binary Search for each of the $m$ elements of list $A$ so that would take $\Theta(m \log n)$ time. Assuming appending to list takes constant-time, Line 5 would take $O(m)$ time in worst-case, which is asymptotically less than or equal to $\Theta(m \log n)$. Initializing the list happens in constant-time at Line 1. So, the whole algorithm takes $\Theta(n \log n + m \log n)$ time. Since, we don't know which of m or n is bigger, we can't drop any of them.

(b) (2 points) Calculate the runtime of `Search2` in $\Theta$ notation, in terms of $m$ and $n$.
$\Theta(mn)$**.** Linear search takes $O(n)$ time in worst-case. We are performing Linear Search for each of the $m$ elements of list $A$ so that would take $\Theta(mn)$ time. Assuming appending to list takes constant-time, Line 5 would take $O(m)$ time in worst-case, which is asymptotically less than or equal to $\Theta(mn)$. Initializing the list happens in constant-time at Line 1. So, the whole algorithm takes $\Theta(mn)$ time. Since, we don't know which of m or n is bigger, we can't drop any of them.

(c) (2 points) When $m \in \Theta(1)$, which algorithm has faster runtime asymptotically?
**Search2**. When $m \in \Theta(1)$, we can assume $m$ is a constant. When $m$ is constant, time complexity for Search1 becomes $\Theta(n \log n)$, and for Search2 becomes $\Theta(n)$, which is certainly faster.

(d) (2 points) When $m \in \Theta(n)$, which algorithm has faster runtime asymptotically?
**Search1**. When $m \in \Theta(n)$, $m$ is a linear function in $n$. We replace $m$ by $cn$, $c$ being a constant,

so the time complexity for Search1 becomes $\Theta(n \log n)$, and for Search2 becomes $\Theta(n^2)$. Therefore, Search1 is certainly faster.

(e) (2 points) Find a function $f(n)$ so that when $m \in \Theta(f(n))$, both algorithms have equal runtime asymptotically.
$f(n) = \log n$. When $m \in \Theta(\log n)$, $m$ is proportional to $\log n$. We replace $m$ by $c \log n$, $c$ being a constant, so the time complexity for Search1 becomes $\Theta(n \log n)$, and for Search2 also it becomes $\Theta(n \log n)$.

4. Show that if $c$ is a positive real number, then $g(n) = 1 + c + c^2 + ... + c^n$ is:

**Solution:** Using sum of geometric progression, we get, $g(n) = \frac{c^{n+1}-1}{c-1}$, $c \neq 1$.
To prove, $g(n) \in \Theta(f(n))$ , we need to show that $g(n) \in O(f(n))$ and $g(n) \in \Omega(f(n))$

(a) (3 points) $\Theta(1)$ if $c < 1$.
When $c < 1$, $g(n) = \frac{1-c^{n+1}}{1-c}$, $\lim_{n\to\infty} g(n) = \frac{1}{1-c}$. $f(n) = 1$. To get upper bound and lower bound:

- Choose $n_1 = 1 > 0$, for $g(n) \geq k_1 \cdot f(n)$ for all $n \geq n_1$. Since $c < 1 \Rightarrow c^{n+1} < c^n$, we get, $k_1 = \frac{1-c^2}{1-c}$ This makes $g(n) \in \Omega(f(n))$
- Choose $k_2 = \frac{1}{1-c} > 0$, for $g(n) \leq k_2 \cdot f(n) = \frac{1}{1-c}$, for all $n \geq n_2$, we get, $c^{n+1} \geq 0$, which is true for all $n$ since $c$ is a positive real number, so we choose $n_2 = 1$, which makes $g(n) \in O(f(n))$

(b) (3 points) $\Theta(n)$ if $c = 1$.
When $c = 1$, $g(n) = n + 1$, $f(n) = n$. To get upper bound and lower bound:

- Choose $n_1 = 1 > 0$ and $k_1 = 1 > 0$, so $g(n) \geq k_1 \cdot f(n)$ for all $n \geq n_1$, hence $g(n) \in \Omega(f(n))$
- Choose $n_2 = 1 > 0$ and $k_2 = 2 > 0$, so $g(n) \leq k_2 \cdot f(n)$ for all $n \geq n_2$, hence $g(n) \in O(f(n))$

(c) (3 points) $\Theta(c^n)$ if $c > 1$.
When $c > 1$, $g(n) = \frac{c^{n+1}-1}{c-1}$. $f(n) = c^n$. To get upper bound and lower bound:

- Choose $k_1 = 1 > 0$, for $g(n) \geq k_1 \cdot f(n)$ for all $n \geq n_1$, we get, $c^n \geq 1$, which is true for all $n$ since $c > 1$. So we choose $n_1 = 1$, which makes $g(n) \in \Omega(f(n))$
- Choose $k_2 = \frac{c}{c-1} > 0$, for $g(n) \leq k_2 \cdot f(n)$, for all $n \geq n_2$, we get, $1 \geq 0$, which is true for all $n$. So we choose $n_2 = 1$, which makes $g(n) \in O(f(n))$

5. The Fibonacci numbers $F_0, F_1, ...,$ are defined by

$$F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$$

(a) (4 points) Use induction to prove that $F_n \geq 2^{0.5n}$ for $n \geq 6$.

**Solution:**

$F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8, F_7 = 13$

Induction Hypothesis: For all $k \geq 6$, Let $P(k)$ be the proposition that $F_k \geq 2^{0.5k}$. We need to show that $P(k)$ is true for all $k \geq 6$.

Base Case: $F_6 = 8 \geq 2^{0.5*6} = 2^3, F_7 = 13 \geq 2^{0.5*7} = 8\sqrt{2} = 11.314$. Hence $P(6)$ and $P(7)$ are true.

Inductive Step: Suppose $P(6), P(7), P(8), ...., P(i-1), P(i)$ are true. We have to show that $P(i+1)$ is also true. From induction, we know that $F_i \geq 2^{0.5i}$ and $F_{i-1} \geq 2^{0.5(i-1)}$. Since, $F_{i+1} = F_i + F_{i-1} \Rightarrow F_{i+1} \geq 2^{0.5i} + 2^{0.5(i-1)} = (\frac{\sqrt{2}+1}{\sqrt{2}}) \cdot 2^{0.5i} \geq \sqrt{2} \cdot 2^{0.5i} = 2^{0.5(i+1)}$. Hence $P(i+1)$ is true.

(b) (4 points) Use induction to prove that $F_n \leq 2^n$ for $n \geq 0$.

**Solution:**

$F_0 = 0, F_1 = 1, F_2 = 1, F_3 = 2, F_4 = 3, F_5 = 5, F_6 = 8, F_7 = 13$

Induction Hypothesis: For all $k \geq 0$, Let $P(k)$ be the proposition that $F_k \leq 2^k$. We need to show that $P(k)$ is true for all $k \geq 0$.

Base Case: $F_0 = 0 \leq 2^0 = 1, F_1 = 1 \leq 2^1 = 2$. Hence $P(0)$ and $P(1)$ are true.

Inductive Step: Suppose $P(0), P(1), P(2), ...., P(i-1), P(i)$ are true. We have to show that $P(i+1)$ is also true. From induction, we know that $F_i \leq 2^i$ and $F_{i-1} \geq 2^{i-1}$. Since, $F_{i+1} = F_i + F_{i-1} \Rightarrow F_{i+1} \leq 2^i + 2^{i-1} = (\frac{3}{2}) \cdot 2^i \leq 2 \cdot 2^i = 2^{i+1}$. Hence $P(i+1)$ is true.

(c) (2 points) What can we conclude about the growth of $F_n$?

**Solution:**

In the above two part (a) and (b), we have seen that $F_n$ is upper bounded by an exponential function ($2^n$), and $F_n$ is also lower bounded by an exponential function ($2^{0.5n}$). Hence, it can be concluded that $F_n$ **grows exponentially**.