CS311 HW1

Jongmin Lee

February 10 2024

1 Problem 1

$\frac{n(n+1)}{2}^2 - \frac{n^2(n^2+1)}{4} + 78 \in O(n^3) \rightarrow \frac{n^2(n+1)^2}{2^2} - \frac{n^2(n^2+1}{4} + 78 \leq C * n^3$
$\rightarrow \frac{n^2(n^2+2n+1)}{4} - \frac{n^2(n^2+1)}{4} + 78 \leq C * n^3 \rightarrow \frac{n^4+2n^3+n^2-n^4-n^2}{4} + 78 \subset C * n^3$
$\rightarrow \frac{2n^3}{4} + 78 \leq C * n^3 \rightarrow 4 * (\frac{2n^3}{4} + 78) \leq 4 * (C * n^3)$
$\rightarrow 2n^3 + 312 \leq 4 * C * n^3 \rightarrow 2n^3 - (2n^3 + 312) \leq 4Cn^3 - 2n^3 \rightarrow 312 \leq n^3(4C - 2)$
Therefore, C is equal to 1 and n is equal to 7 since 7^3 is equal to 343

2 Problem 2

Prove or disprove $2^{2^n} \in O(2^{2n})$
Since we know this is not true because left grows faster than right. We are going to disprove.
If we apply logarithm to this formula, then
$\log C + 2n \geq 2^n \rightarrow (\log C + 2n) - 2n \geq (2^n) - 2n \rightarrow \log C \geq 2^n - 2n$
As you can see $\log C$ is constant and $2^n - 2n$ is keep growing.
Which means that this assumption is wrong
Therefore, $2^{2^n} \notin O(2^{2n})$

3 Problem 3

Prove that any function that is in $O(\log_2(n))$ is also in $O(\log_3(n))$
When we make it to formula we get, $\log_2(n) \in O(\log_3(n))$
In logarithm there is $\log_a(b) = \frac{\log_c b}{\log_c a}$ We can use this to derive the solution.
There exist c, n_0, $\frac{\log n}{\log 2} \leq c * \frac{\log n}{\log 3}$ When we divide $\log n$ from both side we get, $\frac{\log 3}{\log 2} \leq c$ From this we can derive c and n_0 by doing same thing of logarithm's rule reversely which is $c = \log_2 3$ and $n_0 = 1$ since n disappeared.

4 Problem 4

Prove that if $f_1(n) \in O(g_1(n)) and f_2(n) \in O(g_2(n))$ then
$f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$
In different form is
There exist c_1, n_{0_1} $f_1(n) \leq c_1 * g_1(n)$
and there exist c_2, n_{0_2} $f_2(n) \leq c * g_2(n)$
Since c_1 and c_2 is lowest boundary constant, bigger than c_1 and c_2 is not a big problem. Which means we can add it together and make it to $c_3 \rightarrow c_1 + c_2 = c_3$
Also it can be applied for n_{0_1} and n_{0_2} too, because after n_0 it's always bigger than previous. So, $n_{0_1} + n_{0_2} = n_{0_3}$
There exist c_3 and n_{0_3},
We are going to add $f_2(n)$ on both side of $f_1(n) \leq c_3 * g_1(n)$
Then it become, $f_1(n) + f_2(n) \leq c_3 * g_1(n) + f_2(n)$ ——————(1)
and also add $c_3 * g_1(n)$ on both side of $f_2(n) \leq c_3 * g_2(n)$
Then it become, $f_2(n) + c_3 * g_1(n) \leq c_3 * g_2(n) + c_3 * g_1(n)$———(2)
What we can see from this is that both equation (1) and (2) have same part which is $c_3 * g_1(n) + f_2(n)$ which means that we can link two formulas together.
That leads to this formula:
$f_1(n) + f_2(n) \leq c_3 * g_1(n) + f_2(n) \leq c_3 * g_1(n) + c_3 * g_2(n)$
which same as $f_1(n) + f_2(n) \leq c_3 * g_1(n) + c_3 * g_2(n)$
Because of this, $f_1(n) \in O(g_1(n))$ and $f_2(n) \in O(g_2(n))$ then
$f_1(n) + f_2(n) \in O(g_1(n) + g_2(n))$ is right.

5 Problem 5

For this problem first loop with i++ is $\sum\limits_{i=1}^{n}$
Second loop with j- - is $\sum\limits_{j=1}^{n}$
And inner loop with k++ is $\sum\limits_{k=1}^{i+j}$
So if we add it up together equation of runtime for this loop is
$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n}\sum\limits_{k=1}^{i+j} 1 \rightarrow \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} i + j + 1 \rightarrow \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} i + \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} j + \sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} 1$
$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} i = \sum\limits_{i=1}^{n} i * n = n * \sum\limits_{i=1}^{n} i = n * \frac{n(n+1)}{2} = n * \frac{n^2+n}{2} = \frac{n^3+n^2}{2}$
$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} j = \sum\limits_{i=1}^{n} \frac{n(n+1)}{2} = n * (\frac{n(n+1)}{2}) = \frac{n^3+n^2}{2}$
$\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{n} 1 = \sum\limits_{i=1}^{n} 1 = n^2$
So if we add it all up then we get
$2(\frac{n^3+n^2}{2} + \frac{n^3+n^2}{2} + n^2) = 2n^3 + 4n^2$
For worst case runtime which is biggest n, the answer for problem 5 is $O(n^3)$

6 Problem 6

Runtime of inner loop for this problem is $\sum_{j=1}^{n} 1 = n$

Since outer loop is while loop which divide i value by 2 each time we cannot use summation. So we need table.

number of iteration	i value	cost
0	n	n
1	$\frac{n}{2}$	n
2	$\frac{n}{2^2}$	n
...	...	n
k	$\frac{n}{2^k}$	n

And what we can see is last i value is $\frac{n}{2^k}$ since loop ends at $i \geq 2$ situation, smallest value $\frac{n}{2^k}$ can be is 2. Which means $\frac{n}{2^k} = 2$. From here we can use logarithm. If we apply logarithm this equation becomes $\log_2 2 = k$

From logarithm we found k value which is $\log_2 2$. Since cost is constant n that runs $\log_2 2$ times, runtime of this code is $O(\log_2 2n)$