CS6033 Assign No.2

Minghe Yang

February 2021

1 Basic Complexity

1.a
Set $C_2 = 0.1, C_1 = 5$, for $n > n_0 = 100$,
$(c_1 - 1)n^3 - 3n^2 - n + 1 = 4n^3 - 3n^2 - n + 1 > 0$, since it's monotonically increasing and higher than 0 when n=100
$(1 - c_2)n^3 - 3n^2 - n + 1 = 0.9n^3 - 3n^2 - n + 1 > 0$ since it's monotonically increasing and higher than 0 when n=100
So $n^3 - 3n^2 - n + 1 = \Theta(n^3)$

1.b
Set $c = 1, and\ n_0 = 10$, for $n > n_0 = 100$,
$1 * g(n) - f(n) > 0$,and it's Derivative $= 2^n lgn - 2n > 0$, it's monotically increasing.
So $n^2 = O(2^n)$

2.a
let $c = 1, n_0 = 2$
for $n = 2$, $c*g(n) - f(n) = 4 - 1 - 2.828 > 0, (c*g(n) - f(n))' = 2n - 1.5 * n^0.5 = 4 - 1.5 * 1.4 > 0$ and it's a monotically increasing.
$f(n) = O(g(n))$

let $c = 1, n_0 = 1$
for $n = 1$, $c*g(n) - f(n) = 1 - 1 - 1 = -1 < 0, (f(n) - c*g(n))' = 1.5 * 1 - 2 < 0$ it's monotically decreasing.
So $f(n) \neq \Omega(g(n))$

Since it doesn't satisfy $f(n) = \Omega(g(n))$, so $f(n) \neq \Theta(g(n))$. You need both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ to get $f(n) = \Theta(g(n))$

3.a
You can't find one function $f(n)$ like that, Since you need both $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$ to support $f(n) = \Theta(g(n))$

3.b

$f(n) = n^5, g(n) = n^4$
for$n_0 = 5$ *and* $c_0 = 1$, after n_0 you can easily get the conclusion that $f(n) \geq c * g(n)$, so $f(n) = \Omega(g(n))$, and $(f(n) - c(g(n)))' = 5n^4 - 4cn^3$,so $f(n)$ will finally be large enough than $c * (g(n))$, which means $f(n) \neq O(g(n))$.

4.

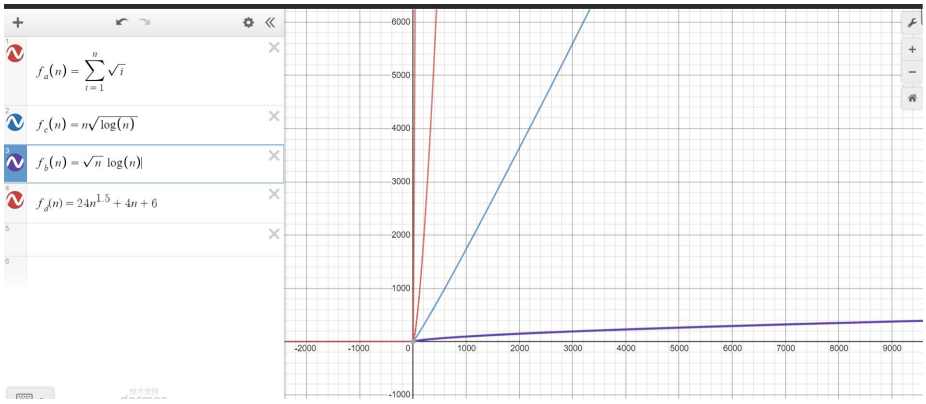
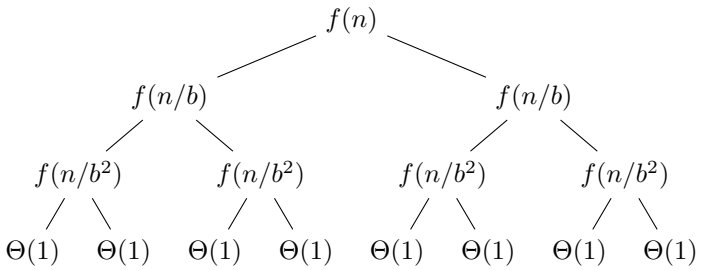


Figure 1: Comparison

$f_b(n) < f_c(n) < f_a(n) < f_d(n)$

2 Master Theorem

1.a



This recursion tree will have a branches, which means a subnodes, depth 1 is $f(n/b)$, but also $af(n/b)$, total depth is $\log_b n$, the last $\Theta(1)$ is also $\Theta(n^{\log_b a})$.

1.b
In depth j we have a^j nodes, with each costs $f(n/b^j)$, so total cost at each depth will be (c) $a^j * f(n/b^j)$.Every leaf cost is $T(1) = \Theta(1)$, with the depth of (a) $\log_b n$ and number of (b) $n^{\log_b a}$.

So add all the the cost together: $\Theta(1)$ + all branches (d) $T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$.

2.a.1
From the assumption that $f(n) = \Theta(n^{\log_b a})$, so we have $f(n/b^j) = \Theta((n/b^j)^{\log_b a})$.
Put it in the eqation above and you'll get:
$g(n) = \Theta(\sum_{j=0}^{log_b n-1} a^j (\frac{n}{b^j})^{\log_b a})$

2.a.3
$\sum_{j=0}^{log_b n-1} a^j (\frac{n}{b^j})^{\log_b a} = n^{\log_b a} \sum_{j=0}^{log_b n-1} *(\frac{a}{b^{\log_b a}})^j = n^{\log_b a} \sum_{j=0}^{log_b n-1} 1 = n^{\log_b a} \log_b n$
So $g(n) = \Theta(n^{\log_b a} \log_b n) = \Theta(n^{\log_b a} * \lg n)$

2.b.2
From the situatiation,$f(n/b^j) = O((n/b^j)^{\log_b a-\epsilon})$
So $g(n) = O(\sum_{j=0}^{log_b n-1} a^j * (\frac{n}{b^j})^{\log_b a-\epsilon})$
$\sum_{j=0}^{log_b n-1} a^j * (\frac{n}{b^j})^{\log_b a-\epsilon} = n^{\log_b a-\epsilon} \sum_{j=0}^{log_b n-1} (\frac{ab^\epsilon}{b^{\log_b a}})^j = n^{\log_b a-\epsilon} \sum_{j=0}^{log_b n-1} (b^\epsilon)^j$
$= n^{\log_b a-\epsilon} (\frac{b^{\epsilon \log_b n}-1}{b^\epsilon-1}) = n^{\log_b a-\epsilon} (\frac{n^\epsilon-1}{b^\epsilon-1})$

2.b.3 Since b, ϵ are constants, we can rewrite the conclusion above to :
$n^{\log_b a-\epsilon} O(n^\epsilon) = O(n^{\log_b a})$, take into $g(n)$, and we'll get $g(n) = O(n^{\log_b a})$

2.c
Set $g(n) = \sum_{j=0}^{log_b n-1} a^j f(n/b^j)$, we can get $a^j f(n/b^j) \leq c^j f(n)$
So $g(n) \leq f(n) \sum_{j=0}^{log_b n-1} c^j \leq f(n) \sum_{j=0}^{\infty} c^j = \frac{1}{1-c} f(n)$
So $g(n) = O(f(n))$
What's more, $g(n) = f(n) + af(\frac{n}{b}) + \cdots + a^{(\log_b n)-1} f(\frac{n}{b^{(\log_b n)-1}}) \geq f(n)$
So $g(n) = \Omega(f(n))$
With both condition satisfied, we can conclude that:
$g(n) = \Theta(f(n))$

2.3
Set constants $a \geq 1, b>1, f(n)$ be a function, and $T(n) = aT(n/b) + f(n)$ since $n = b^i$,
From the question 1 we can get $T(n) = \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$ From the recursion tree.
$\sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$ is all the cost from nodes, and in question 2,
for a constant $\epsilon>0$ we have $f(n) = O(n^{(log_b a)-\epsilon}$, $g(n) = O(n^{log_b a}), T(n) = \Theta(n^{\log_b a}) + O(n^{\log_b a}) = \Theta(n^{\log_b a})$
If $f(n) = \Theta(n^{\log_b a})$, $T(n) = \Theta(n^{\log_b a} \log_n)$
for a constant $c<1$ and all n large enough, $af(n/b) \leq cf(n)$,
Since $f(n) = \Omega(n^{(log_b a)+\epsilon})$, $T(n) = \Theta(n^{\log_b a}) + \Theta f(n) = \Theta(n^{(log_b a)+\epsilon})$

3 Ramanujam numbers

1.

Algorithm 1 Ramanujam numbers

Input: n
Output: *Ramanujam numbers* $\leq n$

```
1: function Ram(n)
2:     i = 1, l = [ ]
3:     while i ≤ n do
4:         intcount = 0
5:         for j ∈ (1, i^(1/3)) do
6:             for k ∈ (j + 1, i^(1/3)) do
7:                 if j^3 + k^3 = i then
8:                     intcount ← intcount + 1
9:                 end if
10:            end for
11:        end for
12:        if intcount ≥ 2 then
13:            l ← l + i
14:        end if
15:        i ← i + 1
16:    end while
17:    return l
18: end function
```

The complexity should be $O(n * n^{1/3} * n^{1/3}) = O(n^{5/3})$

4 Critical Thinking

1.
Name those pirates P1, P2, P3, P4, P5, and P6, while P6 is the first one to propose and P1 be the last.

Start with P1 and P2, at this time P2 get whatever he proposed, so he'll choose all 300 coins for himself and poor have nothing to do with it.

And here comes P3, if he wants to get a 2:1, he'll choose P1 to be his teammate, so 1 coin for him, to vote agree for this design, the proposal will be {299, 0, 1} for P3, P2, and P1.

As for P4, He needs another voter to get 50%. If his proposal is denied, then P3 will win, and P2 get nothing, so P4 should send a coin to P2, the proposal will be {299, 0, 1, 0} for P4, P3, P2, and P1.

If P4's proposal wins, P3 and P1 will get nothing. P5 then comes up with a new idea to let himself win, The proposal will be {298, 0, 1, 0, 1}.

Then P6 should come up with a same one, but give coins to different people to get 50% upvote, The final proposal will be {298, 0, 1, 0, 1, 0} For P6, P5, P4, P3, P2, and P1.