UM-SJTU JOINT INSTITUTE

Introduction to Algorithms
(VE477)

Homework #2

Prof. Manuel

Xinmiao Yu
518021910792

Oct. 2, 2020

Q1.

1.a Prove that there exist three constants c_1, c_2, n_0 such that for $n > n_0$, $c_2 n^3 \leq n^3 - 3n^2 - n + 1 \leq c_1 n^3$. Obviously choose $c_1 = 2, n_0 = 1$, $n^3 - 3n^2 - n + 1 \leq n^3 \leq c_1 n^3$. So $T(n) = \mathcal{O}(n^3)$.

Choose $n_0 = 5$, then $3n^2 \leq \frac{3}{5}n^3$, $n \leq \frac{1}{25}n^3$. So

$$n^3 - 3n^2 - n + 1 \geq n^3 - \frac{3}{5}n^3 - \frac{1}{25}n^3 + 1 \geq \frac{9}{25}n^3 + 1 \geq \frac{1}{5}n^3.$$

Choose $c_2 = \frac{1}{5}$, then $T(n) = \Omega(n^3)$. So $n^3 - 3n^2 - n + 1 = \Theta(n^3)$

1.b Prove that there exist two constants c, n_0 such that for all $n > n_0$, $n^2 < c2^n$, which is same as $2ln(n) < cnln2$. Take the derivative, $\frac{2}{n} < cln2$, so when $n > n_0 = 2$, choose $c = 1$. As $n_0^2 = 2^{n_0}$ and 2^n will grow faster, $n^2 < c2^n$ for specific n_0 and c have been proved, so $n^2 = \mathcal{O}(2^n)$.

2.a $f(n) = \mathcal{O}g(n)$. For $n > n_0 = 2$

$$n\sqrt{n} \leq n * n - 1$$

Choose $c = 1$, it satisfies $f(n) = \mathcal{O}g(n)$.

3.a Such two functions do not exist. When $f(n) = \Theta(g(n))$, existn_0, c_1, c_2 such that for $n > n_0$, $c_2 g(n) \leq f(n) \leq c_1 g(n)$, so the condition of $f(n) = \mathcal{O}(g(n))$ must meet.

3.b f(n) = n^3, g(n) = n^2.

4 $f_2(n) < f_3(n) < f_1(n) < f_4(n)$

As shown in lecture $n > \sqrt{n} > logn$,

$$\lim_{n\to\infty} \frac{\sqrt{n}logn}{n\sqrt{logn}} = \lim_{n\to\infty} \frac{logn}{n} = 1 < \infty$$

so $f_2(n) = \mathcal{O}(f_3(n))$.

As $(\sqrt{x} + \sqrt{y})^2 = x + y + 2\sqrt{xy} > (\sqrt{x+y})^2$,

$$\sum_{i=1}^{n} \sqrt{i} > \sqrt{\sum_{i=1}^{n} i} > \sqrt{\frac{n^2+n}{2}} > \sqrt{n^2} > \sqrt{n}logn$$

So $f_2(n) = \mathcal{O}(f_1(n))$

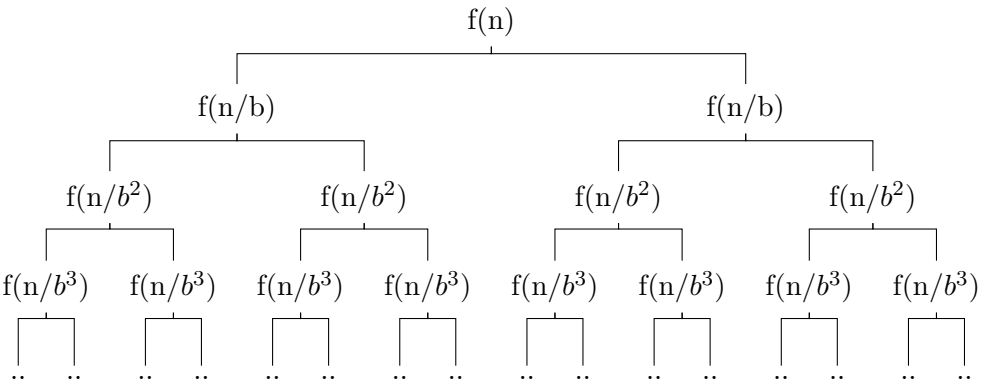
$$\sum_{i=1}^{n} \sqrt{i} < n\sqrt{n} < 24n^{3/2} + 4n + 6$$

$f_1(n) = \mathcal{O}(f_4(n))$.

Then $f_2(n) < f_3(n) < f_1(n) < f_4(n)$ has been proved.

Q2.

1.a Begin with function $f(n)$ and assume $a = 2$



1.b.1 As for each node, the input size is n/b^j. When $n/b^j = 1$, it is the depth of the tree. So depth: $log_b n$

1.b.2 From the graph, the number of leaves at each depth j is a^j.

1.b.3 at each depth j, total cost: $a^j f(n/b^j)$

1.b.4

$$T(n) = \sum_{j=0}^{log_b n} a^j f(n/b^j) = \sum_{j=0}^{log_b n-1} a^j f(n/b^j) + a^{log_b n} f(1)$$
$$= \sum_{j=0}^{log_b n-1} a^j f(n/b^j) + n^{log_b a} f(1) = \sum_{j=0}^{log_b n-1} a^j f(n/b^j) + \Theta(n^{log_b a})$$

2.a.1 From $f(n) = \Theta(n^{log_b a})$, obtain that there exists three constants n_0, c_1, c_2 such that for all $n > n_0$

$$c_1 \cdot n^{log_b a} \leq f(n) \leq c_2 \cdot n^{log_b a}$$

Then if replace n with n/b^j and multiply with a_j then calculate the sum,

$$c_1 \cdot \sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a} \leq g(n) \leq c_2 \cdot \sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a}$$

So

$$g(n) = \Theta\left(\sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a}\right)$$

2.a.3 From the conclusion obtained in 2.a.2,

$$g(n) = \Theta\left(\sum_{j=0}^{\log_b n-1} a^j \left(\frac{n}{b^j}\right)^{\log_b a}\right) = \Theta(n^{\log_b a} \log_b n)$$

2.b.2

$$\begin{aligned}\sum_{j=0}^{\log_b n-1} a^j \left(n/b^j\right)^{\log_b a-\varepsilon} &= n^{\log_b a-\varepsilon}\sum_{j=0}^{\log_b n-1} a^j b^{-j\log_b a} b^{j\varepsilon} = n^{\log_b a-\varepsilon}\sum_{j=0}^{\log_b n-1} a^j a^{-j} b^{j\varepsilon}\\ &= n^{\log_b a-\varepsilon}\sum_{j=0}^{\log_b n-1} b^{\varepsilon j} = n^{\log_b a-\varepsilon}\frac{b^{\varepsilon(\log_b n)}-1}{1-b^{\varepsilon}}\\ &= n^{\log_b a-\varepsilon}\frac{n^{\varepsilon}-1}{b^{\varepsilon}-1}\end{aligned}$$

2.b.3 From 2.b.1 and 2.b.2,

$$g(n) = \mathcal{O}\left(\sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a-\varepsilon}\right) = \mathcal{O}(n^{\log_b a-\varepsilon}\frac{n^{\varepsilon}-1}{b^{\varepsilon}-1})$$

As

$$n^{\log_b a-\varepsilon}\frac{n^{\varepsilon}-1}{b^{\varepsilon}-1} \le n^{\log_b a-\varepsilon}\frac{n^{\varepsilon}}{b^{\varepsilon}-1} = n^{\log_b a}\frac{1}{b^{\varepsilon}-1},$$

where $\frac{1}{b^{\varepsilon}-1}$ is a constant. So,

$$g(n) = \mathcal{O}(n^{\log_b a})$$

2.c.1 From $g(n) = \sum_{j=0}^{log_b n-1} a^j f(n/b^j) = \sum_{j=1}^{log_b n-1} a^j f(n/b^j) + f(n) > f(n)$, we could conclude that $g(n) = \Omega(f(n))$.

2.c.2 From $af(n/b) \le cf(n)$, we could set $x = n/b^{j-1}$, so that

$$a^j(n/b^j) = a^{j-1}\cdot af(x/b) \le a^{j-1}\cdot cf(x) = ca^{j-1}f(n/b^{j-1})$$

So that we could repeat the steps above, until we get $f(n)$.

$$a^j f\left(n/b^j\right) \le c\cdot a^{j-1}f(n/b^{j-1}) \le c^2\cdot a^{j-2}f(n/b^{j-2}) \le \cdots \le c^j f(n)$$

2.c.3 $g(n) = \sum_{j=0}^{log_b n-1} a^j f(n/b^j) \le \sum_{j=0}^{log_b n-1} c^j f(n) \le \frac{1}{1-c} f(n)$, so $g(n) = \mathcal{O}(f(n))$

2.c.4 So to prove that there exist three constants c_1, c_2, n_0 such that for $n > n_0$, $c_2 f(n) \le g(n) \le c_1 f(n)$. As proved in 2.c.3 with $c_1 = \frac{1}{1-c}$, only need to find $c_2 f(n) \le g(n)$

$$g(n) = \sum_{j=0}^{\log n-1} a^j f\left(n/b^j\right) = af\left(n\right) + \sum_{j=1}^{\log n-1} a^j f\left(n/b^j\right) > f(n)$$

so $g(n) = \Omega(f(n))$. Therefore, $g(n) = \Theta(f(n))$.

3 Let $a \ge 1$ and $b > 1$ be constants, and n is an exact power of b. Then T(n) could be defined as

$$T(n) = \begin{cases}\Theta(1) & \text{if} \quad \text{n} = 1\\ aT(n/b) + f(n) & \text{if} \quad \text{n} = b^i\end{cases}$$

where i is a positive integer. Then from the conclusion we obtained,

- $f(n) = \Theta(n^{log_b a})$, $T(n) = g(n) + \mathcal{O}(n^{log_b n}) = \mathcal{O}(n^{log_b n} log_b n) + \mathcal{O}(n^{log_b n}) = \mathcal{O}(n^{log_b n} log_b n)$
- when $f(n) = O(n^{log_b a - \varepsilon})$, same procedure as above.
- $f(n) \geq \frac{a}{c} f(n/b)$, because $f(n) = \Omega(n^{log_b a + \varepsilon})$, take $g(n) = \Theta(f(n))$ only.

$$T(n) = \begin{cases} \Theta(n^{log_b a} log_b n) & f(n) = \Theta(n^{log_b a}) \\ \Theta(n^{log_b a}) & f(n) = O(n^{log_b a - \varepsilon}) \\ \Theta(f(n)) & f(n) \geq \dfrac{a}{c} f(n/b) \end{cases}$$

Q3.

Algorithm 1: Ramanujam numbers

Input : positive integer n

Output: all Ramanujam numbers smaller or equal to n

```
1  Function FindRamanujam(n):
2      i ← 1 ;
3      cubeList ← [ ];
4      while i < ∛n do
5          push i^3 into cubeList;
6      end while
7      sumList ← [ ];
8      for j ← 1 to ⌈∛n⌉ do
9          for k ← 1 to ⌈∛n⌉ do
10             push j + k into sumList;
11         end for
12     end for
13     for all elements in the sumList do
14         count the time they occur;
15     end for
16     result ← elements that occur twice;
17     return result
18 end
```

- Find cube for all numbers that smaller than $\sqrt[3]{n} = \mathcal{O}(n^{1/3})$
- Sum all pairs in the cubeList = $\mathcal{O}(n^{2/3})$
- Find all numbers that occur twice = $\mathcal{O}(n)$

So the total time complexity should be $\mathcal{O}(n)$.

Q4.

The six pirates is P1, P2, P3, P4, P5, P6 and P1 is the senior-most pirate.
We consider from the situation such that only one pirate survive. Because for those pirates with proper thinking abilities, they would consider all the possible distribution methods to decide whether they should vote.

1. Only P6 alive, P6 will have 300 coins.

2. P5 and P6 alive. P5 will vote YES for any distribution he proposed (1 out of 2), so any distribution method could be accepted. Then distribution should be (P5, P6) have (300, 0) respectively,

3. P4, P5 and P6 alive. As P4 knows P6 will have no coin if P4 dies, P4 will give P6 one coin. Then 2 out of 3 will vote YES, P5 will not vote YES for P4 as he always could have better income afterwards. Then distribution should be (P4, P5, P6) have (299, 0, 1) respectively.

4. P3, P4, P5 and P6 alive. P3 need to get support from P5, as P5 obtain nothing in P4's distribution. Then with 2 out of 4 YES, P3's distribution could be accepted. Then distribution should be (P3, P4, P5, P6) have (299, 0, 1, 0) respectively.

5. P2, P3, P4, P5 and P6 alive. P2 need to get support from P4 and P6, because they have nothing in P3's distribution. then 3 out of 5 YES will make it accepted. Then distribution should be (P2, P3, P4, P5, P6) have (298, 0, 1, 0, 1) respectively.

6. All alive. P1 need to get support from P3 and P5. 3 out of 6 YES will make it accepted. Then distribution should be (P1, P2, P3, P4, P5, P6) have (298, 0, 1, 0, 1, 0) respectively.

Number of alive pirates / coins for each pirates	1	2	3	4	5	6
1	\	\	\	\	\	298
2	\	\	\	\	298	0
3	\	\	\	299	0	1
4	\	\	299	0	1	0
5	\	300	0	1	0	1
6	300	0	1	0	1	0