VE477 Introduction to Algorithms

Homework 2

Taoyue Xia, 518370910087

2021/10/01

Ex. 1 — Basic complexity

1. a) To prove this statement, it is equivalent to prove that $c_1 n^3 \leq n^3 - 3n^2 - n + 1 \leq c_2 n^3$ holds when $n \to \infty$, where c_1 and c_2 are two positive constant number.

   Firstly, we can find that $n^3 - 3n^2 - n + 1 \leq n^3$ when $n \geq 1$. Therefore, $c_2 = 1$ and $n^3 - 3n^2 - n + 1 = \mathcal{O}(n^3)$.

   Then, let $f(n) = (1 - c_1)n^3 - 3n^2 - n + 1$, then take its derivative $f'(n) = 3(1 - c_1)n^2 - 6n - 1$. We can see that $f(n)$ is strictly increasing on $[\frac{1}{1-c_1}, +\infty)$. Take c_1 as $\frac{1}{4}$. After some calculation, we can have $f(n) = (1 - c_1)n^3 - 3n^2 - n + 1 > 0$ for $n \geq 5$. Therefore, $\frac{1}{4}n^3 \leq n^3 - 3^n - n + 1$ for $n \geq 5$, indicating that $n^3 - 3n^2 - n + 1 = \Omega(n^3)$.

   Combining the two results, we can finally get to the conclusion that

   $$n^3 - 3n^2 - n + 1 = \mathbf{\Theta}(n^3)$$

   b) To prove the statement, it is equivalent to prove that $n^2 \leq c2^n$ for some constant c and $n \to \infty$. Take the logarithm of both sides, thus we need to prove that $2\log(n) \leq cn \log 2$.

   It is common knowledge that $\log(n) \leq n$ when $n \geq 1$. Therefore, we just need by choosing $c = \frac{2}{\log 2}$, we can have $2\log(n) \leq cn \log 2$ for all $n \geq 1$. Therefore, $n^2 = \mathcal{O}(2^n)$. Proof done.

   c) Firstly, if $a = 0$, it is obviously true, since $n^b \leq n^b \leq n^b$.

   Then, when $a > 0$, we see that

   $$f(n) = (n + a)^b = \sum_{i=0}^{b} \binom{b}{i} a^i n^{b-i}$$

   $$f(n) \leq (\sum_{i=0}^{b} \binom{b}{i} a^i) n^b$$

   So for $c_1 = (\sum_{i=0}^{b} \binom{b}{i} a^i)$, $(n + a)^b \leq c_1 n^b$. Thus, $(n + a)^b = \mathcal{O}(n^b)$. And since $a > 0$, $(n + a)^b \geq n^b$, so $(n + a)^b = \Omega(n^b)$. Therefore, combining the two results, we can conclude that $(n + a)^b = \mathbf{\Theta}(n^b)$.

When $a < 0$, firstly, same as the above condition, for $c_1 = \sum_{i=0}^{b} \binom{b}{i} a^i$, $(n+a)^b \leq c_1 n^b$, indicating that $(n+a)^b = \mathcal{O}(n^b)$. Also, similar to the question a), for any finite N, we can find a c_2 such that when $n \geq N$, $(n+a)^b \geq c_2 n^b$. It needs attention that $0 < c_2 < 1$. Thus, $(n+a)^b = \mathbf{\Theta}(n^b)$.

Since the above three conditions are all met, we can conclude that $(n+a)^b = \mathbf{\Theta}(n^b)$. Proof done.

2. (a) $f(n) = \mathcal{O}(g(n))$. Construct $h(n) = n^2 - 1 - n\sqrt{n}$, then take its derivative, we can get:

$$h'(n) = 2n - (\sqrt{n} + \frac{\sqrt{n}}{2}) = 2n - \frac{3}{2}\sqrt{n}$$

It is obvious that for $n \geq 1$, $h(n)$ is strictly increasing. Since when $n \geq 2 \Rightarrow h(n) \geq 0$, thus $n\sqrt{n} < n^2 - 1$, we can have $f(n) < g(n)$. Therefore, $f(n) = \mathcal{O}(g(n))$. Proof done.

(b) $f(n) = \Omega(g(n))$. Construct $h(n) = f(n) - g(n) = 2^n - n^2 - n^4 - n^2$, then take its logarithm, we can get:

$$s(n) = \log h(n) = n\log 2 - 2\log n - 4\log n - 2\log n = n\log 2 - 8\log n$$

Then take the derivative of $s(n)$, we can get, $s'(n) = \log 2 - 8/n$. For $n \geq 14$, $s'(n) > 0$, therefore, $h(n)$ is strictly increasing on $[14, +\infty)$.

Therefore, when $n \geq 17$, $h(n) > 0$, which indicates that $f(n) > g(n)$ when $n \to +\infty$. In conclusion, $f(n) = \Omega(g(n))$. Proof done.

3. a) Since the requirement for $f(n) = \mathbf{\Theta}(g(n))$ is $f(n) = \mathcal{O}(n)$ and $f(n) = \Omega(g(n))$ at the same time, the statement is false, so we cannot find such two functions.

b) From problem 2(b), $f(n) = 2^n - n^2$ and $g(n) = n^4 + n^2$ can meet the requirement for $f(n) = \Omega(g(n))$ and $f(n) \neq \mathcal{O}(g(n))$.

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

First we construct $g(n) = f_2(n) - f_3(n) = \sqrt{n}\log n - n\sqrt{\log n}$. Then take its derivative,

we can get:

$$\begin{aligned} g'(n) &= \frac{\log n}{2\sqrt{n}} + \frac{\sqrt{n}}{n} - (\sqrt{\log n} + n \cdot \frac{1}{2\sqrt{\log n}} \cdot \frac{1}{n}) \\ &= \frac{\log n + 2}{2\sqrt{n}} - \frac{2\log n + 1}{2\sqrt{\log n}} \\ &= \frac{\log n\sqrt{\log n} + 2\sqrt{\log n} - 2\sqrt{n}\log n - \sqrt{n}}{2\sqrt{n\log n}} \\ &= \frac{\log n(\sqrt{\log n} - 2\sqrt{n}) + (2\sqrt{\log n} - \sqrt{n})}{2\sqrt{n\log n}} \end{aligned}$$

From the slides we know that $\log n < \sqrt{n} < n$, which indicates that $\sqrt{\log n} < \sqrt{n}$. Thus, $g'(n) < 0$ for $n \geq 1$. In this sense, we can know that for $n \to +\infty$, $f_2(n) < f_3(n)$, which means that $f_2(n) = \mathcal{O}(f_3(n))$.

For $f_1(n) = \sum_{i=1}^{n} \sqrt{i}$, we can take its square, and get:

$$\begin{aligned} f_1^2(n) = (\sum_{i=1}^{n} \sqrt{i})^2 &= \sum_{i=1}^{n} i + 2 \sum_{1 \leq i < j \leq n} \sqrt{ij} \\ &\geq \frac{n^2 + n}{2} + 2\sum_{i=1}^{n-1} \sqrt{i^2}(n - i) \\ &= \frac{n^2 + n}{2} + 2\sum_{i=1}^{n-1} (ni - i^2) \\ &= \frac{n^2 + n}{2} + 2(n \cdot \frac{n^2 - n}{2} - \frac{n(n+1)(2n+1)}{6}) \\ &= \boldsymbol{\Theta}(n^3) \end{aligned}$$

Then we take the square of $f_3(n)$, which gives us $f_3^2(n) = n^2 \log n < n^2\sqrt{n} < n^3$. Therefore, we can conclude that $f_3(n) < f_1(n)$.

For $f_1(n)$, we know that:

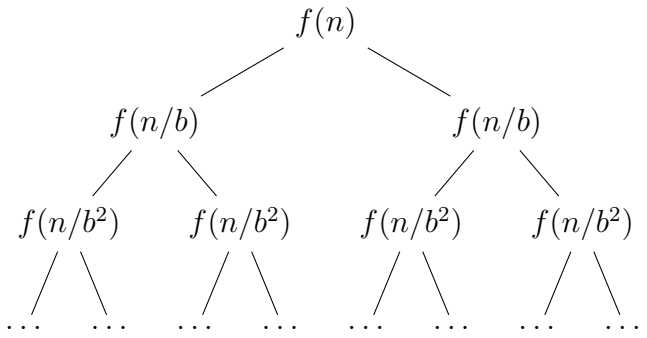
$$\sum_{i=1}^{n} \sqrt{i} \leq \frac{2}{3}(n + \frac{1}{2})^{\frac{3}{2}}$$

Since $f_4(n) = 24n^{\frac{3}{2}} + 4n + 6$, we can simply find that $f_1(n) < f_4(n)$.

According to all the calculations above, we can finally conclude that $f_2(n) < f_3(n) < f_1(n) < f_4(n)$. Proof done.

Ex. 2 — Master Theorem

1. a) The recurrence tree is shown below:



b) i) Let the depth of tree be d, then $n/b^d = 1$ when it reaches the bottom. Therefore, the depth of the tree is $\log_b n$.

ii) The number of leaves is a to the power of depth, which is $a^{\log_b n}$.

iii) The total cost for all the nodes at each depth is the cost of each node at that depth multiplied by the number of nodes. Suppose that the depth is d, so the total cost is $a^d f(n/b^d)$.

iv) The overall cost is the sum of the costs of all the depths, which is calculated as:

$$\begin{aligned} T(n) &= \sum_{i=0}^{\log_b n} a^i f(n/b^i) \\ &= a^{\log_b n} f(n/b^{\log_b n}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i) \\ &= n^{\log_b a} f(1) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i) \\ &= \Theta(n^{\log_b a}) + \sum_{i=0}^{\log_b n - 1} a^i f(n/b^i) \end{aligned}$$

2. (a) i) Since $f(n) = \Theta(n^{\log_b a})$, we can find some constant c_1, c_2, such that:

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

Then, by substituting n with n/b^j, for $0 \leq j \leq \log_b n - 1$, and multiplied by a^j, we can have:

$$c_1(a^j(\frac{n}{b^j})^{\log_b a}) \leq a^j f(\frac{n}{b^j}) \leq c_2(a^j(\frac{n}{b^j})^{\log_b a})$$

Calculating the sum for j from 0 to $\log_b n - 1$, we can get:

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

Since $g(n) = \sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$, we can finally reach the conclusion that $g(n) = \mathbf{\Theta}(\sum_{j=0}^{\log_b n-1} a^j(\frac{n}{b^j})^{\log_b a})$. Proof done.

ii) Firstly, we can do some formula transformation:

$$a^j(\frac{n}{b^j})^{\log_b a} = a^j \frac{n^{\log_b a}}{(b^j)^{\log_b a}} = a^j \frac{n^{\log_b a}}{(b^{\log_b a})^j} = a^j \frac{n^{\log_b a}}{a^j} = n^{\log_b a}$$

We can find that the deduced expression does not contain j. Therefore, we can simply get to the point:

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

Proof done.

iii) From problem 2.a.i, we get:

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

From problem 2.a.ii, we get:

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

Combining the two formulas, we can finally get:

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

Therefore, $g(n) = \mathbf{\Theta}(n^{\log_b a} \log n)$. Proof done.

(b) i) Since $f(n) = \mathcal{O}(n^{\log_b a-\varepsilon})$, we can find some constant c, such that:

$$f(n) \leq cn^{\log_b a-\varepsilon}$$

Then, by substituting n with n/b^j, for $0 \leq j \leq \log_b n - 1$, and multiplied by a^j, we can have:

$$a^j f(\frac{n}{b^j}) \leq c_2(a^j(\frac{n}{b^j})^{\log_b a-\varepsilon})$$

Calculating the sum for j from 0 to $\log_b n - 1$, we can get:

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

Since $g(n) = \sum_{j=0}^{\log_b n-1} a^j f(n/b^j)$, we can finally reach the conclusion that

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

Proof done.

ii) Firstly, we can do some formula transformation:

$$\begin{aligned} a^j(\frac{n}{b^j})^{\log_b a-\varepsilon} &= a^j \frac{n^{\log_b a-\varepsilon}}{(b^j)^{\log_b a-\varepsilon}} \\ &= a^j \frac{n^{\log_b a-\varepsilon}}{(b^{\log_b a-\varepsilon})^j} \\ &= a^j \frac{n^{\log_b a-\varepsilon}}{(\frac{a}{b^\varepsilon})^j} \\ &= n^{\log_b a-\varepsilon} b^{j\varepsilon} \end{aligned}$$

Then we calculate the sum for $0 \leq j \leq \log_b n - 1$:

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

Proof done.

iii) From problem 2.b.i, we get:

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

From problem 2.b.ii, we get:

$$\sum_{j=0}^{\log_b n-1} a^j (\frac{n}{b^j})^{\log_b a-\varepsilon} = \frac{n^\varepsilon - 1}{b^\varepsilon - 1} n^{\log_b a-\varepsilon}$$

Combine the two formulas, we can get:

$$g(n) = \mathcal{O}(\frac{n^\varepsilon - 1}{b^\varepsilon - 1} n^{\log_b a-\varepsilon})$$

This tells us that there exists some constant c such that:

$$g(n) \leq c(\frac{n^\varepsilon - 1}{b^\varepsilon - 1} n^{\log_b a-\varepsilon})$$

Since $\varepsilon > 0$, $n^\varepsilon - 1 > 0$ and $b^\varepsilon - 1 > 0$, thus:

$$\frac{n^\varepsilon - 1}{b^\varepsilon - 1} n^{\log_b a-\varepsilon} < \frac{n^\varepsilon}{b^\varepsilon - 1} n^{\log_b a-\varepsilon} < n^{\log_b a} \cdot \frac{1}{b^\varepsilon - 1}$$

where $1/(b^\varepsilon - 1)$ is a constant. Therefore, we can have the following:

$$g(n) \leq c(\frac{n^\varepsilon - 1}{b^\varepsilon - 1} n^{\log_b a-\varepsilon}) < \frac{c}{b^\varepsilon - 1} n^{\log_b a}$$

So we can conclude that $g(n) = \mathcal{O}(n^{\log_b a})$. Proof done.

(c) i. From the expression of $g(n)$, we can deduce that:

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

Therefore, $g(n) = \Omega(f(n))$. Proof done.

ii. Since $af(n/b) \leq cf(n)$, by changing n to n/b^{j-1} and multiplying a^{j-1} to both sides, we can get $af(n/b^i) \leq cf(n/b^{i-1}))$ for $1 \leq i \leq \log_b n - 1$. Therefore, we can do the following recurrence:

$$a^j f(\frac{n}{b^j}) \leq c \cdot a^{j-1} f(\frac{n}{b^{j-1}}) \leq c^2 \cdot a^{j-2} f(\frac{n}{b^{j-2}}) \leq \cdots \leq c^{j-1} \cdot af(\frac{n}{b}) \leq c^j f(n)$$

Proof done.

iii. From the definition of $g(n)$ and the result of problem 2.c.ii, we can have:

$$g(n) = \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j) \leq \sum_{j=0}^{\log_b n - 1} c^j f(n) = \frac{1 - c^{\log_b n}}{1 - c} f(n)$$

Since $\frac{1-c^{\log_b n}}{1-c}$ is a constant, we can finally say that $g(n) = \mathcal{O}(f(n))$. Proof done.

iv. Combining the conclusions of problem 2.c.i $g(n) = \Omega(f(n))$ and 2.c.iii $g(n) = \mathcal{O}(f(n))$, we can simply say that $g(n) = \mathbf{\Theta}(f(n))$. Proof done.

3. Let $a \geq 1$, $b > 1$ be two constants, $f(n)$ be a function, and $T(n) = aT(n/b) + f(n)$, where $n = b^i$ with i a positive integer, and $T(1) = O(1)$ be a recurrence relation over the positive integers.

Firstly, for $f(n) = \mathbf{\Theta}(n^{\log_b a})$, we will have:

$$\begin{aligned} T(n) &= \sum_{j=0}^{\log_b n - 1} a^j f(\frac{n}{b^j}) + \sum_{j=0}^{i-1} f(n/b^j) \\ &= \mathbf{\Theta}(n^{\log_b a} \log n) + \mathbf{\Theta}(n^{\log_b a}) \\ &= \mathbf{\Theta}(n^{\log_b a} \log n) \end{aligned}$$

Secondly, for $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$, we will have:

$$\begin{aligned} T(n) &= \sum_{j=0}^{\log_b n - 1} a^j f(\frac{n}{b^j}) + \sum_{j=0}^{i-1} f(n/b^j) \\ &= \mathcal{O}(n^{\log_b a}) + \mathcal{O}(n^{\log_b a - \varepsilon}) \\ &= \mathcal{O}(n^{\log_b a}) \end{aligned}$$

Finally, when $af(n/b) \leq cf(n)$ for some $c < 1$ and $f(n) = \Omega(n^{\log_b a + \varepsilon})$, since $g(n) = \mathbf{\Theta}(f(n))$, we will have $T(n) = \mathbf{\Theta}(f(n))$.

In all, we can sum up the Master Theorem as:

$$T(n) = \begin{cases} \mathbf{\Theta}(n^{\log_b a} \log n) & \text{if } f(n) = \mathbf{\Theta}(n^{\log_b a}) \\ \mathbf{\Theta}(n^{\log_b a}) & \text{if } f(n) = \mathcal{O}(n^{\log_b a - \varepsilon}), \varepsilon > 0 \\ \mathbf{\Theta}(f(n)) & \text{if } f(n) = \Omega(n^{\log_b a + \varepsilon}), \varepsilon > 0, \text{ and } af(n/b) \leq cf(n), c < 1 \end{cases}$$

Proof done.

Ex. 3 — Ramanujam numbers

Algorithm 1: Ramanujam numbers detection

Input : a positive integer n

Output: a list containing all the Ramanujam numbers less or equal to n

```
1  Function DetectRamanujam(n):
2      sumList ← [ ];
3      for i ← 1 to ⌊∛n⌋ do
4          if j^3 + (j + 1)^3 > n then
5              break;
6          end
7          for j ← i + 1 to ⌊∛n⌋ do
8              if i^3 + j^3 > n then
9                  break;
10             end
11             k ← i^3 + j^3;
12             push k into sumList;
13         end
14     end
15     Rams ← the list of all numbers in sumList that occur twice;
16     return Rams
17 end
```

The time complexity of constructing the **sumList** is $\mathcal{O}(n^{2/3})$.

The time complexity of finding the Ramanujam numbers from sumList is $\mathcal{O}(n^{2/3})$ if using a dictionary for judging.

Therefore, the total complexity is $\mathcal{O}(n^{2/3})$.

Ex. 4 — Critical thinking

Denote the six pirates as A, B, C, D, E, F, and A with the highest seniority, while E the lowest. Then, all the possible situations will be listed above:

1. Only E and F survive the game. Since E can vote agree and meet the requirement for half agreement, E will keep all 300 pieces of gold himself, and F will get nothing.

2. If D, E and F survive. To avoid being thrown overboard, D should get support from F, since if D leaves, F will get nothing. Therefore, D will give F 1 piece of gold, and keep 299 himself, while E will get nothing.

3. If C, D, E and F survive. C needs to get one support so that he won't be thrown overboard. If C leaves, E will get nothing as the former condition, so C can give E 1 piece of gold, and keep 299 himself, while D and F will get nothing.

4. If B, C, D, E and F survive. B needs to get two supports so that he won't be thrown aboard. If B leaves, according to the former condition, D an F will get nothing, so B can give D and F 1 piece of gold each, and keep the rest 298 himself, while C and E will get nothing.

5. If all the pirates survive. A also needs to get two supports so that he won't be thrown aboard. Since if A leaves, C and E will get nothing, A can give C and E 1 piece of gold each, and keep the rest 298 himself, while B, D and F get nothing.

The following table will show in simplicity:

gold \ pirate / cond	A	B	C	D	E	F
1	\	\	\	\	300	0
2	\	\	\	299	0	1
3	\	\	299	0	1	0
4	\	298	0	1	0	1
5	298	0	1	0	1	0

*note: "cond" denotes for the above conditions.