

# 6.046/18.410 Problem Set 1

Yijun Jiang

Collaborator: Hengyun Zhou, Eric Lau

October 14, 2015

## 1 Problem 1-1: 6.006 Review

### 1.1 Part (a)

**1.1.1 Suppose  $f(n) = \Theta(g(n))$ , then  $2^{f(n)} = \Theta(2^{g(n)})$**

This statement is FALSE.

Consider  $f(n) = n^2$  and  $g(n) = n^2 + n$ . Obviously we have  $f(n) = \Theta(g(n))$ . But  $2^{g(n)} = 2^n 2^{f(n)}$ , so it is impossible to find  $C_1 \geq 0$  and  $n_0 \in \mathbb{N}$  such that  $\forall n > n_0, C_1 \cdot 2^{g(n)} \leq 2^{f(n)}$ .

In this case,  $2^{g(n)}$  is NOT an asymptotic lower bound of  $2^{f(n)}$ . So the original statement is false.

**1.1.2 For any constants  $a, b > 0$ ,  $af(n) + bg(n) = \Theta(\max(f(n), g(n)))$**

This statement is TRUE.

Let  $C_1 = \min(a, b)$  and  $C_2 = a + b$ . Clearly we have  $C_1, C_2 > 0$  and

$$0 \leq C_1 \max(f(n), g(n)) \leq af(n) + bg(n) \leq C_2 \max(f(n), g(n))$$

which means that  $af(n) + bg(n) = \Theta(\max(f(n), g(n)))$ .

**1.1.3 Suppose  $f(n) = o(1)$ , then  $f(n)g(n) = o(1)$**

This statement is FALSE.

Consider  $f(n) = n^{-1}$  and  $g(n) = n$ . We have  $f(n) = o(1)$ , but  $f(n)g(n) = 1 \neq o(1)$ .

### 1.1.4 Rank functions by order of growth

The ordering (growth rate from large to small, i.e.  $g_k = \Omega(g_{k+1})$  for  $k = 1, 2, \dots, 11$ ) is given below. The proof can be found after this list.

$$\begin{aligned}
 g_1 &= n! \\
 g_2 &= 4^n \\
 g_3 &= 2^n \\
 g_4 &= 3^{\log^2 n} \\
 g_5 &= (\log n)^{\log n} = g_6 = n^{\log \log n} \\
 g_7 &= n^{10} \\
 g_8 &= n^3 \\
 g_9 &= n \log n \\
 g_{10} &= \sum_{k=1}^n \log k \\
 g_{11} &= \log \log n \\
 g_{12} &= 100000^{100000000}
 \end{aligned}$$

$g_5 = (\log n)^{\log n}$  and  $g_6 = n^{\log \log n}$  belong to the same equivalent class. In fact, they are equal.  $g_9 = n \log n$  and  $g_{10} = \sum_{k=1}^n \log k$  belong to the same equivalent class. Each of the remaining functions is partitioned into its own equivalent class.

**Proof:**

To compare the first 7 functions in the list, we can take log first.

$$\begin{aligned}
 f_1 &= \log g_1 \approx n \log n - n \text{ [Stirling approx.]} \\
 f_2 &= \log g_2 = n \log 4 \\
 f_3 &= \log g_3 = n \log 2 \\
 f_4 &= \log g_4 = \log^2 n \log 3 \\
 f_5 &= \log g_5 = \log n \log \log n \\
 f_6 &= \log g_6 = \log n \log \log n \\
 f_7 &= \log g_7 = 10 \log n
 \end{aligned}$$

Then it is clear that  $g_1 = \Omega(g_2), \dots, g_6 = \Omega(g_7)$ . Moreover, we notice that  $g_5 = g_6$ . The rest of the functions can be compared without taking log. The only slightly tricky one is  $g_{10} = \sum_{k=1}^n \log k$ . Notice that  $\log n \leq g_{10} \leq n \log n$ , then the comparisons can be made.

Finally, I will prove  $g_9 = \Theta(g_{10})$ . Since  $y = \log x$  is concave, an integral can be used as the lower bound of the sum:

$$g_{10} \geq (\ln 2)^{-1} \int_1^n \ln x dx = n \log n - \frac{n-1}{\ln 2}$$

On the other hand  $g_{10} \leq n \log n$ . So  $g_{10} = \Theta(n \log n)$ , and then  $g_9 = \Theta(g_{10})$ .

## 1.2 Recurrences

### 1.2.1 $T(n) = 10T(n/3) + n^2$

$$T(n) = \Theta(n^{\log_3 10}).$$

**Proof:**

Use the master theorem (case 1).  $a = 10$ ,  $b = 3$ ,  $\log_b a = \log_3 10 > 2$ . So  $n^2 = O(n^{\log_b a - \epsilon})$  for some  $\epsilon > 0$ . Therefore,  $T(n) = \Theta(n^{\log_3 10})$ .

**1.2.2**  $T(n) = 9T(n/3) + n^2 \log n$ 

$$T(n) = \Theta(n^2 \log^2 n).$$

**Proof:**

Use the master theorem (case 2).  $a = 9$ ,  $b = 3$ ,  $\log_b a = 2$ . So  $n^2 \log n = \Theta(n^{\log_b a} \log n)$ . Therefore,  $T(n) = \Theta(n^2 \log^2 n)$ .

**1.2.3**  $T(n) = T(\sqrt{n}) + \log n$ 

$$T(n) = \Theta(\log n).$$

**Proof:**

$$\begin{aligned} T(n) &= T(\sqrt{n}) + \log n \\ &= T(n^{1/4}) + \log n + \frac{1}{2} \log n \\ &= \dots \\ &= \Theta(1) + \left(1 + \frac{1}{2} + \frac{1}{2^2} + \dots\right) \log n \\ &= \Theta(\log n) \end{aligned}$$

**1.2.4**  $T(n) = T(n/4) + T(n/2) + n$ 

$$T(n) = \Theta(n).$$

**Proof:**

The recurrence is linear. So the guess is  $T(n) = \Theta(n)$ . To prove this, use the substitution method. Suppose  $C_1 n \leq T(n) \leq C_2 n$ , where  $C_1$  and  $C_2$  are positive. Then for the lower bound we have

$$\begin{aligned} T(n) &= T(n/4) + T(n/2) + n \\ &\geq \left(\frac{1}{4}C_1 + \frac{1}{2}C_1 + 1\right)n \\ &= \left(\frac{3}{4}C_1 + 1\right)n \\ [desired] &\geq C_1 n \end{aligned}$$

Obviously, if  $C_1 \leq 4$ , the desired inequality holds. On the other hand, for the upper bound,

$$\begin{aligned} T(n) &= T(n/4) + T(n/2) + n \\ &\leq \left(\frac{1}{4}C_2 + \frac{1}{2}C_2 + 1\right)n \\ &= \left(\frac{3}{4}C_2 + 1\right)n \\ [desired] &\leq C_2 n \end{aligned}$$

Obviously, if  $C_2 \geq 4$ , the desired inequality holds. This completes the prove that  $T(n) = \Theta(n)$ .

**1.2.5**  $T(n) = T(2n/3) + T(n/3) + n \log n$ 

$$T(n) = \Theta(n \log^2 n).$$

**Proof**

Use the recursion tree method. The tree is shown below (only the coefficients are written out). We notice that  $2/3 + 1/3 = 1$ , thus the coefficients in each level sum up to unity. This feature guarantees that there are  $\Theta(n)$  leaves in the tree, eaching contributing constant runtime. Moreover, each level contributes  $\Theta(n \log n)$

since the sum of coefficients in a level is unity. And there are  $\Theta(\log n)$  levels (bounded between  $\log_{3/2} n$  and  $\log_3 n$ ). In conclusion, we have

$$T(n) = \Theta(n) + \Theta(n \log n) \Theta(\log n) = \Theta(n \log^2 n)$$

