CS170–Spring 2019 — Homework 2 Solutions

Ran Liao, SID 3034504227

Collaborators: YiLin Wu, YeFan Zhou

1 Study Group

Name	SID
Ran Liao	3034504227
YiLin Wu	3034503863
YeFan Zhou	3034503824

2 Asymptotic Complexity Comparisons

(a) $3 \le 7 \le 2 \le 5 \le 4 \le 9 \le 8 \le 6 \le 1$

(b) (i) $\log_3 n = \Theta(\log_4 n)$
Proof:

$$\log_3 n = \log_{(4^{\log_4 3})} n = (\frac{1}{\log_4 3}) \log_4 n = \Theta(\log_4 n)$$

(ii) $n\log(n^4) = O(n^2\log(n^3))$
Proof:

$$\lim_{n\to+\infty} \frac{n^2\log(n^3)}{n\log(n^4)} = \lim_{n\to+\infty} \frac{3n^2\log n}{4n\log n} = \lim_{n\to+\infty} \frac{3}{4}n = \infty$$

(iii) $\sqrt{n} = \Omega((\log n)^3)$
Proof: (Use L'Hôpital's rule)

$$\begin{aligned}\lim_{n\to+\infty} \frac{\sqrt{n}}{(\log n)^3} &= \lim_{n\to+\infty} \frac{\frac{1}{2\sqrt{n}}}{3(\log n)^2\frac{1}{n}} = \lim_{n\to+\infty} \frac{\sqrt{n}}{(\log n)^2} \\ &= \lim_{n\to+\infty} \frac{\frac{1}{2\sqrt{n}}}{2(\log n)\frac{1}{n}} = \lim_{n\to+\infty} \frac{\sqrt{n}}{\log n} \\ &= \lim_{n\to+\infty} \frac{\frac{1}{2\sqrt{n}}}{\frac{1}{n}} = \lim_{n\to+\infty} \sqrt{n} = \infty\end{aligned}$$

(iv) $2^n = \Theta(2^{n+1})$
Proof:

$$2^{n+1} = 2 * 2^n = \Theta(2^n)$$

(v) $n = \Omega((\log n)^{\log\log n})$
Proof:(Use L'Hôpital's rule)
Take the logarithm of both functions

$$\log(\log n)^{\log\log n} = (\log\log n)^2$$

Therefore, it is equivalent to compare $(\log\log n)^2$ and $\log n$

$$\begin{aligned}\lim_{n\to+\infty} \frac{(\log\log n)^2}{\log n} &= \lim_{n\to+\infty} \frac{2(\log\log n)\frac{1}{\log n}\frac{1}{n}}{\frac{1}{n}} = \lim_{n\to+\infty} \frac{2(\log\log n)}{\log n} \\ &= \lim_{n\to+\infty} \frac{\frac{1}{\log n}\frac{1}{n}}{\frac{1}{n}} = \lim_{n\to+\infty} \frac{1}{\log n} = 0\end{aligned}$$

(vi) $n + \log n = \Theta(n + (\log n)^2)$
Proof:(Use L'Hôpital's rule)

$$\begin{aligned}\lim_{n\to+\infty} \frac{n+\log n}{n+(\log n)^2} &= \lim_{n\to+\infty} \frac{1+\frac{1}{n}}{1+2(\log n)\frac{1}{n}} = \lim_{n\to+\infty} \frac{n+1}{n+2\log n} \\ &= \lim_{n\to+\infty} \frac{1}{1+\frac{2}{n}} = 1\end{aligned}$$

(vii) $\log(n!) = O(n\log n)$
Proof: Since $n\log n = log(n^n)$, it's equivalent to compare $n!$ and n^n. Obviously, $n! = O(n^n)$.

3 In Between Functions

Disproof:

Consider the following function, this is a counterexample to the claim.

$$f(n) = \lfloor 2^{\sqrt{n}} \rfloor$$

Since for $\forall c > 0$

$$\lim_{n \to +\infty} \frac{\lfloor 2^{\sqrt{n}} \rfloor}{n^c} = +\infty \tag{1}$$

By (1), first part of claim doesn't hold. This can be proved by keep using L'Hôpital's rule.

In the meantime, for $\forall \alpha > 1$

$$\lim_{n \to +\infty} \frac{\alpha^n}{\lfloor 2^{\sqrt{n}} \rfloor} = +\infty \tag{2}$$

By (2), second part of claim doesn't hold. Again, this can be proved similarly with L'Hôpital's rule.

4 Bit Counter

n	# Total Flips
1	1
2	4
3	11
4	26

There're two stages when counting from 0 to $2^n - 1$ for a n-bit counter.

- Stage1 : when most-significant bit is 0
- Stage2 : when most-significant bit is 1

Suppose $f(n)$ represents the total number of flips need for n-bit long counter. Both in stage 1 and 2, $f(n-1)$ flips is needed to flip all bits to 1 except the most-significant bit. When switching between stage 1 and 2, another n flips is introduced. 1 flip to switch the most-significant bit from 0 to 1. $n-1$ flips needed to switch all other bits to 0. Therefore, we have the following recursive formula:

$$f(n) = 2f(n-1) + n$$

To solve this formula, we keep using it recursively, providing $f(1) = 1$ as base case:

$$\begin{aligned}
f(n) &= 2f(n-1) + n \\
&= 4f(n-2) + 2(n-1) + n \\
&= 8f(n-3) + 4(n-2) + 2(n-1) + n \\
&= 2^k f(n-k) + 2^{k-1}(n-k+1) + \ldots + 2^2(n-2) + 2^1(n-1) + 2^0(n-0) \\
&= 2^{n-1} f(1) + 2^{n-2}(2) + \ldots + 2^2(n-2) + 2^1(n-1) + 2^0(n-0) \\
&= 2^{n-1}(1) + 2^{n-2}(2) + \ldots + 2^2(n-2) + 2^1(n-1) + 2^0(n-0)
\end{aligned} \tag{3}$$

Then we multiply it by 2:

$$2f(n) = 2^n(1) + 2^{n-1}(2) + \ldots + 2^3(n-2) + 2^2(n-1) + 2^1(n-0) \tag{4}$$

By (4) - (3), we have

$$\begin{aligned}
2f(n) - f(n) &= 2^n + 2^{n-1} + \ldots + 2^1 - n \\
f(n) &= \frac{2(1-2^n)}{1-2} - n \\
&= 2^{n+1} - n - 2 \\
&= \Theta(2^n)
\end{aligned}$$

5 Recurrence Relations

Master theorem:

Suppose $a, b, c \in \mathbb{R}^+$, $b > 1$ and $T(1) = 1$

Given

$$T(n) = aT(\frac{n}{b}) + \Theta(n^c),$$

We have

$$T[n] = \begin{cases} \Theta(n^{\log_b a}) & c < \log_b a \\ \Theta(n^c \log_2 n) & c = \log_b a \\ \Theta(n^c) & c > \log_b a \end{cases}$$

(a) Let $a = 4, b = 2, c = 1$. Since $\log_b a = log_2 4 = 2 > c$, $T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$.

(b) Let $a = 4, b = 3, c = 2$. Since $\log_b a = log_3 4 < c$, $T(n) = \Theta(n^c) = \Theta(n^2)$.

(c)

$$\begin{aligned} T(n) &= T(\sqrt{n}) + 1 \\ &= T(n^{\frac{1}{2^1}}) + 1 \\ &= T(n^{\frac{1}{2^2}}) + 1 + 1 \\ &= T(n^{\frac{1}{2^k}}) + k \end{aligned}$$

This recursion process ends when $n^{\frac{1}{2^k}}$ reaches 2.

$$\begin{aligned} n^{\frac{1}{2^k}} &= 2 \\ \frac{1}{2^k} &= \log_n 2 \\ 2^k &= \log_2 n \\ k &= \log_2 \log_2 n \end{aligned}$$

Therefore, $T(n) = \Theta(\log \log n)$

6 Hadamard matrices

(a)

$$H_0 = \begin{bmatrix} 1 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

(b)

$$H_2 v = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 4 \end{bmatrix}$$

(c)

$$\begin{aligned} u_1 &= H_1(v_1 + v_2) \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} u_2 &= H_1(v_1 - v_2) \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \left(\begin{bmatrix} 1 \\ -1 \end{bmatrix} - \begin{bmatrix} -1 \\ 1 \end{bmatrix} \right) \\ &= \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 2 \\ -2 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ 4 \end{bmatrix} \end{aligned}$$

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 4 \end{bmatrix}$$

u is identical to $H_2 v$

(d)

$$
\begin{aligned}
H_k v &= \begin{bmatrix} H_{k-1} & H_{k-1} \\ H_{k-1} & -H_{k-1} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \\
&= \begin{bmatrix} H_{k-1}v_1 + H_{k-1}v_2 \\ H_{k-1}v_1 - H_{k-1}v_2 \end{bmatrix} \\
&= \begin{bmatrix} H_{k-1}(v_1 + v_2) \\ H_{k-1}(v_1 - v_2) \end{bmatrix}
\end{aligned} \tag{5}
$$

(e) (i) **Main idea**

This is a recursive algorithm.

If the length of v is 1, just return $H_0 v$ immediately. This is the base case.

Otherwise, let v_1 and v_2 be the top and bottom half of the vector v, respectively. Invoke this algorithm recursively to compute $H_{k-1}v_1$ and $H_{k-1}v_2$. Return $\begin{bmatrix} H_{k-1}v_1 + H_{k-1}v_2 \\ H_{k-1}v_1 - H_{k-1}v_2 \end{bmatrix}$ as result.

(ii) **Proof of correctness**

By (5), this strategy is correct mathematically.

(iii) **Running time**

This algorithm divides the problem into 2 subproblems, thus reducing the problem size by factor 2. Adding several 1-d vectors will cost $O(n)$ operations. Thus, we have the following formula.

$$T(n) = 2T(\frac{n}{2}) + O(n)$$

By Master theorem, $T(n) = O(n \log n)$.

7 Fastest Winning Strategys

(a)