به نام خدا

پاسخ تمرین‌های فصل اول — دکتر مرجان کائدی

دانشکده مهندسی کامپیوتر دانشگاه اصفهان — نیمسال دوم ۱۳۹۹-۱۴۰۰

.۱

$\frac{1}{5}n^{\left(\frac{1}{\log n}\right)} < e^{20} < \sqrt{\log n} < \ln(\ln n) < \log n^2 < (\log n)^2 < 2^{\log n} < 12(\log n)! < 10n\sqrt{n} <$
$n^{15} + 20n + 25\log n < e^n < n! + n < 100n^n + \pi$

.۲

1. $c_1 \times n^3 \le (n + \log n)^3 \le c_2 \times n^3\ for\ n \ge N \xrightarrow{c_1=1,\ c_2=2,\ N=0}$
   $n^3 \le n^3 + (\log n)^3 + 3n^2\log n + 3n(\log n)^2 \le 2n^3$ *always true*
2. $n\sqrt{n} \le c \times n^n\ for\ n \ge N \xrightarrow{c=1,\ N=2} n^{1.5} \le n^n \Rightarrow 1.5 \le n$ *always true*
3. $c_1 \times n^3 \le \sum_{i=1}^{n}\sum_{j=1}^{i} j \le c_2 \times n^3\ for\ n \ge N \xrightarrow{c=\frac{1}{6},\ c_2=\frac{1}{3},\ N=4} \frac{1}{6}n^3 \le \sum_{i=1}^{n}\frac{i(i+1)}{2} \le \frac{1}{3}n^3$
   $\Rightarrow \frac{1}{6}n^3 \le \frac{1}{2}\sum_{i=1}^{n} i^2 + i \le \frac{1}{3}n^3 \Rightarrow \frac{1}{2}\left(\frac{n(n+1)(2n+1)}{6} + \frac{n(n+1)}{2}\right) \ge \frac{1}{6}n^3$
   $\Rightarrow n^3 \le n^3 + 3n^2 + 2n \le n^3$ *always true*
4. $c_1 \times f(n) \le f(n) + o(f(n)) \le c_2 \times f(n) for\ n \ge N \xrightarrow{c_1=\frac{1}{2},\ c_2=\frac{3}{2},\ N=1}$
   $\frac{1}{2}f(n) \le (1+\epsilon)f(n) \le \frac{3}{2}f(n)\ where\ \epsilon \le \frac{1}{2}$ *always true*
5. $f(n) + g(n) \le c \times max(f(n), g(n))\ for\ n \ge N \xrightarrow{c=2,\ N=1}$
   $f(n) + g(n) \le 2 \times max(f(n), g(n))$ *always true*
6. $f^2(n) \ge c \times f(n) for\ n \ge N \xrightarrow{c=1,\ N=0} f^2(n) \ge f(n)$ *always true*

.۳

1. *false*, $if\ f(n) = 2n\ and\ g(n) = n \Rightarrow O(2^{2n} = 4^n) \notin O(2^n)$
2. *false*, $if\ f(n) = \frac{1}{n} \Rightarrow \frac{1}{n} \notin O(\frac{1}{n^2})$
3. *false*, $if\ f(n) = n\ and\ g(n) = n^2 \Rightarrow n + n^2 \notin \Theta(\min(n, n^2) = n)$
4. *false*, $if\ f(n) = 4^n \Rightarrow 4^n \notin \Theta(4^{\frac{n}{2}} = 2^n)$
5. *true*, $if\ f(n) \in O(g(n)) \Rightarrow 0 \le f(n) \le c_1 \times g(n)\ for\ all\ n > n_0$
   $\overset{log}{\Rightarrow} 0 \le \log(f(n)) \le c_2 \times \log(g(n)) \Rightarrow \log(f(n)) \le c_2 \times \log(k \times f(n))\ for\ all\ n > n_0$

۴.

الف)

$$T_A(n) = 0.1n^2 \, log\, n \Longrightarrow \mathrm{O}(n^2 \log n)$$
$$T_B(n) = 2.5n^2 \Longrightarrow \mathrm{O}(n^2)$$

بنابراین الگوریتم B بهتر می‌باشد.

ب)

$$2.5n^2 \le 0.1n^2 \, log\, n \Longrightarrow 2.5 \le 0.1 \, log\, n \Longrightarrow 25 \le log\, n \Longrightarrow n \ge 2^{25} \Longrightarrow n_0 = 2^{25}$$

ج) با توجه به قسمت ب، اگر $n \le 2^9$ باشد آنگاه الگوریتم A بهتر است.

۵. برای محاسبه باید از عدد X شروع کرده و آن را در خودش ضرب کنیم. سپس همنین کار را برای حاصل به دست آمده انجام دهیم و آن قدر این کار را ادامه دهیم تا به عدد خواسته شده برسیم. بنابراین داریم:

$$X \times \mathrm{X} = \mathrm{X}^2$$
$$\Longrightarrow \mathrm{X}^2 \times \mathrm{X}^2 = \mathrm{X}^4$$
$$\Longrightarrow \mathrm{X}^4 \times \mathrm{X}^4 = \mathrm{X}^{16}$$
$$\Longrightarrow \mathrm{X}^{16} \times \mathrm{X}^{16} = \mathrm{X}^{32}$$
$$\Longrightarrow \mathrm{X}^{32} \times \mathrm{X}^{32} = \mathrm{X}^{64}$$

در نتیجه حداقل به 5 بار عمل ضرب نیاز داریم.

۶.

الف)

تعداد دفعات اجرای حلقه داخلی ثابت یعنی 51 بار می‌باشد، پیچیدگی تکه کد را محاسبه می‌کنیم:

$$T(n) = \sum_{i=0}^{n-1} \sum_{j=0}^{50} 1 = \sum_{i=0}^{n-1} 51 = 51n$$

ب)

- اگر i=n باشد، آنگاه عمل اصلی 1 بار انجام می‌شود.
- اگر i=$\frac{n}{2}$ باشد، آنگاه عمل اصلی 2 بار انجام می‌شود.
- اگر i=$\frac{n}{4}$ باشد، آنگاه عمل اصلی 3 بار انجام می‌شود.
- ⋮

در نتیجه به ازاء هر i، عمل اصلی به اندازه $\lfloor \log_2 i \rfloor + 1$ بار انجام می‌شود.

حال با فرض توان 2 بودن n، پیچیدگی تکه کد را محاسبه می‌کنیم:

$$T(n) = 1 + 2 + 3 + \cdots + \lfloor log_2 n \rfloor + 1 = \frac{(\lfloor log_2 n \rfloor + 1) \times (\lfloor log_2 n \rfloor + 2)}{2}$$

* برای محاسبه‌ی تقریبی و سریع می‌توان گفت چون دو حلقه از یک دیگر مستقل عمل می‌کنند و هر یک دارای پیچیدگی $log_2 n$ هستند بنابراین پیچیدگی آن‌ها در هم ضرب می‌شود و داریم: $T(n) = log_2 n \times log_2 n$

ج)

مقادیر متمایزی که i می‌تواند داشته باشد: $2, 4, 16, 256, \ldots = 2^{2^0}, 2^{2^1}, 2^{2^2}, 2^{2^3}, \ldots$

اگر k تعداد دفعات اجرای حلقه باشد، پیچیدگی تکه کد را محاسبه می‌کنیم:

$$2^{2^k} \le n \Rightarrow 2^k \le log_2 n \Rightarrow k \le log_2(log_2 n) \Rightarrow T(n) = log_2(log_2 n)$$

د)

- اگر i=1 باشد، آنگاه عمل اصلی 1 بار انجام می‌شود.
- اگر i=2 باشد، آنگاه عمل اصلی 2 بار انجام می‌شود.
- اگر i=3 باشد، آنگاه عمل اصلی 2 بار انجام می‌شود.
- اگر i=4 باشد، آنگاه عمل اصلی 3 بار انجام می‌شود.
- $\vdots$

در نتیجه به ازاء هر i، عمل اصلی به اندازه $\lfloor \log_2 i \rfloor + 1$ بار انجام می‌شود.

حال با فرض توان 2 بودن n، پیچیدگی تکه کد را محاسبه می‌کنیم:

$$T(n) = \sum_{i=1}^{n} \lfloor log_2 i \rfloor + 1 = \sum_{i=1}^{n} \lfloor log_2 i \rfloor + \sum_{i=1}^{n} 1 = log_2 1 + log_2 2 + \cdots + log_2 n + n = log_2(n!) + n$$

موفق باشید.