پاسخ تمرین شماره ۱

دانشکده مهندسی برق و کامپیوتر

پیچیدگی زمانی
و روابط بازگشتی

ساختمان های داده و الگوریتم - پاییز 1400

استاد: دکتر هشام فیلی

طراح تمرین : **علی زارع**

---

۱)

الف) مثال نقض:

$$f(n) = n^3 = O(n^3)$$

$$g(n) = n^2 = O(n^3)$$

$$\frac{n^3}{n^2} = n \neq O(1)$$

ب) اثبات:

چون $f(n) = O(g(n))$ در نتیجه یک c و n0 وجود دارند که اگر $n > n_0$ باشد خواهیم داشت $f(n) \leq cg(n)$. در نتیجه داریم:

$$g(n) \geq \frac{1}{c}f(n) \rightarrow g(n) = \Omega(f(n))$$

ج) مثال نقض:

$$f(n) = n^2$$

$$g(n) = n^3$$

$$log(n^2) = \Theta(log(n^3))$$

$$n^2 \neq \Theta(n^3)$$

۲)

الف)

متغیر حلقه هربار ۲ برابر می‌شود در نتیجه logn بار اجرا می‌شود. $O(logn)$

ب)

حلقه بیرونی n بار اجرا می‌شود و حلقه درونی $log_4(i^2)$ بار اجرا می‌شود. در نتیجه:

(یادآوری کنیم که توان پایه و ورودی لگاریتم تبدیل به صورت و مخرج ضریب آن می شوند. همچنین جمع تعدادی لگاریتم ها در پایه یکسان تبدیل به لگاریتم ضرب آن‌ها می شود)

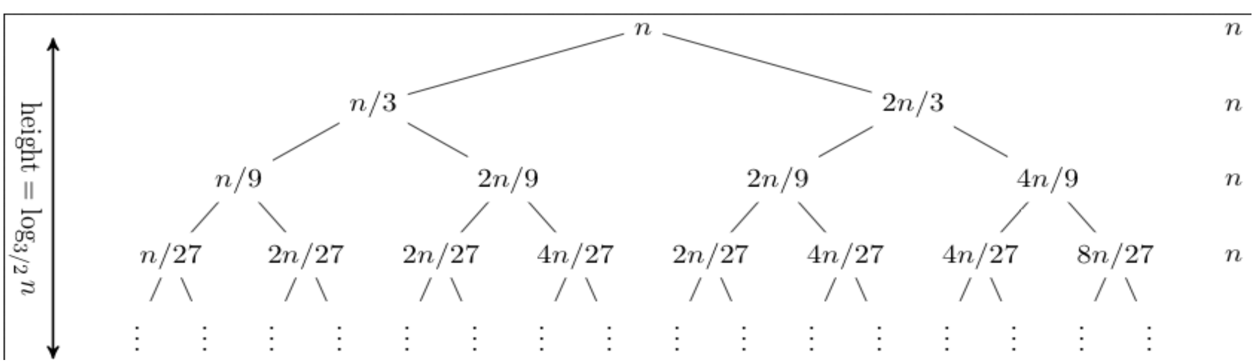
$$\sum_{i=0}^{n} \frac{2}{2} * log_2(i) = log(n!) = O(nlogn)$$

ج)

حلقه درونی $\frac{n^2}{2}$بار اجرا می‌شود. همچنین متغیر حلقه بیرونی به صورت نمایی درون حلقه دوم رشد می‌کند، در نتیجه تنها یک بار اجرا می‌شود. $O(n^2)$

۳)

الف) از روش درخت تکرار استفاده می‌کنیم:



ولی درخت متوازن نیست و سمت راست آن عمیق‌تر است. در نتیجه طولانی‌ترین مسیر را در نظر می‌گیریم که طول آن $log_{\frac{3}{2}}n$ است. پس پیچیدگی زمانی این رابطه برابر $O(nlogn)$است.

ب) حل با استفاده از قضیه اصلی: در این سوال $f(n) \ = \ O(n^{log_b^a}) = O(n^{log_7^{49}})$ است، پس با توجه به قضیه اصلی جواب نهایی ما برابر است با: $T(n) \ = \ \Theta(n^2)$

ج) حل با استفاده از قضیه اصلی: در این سوال $f(n) \ = \ \Theta(n^{log_b^a}) = \Theta(n^{log_3^9})$ است، پس با توجه به قضیه اصلی جواب نهایی ما برابر است با: $T(n) \ = \ \Theta(n^2 log(n))$

۴)

برای حل این سوال می‌توان براساس دست دادن نفر ۱ با هر نفر، میز را به دو قسمت تقسیم کرد و به صورت بازگشتی تعداد حالت‌های ممکن دو طرف را حساب و در هم ضرب کرد.

بدیهی است که اگر در یک سمت تعداد فردی از افراد بماند ممکن نیست جوابی پیدا شود و همچنین اگر صفر نفر بماند تنها یک حالت وجود دارد. (حالت‌های پایه)

```
int handshake(n) {
    if (n % 2 == 1)
        return 0;
    if (n == 0)
        return 1;

    int ans = 0;
    for (int i = 2; i < n + 1; i += 2)
        ans += handshake(i-2) * handshake(n-i);
    return ans;
}
```

چون می‌دانیم تنها nهای زوج را نیاز داریم در نتیجه متغیر حلقه را هر بار ۲ قدم جلو می‌بریم.

با یکبار صدا شدن تابع با پارامتر n، در حلقه n بار تابع به شکل بازگشتی صدا خواهد شد. (حلقه $n/2$بار اجرا می‌شود و در هر اجرا ۲ بار تابع صدا می‌شود.)

به طور دقیق‌تر به شکل زیر خواهد بود:

$f(n) = 2 * (f(0) + f(2) + ... + f(n - 2))$

همین رابطه را برای $n - 2$ نیز می‌نویسیم:

$f(n - 2) = 2 * (f(0) + f(2) + ... + f(n - 4))$

اگر این دو رابطه‌را از هم کم کنیم داریم:

$f(n) - f(n - 2) = 2 * f(n - 2)$

$\rightarrow f(n) = 3 * f(n - 2)$

$= 3 * (3 * f(n - 4)) = 3^2 * (3 * f(n - 6) = ... = 3^{n/2} * f(0)$

در نتیجه پیچیدگی زمانی این تابع $O(3^n)$است.

۵)

$$\sum_{i=0}^{\frac{n}{2}} n - 2i = (n - 0) + (n - 2) + (n - 4) + ... + (n - n)$$

$$= \frac{n}{2} \times n - (2 + 4 + ... + n) = \frac{n^2}{2} - \frac{n}{2}(\frac{n}{2} + 1) = O(n^2)$$

$$n^n < 2^n < \sum_{i=0}^{\frac{n}{2}} n - 2i < log(n!) < \sqrt[10]{n} < log(n)$$

۶)

الف)

$T(n) = 4T(n/3) + O(nlogn)$

با استفاده از قضیه اصلی خواهیم داشت $\Theta(n^{log_3(4)})$

ب)

$T(n) = T(n - 2) + O(n^2)$

اگر رابطه را چند باز کنیم می‌فهمیم که داریم:

$$T(n) = n^2 + (n-2)^2 + (n-4)^2 + \dots$$

$$\rightarrow \sum_{i=0}^{i=n/2} (n-2i)^2 = \sum_{i=0}^{i=n/2} n^2 + 4i^2 - 4ni$$

$$= \frac{n^3}{2} + 4\left(\frac{\frac{n}{2}\left(\frac{n}{2}+2\right)\left(\frac{n}{2}+1\right)}{6}\right) - 4n * \frac{n}{2}\left(\frac{\frac{n}{2}+1}{2}\right)$$

می‌بینیم که بزرگترین درجه n از ۳ است در نتیجه پیچیدگی عبارت $\Theta(n^3)$است.