باسمه تعالی
دانشگاه تهران - دانشکده مهندسی برق و کامپیوتر

ساختمان‌داده‌ها و الگوریتم‌ها

تمرین اول - پیچیدگی و الگوریتم های بازگشتی

حامد میرامیرخانی، ماردین نیچی

تاریخ تحویل: ۱۴۰۲/۸/۵

۱. ۲۰ نمره

پیچیدگی زمانی قطعه کدهای زیر را محاسبه کنید.

الف)

```
while (n > 0) {
    for (int j = 0; j < n; j++)
        System.out.println("*");
    n = n / 2;
}
```

ب)

```
sum=0;
for(i=1; i<n;i++)
    for( j = 1; j < i * i; j++ )
        if( j % i == 0 )
            for( k = 0; k < j; k++ )
                sum++;
```

ج)

```
for(i = n; i > 2; i = i * 1/5)
    for (int j = 1; j < n; j++)
        for (int k = 1; k < n; k++)
            j*=2;
```

پاسخ :

الف) درونی ترین حلقه در مرحله ی اول n بار، در مرحله ی دوم $\frac{n}{۲}$ بار و به همین ترتیب تا در نهایت ۱ بار اجرا میشود بعبارتی تعداد اجرای آن معادل است با:

$$\sum_{i=۰}^{log(n)} \frac{n}{۲^i} = ۲n - ۱$$

بنابراین پیچیدگی زمانی این قطعه کد معادل $O(n)$ خواهد بود.

ب) شرط if زمانی برقرار است که j مضربی از i باشد پس اگر j از ۱ تا $i^۲$ پیمایش شود درونی ترین حلقه بار اول i بار، بار دوم $۲i$ و به همین ترتیب تکرار می شود تا در نهایت به $i^۲ - i$ برسد. پس تعداد تکرار درونی ترین حلقه عبارت است از :

$$\sum_{i=۱}^{n-۱}[(i-۱)\sum_{k=۰}^{i^۲-۲}۱] = \sum_{i=۱}^{n-۱}(i-۱)(i^۲-۱) < \sum_{i=۱}^{n-۱} i^۳ = \frac{(n-۱)^۲ \times n^۲}{۴}$$

بنابراین پیچیدگی زمانی این قطعه کد معادل $O(n^۴)$ خواهد بود.

ج) درونی ترین حلقه n مرتبه اجرا میشود و با هر بار اجرا مقدار j دوبرابر میشود. بنابراین دفعه ی دومی که شرط $j < n$ چک میشود $۲^n - ۱ > n$ خواهد بود و برنامه خاتمه پیدا میکند. حلقه ی بیرونی $log(n)$ بار اجرا شود. بنابراین پیچیدگی زمانی این الگوریتم معادل $O(nlogn)$ خواهد بود.

۲. ۲۰ نمره

روابط زیر را رد یا اثبات کنید.

الف) $f(n) = O(g(n)) \Rightarrow ۲^{f(n)} = O(۲^{g(n)})$

ب) $f(n) \in O(g(n)) \Rightarrow g(n) \in \Omega(f(n))$

ج) $f(n) \neq O(g(n)) \Rightarrow f(n) = \Omega(g(n))$

پاسخ :

الف) نادرست

مثال نقض:

$$f(n) = ۲n, g(n) = n \Rightarrow ۲^{۲n} \neq O(۲^n)$$

ب) درست

اثبات:

$$f(n) = O(g(n)) : \exists c, n_۰; n > n_۰$$
$$\Rightarrow f(n) \leq c(g(n)) \Rightarrow g(n) \geq \frac{۱}{c} f(n) \Rightarrow g(n) = \Omega(f(n))$$

ج) نادرست

مثال نقض:
دو تابع زیر را تعریف میکنیم:

$$f(x) = \begin{cases} ۰, & \text{n} = ۲k+۱ \\ ۱, & \text{otherwise} \end{cases}$$

$$g(x) = \begin{cases} ۱, & \text{n} = ۲k+۱ \\ ۰, & \text{otherwise} \end{cases}$$

$$f(n) = O(g(n)) : \exists c, n. > ۰; \forall n > n. \Rightarrow f(n) \leq c(g(n))$$

فرض کنید $n = zn. + ۱$ بنابراین طبق تعریف $f(n) = ۰, g(n) = ۱$ که چنین c ای وجود ندارد که بتواند رابطه را برقرار سازد.
به صورت مشابه نشان میدهیم برای دو تابع مثال زده شده رابطه $f(n) = \Omega(g(n))$ نیز برقرار نخواهد بود.

۳. ۲۰ نمره

پیچیدگی روابط بازگشتی زیر را با استفاده از روش‌های ممکن به دست آورید.

الف) $T(n) = T(\sqrt{n}) + n$

ب) $T(n) = \sqrt{n}T(\sqrt{n}) + nlog(logn)$

ج) $T(n) = T(\frac{n}{۲}) + n(۵ - cos(n))$

پاسخ :

الف) با استفاده از روش تغییر متغیر سوال را حل می‌کنیم.

$$n = ۲^m \implies T(۲^m) = T(۲^{\frac{m}{۲}}) + ۲^m$$

$$S(m) = T(۲^m) = S(\frac{m}{۲}) + ۲^m$$

حال پیچیدگی $S(m)$ را با استفاده از قضیه اصلی به دست می‌آوریم:

$$a = ۱, b = ۲, f(m) = ۲^m, \log_۲ ۱ = ۰$$

$$\implies S(m) = \Theta(۲^m)$$

حال طبق تغییر متغیر اولیه، برای $T(n)$ خواهیم داشت:

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

ب)

$$\frac{T(n)}{n} = \frac{T(\sqrt{n})}{\sqrt{n}} + log(logn)$$

$$F(n) = \frac{T(n)}{n} \rightarrow F(n) = F(\sqrt{n}) + log(logn)$$

$$n = ۲^{۲^m} \rightarrow F(۲^{۲^m}) = F(۲^{۲^{m-۱}}) + m$$

$$G(m) = F(۲^{۲^m}) \to G(m) = G(m-۱) + m = G(m-۲) + G(m-۱) + m$$
$$= G(۱ + ۲ + ... + m) \to G(m) = O(m^۲)$$
$$G(m) = O(m^۲) = F(۲^{۲^m}) = F(n) = O(log(log(n))^۲)) = \frac{T(n)}{n}$$
$$T(n) = O(nlog(log(n))^۲))$$

ج) نمیتوانیم از قضیه اصلی استفاده کنیم چرا که شرط regularity نقض شده است. داریم:

$$|cos(n)| \geq ۱ \to -۱ \leq cos(n) \leq ۱ \to ۴ \leq ۵ - cos(n) \leq ۶$$

چون عبارت $۵ - cos(n)$ بین دو عدد ثابت قرار گرفته پس میتوان رابطه را بصورت زیر بازنویسی کرد:

$$T(n) = T(\frac{n}{۲}) + \Theta(n)$$

حالا میتوانیم از قضیه اصلی استفاده کنیم:

$$a = ۱, b = ۲ \to c = \log_۲ ۱ = ۰ \to T(n) = \Theta(n)$$

۴. ۱۰ نمره

توابع زیر را براساس پیچیدگی زمانی مرتب کنید.

الف) $log(n), n^n, \sum_{i=۱}^{\frac{n}{۲}} n - ۲i, log(n!), ۲^n, \sqrt[۱۰]{n}$

ب) $nlog(log(n)), n^{log(log(n))}, n^{\pi}, (log(n))!$

ج) $\sum_{i=۰}^{n} \frac{n^i}{i!}, n۲^n, n^{nloglogn}, logn!, n^۲, ۲^n$

پاسخ:

الف)

$$\sum_{i=۱}^{\frac{n}{۲}} n - ۲i = (n-۰) + (n-۲) + (n-۴) + ... + ۰ =$$
$$\frac{n}{۲} \times n - (۲ + ۴ + .. + n) = O(n^۲)$$
$$\Rightarrow n^n > ۲^n > \sum_{i=۱}^{\frac{n}{۲}} n - ۲i > log(n!) > \sqrt[۱۰]{n} > log(n)$$

ب) می‌توانیم از تغییر متغیر $n = ۲^m$ استفاده کنیم:

$$nlog(log(n)) \to ۲^m log(log(۲^m)) = ۲^m log(m)$$
$$n^{log(log(n))} \to (۲^m)^{log(log(۲^m))} = ۲^{mlog(m)} = (۲^{log(m)})^m \approx m^m$$
$$n^{\pi} \to ۲^{m\pi}$$
$$(log(n))! \to (log(۲^m))! = m!$$

$$\Rightarrow nlog(log(n)) < n^{\pi} < (log(n))! < n^{log(log(n))}$$

ج)

$$logn! < n^{۲} < ۲^{n} < n۲^{n} < \sum_{i=۰}^{n} \frac{n^{i}}{i!} < n^{nloglogn}$$

۵. ۲۰ نمره

پیچیدگی زمانی توابع بازگشتی زیر را محاسبه کنید.

الف)

```
func(n) {
    if n <= 1
        return
    for i in 1 to n*n:
        //O(1)
    func(n-2);
}
```

ب)

```
func(a, b):
    if(b == 0):
        return a
    return func(b, a % b)
```

پاسخ :

الف)

$$T(n) = T(n-۲) + O(n^{۲})$$

$$\Rightarrow T(n) = n^{۲} + (n-۲)^{۲} + (n-۴)^{۲} + ...$$

$$\sum_{i=۱}^{\frac{n}{۲}} (n-۲i)^{۲} = \sum_{i=۱}^{\frac{n}{۲}} n^{۲} + ۴i^{۲} - ۴ni = \frac{n^{۳}}{۲} + ۴(\frac{\frac{n}{۲} \times (\frac{n}{۲}+۱) \times (\frac{n}{۲}+۲)}{۶}) - ۴n \times \frac{n}{۲} \times (\frac{(\frac{n}{۲}+۱)}{۲})$$

بزرگترین درجه عبارت بدست آمده ۳ است پس پیچیدگی زمانی این قطعه کد $O(n^{۳})$ است.

ب)

در اجرای بعدی جای a و b عوض میشود و همواره ورودی اول از ورودی دوم بزرگتر میشود. https://www.overleaf.com/project/۶۵۲۷c۱۵۰۴۷۱۷۸۶c۶۹۰۰ec۴۸c

همینطور اگر $a \geq b$ باشد باقیمانده a بر b از $\frac{a}{۲}$ کوچکتر میشود:

$$b < \frac{a}{۲} \Rightarrow a \% b < b < \frac{a}{۲}$$

$$b = \frac{a}{۲} \Rightarrow a \,\%\, b = ۰ < \frac{a}{۲}$$
$$b > \frac{a}{۲} \Rightarrow a \,\%\, b \leq a - b < \frac{a}{۲}$$

بعد از دو گام متوالی اجرا داریم: $a, b \to a \,\%\, b, b \,\%\, (a \,\%\, b)$ بعبارتی هر دو ورودی حداقل نصف میشوند. پس خواهیم داشت:

$$T(n) = ۲ \times min(\log a, \log b) + ۱ = O(min(\log a, \log b))$$

۶. ۱۰ نمره

فرض کنید یک دنباله ی n تایی شامل $a_۱, a_۲, ..., a_n$ از اعداد در اختیار دارید. الگوریتمی از مرتبه $O(n)$ طراحی کنید که بتواند حاصل عبارت مقابل را محاسبه کند.

$$\sum_{۱ \leq l \leq r \leq n} f(l, r)$$

که $f(l, r)$ را اینگونه تعریف می‌کنیم:

$$f(l, r) = \sum_{i=l}^{r} a_i$$

پاسخ :

محاسبه میکنیم که هر a_i در چند بازه قرار دارد.اگر درایه ی iام در بازه ای قرار داشته باشد آن بازه قبل از درایه ی i شروع شده و بعد از آن به پایان رسیده است. برای شروع بازه ی قبل از درایه ی iام i حالت و برای پایان یافتن بازه ی بعد از درایه ی i، $n - i + ۱$ حالت داریم.پس درایه ی iدر $i \times (n - i + ۱)$ بازه قرار دارد. پس میتوانیم مسئله را اینگونه بنویسیم:

$$\sum_{۱ \leq l \leq r \leq n} \sum_{i=l}^{r} a_i = \sum_{i=۱}^{n} i \times (n - i + ۱) \times a_i$$

تعداد تکرار عملیات معادل با $c \times n$ خواهد بود پس پیچیدگی زمانی از مرتبه ی $O(n)$ میشود.