به نام خدا

ساختمان داده‌ها و الگوریتم‌ها (۴۰۲۵۴)
دانشگاه صنعتی شریف
مدرس: مهدی صفرنژاد

تمرین اوّل
انتشار: ۱۱ مهر ۰۰

تفکر الگوریتمی

سوالات را با دقت بخوانید و روی همه آن هاوقت بگذارید. تمرین های تئوری تحویل گرفته نمی‌شوند اما از آن هاسوالات کوییز مشخص می‌شود. بنابراین روی سوالات به خوبی فکر کنید و در کلاس های حل تمرین مربوطه شرکت کنید.

---

۱ تمرینات تئوری

سؤال ۱. با مثال نقض، نادرستی گزاره‌های زیر را نشان دهید.

آ. توابع $f : \mathbb{N} \to \mathbb{N}$ و $g : \mathbb{N} \to \mathbb{N}$ وجود ندارند که $f(n) + g(n) \neq \mathcal{O}(f(n))$ و $f(n) + g(n) \neq \mathcal{O}(g(n))$ باشد.

ب. به ازای هر تابع $f : \mathbb{N} \to \mathbb{N}$، آنگاه $f(n) = \Theta(f(\frac{n}{۲}))$.

پ. به ازای توابع $f_i : \mathbb{N} \to \mathbb{N}$، که در آن $i \in \{۱, ۲, \ldots, n\}$ باشد، آنگاه $\sum_{i=۱}^{n} f_i(n) = \mathcal{O}\left(\max_i\{f_i(n)\}\right)$.

پاسخ: آ. گزاره نادرست است، زیرا ابتدا فرض کنید $f(n) = \left\{ \begin{array}{ll} ۱, & n = ۲k + ۱ \\ n, & n = ۲k \end{array} \right\}$ و $g(n) = \left\{ \begin{array}{ll} n, & n = ۲k + ۱ \\ ۱, & n = ۲k \end{array} \right\}$ باشد. هم‌چنین فرض کنید

$$h(n) = f(n) + g(n)$$

در نتیجه

$$h(n) = n + ۱$$

باشد، واضح است که

$$h(n) \neq \mathcal{O}(f(n)) \wedge h(n) \neq \mathcal{O}(g(n)).$$

در واقع به ازای هر $n.$ و c، $n \geq n.$ وجود دارد که $h(n) > cf(n)$ و $h(n) > cg(n)$.

ب. گزاره نادرست است، زیرا فرض کنید $f(n) = ۲^n$ باشد آنگاه $f(\frac{n}{۲}) = \sqrt{۲}^n$ خواهد بود که بیانگر این می‌باشد که $f(n) \neq \Theta(f(\frac{n}{۲}))$.

پ. گزاره نادرست است، زیرا فرض کنید

$$\forall i \quad f_i(n) = ۱$$

باشد، واضح است که

$$\sum_{i=۱}^{n} f_i(n) = n \neq \mathcal{O}(۱).$$

سؤال ۲. پیچیدگی زمانی قطعه کد های زیر را بیابید.

آ.

```
for (int i = 0; i < n; i++) {
    for (int j = n; j > i; j--) {
        //O(1)
    }
}
```

ب.

```
for (int i = n/2; i <= n; i++) {
    for (int j = 2; j <= n; j *= 2) {
        //O(1)
    }
}
```

پ.

```
int j = 1;
int i = 0;
while (i < n) {
    //O(1)
    i += j;
    j++;
}
```

ت.

```
for (int i = 1; i <= n; i++) {
    for (int j = 1; j < n; j+=i) {
      //O(1)
    }
}
```

ث.

```
for (int i = n; i > 0; i/=2) {
    for (int j = 0; j < i; j++) {
        //O(1)
    }
}
```

پاسخ:

آ. پاسخ :

$$O(n^۲)$$

راه حل: $n + n - ۱ + \ldots + ۲ + ۱ = \frac{n(n-۱)}{۲} = O(n^۲)$

ب.

پاسخ: $O(n\log n)$

راه حل:

حلقه اول $n/۲$ بار و حلقه درونی $\log n$ بار اجرا می شود

پ.

پاسخ:$O(\sqrt{n})$

راه حل: i در هر مرحله جمع اعداد ۱ تا j است اگر j تا k پیش برود (برنامه k بار اجرا می شود)

$$۱ + ۲ + \ldots + k - ۱ + k = \frac{k\left(k+۱\right)}{۲}$$

و حلقه زمانی پایان میابد که :

$$\frac{k\left(k+۱\right)}{۲} > n$$

در نتیجه

$$k = O\left(\sqrt{n}\right).$$

ت.

پاسخ: $O(n\log n)$

راه حل: حلقه در مرحله اول n بار، در مرحله دوم n/۲ بار و ... اجرا می شود

پس داریم:

$$O\left(s\right) = O\left(n + \frac{n}{۲} + \frac{n}{۳} + \frac{n}{۴} + \ldots + \frac{n}{n}\right) = O(n\left(۱ + \frac{۱}{۲} + \frac{۱}{۳} + \ldots + \frac{۱}{n}\right) = O(n\left(\sum_{i=۱}^{n}\frac{۱}{i}\right))$$

حال $O\left(\sum_{i=۱}^{n}\frac{۱}{i}\right)$ را حساب می کنیم:

$$\sum_{i=۱}^{n}\frac{۱}{i} \leq \int_{۱}^{n}\frac{۱}{x}\,dx = \ln\left(n\right)$$

$$\sum_{x=۱}^{n}\frac{۱}{x+۱} \leq \int_{۱}^{n}\frac{۱}{x}\,dx = \ln n \leq \sum_{x=۱}^{n}\frac{۱}{x}$$

$$O\left(\frac{۱}{۱}+\frac{۱}{۲}+\ldots+\frac{۱}{n}\right)=O(\log n).$$

ث.

پاسخ : $O\left(n\right)$

راه حل:

$$O\left(n\left(۱+\frac{۱}{۲}+\frac{۱}{۴}+\ldots\right)\right)=O\left(۲n\right)=O\left(n\right).$$

سؤال ۳. با استفاده از روش های دلخواه پیچیدگی زمانی رابطه های بازگشتی زیر را پیدا کنید.

آ. $T\left(n\right) = ۲T\left(\lfloor\sqrt{n}\rfloor\right) + \log(n)$

ب. $T\left(n\right) = T\left(\frac{n}{۳}\right) + T\left(\frac{۲n}{۳}\right) + O(n)$

پ. $T\left(n\right) = \left(\frac{۲}{n}\right)\left(T\left(۰\right) + \ldots + T\left(n - ۱\right)\right) + c$
$T(۰) = ۰$

پاسخ:

آ. از تغییر متغیر $m = \log\left(n\right)$ استفاده می کنیم:

$$T\left(۲^m\right) = ۲T\left(۲^{\frac{m}{۲}}\right) + m.$$

سپس قرار میدهیم $T\left(۲^m\right) = S\left(m\right)$

در نتیجه:

$$S(m) = ۲S(\frac{m}{۲}) + m$$

طبق قضیه اصلی:

$$S\left(m\right) = T\left(۲^m\right) = m\log\left(m\right)$$

$$T\left(n\right) = \log\left(n\right)\log\left(\log\left(n\right)\right)$$

ب. اگر درخت مربوط به $T(n)$ را رسم کنیم، بلندترین شاخه دارای ارتفاع $log\frac{n}{۲}$ و کوتاه ترین شاخه دارای ارتفاع $\log\frac{n}{۳}$ است.
در نتیجه:

$$T\left(n\right) = O\left(n\log_{\frac{۳}{۲}} n\right) = O\left(\log_{\frac{۳}{۲}} ۳ \times n\log_{۳} n\right) = O\left(n\log\left(n\right)\right)$$

$$T\left(n\right) = \Omega\left(n\log_{۳} n\right) = \Omega\left(n\log\left(n\right)\right)$$

در نتیجه:

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

پ. با توجه به رابطه‌ی داده شده داریم:

$$nT(n) = ۲\left(T(۰) + T(۱) + \ldots + T(n-۲) + T(n-۱)\right) + cn$$
$$(n-۱)(T(n-۱) = ۲(T(۰) + T(۱) + \ldots + T(n-۲)) + c(n-۱)$$

با تفاضل این دو رابطه از هم نتیجه می شود:

$$n(T(n)) - (n-۱)T(n-۱) = ۲T(n-۱) + c$$
$$nT(n) = (n+۱)T(n-۱) + c$$
$$\frac{T(n)}{n+۱} = \frac{T(n-۱)}{n} + \frac{c}{n(n+۱)}$$
$$\frac{T(n)}{n+۱} = \frac{T(n-۱)}{n} + \frac{c}{n(n+۱)}$$
$$\ldots\ \ldots\ \ldots\ \ldots$$
$$\frac{T(۲)}{۳} = \frac{T(۱)}{۲} + \frac{c}{۲/۳}$$
$$\frac{T(۱)}{۲} = \frac{T(۰)}{۱} + \frac{c}{۱/۲}$$
$$\frac{T(n)}{n+۱} = \frac{c}{۱\times۲} + \frac{c}{۲\times۳} + \ldots + \frac{c}{(n-۱)n} + \frac{c}{n(n+۱)} = c.\frac{n}{n+۱}$$
$$T(n) = cn = O(n)$$

سؤال ۴. رابطه ی بازگشتی مربوط به تکه کد زیر را پیدا کنید و سپس با روشی دلخواه پیچیدگی زمانی آن را بدست آورید.

```
int gcd(int a, int b)
{
    if(a == b)
                return a;
    if(a > b)
                gcd(a % b, b);
    else
                gcd(a, b % a);
}
```

پاسخ:

در هر مرحله تابع gcd یکی ازآرگمان ها را نصف می کند(حداکثر). به طور مثال:

اگر $ba/۲$ باشد در مرحله بعد داریم $a' = b$ و $b' < a/۲$ چرا که % به اندازه b یا بیشتر از آن از a کم می کند

اگر $b < a/۲$ باشد در مرحله بعد داریم $a' = b$ و $b' < a/۲$ چرا که % حداکثر $b - ۱$ را برمیگرداند

در نتیجه در هر مرحلهي بازگشتی تابع gcd حداکثر یکی از عبارت ها را نصف می‌کند. در نتیجه پیچیدگی زمانی این قطعه کد برابر با $O\left(\log\left(n\right)\right)$ که در این جا n ماکسیمم a یا b است.

اگر $g\left(x,y\right) \leq T\left(n\right)$ در صورتی که $n = xyd = \frac{x}{x \ mod \ y}$ باشد:

$$T\left(n\right) = T\left(\tfrac{n}{d}\right) + c$$
$$T\left(n\right) = T\left(\tfrac{n}{d^۲}\right) + ۲c$$

$$T\left(n\right) = T\left(\frac{n}{d^k}\right) + kc$$

$$\text{When } \frac{n}{d^k} = ۱ \implies n = d^k \implies k = \log_d^n$$

$$T\left(n\right) = T\left(۱\right) + c\log_d^n$$

$$T\left(xy\right) = ۱ + c\log_d^{xy}$$

$$\implies g\left(x,y\right) \leq T\left(xy\right)$$

$$\implies g(x,y) \,\epsilon\, O(\log \ d(xy)).$$

سؤال ۵. می‌خواهیم در رشته ای از حروف کوچک انگلیسی، بلندترین زیر رشته‌ای که در آن هیچ حرفی دو بار تکرار نشده است را پیدا کنیم. شبه‌کدی بنویسید که این کار را در مرتبه $O(n)$ انجام دهد.(برای راحتی می‌توانید فرض کنید که رشته به صورت آرایه‌ای از اعداد ۰ تا ۲۵ به شما داده شده است)

پاسخ:

Algorithm 1 finding the length of longest substring with unique characters

```
procedure LONGESTUNIQUESUBSTTR(str[])
    result ← 0, i ← 0

    lastIndex ← int[26]

    while i < 26 do

        lastIndex[i] ← −1

        i+ = 1

    end while
    i ← 0, j ← 0
    while i < str.length do
        j ← j > lastIndex[str[i]] + 1?j : lastIndex[str[j]] + 1
        result ← result > i − j + 1?result : i − j + 1
        lastIndex[str[i]] ← i
        i+ = 1
    end while
    return result
end procedure
```

سؤال ۶. آرایه $w[۱..n]$ داده شده است که حاوی n عدد طبیعی است. $idx[i]$ را این‌گونه تعریف می‌کنیم:

$$idx[i] = \max\{j : ۱ \leq j < i \;\; \wedge \;\; w[j] \leq w[i]\}.$$

الگوریتم زیر را برای پیدا کردن idx اجرا خواهیم کرد:

Algorithm 2 Checking correctness and termination

```
procedure ALG(w[1..n], idx[1..n])
    idx[1] = −1
    for i=2 to n do
        j = i − 1
        while w[j] > w[i] ∧ j ≠ −1 do
            j = idx[j]
        end while
        idx[i] = j
    end for
end procedure
```

آ. نشان دهید الگوریتم بالا خاتمه‌پذیر است.

ب. نشان دهید الگوریتم پاسخ صحیح را تولید می‌کند.

پ. مرتبه زمانی شبه‌کد را پیدا کنید. (راهنمایی: تعداد دفعات رجوع به هر خانه آرایه محاسبه نمایید.)

پاسخ:

آ. الگوریتم خاتمه‌پذیر است، زیرا در هر بار اجرای حلقه داخلی، مقدار شمارنده j مقداری نزولی نسبت به حالت قبلی خواهد داشت و در نتیجه حلقه داخلی با شمارنده j در ابتدا از $i - ۱$ شروع خواهد شد و در بدترین حالت هر بار یک واحد از آن کم خواهد شد تا به مقدار -۱ برسد و حلقه داخلی خاتمه یابد. از طرفی حلقه بیرونی نیز هر بار یک واحد افزایش می‌یابد تا به n برسد. در نتیجه الگوریتم حتماً خاتمه پذیر خواهد بود.

ب. هدف مسئله، این است که به ازای هر i، مقدار $idx[i]$ برابر با بزرگ‌ترین jای باشد که $j < i$ باشد و $w[j] \leq w[i]$ باشد. از طرفی مطابق الگوریتم، حلقه داخلی نیز ابتدا از $i - ۱$ شروع کرده و هر بار مقداری کاهشی نسبت به مقدار قبلی خواهد داشت. با توجه به شرط مسئله، اولین j که شرط مسئله را ارضا کند به عنوان $idx[i]$ شناخته خواهد شد که درست است. زیرا مقدار آن بیشینه است و همچنین شرط $w[j] \leq w[i]$ نیز برقرار است. پس الگوریتم به درستی کار می‌کند.

پ. برای یافتن مرتبه زمانی شبه‌کد این‌گونه عمل خواهیم کرد، با به توجه حلقه‌ها، الگوریتم به هر خانه آرایه idx حداکثر دو بار مراجعه خواهد کرد، بار اول برای یافتن j مورد نظر و بار دوم نیز هنگامی خواهد بود که در گام‌های بعدی حلقه ممکن است به آن خانه آرایه idx مراجعه کند. در نظر داشته باشید هنگامی که در اجرای kام که $i < k$ است، به آرایه $idx[i]$ مراجعه

صورت گیرد،در اجراهای بعدی یعنی اجرای $k < t$ به $idx[i]$ مراجعه‌ای صورت نخواهد گرفت و ممکن است به خانه‌ای قبل از $idx[i]$ مراجعه صورت گیرد، زیرا در درون حلقه، مقدار شمارنده $j = idx[k]$ خواهد شد. پس در نتیجه به هر خانه آرایه idx به تعداد بار ثابتی مراجعه می‌شود، از طرفی، طول آرایه idx برابر با n است، پس مرتبه شبه کد بالا $\mathcal{O}(n)$ است.

موفق باشید