جواب تکلیف سری اول

طراحی الگوریتم
دانشکده ریاضی. دانشگاه صنعتی خواجه نصیرالدین طوسی. پاییز ۱۴۰۲

۱. در قطعه کد زیر، خطهای سوم و پنجم هر کدام چند بار اجرا می‌شوند؟

```
0. j = 1
1. for i in range(1,n):
2.   while (j < n) :
3.     j = j * 2
4.     for k in range(1,j):
5.       count = count + 1
```

بستگی به مقدار اولیه j دارد. با فرض اینکه مقدار اولیه j برابر با 1 باشد، خط سوم به تعداد $\lceil \log_2 n \rceil$ بار در کل اجرا می‌شود. مقدار j که به n رسید، دیگر حلقه while اجرا نمی‌شود. خط پنجم به تعداد

$$2 + 4 + 8 + \ldots + 2^r$$

اجرا می‌شود. اینجا $r = \lceil \log_2 n \rceil$.

۲. برای هر یک از دنباله‌های زیر تعداد مقایسه‌ای که الگوریتم مرتب سازی درجی، حبابی و ادغامی انجام می‌دهند را با استفاده از نماد $\Theta()$ بنویسید. دقت کنید هر دنباله جایگشتی از $1, 2, \ldots, n$ است.

(آ) $2, 3, 4, \cdots, n, 1$

- حبابی $\Theta(n^2)$
- درجی $\Theta(n)$
- ادغامی $\Theta(n \log n)$

(ب) $n, 1, 2, \cdots, n-1$

- حبابی $\Theta(n^2)$
- درجی $\Theta(n^2)$
- ادغامی $\Theta(n \log n)$

(ج) $1, 2, 3, \cdots, k, n, n-1, n-2, \cdots, k+1$

- حبابی $\Theta(n(n-k+1))$
- درجی $\Theta(k + (n-k)^2)$
- ادغامی $\Theta(n \log n)$

(د) $k, k-1, \cdots, 3, 2, 1, k+1, k+2, \cdots, n$

- حبابی $\Theta(k(n-1))$

- درجی $\Theta(k^2+n)$
- ادغامی $\Theta(n\log n)$

۳. توابع زیر را بر اساس کران مجانبی $\Theta()$ مرتب کنید.

$$f_1(n)=n^{1.5} \qquad f_2(n)=2n+\log n+3 \qquad f_3(n)=n^{\log n}$$

$$f_4(n)=2^n \qquad f_5(n)=\log(n!) \qquad f_6(n)=(\log n)!$$

- $2n+3=\Theta(n)$
- $\log(n!)=\Theta(n\log n)$
- $n^{2.5}=\Theta(n^{2.5})$
- $(\log n)!=\Theta(n^{\log\log n})$
- $n^{\log n}=\Theta(n^{\log n})=\Theta(2^{(\log^2 n)})$
- $2^n=\Theta(2^n)$

۴. برای هر کدام از حالات زیر یک تابع f مثال بزنید.

(آ) $f(n+1)\in O(f(n))$

$f(n)=n$

(ب) $f(n+1)\notin O(f(n))$

$f(n)=n!$

(ج) $f(2n)\notin O(f(n))$ ولی $f(n+1)\in O(f(n))$

$f(n)=2^n$

(د) $f(2^n)\in O(f(n))$

$f(n)=\log^* n$ $\quad$ $\log^* n$ برابر با تعداد دفعاتی است که تابع $\log_2$ روی n اعمال شود تا اینکه به کمتر یا مساوی 1 برسیم.

۵. فرض کنید $f(n)\in O(g(n))$. مشخص کنید گزاره‌های زیر درست هستند یا نادرست. مثال نقض بیاورید یا اثبات کنید.

(آ) $\log f(n)\in O(\log g(n))$

درست است. داریم

$$\exists c,n_0 \mid \forall n\geq n_0 \ \ f(n)\leq cg(n)$$

چون $\log$ یک تابع صعودی است پس:

$$\exists c,n_0 \mid \forall n\geq n_0 \ \ \log f(n)\leq \log(cg(n))=\log c+\log g(n)$$

می‌توان ثابتهای جدید c' و n'_0 را پیدا کرد بطوریکه

$$\forall n\geq n'_0, f(n)\leq c'g(n)$$

(ب) $2^{f(n)} \in O(2^{g(n)})$

نادرست است. مثال نقض $f(n) = 2n, g(n) = n$

(ج) $f^2(n) \in O(g^2(n))$

درست است. داریم

$$\exists c, n_0 \mid \forall n \geq n_0 \quad f(n) \leq cg(n)$$

پس:

$$\exists c, n_0 \mid \forall n \geq n_0 \quad f^2(n) \leq (cg(n))^2 = c^2 g^2(n)$$

می‌توان ثابتهای جدید $c' = c^2$ و $n'_0 = n_0$ را پیدا کرد بطوریکه

$$\forall n \geq n'_0, f(n) \leq c' g(n)$$

۶. الگوریتم اقلیدس برای پیدا کردن بزرگترین مقسوم علیه مشترک بین دو عدد صحیح a و b را در نظر بگیرید. توصیف این الگوریتم بصورت بازگشتی بصورت زیر است.

```
procedure gcd(a,b)   # a >= b
x := a
y := b
r := x mod y
if (r == 0)
   return b
else
  x := y
  y := r
  return gcd(x,y)
```

فرض کنید $n = \max\{a, b\}$. نشان دهید اگر $T(n)$ زمان اجرای الگوریتم اقلیدس (در بدترین حالت) باشد آنگاه $T(n) \in \Theta(\log n)$ است.

راهنمایی: برای کران پایین $T(n)$ ، اعداد فیبوناچی را در نظر بگیرید.

ابتدا نشان می‌دهیم $T(n) = O(\log n)$. نشان می‌دهیم بعد از هر دو گام متوالی الگوریتم اقلیدس اندازه عدد ورودی حداقل نصف می‌شود. از این نتیجه می‌شود: $T(n) \leq 2\log n$. برای اثبات این فرض کنید $a = \max\{a, b\}$. اگر $b \leq a/2$، بعد از یک تقسیم هر دو عدد مرحله بعدی حداکثر نصف a هستند. اگر $b > a/2$، آنگاه باقیمانده تقسیم a بر b از نصف a کمتر است. در نتیجه در مرحله بعدی یکی از اعداد (عدد کوچکتر) از نصف a کمتر است. بعد از دو تقسیم هر دو عدد کمتر از نصف a خواهند شد. مثال:

$$(1) \quad a = 13, b = 8$$

$$(2) \quad a = 8, b = 5$$

$$(3) \quad a = 5, b = 3$$

$$(4) \quad a = 3, b = 2$$

$$(5) \quad a = 2, b = 1 \qquad \text{finish}$$

برای اثبات $T(n) = \Omega(\log n)$ اعداد فیبوناچی را در نظر می‌گیریم. کافی است نشان دهیم F_n و F_{n+1} نسبت به هم اول هستند. این را با استقرا می‌توان نشان داد.

$$\checkmark \qquad F_1 = 1, F_2 = 1 \qquad \text{پایه استقرا}$$

$$(F_n, F_{n-1}) = 1 \qquad \text{فرض استقرا}$$
$$(F_{n+1}, F_n) = 1 \qquad \text{حکم استقرا}$$

فرض کنید F_{n+1} و F_n نسبت به هم اول نباشند و یک مقسوم علیه مشترک بزرگتر از 1 داشته باشند. پس

$$F_{n+1} = mk \qquad F_n = m'k$$

اما

$$F_{n-1} = F_{n+1} - F_n = (m - m')k$$

در نتیجه F_{n-1} هم مقسوم علیه مشترک k را دارد. پس F_n و F_{n-1} نسبت به هم اول نیستند و این متناقض با فرض استقرا است. پس حکم درست است.

از توضیحات بالا نتیجه می‌شود که برای اعداد ورودی F_n و F_{n+1} الگوریتم اقلیدس به تعداد n تقسیم انجام می‌دهد. چون $F_n \geq (1.5)^n$ پس تعداد تقسیمها با اندازه عدد ورودی نسبت لگاریتمی دارد.

۷. در الگوریتم مرتب سازی ادغامی نشان دهید اگر به جای تقسیم به دو قسمت تقریبا مساوی، لیست شامل n عدد را به دو قسمت با اندازه‌های $\lceil n/3 \rceil$ و $\lfloor 2n/3 \rfloor$ تقسیم کنیم زمان اجرای الگوریتم باز هم $\Theta(n \log n)$ خواهد بود.

اگر $T(n)$ زمان اجرای الگوریتم باشد داریم

$$T(n) \leq \begin{cases} T(\lfloor 2n/3 \rfloor) + T(\lceil n/3 \rceil) + O(n) & n > 3 \\ O(1) & n \leq 3 \end{cases}$$

با استفاده از تکنیک درخت بازگشت می‌توان نشان داد که $T(n) = \Theta(n \log n)$.

۸. در الگوریتم SELECT که در کلاس ارائه شد نشان دهید اگر اندازه گروه ها 7 باشد، زمان اجرای الگوریتم کماکان $\Theta(n)$ خواهد بود. ~~اما اگر اندازه هر گروه 3 باشد زمان اجرا از مرتبه $\Theta(n \log n)$ خواهد بود.~~

تصحیح: در واقع هنوز اثبات نشده است که برای گروه ۳ تایی زمان اجرا حتما بالاتر از $O(n)$ است. فقط می‌توانیم بگوییم که با تحلیل مشابه نمی‌توان اثبات کرد که زمان اجرا در این حالت لزوما خطی است. مقاله زیر را ببینید.

https://arxiv.org/pdf/1409.3600.pdf

برای حالت ۷ تایی داریم

$$T(n) \leq \begin{cases} T(\lceil n/7 \rceil) + T(\max\{|L|, |R|\}) + O(n) & n > b \\ O(1) & n \leq b \end{cases}$$

مشابه آنچه در برای حالت ۵ تایی گفته شد، اینجا می‌توان نشان داد که $\max\{|L|, |R|\} \leq \frac{5}{7}n + 8$. پس بدست می آید

$$T(n) \leq \begin{cases} T(\lceil n/7 \rceil) + T(\frac{5}{7}n + 8) + O(n) & n > b \\ O(1) & n \leq b \end{cases}$$

باز هم مشابه حالت ۵ تایی، با استفاده از حدس و استقرا می‌توان نشان داد که عدد ثابت c وجود که برای $n \geq b$ همواره $T(n) \leq cn$. این یعنی اینکه $T(n) = \Theta(n)$.

اما برای گروه های ۳ تایی، داریم

$$T(n) \leq \begin{cases} T(\lceil n/3 \rceil) + T(\max\{|L|, |R|\}) + O(n) & n > b \\ O(1) & n \leq b \end{cases}$$

اینجا با همان تحلیل بدست می‌آید $\max\{|L|, |R|\} \leq \frac{2}{3}n + 1$ و در نتیجه می‌توان گفت

$$T(n) \leq \begin{cases} T(\lceil n/3 \rceil) + T(\frac{2}{3}n + 1) + O(n) & n > b \\ O(1) & n \leq b \end{cases}$$

بعد از حل این رابطه بازگشتی، مانند مورد مرتب سازی ادغامی، فقط می‌توان گفت $T(n) = O(n \log n)$ لذا نمی توان گفت که زمان اجرا لزوما خطی است.

۹. فرض کنید دو X و Y دو لیست مرتب باشند که هر کدام شامل n عنصر هستند. چگونه می‌توان میانه اجتماع X و Y را در زمان $O(\log n)$ پیدا کرد؟

ایده کلی: کافی است هر بار میانه‌های دو لیست مرتب را مقایسه کنیم. فرض کنید m_x میانه X و m_y میانه لیست Y باشد.

- اگر $m_x = m_y$ آنگاه m_x جواب مسئله است.
- اگر $m_x > m_y$ آنگاه عناصر بزرگتر از m_x در لیست X نمی‌توانند جواب مسئله باشند. بطور مشابه عناصر کمتر از m_y در لیست Y نمی‌توانند جواب مسئله باشند. پس نصف عناصر از دایره جواب خارج می‌شوند.
- حالت $m_x < m_y$ مشابه حالت قبل است و نصف عناصر دو لیست را می‌توان کنار گذاشت.

برای عناصر باقیمانده، دوباره وسط دو لیست را مقایسه می‌کنیم و به همین طریق جلو می‌رویم. مشابه الگوریتم جستجوی باینری می‌توان استدلال کرد که تعداد مقایسه‌ها $O(\log n)$ است.