۹۷۲۴۴۳۰۸۱ امیر مسعود شاکر

تمرین ۱

Subject:
Year Month Date ( )

(1

a) $T(n) = 4\sqrt{n}\, T(\sqrt{n}) + 2n^2$

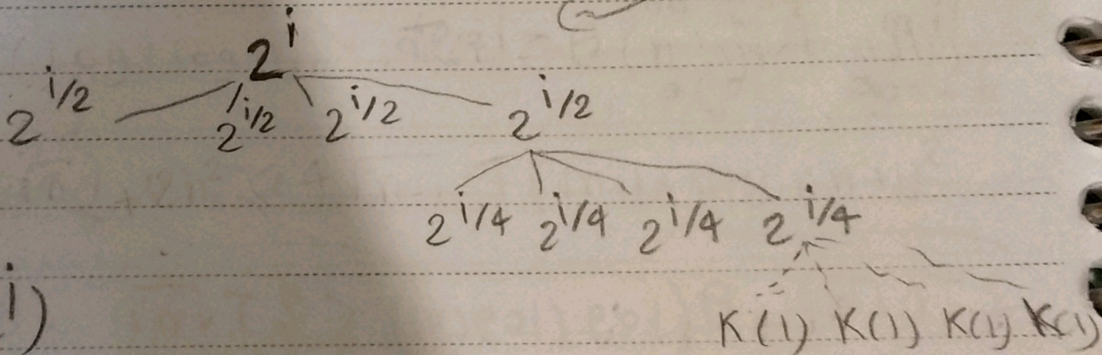
$n = 2^i \rightarrow i = \log n$

$T(n) = n\,G(n) = 4\sqrt{n} \times \sqrt{n}\, G(\sqrt{n}) + 2n^2$

$\rightarrow G(n) = 4\,G(\sqrt{n}) + 2n$

$\xrightarrow{n=2^i} G(2^i) = 4\,G(2^{i/2}) + 2^{i+1}$

$K(i) = G(2^i) \rightarrow K(i) = 4K(\frac{i}{2}) + 2^{i+1}$



$K(i) \in O(2^i)$

$\rightarrow G(2^i) \in O(2^i) \longrightarrow G(n) \in O(n)$

$\xrightarrow{T(n)=nG(n)} T(n) \in O(n^2)$

Subject:
Year. Month. Date. ( )

b) $T(n) = 2T(\sqrt{n}) + \Theta(\log(\log(n)))$ با smoothness

$n = 2^i \rightarrow i = \log n$

$T(2^i) = 2T(2^{i/2}) + \Theta(\log(i))$

$G(i) = T(2^i) \rightarrow G(i) = 2G(i/2) + \Theta(\log(i))$

برای استفاده از قضیه‌ی smoothness به منظور حل رابطه‌ی بازگشتی، نیاز داریم

ثابت کنیم $\log(n)$ یک تابع b-smooth است و $T(n)$ غیر کاهشی است.

اثبات b-smooth بودن $\log(n)$:

$$\lim_{n \to \infty} \frac{\log(bn)}{\log n} = \log b \checkmark$$

اثبات غیر کاهشی بودن $T(n)$:

$$\overbrace{2T(\sqrt{n}) + \Theta(\log(\log(n)))}^{T(n)} \leq \overbrace{2T\sqrt{n+1} + \Theta(\log(\log(n+1)))}^{T(n+1)} \checkmark$$

با اثبات‌های انجام شده در بالا، طبق قضیه smoothness می‌توان گفت که

به ازای هر n رابطه‌ی $T(n) \in O(\log n)$ برقرار است.

Subject:
Year. Month. Date. ( )

$b_1 = 4, d_1 = 1$

c) $T(n) - 3T(n-1) = 4^n(2n+1)$

$T(0) = 0, T(1) = 12$ با معادلات همگن و ناهمگن

$P(x) = (x-3)(x-4)^2$ $(r_1 = 3, m_1 = 1, r_2 = 4, m_2 = 2)$

$T(n) = c_1 3^n + c_2 4^n + c_3 n 4^n$

جایگذاری: $c_1 3^n + c_2 4^n + c_3 n 4^n - 3\left(\frac{c_1}{3} 3^n + \frac{c_2}{4} 4^n + \frac{c_3}{4}(n-1)4^n\right) = 4^n(2n+1)$

$\rightarrow \frac{1}{4} c_2 4^n + c_3 n 4^n - \frac{3}{4} c_3 n 4^n + \frac{3}{4} c_3 4^n - 4^n(2n+1) = 0$

جایگذاری مقادیر اولیه $\rightarrow \frac{c_2}{4} + \frac{3}{4} c_3 - 1 = 0 \Rightarrow c_2 + 3c_3 = 4$

$\rightarrow c_2 + 4c_3 - 12 = 12$ $\quad c_2 + 4c_3 = 24$

$\rightarrow c_1 = 56, c_2 = -56, c_3 = 20$

Subject:
Year. Month. Date. ( )

۲) کد سمت چپ:

$m = 0$

$f(n,m) \to f(n, m/2) \to f(n, \frac{m}{2^2}) \to \dots \to f(n, \frac{m}{2^i})$

با توجه به تقسیم شدن متوالی m بر ۲، می‌توان تابع بازگشتی را به صورت زیر نوشت:

$T(m) = T(\frac{m}{2}) +$ عدد ثابت

از master theorem داریم:

$T(m) = \overset{1}{a}\, T(\frac{m}{\underset{2}{b}}) + m^{\overset{0}{c}}$

$\to T(m) \in \theta(n^c \log m) \to \underline{T(m) \in \theta(\log m)}$

کد سمت راست:

۱) اگر $n \geq m$ باشد:

حالت اول: $m > \frac{n}{2} \to n\%m = n - m \to n\%m < n - \frac{n}{2}$

$\to n\%m < n/2$

حالت دوم: $m \leq \frac{n}{2} \to n\%m < m \to n\%m < \frac{n}{2}$

پس اگر $n \geq m$ باشد، در هر صورت رابطه $n\%m < \frac{n}{2}$ برقرار است.

Subject:
Year. Month. Date ( )

۲) اگر $n < m$ باشد:

شمارنده	مقدار داخل n	مقدار داخل m		
۱	n	m		
۲	m	n		
۳	n	$\frac{m}{2}$		
۴	$\frac{m}{2}$	$\frac{n}{2}$		
۵	$\frac{n}{2}$	$\frac{m}{4}$	$2i-1$	$\frac{m}{2^i}$
۶	$\frac{m}{4}$	$\frac{n}{4}$	$2i$	$\frac{n}{2^i}$
⋮	⋮	⋮		

$$n \% \frac{m}{2} = n - \frac{m}{2} < \frac{3n}{4}$$

$$n \% \frac{m}{2} < \frac{m}{2} < \frac{n}{2}$$

با فرض $\frac{m}{2^i} > 0$ $\Rightarrow \frac{m}{2^i} \geq 1 \Rightarrow m \geq 2^i \Rightarrow \log m \geq i$

$$\Rightarrow \underline{t(n) \in O(\log m)}$$

Subject:
Year. Month. Date. ( )

۳)

۳-۱)

```
def transfer(s1, s2, n):
    for i in range(n):
        t = s1[-1]
        s1.pop()
        s2.append(t)

def stack_reverse(s, n):
    s2 = []
    for i in range(n):
        x = s[-1]
        s.pop()
        transfer(s, s2, n-i-1)
        s.append(x)
        transfer(s2, s, n-i-1)
```

پیچیدگی زمانی برنامه $O(n^2)$ است.

۳-۳) با استفاده از روش استقرا سعی می کنیم اثبات را انجام دهیم:

حالت اول: $h = 0 \rightarrow n = 2^{h+1} - 1 = 2^{0+1} - 1 = 1$

$\varepsilon_h = n - h - 1 = 1 - 0 - 1 = 0$ ✓

فرض استقرا: حکم برای $h = k$ برقرار است. حال ثابت می کنیم برای $h = k+1$ نیز برقرار است:

$h = k \rightarrow n = 2^{h+1} - 1$

$\varepsilon_h = n - h - 1 = 2^{h+1} - 1 - k - 1$

در حالت $h = K+1$ ، نسبت به حالت $h = K$ ، یک ردیف بیشتر با تعداد 2^{K+1} برگ داریم. در نتیجه تعداد گره‌های غیر برگ برابر می‌شود با حالت $h = K$. اما ارتفاع ما در این حالت به اندازه یک واحد بیشتر است.

$$n' = 2^{K+1} - 1 + 2^{K+1} = 2^{K+2} - 1$$

$$\Sigma'_h = 2^{K+1} - 1 - K - 1 + 2^{K+1} - 1 = (\overbrace{2^{K+2} - 1}^{n}) - \overbrace{(K+1)}^{h} - 1$$

پس برای حالت $h = K+1$ هم حکم برقرار است. در نتیجه اثبات کامل است.

۳-۳)

$\lceil n/2 \rceil$

```
def find_min(heap, n):
    min_element = heap[n // 2]
    for i in range(1 + n//2, n):
        min_element = min(min_element,
                          heap[i])
    return min_element
```

پیچیدگی زمانی برنامه $\Theta(n)$ است.

۴-۳) در بهترین حالت با استفاده از recursion یا Function callstack، تعداد استک ها را به یک عدد کاهش می دهیم:

```
class queue:

    def __init__(self):
        self.s = []

    def enqueue(self, data):
        self.s.append(data)

    def dequeue(self):
        if len(self.s) <= 0:
            return
        x = self.s[len(self.s) - 1]
        self.s.pop()
        if len(self.s) <= 0:
            return x
        item = self.dequeue()
        self.s.append(x)
        return item
```

پیچیدگی زمانی عمل push، $O(1)$ است و پیچیدگی زمانی عمل پاپ برابر $O(n)$ است.