מטלה 1- מבני נתונים

יום שישי 01 מאי 2020   11:43

שאלה 1: לכל אחת מן הטענות הבאות, קבעו אם היא נכונה או לא נכונה, והוכיחו את תשובתכם.

- **טענה 1:** יהיו $f(n) = \left(\sqrt{n}\right)^{\sqrt{n}}, g(n) = \left(\sqrt{n}\right)!$, אזי:

$$\log(f(n)) = \Theta(\log(g(n)))$$

כלומר:

$$\log(f(n)) = O(\log(g(n)))$$

וגם

$$\log(f(n)) = \Omega(\log(g(n)))$$

כיוון 1: צ"ל: קיימים קבועים $c, n_0 > 0$ כך שלכל $n > n_0$ מתקיים:

$$\log((\sqrt{n})!) \le c \cdot \log\left((\sqrt{n})^{\sqrt{n}}\right)$$

מיידי ונכון עבור $c=1, n_0=1$ (למשל)

כי הרי:

$$\log\left(\sqrt{n}\cdot(\sqrt{n}-1)\cdot(\sqrt{n}-2)\cdots 2\cdot 1\right) < \log\left(\sqrt{n}\cdot\sqrt{n}\cdot\sqrt{n}\cdots\sqrt{n}\right)$$

ולכן: $\log(f(n)) = \Omega \log(g(n))$

כיוון 2: צ"ל קיימים קבועים $c, n_0 > 0$ כך שלכל $n > n_0$ מתקיים:

$$\log((\sqrt{n})!) \ge c \cdot \log\left((\sqrt{n})^{\sqrt{n}}\right)$$

וזה נכון עבור $c = n_0 = 4$ (למשל).

כי:

$$\log((\sqrt{n})!) = \log\left(\sqrt{n}(\sqrt{n}-1)\cdot(\sqrt{n}-2)\cdots\left\lceil\frac{\sqrt{n}}{2}\right\rceil\cdots 2\cdot 1\right)$$

$$\ge \log\left(\sqrt{n}\cdot(\sqrt{n}-1)\cdots\left\lceil\frac{\sqrt{n}}{2}\right\rceil\right)$$

$$\ge \log\left(\left\lceil\frac{\sqrt{n}}{2}\right\rceil\cdot\left\lceil\frac{\sqrt{n}}{2}\right\rceil\cdots\left\lceil\frac{\sqrt{n}}{2}\right\rceil\right)$$

$$= \log\left(\left\lceil\frac{\sqrt{n}}{2}\right\rceil^{\left\lceil\frac{\sqrt{n}}{2}\right\rceil}\right) \ge \log\left(\frac{\sqrt{n}}{2}^{\frac{\sqrt{n}}{2}}\right) = \frac{\sqrt{n}}{2}\log\left(\frac{\sqrt{n}}{2}\right)$$

$$= \frac{\sqrt{n}}{2}\left(\log\sqrt{n} - \log 2\right) = \frac{\sqrt{n}}{2}\left(\log\sqrt{n} - 1\right)$$

$$\ge \frac{\sqrt{n}}{2}\left(\log(\sqrt{n}) - \frac{1}{2}\log(\sqrt{n})\right) = \sqrt{n}\cdot\left(\frac{1}{2}\log(\sqrt{n}) - \frac{1}{4}\log(\sqrt{n})\right)$$

$$= \frac{1}{4}\sqrt{n}\log(\sqrt{n}) = \frac{1}{4}\log\left(\sqrt{n}^{\sqrt{n}}\right)$$

ולכן: $\log(f(n)) = O(\log(g(n)))$

ובשילוב שני הכיוונים קיבלנו וסיכום כי: $\log(f(n)) = \Theta(\log(g(n)))$

- **<u>טענה 2:</u>** תהי T פונקציה המוגדרת רקורסיבית באופן הבא:

$$T(n) = \left(T\left(\frac{n}{2}\right)\right)^2 \cdot 2^n, T(1) = 2$$

אזי:

$$T(n) = 2^{\Theta(n)}$$

<u>כלומר:</u>

$$T(n) = 2^{O(n)}$$

וגם

$$T(n) = 2^{\Omega(n)}$$

נפשט את הפונקציה T:

$$T(n) = T\left(\frac{n}{2}\right) \cdot T\left(\frac{n}{2}\right) \cdot 2^n$$

$$\cdot\ T\left(\frac{n}{2}\right) = T\left(\frac{n}{4}\right) \cdot T\left(\frac{n}{4}\right) \cdot 2^n$$

$$\cdot\ T\left(\frac{n}{4}\right) = T\left(\frac{n}{8}\right) \cdot T\left(\frac{n}{8}\right) \cdot 2^n \ \ldots$$

ובאופן כללי:

$$T(n) = \left(T\left(\frac{n}{2^k}\right)\right)^{2^k} \cdot 2^{n \cdot k}$$

נבדוק מהו k עבורו יעצר האלגוריתם:

$$\frac{n}{2^k} = 1 \Rightarrow n = 2^k \Rightarrow k = \log(n)$$

$$\left(T\left(\frac{n}{2^{\log n}}\right)\right)^{2\log n} \cdot 2^{n \log n}$$

$$= \left(T\left(\frac{n}{n}\right)\right)^{2\log n} \cdot 2^{n\log n} = 2^{2\log n} \cdot 2^{n \log n}$$

$$= 2^n\left(2^n + 2^{\log n}\right) = \left(2^n\right)^2 + 2^{\log n}$$

<u>צ"ל:</u> קיימים קבועים חיוביים c, כך שלכל n מתקיים:

$$2^{\Omega(n)} \geq c \cdot \left(2^{n^2} + 2^{\log n}\right)$$

כלומר

$$2^{\Omega(n)} \geq c \cdot \left(2^{n^2} + n\right)$$

ולא קיים c עבורו זה יהיה נכון כי אנחנו רואים שהמעריך של 2 בצד שמאל הוא $\Theta(n^2)$ ובצד ימין הוא $O(n)$ ולא משנה איזה קבוע נבחר, תמיד צד ימין יגדל יותר מהר ולא יתקיים השוויון.

• **טענה 3:** אם הפונקציות f ו- g הן פונקציות **מונוטוניות עולות** כך שמתקיים:

$$f(g(n)) = O(n), \qquad f(n) = \Omega(n)$$

אזי:

$$g(n) = O(n)$$

נתון: 1) $f(g(n)) = O(n)$ ולכן: $f(g(n)) \le C_1 \cdot n$

2) $f(n) = \Omega(n)$ ולכן: $f(n) \ge c_2 \cdot n$

לפי 1,2 מתקיים: $C_2 \cdot g(n) \le f(g(n)) \le C_1 \cdot n$

ולכן: $C_2 \cdot g(n) \le C_1 \cdot n$

נחלק ב- c_2 : $g(n) \le \frac{C_1}{C_2} \cdot n$

והרי $\frac{C_1}{C_2}$ הוא קבוע כלשהו לכן מתקיים: $g(n)=O(n)$.

שאלה 2:

נתונות הפונקציות הבאות:

2 $f_1(n) = 2019^{2020}$	7 $f_2(n) = 2^{3^n}$	1 $f_3(n) = \frac{1}{n^{2020}}$
4 $f_4(n) = 2^{\log_{\sqrt{2}} n}$	5 $f_5(n) = 4^n$	6 $f_6(n) = n^n$
3 $f_7(n) = \log_3(3^n n^2)$		

סדרו את הפונקציות לפי סדר אסימפטוטי ($O(\dots)$, מן ה"קטנה" ל"גדולה". אם מתקיים $f_i = \Theta(f_j)$ ציינו זאת, הוכיחו פורמלית את תשובותיכם.

$$f_3(n) = O(f_1(n)) = O(f_7(n)) = O(f_4(n)) = O(f_5(n)) = O(f_6(n)) = O(f_2(n))$$

הוכחה 1)

$$\frac{1}{n^{2020}} \le C \cdot 2019^{2020}$$

מתקיים עבור $C=1$, ולכל $n>0$

$$2019^{2020} \leq C \cdot \log_3(3^n n^2)$$

מתקיים עבור $C = 2019^{2020}$ וכל $n > 0$.

הוכחה ②

$$\log_3(3^n n^2) = \log_3(3^n) + \log(n^2)$$
$$= n \cdot \log_3(3) + 2 \cdot \log(n)$$
$$= n + 2\log(n) \leq C \cdot 2^{\log_{\sqrt{2}}(n)}$$
$$= C \cdot (\sqrt{2} \cdot \sqrt{2})^{\log_{\sqrt{2}}(n)}$$
$$= C \cdot \sqrt{2}^{\log_{\sqrt{2}}(n)} \cdot \sqrt{2}^{\log_{\sqrt{2}}(n)}$$
$$= C \cdot n \cdot n =$$
$$= C \cdot n^2$$

מתקיים לכל $n > n_0 = 3$, $C = 3$ (למשל). ולכן: $f_7(n) = O(f_4(n))$

הוכחה ③

$$2^{\log_{\sqrt{2}} n} = n^2 \leq C \cdot 4^n$$

מתקיים לכל $n > n_0 = 1$, $C = 1$ (למשל) ולכן: $f_4(n) = O(f_5(n))$

הוכחה ④

$$4^n \leq C \cdot n^n$$
$$4 \cdot 4 \cdot 4 \cdots 4 \leq C \cdot n \cdot n \cdot n \cdots n$$

מתקיים לכל $n > n_0 = 4$ ו- $C = 1$ (למשל) ולכן: $f_5(n) = O(f_6(n))$.

הוכחה ⑤

$$n^n \leq C \cdot 2^{3^n} \quad / \log(\,)$$
$$\log(n^n) = n\log n \leq c \cdot \log(2^{3^n}) = 3^n \log 2$$

מתקיים לכל $n > n_0 = 2$ ו- $C = 1$ (למשל).

הוכחה ⑥

שאלה 3: חשבו את זמני הריצה של הפונקציות הבאות:

```java
//A:
public static void mystery1(int n) {
int k = n;
    while (k > 2) {
        System.out.println(k);
        k = (int) (Math.pow(k, 1 / 3));
    }
}
```

נתאר את הנעה דרך המשוואה:

$$2^{3^x} = n \quad / \log(\,)$$
$$\log 2^{3^x} = \log n \Rightarrow 3^x \cdot O(1) = \log n \quad / \log(\,)$$
$$\Rightarrow \log 3^x = \log\log n \Rightarrow x \cdot O(1) = \log\log n$$

כאשר תיארנו באמצעות x את "הקפיצות" שהתוכנית עושה "נכנסת ומאה"

כאשר תיארנו בדוגמה x את "הקפיצות" שהתוכנית עושה "בגישה ימנית"
ולכן לסיכום גם נדע כי התוכנית רצה בסדר גודל של loglogn פעמים

// B:

```java
public static void dunno1(int n) {
        int i, j;
        for (i = 1; i <= n; i++) {
                for (j = 1; j <= i; j++) {
                        mystery1(n);
                }
        }
}
```

הלולאה הפנימית רצה עד הלולאה החיצונית באופן הבא: $1+2+3 \cdots + n$
ולפי נוסחת סדרה חשבונית זה שווה ל: $\frac{n(1+n)}{2}$
בכל "סיבוב" מתבצעות סדר גודל של $\log\log n$ (לפי A)
פעולות ולכן בסך הכל התוכנית תבצע סדר גודל של $\log\log n \cdot \frac{n(1+n)}{2} = O(n^2 \log\log n)$
פעולות.

//C:

```java
public static void mystery2(int i) {
        int x;
        x = 1;
        while (x < i) {
                System.out.println(x);
                x = x * 2;
        }
        while (x > 2) {
            x = (int) (Math.pow(x, 1 / 2));
            System.out.println(x);
        }
}
```

הלולאה הראשונה רצה בקפיצות $1,2,4,8,16 \ldots$ עד שx מגיע ל-i, כלומר הלולאה הראשונה
מבצעת סדר גודל של $\log(i)$ פעולות.
הלולאה השנייה רצה בדיוק אותו כמו תוכנית A, ולכן היא מבצעת סדר גודל של $\log\log(i)$ פעולות
$O(\log(i) + \log\log(i)) = O(\log i)$

במילים אחרות, הלולאה השנייה היא "זניחה" ביחס לראשונה.

// D:

```java
public static void dunno2(int n) {
        int i;
        for (i = n; i >= 1; i--) {
                mystery2(i);
        }
}
```

באיטרציה הראשונה של הלולאה מתבצעות $O(\log(n))$
פעולות, בשנייה: $O(\log(n-1))$ וכן הלאה עד 1
ולכן: $\log(1) + \log(2) + \log(3) + \cdots + \log(n)$
$= \log(1 \cdot 2 \cdot 3 \cdots n) = O(\log(n!))$
(לפי חוקי לוגים)

שאלה 4

```java
public static int Search(int[] arr, int k){
    int s = arr.length;
    if(k>arr[s-1] || k<arr[0]) return -1;
    int i=1;
    while(i<s && arr[i] <= k) i*=2;
    return binarySearch(arr, i/2, Math.min(i, s-1), k);
}
```

שאלה 4

```java
public static int Search(int[] arr, int k){
    int s = arr.length;
    if(k>arr[s-1] || k<arr[0]) return -1;
    int i=1;
    while(i<s && arr[i] <= k) i*=2;
    return binarySearch(arr, i/2, Math.min(i, s-1), k);
}

private static int binarySearch(int[] a, int l, int h, int val) {
    int m = (l+h)/2;                        //O(1)
    if(h<l) return -1;
    if(a[m]==val) return m;                 //O(1)
    if(a[m]>val)  return binarySearch(a, l, m-1, val);
    else          return binarySearch(a, m+1, h, val);
}
```

אלגוריתם החיפוש שבניתי מותאם בעיקר לחיפוש במבנה נתונים ממוין דבר אחר כאשר לא ידוע חסם חסם עליון לחיפוש.

בחלק הראשון האלגוריתם רץ עם אינדקס i בקפיצות של 1,2,4,8,16...
והוא רץ עד האינדקס הראשון שגדול או שווה ל-k סה"כ החלק הזה
דורש $\lceil \log(i) \rceil$ חזרות, כלומר $O(\log(i))$. (כאשר i כמו שמתואר בשאלה)

בחלק השני נבצע חיפוש בינארי על הטווח $[2^{\log(i)-1}, 2^{\log(i)}]$
שגודלו הינו: $2^{\log(i)} - 2^{\log(i)-1} = 2^{\log(i)-1}$
חיפוש בינארי לוקח במקרה כמו סדר גודל של $\log n$ פעולות
ובפרט, בגודל התחום נבצע סדר גודל של:

$$\log(2^{\log(i)-1}) = (\log(i)-1)\cdot\log(2) = \underline{O(\log(i))}$$

פעולות.

סדר האינדקסים: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

K

חיפוש בינארי