פתרון תרגיל מספר 8 - אלגוריתמים

שם: מיכאל גרינבאום, **ת.ז:** 211747639; **שם:** גיא לוי, **ת.ז:** 211744636

4 בנובמבר 2019

1. **פתרון:**

(א) **צ"ל:** האם $\sqrt{\log(n)} = \Theta\left(\log\left(\sqrt{n}\right)\right)$?

הוכחה:

לא נכון! נניח בשלילה כי מתקיים $\sqrt{\log(n)} = \Theta\left(\log\left(\sqrt{n}\right)\right)$, בפרט מתקיים $\sqrt{\log(n)} = \Omega\left(\log\left(\sqrt{n}\right)\right)$, ולכן מהגדרה $\exists n_0 \in \mathbb{N}, c > 0$ כך ש- $\forall n > n_0$ מתקיים

$$\frac{c}{2} \cdot \log(n) = c \cdot \log\left(n^{\frac{1}{2}}\right) = c \cdot \log\left(\sqrt{n}\right) \leq \sqrt{\log(n)}$$
$$\Rightarrow \frac{c}{2} \leq \frac{\sqrt{\log(n)}}{\log(n)} = \frac{1}{\sqrt{\log(n)}}$$

עתה נשים לב כי $\lim_{n\to\infty} \frac{1}{\sqrt{\log(n)}} = 0$, ולכן ממונוטוניות הגבול מתקיים

$$\frac{c}{2} = \lim_{n\to\infty} \frac{c}{2} \leq \lim_{n\to\infty} \frac{1}{\sqrt{\log(n)}} = 0 \Rightarrow \frac{c}{2} \leq 0 \Rightarrow \boxed{c \leq 0}$$

בסתירה להנחה ש- $c > 0$, ולכן $\boxed{\sqrt{\log(n)} \neq \Theta\left(\log\left(\sqrt{n}\right)\right)}$

מ.ש.ל.א.☺

(ב) **צ"ל:** האם $\sum_{j=1}^{n^2} \frac{n}{2^j} = \Omega\left(4n^2\right)$?

הוכחה:

לא נכון! נניח בשלילה כי מתקיים $\sum_{j=1}^{n^2} \frac{n}{2^j} = \Omega\left(4n^2\right)$, ולכן מהגדרה $\exists n_0 \in \mathbb{N}, c > 0$ כך ש- $\forall n > n_0$ מתקיים

$$c \cdot 4n^2 \leq \sum_{j=1}^{n^2} \frac{n}{2^j} = n \cdot \sum_{j=1}^{n^2} \frac{1}{2^j} \leq n \cdot \sum_{j=1}^{\infty} \frac{1}{2^j} \overset{(1)}{=} n$$
$$\Rightarrow 4c \cdot n \leq 1$$

כאשר (1) מתקיים מסכום אינסופי של סדרה הנדסית עבור מנה שווה $\frac{1}{2}$. לכן עבור כל $\max\left\{n_0, \frac{1}{4c}\right\} < n \in \mathbb{N}$ נקבל סתירה ל$4c \cdot n \leq 1$ ולכן $\boxed{\sum_{j=1}^{n^2} \frac{n}{2^j} = \Omega\left(4n^2\right)}$

מ.ש.ל.ב.☺

(ג) **צ"ל:** האם $n^n = \Theta\left(n^{\log(n)}\right)$?
הוכחה:
לא נכון! נניח בשלילה כי מתקיים $n^n = \Theta\left(n^{\log(n)}\right)$, ולכן בפרט $n^n = O\left(n^{\log(n)}\right)$, ולכן מהגדרה $\exists n_0 \in \mathbb{N}, c > 0$ כך ש־ $\forall n > n_0$ מתקיים

$$n^n \leq c \cdot n^{\log(n)} \Rightarrow \frac{1}{c} \leq \frac{n^{\log(n)}}{n^n} = n^{\log(n)-n}$$

עתה נשים לב כי

$$\lim_{n\to\infty} (\log(n) - n) = -\infty \Rightarrow \lim_{n\to\infty} (\log(n) - n) \cdot \log(n) \overset{\star}{=} -\infty$$
$$\Rightarrow \lim_{n\to\infty} n^{\log(n)-n} = \lim_{n\to\infty} e^{\log\left(n^{\log(n)-n}\right)} = \lim_{n\to\infty} e^{(\log(n)-n)\cdot\log(n)} \overset{\star}{=} e^{-\infty} = 0$$

ולכן ממונוטוניות הגבול מתקיים

$$\frac{1}{c} = \lim_{n\to\infty} \frac{1}{c} \leq \lim_{n\to\infty} n^{\log(n)-n} = 0 \Rightarrow \frac{1}{c} \leq 0 \Rightarrow \boxed{c \leq 0}$$

בסתירה להנחה ש־ $c > 0$, לכן $\boxed{n^n \neq \Theta\left(n^{\log(n)}\right)}$

מ.ש.ל.ג.☺

(ד) **צ"ל:** האם $f(n) = \Omega(g(n))$ או $g(n) = \Omega(f(n))$ כאשר f, g מונוטוניות עולות?
הוכחה:
לא נכון! נגדיר:

$$f(n) = \begin{cases} 2^{2^n} & n \text{ is even} \\ 2^{2^{n-1}} & n \text{ is odd} \end{cases}, g(n) = \begin{cases} 2^{2^{n-1}} & n \text{ is even} \\ 2^{2^n} & n \text{ is odd} \end{cases}$$

תחילה נראה שf מונוטונית עולה: יהי $1 < n \in \mathbb{N}$. אם n זוגי מתקיים $f(n) = 2^{2^n} > 2^{2^{n-2}} = f(n-1)$, ואם n אי זוגי מתקיים כי $f(n) = 2^{2^{n-2}} = f(n-1)$,
לכן קיבלנו כי

$$\forall n \in \mathbb{N}, n > 1 : f(n) \geq f(n-1)$$

כלומר מהגדרה f מונוטונית עולה. באופן דומה (עד כדי החלפה בין זוגי לאי זוגי) נקבל כי g מונוטונית עולה.
נניח בשלילה כי $g(n) = \Omega(f(n))$, כלומר $\exists n_0 \in \mathbb{N}, c > 0$ כך ש־ $\forall n > n_0$ מתקיים

$$c \cdot f(n) \leq g(n) \Rightarrow c \leq \frac{g(n)}{f(n)} = \begin{cases} \frac{2^{2^{n-1}}}{2^{2^n}} & n \text{ is even} \\ \frac{2^{2^n}}{2^{2^{n-1}}} & n \text{ is odd} \end{cases}$$

עתה נשים לב כי ממונוטוניות הגבול מתקיים

$$c \leq \frac{g(2n)}{f(2n)} = \frac{2^{2^{2n-1}}}{2^{2^{2n}}} = \frac{1}{2^{2^{2n-1}}}$$
$$\Rightarrow c = \lim_{n\to\infty} c \leq \lim_{n\to\infty} \frac{g(2n)}{f(2n)} = \lim_{n\to\infty} \frac{1}{2^{2^{2n-1}}} = 0 \Rightarrow \boxed{c \leq 0}$$

סתירה להנחה כי $c > 0$, ולכן $\boxed{g(n) \neq \Omega(f(n))}$,
עתה נניח בשלילה כי $f(n) = \Omega(g(n))$, כלומר $\exists n_0 \in \mathbb{N}, c > 0$ כך ש־ $\forall n > n_0$ מתקיים

$$c \cdot g(n) \leq f(n) \Rightarrow c \leq \frac{f(n)}{g(n)} = \begin{cases} \frac{2^{2^n}}{2^{2^{n-1}}} & n \text{ is even} \\ \frac{2^{2^{n-1}}}{2^{2^n}} & n \text{ is odd} \end{cases}$$

עתה נשים לב כי ממונוטוניות הגבול מתקיים

$$c \leq \frac{f(2n+1)}{g(2n+1)} = \frac{2^{2^{(2n+1)-1}}}{2^{2^{(2n+1)}}} = \frac{1}{2^{2^{2n}}}$$
$$\Rightarrow c = \lim_{n\to\infty} c \leq \lim_{n\to\infty} \frac{g(2n+1)}{f(2n+1)} = \lim_{n\to\infty} \frac{1}{2^{2^{2n}}} = 0 \Rightarrow \boxed{c \leq 0}$$

סתירה להנחה כי $c > 0$, ולכן $\boxed{f(n) \neq \Omega(g(n))}$,
לכן f, g עומדות בדרישות וגם $f(n) \neq \Omega(g(n))$ וגם $g(n) \neq \Omega(f(n))$

מ.ש.ל.ד.☺

(ה) **צ"ל:** האם $\underbrace{\log\log\ldots\log(n)}_{\text{m times}} = O(n)$?

הוכחה:

נכון! נשים לב כי $\forall n \in \mathbb{N}$

$$\underbrace{\log\log\ldots\log(n)}_{\text{m times}} \overset{\log(n)\leq n}{\leq} \underbrace{\log\log\ldots\log(n)}_{\text{m-1 times}} \overset{\text{induction}}{\leq} \log(n) \leq n$$

כלומר עבור $c = 1$, $n_0 = 1$, $\forall n \in \mathbb{N} > n_0$ מתקיים $\underbrace{\log\log\ldots\log(n)}_{\text{m times}} \leq c \cdot n$, כלומר $\boxed{\underbrace{\log\log\ldots\log(n)}_{\text{m times}} = O(n)}$

מ.ש.ל.ה.☺

2. **פתרון:**

(א) **צ"ל:** מה הטעות בהוכחה האינדוקטיבית?

הוכחה:

הבעיה היא שבצעד האינדוקטיבי, השימוש בהנחה לא תקין. אם המטבע המזויף יצא החוצה, אז אנחנו מביאים n מטבעות תקינים להנחה. והטענה על n נכונה **רק** אם יש **בדיוק** מטבע מזויף אחד, ואנחנו עלולים להביא ללא מטבע מזויף.

מ.ש.ל.א.☺

(ב) **צ"ל:** מה הטעות בהוכחה האינדוקטיבית?

הוכחה:

הבעיה היא שההוכחה מניחה שכל עץ בעל n קודקודים נוצר מלקיחה של עץ עם $n-1$ קודקודים וחיבור קודקוד חדש לקודקוד 1 בגרף.

מ.ש.ל.ב.☺

(ג) **צ"ל:** מה הטעות בהוכחה האינדוקטיבית?

הוכחה:

הבעיה היא שהטענה נכונה עבור טבעיים **לא כולל 0**, אבל בצעד האינדוקציה עבור $n = 2, x = 2, y = 1$ לדוגמא, אנחנו משתמשים בהנחת האינדוקציה על $x - 1 = 1, y - 1 = 0$, כשהטענה לא הוכחה עבור **טבעיים כולל 0**

מ.ש.ל.ג.☺

3. **צ"ל:** $mergeSort$ חסום על ידי $O(n\log(n))$

הוכחה:

תחילה נחשב את היעילות של merge, נשים לב כי

$$T_{\text{merge}}(n,m) = 3 + (n+m)\cdot 4 = O(n+m)$$

עתה נשים לב שנוסחת הרקורסיה של mergeSort היא

$$T_{\text{mergeSort}}(n) = 2\cdot T_{\text{mergeSort}}\left(\frac{n}{2}\right) + T_{\text{merge}}\left(\frac{n}{2},\frac{n}{2}\right)$$

מאחר שמתקיים $T_{\text{merge}}\left(\frac{n}{2},\frac{n}{2}\right)=O\left(n\right)$ הרי כי

$$\exists\left(c_1>0,1<n_0\in\mathbb{N}\right)\text{ such that }\forall n\geq n_0:T_{\text{merge}}\left(\frac{n}{2},\frac{n}{2}\right)\leq c_1\cdot n$$

עתה נוכיח באינדוקציה כי $T_{\text{mergeSort}}\left(n\right)\leq c\cdot n\log\left(n\right)$, כאשר נבחר $c=\max\left\{c_1,T_{\text{mergeSort}}\left(n_0\right)\right\}$
<u>בסיס האינדוקציה:</u> $n=n_0$ נשים לב כי מבחירה של c מתקיים:

$$T_{\text{mergeSort}}\left(n_0\right)\leq c\leq c\cdot n\cdot\log\left(n\right)$$

<u>צעד האינדוקציה:</u> נניח שהטענה נכונה לכל $k<n$ ונוכיח ל־n:

$$\begin{aligned}T_{\text{mergeSort}}\left(n\right)&=2\cdot T_{\text{mergeSort}}\left(\frac{n}{2}\right)+T_{\text{merge}}\left(\frac{n}{2},\frac{n}{2}\right)\leq 2\cdot T_{\text{mergeSort}}\left(\frac{n}{2}\right)+c_1\cdot n\\&\overset{\text{induction}}{\leq}2\cdot\left(c\cdot\frac{n}{2}\cdot\log\left(\frac{n}{2}\right)\right)+c_1\cdot n=c\cdot n\log\left(n\right)-cn+c_1n\\&\overset{c\geq c_1}{\leq}c\cdot n\log\left(n\right)\end{aligned}$$

כלומר הראנו כי עבור הבחירה לעיל של n_0,c מתקיים $\forall n\geq n_0$ מתקיים $T_{\text{mergeSort}}\left(n\right)\leq c\cdot n\log\left(n\right)$, לכן מהגדרה $\boxed{T_{\text{mergeSort}}\left(n\right)=O\left(n\log\left(n\right)\right)}$

מ.ש.ל.☺

4. **פתרון:**

(א) **צ"ל:** אלגוריתם יעיל לחיבור
הוכחה:
תחילה נראה את האלגוריתם

```
Sum(a,b):
    c = 0
    c[0], carry = a[0] + b[0]
    for i in range(1, t):
      c[i], carry = Sum_with_carry(a[i], b[i], carry)
    return c
Sum_with_carry(a, b, c):
    sum, carry1 = a + b
    sum, carry2 = sum + c
    return sum, (carry1 or carry2)
```

עתה נראה שהוא יעיל! ראשית נשים לב כי פעולת חיבור $'+'$ בין שני ביטים המחזירה carryיעילה (למעשה $O\left(1\right)$, על־ידי שימוש בשערים לוגיים בסיסיים - הסכום על־ידי שער xor והשארית על־ידי שער and). עתה נשים לב שSum_with_carry הוא יעיל כי הוא מבצע 7 פעולות על ביטים
ולכן $T_{\text{Sum\_with\_carry}}\left(2t\right)=O\left(1\right)=O\left(\left(2t\right)^0\right)$ שזה פולינומיאלי ולכן יעיל.
עתה נשים לב כי

$$T_{\text{Sum}}\left(2t\right)=2+t+\left(t-1\right)\cdot\left(T_{\text{Sum\_with\_carry}}\left(2t\right)+2\right)=2+t+9\left(t-1\right)=10t-7=O\left(2t\right)=O\left(\left(2t\right)^1\right)$$

כלומר Sum הוא גם אלגוריתם יעיל

מ.ש.ל.א.☺

(ב) **צ"ל:** אלגוריתם יעיל לכפל
הוכחה:
תחילה נראה את האלגוריתם

```
Mul(a,b):
   shifts = []
   shifts[0] = a
   for i in range(1,t):
     shifts[i] = Shift_left(shifts[i-1])
   sum = 0
   for i in range(0,t):
     if b[i] == 1:
       sum = Sum(sum, shifts[i])
   return sum
Shift_left(a):
   c = 0
   for i in range(1, t):
     c[i] = a[i-1]
   return c
```

תחילה נשים לב כי

$$T_{\text{Shift\_left}}(t) = t + (t-1) = 2t - 1 = O(2t)$$

עתה נשים לב כי

$$\begin{aligned} T_{\text{Mul}}(2t) &= t + t \cdot \left(t + T_{\text{Shift\_left}}(t)\right) + t + t \cdot (1 + T_{\text{Sum}}(2t)) = t + t^2 + t \cdot O(2t) + t + t + t \cdot O(2t) \\ &= O\left(2t^2\right) + 4t = O\left(2t^2\right) = O\left((2t)^2\right) \end{aligned}$$

לכן $T_{\text{Mul}}(2t)$ הוא יעיל

מ.ש.ל.ב.☺

5. **פתרון:**

(א) **צ"ל:** אלגוריתם שמשתמש ב$\log$ פעולות כפל לחישוב a^n
הוכחה:
תחילה נראה את האלגוריתם

```
Power(a,k):
   if k == 1:
     return a
   k_divide_2 = Shift_Right(k)
   if k[0] == 0:  #k is even
     p = Power(a, k_divide_2)
     return Mul(p, p)
   p = Power(a, k_divide_2)
   return Mul(a, Mul(p, p))
Shift_Right(a):
   c = 0
   for i in range(0, t-1):
     c[i] = a[i+1]
   return c
```

נוכיח באינדוקציה כי $\text{Power}(a,n)$ משתמש לכל היותר ב$2\log(n)$ פעולות כפל.
בסיס האינדוקציה: עבור $n=1$, האלגוריתם משתמש ב־ 0 פעולות כפל שזה לכל היותר $0 = 2\log(1) = 2\log(n)$.
צעד האינדוקציה: נניח ש$\text{Power}(a,k)$ משתמש לכל היותר ב־ $2\log(k)$ נכונה לכל $k<n$ ונוכיח לn
אם n זוגי, אז הפונקציה משתמשת ב1 ועוד כמות האלה ש$\text{Power}\left(a,\frac{k}{2}\right)$, כלומר לכל היותר

$$1+\text{Power}\left(a,\frac{k}{2}\right)\le 1+2\log\left(\frac{n}{2}\right)=2\log(n)-1\le 2\log(n)$$

אם n אי זוגי, אז הפונקציה משתמשת 21 ועוד כמות האלה ש$\text{Power}\left(a,\frac{k}{2}\right)$, כלומר לכל היותר

$$2+\text{Power}\left(a,\frac{k}{2}\right)\le 2+2\log\left(\frac{n}{2}\right)=2\log(n)$$

כלומר $\text{Power}\,(a,n)$ משתמש לכל היותר ב$2\log(n)=O(\log(n))$ פעולות כפל

מ.ש.ל.א.☺

(ב) **צ"ל:** אלגוריתם יעיל למציאת x כך ש$x^2=n$
הוכחה:
תחילה נראה את האלגוריתם

```
sqrt(n):
   return sqrt_helper(0, n, n)
sqrt_helper(a, b, n):
   sum = Sum(a, b)
   mid = Shift_Right(sum) # (a+b)/2
   guess = Mul(mid, mid)
   if guess == n:
     return mid
   else if guess > n:
     return sqrt_helper(a, mid - 1, n)
   return sqrt_helper(mid + 1, b, n)   // if guess < n
```

נראה שT_{sqrt} יעיל: לפני שננתח את זמן הריצה נשים לב כי פעולות של השוואה בין מספרים ($<,>,==$) יהיה השוואה בין הביטים מימין לשמאל ־ ולכן זמן הריצה ייקח $O(\log(n))$ (כמות הביטים כאורך הקלט). נשים לב כי אורך הקלט של sqrt_helper זה גודל המערך ונשים לב כי :

$$\begin{aligned}T_{\text{sqrt}}(n)=T_{\text{sqrt\_helper}}(n)&=\underbrace{O(\log(n))}_{T_{\text{Sum}}}+\underbrace{O(\log(n))}_{T_{\text{Comperison}}}+\underbrace{O(\log(n))}_{T_{\text{Shift\_Right}}}+\underbrace{O\left(\log^2(n)\right)}_{T_{\text{Mul}}}+T_{\text{sqrt\_helper}}\left(\frac{n}{2}\right)\\&=O\left(\log^2(n)\right)+T_{\text{sqrt\_helper}}\left(\frac{n}{2}\right)\le c\cdot\log^2(n)+T_{\text{sqrt\_helper}}\left(\frac{n}{2}\right)\end{aligned}$$

כאשר אי השוויון האחרון נכון עבור $c>0, n_0\in\mathbb{N}$ (שמותאמים לחסם שבמשוואה). נשים לב כי נאכל לבחור $n_0=1$ ובהתאמה נאכל לשנות את c (זמן הריצה של הפונקציות שאנחנו משתמשים בהם לא קופץ לערך בהתחלה של זמן הריצה כך שנצטרך לבחור n_0 מספיק גדול) נוכיח באינודקציה כי $T_{\text{sqrt\_helper}}(n)=c\cdot\log^3(n)$ לכל $n\ge 4$:
בסיס האינדוקציה: עבור $n=4$ מתקיים מתקיים

$$T(4)=c\cdot\log^2(4)+c\cdot\log^2(2)+c\cdot\log^2(1)=3c\le c\cdot\log^3(4)=8c$$

הדרוש.
צעד האינדוקציה: נניח את נכונות הטענה לכל $k<n$ ונוכיח ל־ n

$$\begin{aligned}T_{\text{sqrt\_helper}}(n)&\le c\cdot\log^2(n)+T_{\text{sqrt\_helper}}\left(\frac{n}{2}\right)\le c\cdot\log^2(n)+c\cdot\log^3\left(\frac{n}{2}\right)=c\cdot\left(\log^2(n)+\log^3(n)-3\log^2(n)+3\log(n)-1\right)\\&=c\cdot\left(\log^3(n)-2\log^2(n)+3\log(n)-1\right)\end{aligned}$$

נשים לב כי

$$2\log^2(n) \geq 3\log(n) \Leftrightarrow 2\log(n) \geq 3 \Leftrightarrow \log(n) \geq \frac{3}{2}$$

שנכון עבור $n \geq 4$ ולכן סך הכל נקבל כי

$$T_{\text{sqrt\_helper}}(n) \leq c \cdot \left(\log^3(n) - 1\right) = c \cdot \log^3(n)$$

כדרוש לצעד האינדוקציה. לכן נסיק מהטענה כי $T_{\text{sqrt}}(n) = T_{\text{sqrt\_helper}}(n) = O\left(\log^3(n)\right)$, כלומר sqrt יעיל כדרוש

מ.ש.ל.ב.☺