DATA STRUCTURES

עבודה מספר 1

שאלה 1:

מכיוון שלכל 3 פונקציות h(n) ,g(n) ,f(n)
אם f(n)=O(g(n)) וגם g(n) = O(h (n))
אז מתקיים: f(n) = O(h(n))

אז מספיק להוכיח את הסדר לכל זוג פונקציות.

פונקציות מספר 1) ו-4): קיים c>0 קבוע וN טבעי כך שלכל n>N מתקיים:

$$\frac{1}{n^2} \leq c \ \cdot 2019$$

וזה מתקיים כאשר c=1 ולכל n>=1
לכן:

$$\frac{1}{n^2} = O\ (2019) = O(1)$$

פונקציות מספר 9) ו-1): קיים c>0 קבוע וN טבעי כך שלכל n>N מתקיים:

$$2019 \leq \log\ (n^{\frac{2}{3}}) = \frac{2}{3}\log n\ \leq c\ \cdot \log n$$

וזה מתקיים כאשר c=1 ולכל $n > 2^{2019}$ ולכן:

$$2019 = O(\log(n))$$

פונקציות מספר 11) ו-9) : קיים c>0 קבוע וN טבעי כך שלכל n>N מתקיים :

$$\log\ (n^{\frac{2}{3}}) = \frac{2}{3}\log n\ \le c\cdot(\log n^{100}\ ) = c\cdot 100\log n$$

וזה מתקיים כאשר c=1 ולכל n>1 ולכן :

$$\log\ (n^{\frac{2}{3}}) = O(\log(n))$$

נוכיח בנוסף שפונקציה מספר 11 היא אומגה של פונקציה 9 :

$$\log\ \left(n^{\frac{2}{3}}\right) = \frac{2}{3}\log n \ge c\cdot\log n^{100} = c\cdot 100\log n$$

זה מתקיים כאשר $c = \frac{1}{300}$ ולכל n>1 ולכן :

$$f9 = \theta(f11)$$

פונקציות מספר 10) ו-11) : קיים c>0 קבוע וN טבעי כך שלכל n>N מתקיים :

$$\begin{aligned}\log\ (n^{100}) &= 100\log n\ \le c\cdot(\log 3^n\cdot n^2\ )\\ &= c\cdot(\log_3 3^n + \log n^2) = c\cdot(n\cdot\log_3 3 + 2\cdot\log n)\\ &\le c\cdot(2+1)\cdot n\end{aligned}$$

וזה מתקיים כאשר c=100 ולכל n>2 ולכן :

$$\log\ (n^{100}) = O(n)$$

פונקציות מספר 12) ו-10) : קיים c>0 קבוע וN טבעי כך שלכל n>N מתקיים :

$$\log_3(3^n\cdot n^2) = (\log_3 3^n + \log n^2) = (n\cdot\log_3 3 + 2\cdot\log_3 n) = n + 2\log_3 n \le c\cdot(n^2 + n\cdot\ln n^{10} + n + 10) = c\cdot(n^2 + n\cdot 10\ln n + n + 10) \le c\cdot(n^2 + 10n^2 + n^2 + 10n^2) = c\cdot 22n^2$$

מתקיים כאשר c=1 ולכל n>2 ולכן :

$$\log_3(3^n\cdot n^2) = O(n^2)$$

פונקציות מספר 2) ו-12) : קיים c>0 קבוע וN טבעי כך שלכל n>N מתקיים :

$$n^2 + n \cdot \log n^{10} + n + 10 \le n^2 + 10n \cdot n + 10n^2 + n^2 = 22n^2$$
$$\le\ c \cdot 2^{\log_{\sqrt{2}} n} = c \cdot 2^{\frac{logn}{log\sqrt{2}}} = c \cdot 2^{2 \cdot \log n} = c \cdot n^2$$

מתקיים כאשר c=23 ולכל n>1 ולכן :

$$n^2 + n \cdot \log n^{10} + n + 10 = O(n^2)$$

בנוסף נראה כי פונקציה מספר 2 היא אומגה של פונקציה מספר 12 :

$$n^2 + n \cdot \log n^{10} + n + 10 \ge n^2 \ge c \cdot 2^{\log_{\sqrt{2}} n} = c \cdot 2^{\frac{logn}{log\sqrt{2}}} = c \cdot 2^{2 \cdot \log n} = c \cdot n^2$$

וזה מתקיים לכל $c = \frac{1}{2}$ ולכל $n>1$ ולכן :

$$f12 = \theta(f2)$$

פונקציות מספר 3) ו-2) :

$$\lim_{n\to\infty} \frac{2^{\log_{\sqrt{2}} n}}{2^{\sqrt{n}}} = \lim_{n\to\infty} \frac{2^{\frac{logn}{log\sqrt{2}}}}{2^{\sqrt{n}}} = \lim_{n\to\infty} \frac{2^{2logn}}{2^{\sqrt{n}}} = \lim_{n\to\infty} \frac{n^2}{2^{\sqrt{n}}} = \lim_{n\to\infty} \frac{n^{1.5}}{ln2\ \cdot 2^{\sqrt{n}}}$$
$$= \lim_{n\to\infty} \frac{1.5n}{2 \cdot ln2 \ln 2 \cdot 2^{\sqrt{n}}} = \lim_{n\to\infty} \frac{1.5\sqrt{n}}{2 \cdot 2 \cdot \ln 2 \ln 2 \ln 2\ 2^{\sqrt{n}}}$$
$$= \lim_{n\to\infty} \frac{1.5}{2 \cdot 2 \cdot \ln 2 \ln 2 \ln 2\ \ \ln 2\ 2^{\sqrt{n}}} = 0$$

ולכן :

$$2^{\log_{\sqrt{2}} n} = O(2^{\sqrt{n}})$$

פונקציות מספר 5) ו-3) :

$$2^{\sqrt{n}} \le c \cdot 4^n = c \cdot 2^{2n}$$

מתקיים כאשר c=2 ולכל n>1 ולכן :

$$2^{\sqrt{n}} = O\ (4^n)$$

פונקציות מספר 7) ו-5) :

$$\lim_{n\to\infty} \frac{4^n}{n^n} = \lim_{n\to\infty} (\frac{4}{n})^n = 0$$

ולכן:

$$4^n = O(n^n)$$

פונקציות מספר 8) ו-7):

$$n^n \le c \cdot 3^{2^n}$$

$$2^{\log n^n} \le c \cdot 2^{\log 3^{2^n}}$$

$$2^{n \log n} \le c \cdot 2^{2^n \log 3}$$

כאשר c=1 :

$$n \log n \ \le 2^n \log 3$$

$$2^{\log n \log n} \ \le 2^{\log 2^n \log 3}$$

$$2^{\log n \log n} \ \le 2^{\text{n} \log 2 \log 3}$$

$$\log(n \cdot \log n) \le n \cdot log 3$$

$$\log(n \cdot \log n) \le \log(n^2) = 2 \cdot \log n \le n \ \cdot \log 3$$

וזה מתקיים לכל $n>2$

ולכן:

$$n^n = O\left(3^{2^n}\right)$$

פונקציות מספר 8) ו-6):

$$3^{2^n} \le c \cdot 2^{3^n}$$

$$2^{\log(3^{2^n})} \le c \cdot 2^{\log(2^{3^n})}$$

$$2^{2^n \log 3} \le c \cdot 2^{3^n \cdot \log 2}$$

$$2^{2^n \log 3} \le 2^{2 * 2^n} \le c \cdot 2^{3^n}$$

כאשר $c=1$:

$$2 * 2^n \le 3^n \Rightarrow \frac{\log 2}{\log \frac{3}{2}} > n$$

ולכל $n>4$ זה מתקיים ולכן:

$$3^{2^n} = O(2^{3^n})$$

שאלה 2:

א. הטענה נכונה, נוכיח:

נגדיר:

$$f(n)\begin{cases} n^2, & \text{זוגי } n \\ 1, & \text{אי} - \text{זוגי } n \end{cases}$$

נוכיח כי עבור $k = 1$

$$f(n-1) \neq \theta(f(n))$$

כלומר צריך להוכיח:

$$f(n-1) \neq O(f(n))$$

$$f(n-1) \neq \Omega\big(f(n)\big)$$

הוכחת $f(n-1) \neq O(f(n))$

נניח בשלילה שקיים C ו n_0 שעבורו מתקיים:

$$f(n-1) \leq c * f(n)$$

נראה שהדבר לא ייתכן.

עבור n_0 אי-זוגי מתקיים:

$c*f(n_0)=c*1=c$

אם n_0 אי זוגי אז n_0-1 זוגי ולכן:

$f(n_0-1)=(n_0-1)^2=n_0^2-2n_0+1$ כלומר, $n_0^2-2n_0+1\leq c$

אין זה נכון , ניתן למצוא n_0 כך שהאי שוויון למעלה לא מתקיים, כלומר קיים $n_0<n$ כך ש – $n_0^2-2n_0+1\geq c$ לכל c.

נניח בשלילה שקיים C ו n_0 שעבורו מתקיים:

$$f(n-1) \geq c * f(n)$$

נראה שהדבר לא ייתכן.

עבור n_0 זוגי מתקיים:

$c*f(n_0)=c*n_0^2$

אם n_0 זוגי אז n_0-1 אי - זוגי ולכן:

$f(n_0-1)= 1$ ובוודאי קיים c ו- $n>n_0$ כך ש:

$1\le c*n^2$

מ.ש.ל.

ב.

הטענה שגויה, לא קיימות פונקציות המקיימות את הדרישות נראה זאת:

דרישה 1:

$$\left(f(n)\right)^2 = O(f(n)) \Rightarrow \lim_{n\to\infty} \sup \left| \frac{\left(f(n)\right)^2}{f(n)} \right| < \infty$$

$$\lim_{n\to\infty} \sup|f(n)| < \infty$$

דרישה 2:

$$f(n) = \Omega \log(\log(n)) \Rightarrow \lim_{n\to\infty} \inf\left|\frac{f(n)}{g(n)}\right| > 0 \Rightarrow \lim_{n\to\infty} \inf \left| \frac{f(n)}{\log(\log(n)} \right| > 0$$

נניח שדרישה 1 מתקיימת, ונראה שלא ייתכן שדרישה 2 נכונה.

נניח: $\lim_{n\to\infty} \sup|f(n)| < \infty$

הגבול הנ"ל מתכנס נסמן אותו ב- L.

$$\lim_{n\to\infty} \inf\left|\frac{f(n)}{\log(\log(n))}\right| \to 0 \le \frac{\lim_{n\to\infty} \inf|f(n)|}{\lim_{n\to\infty} \log(\log(\mathrm{n}))} \le \frac{\lim_{n\to\infty} \sup|f(n)|}{\lim_{n\to\infty} \log(\log(\mathrm{n}))}$$

נשים לב כי כאשר $n \to \infty$, $\log(\log(n)) \to \infty,$ נשים לב ש - $\frac{\lim_{n\to\infty} \inf|f(n)|}{\lim_{n\to\infty} \log(\log(\mathrm{n}))}$

תמיד חיובי כי הפונקציות בערך מוחלט.

ולכן:

$$\frac{\lim_{n\to\infty} \sup|f(n)|}{\lim_{n\to\infty} \log(\log(\mathrm{n}))} = \frac{L}{\lim_{n\to\infty} \log(\log(n)} = \frac{L}{\infty} = 0$$

לפי כלל הסנדוויץ' גם $\frac{\lim_{n\to\infty} \inf|f(n)|}{\lim_{n\to\infty} \log(\log(\mathrm{n}))}$ הגבול הזה שואף ל $-$ 0.

ולכן תנאי 2 לא מתקיים.

ג. הטענה נכונה, נוכיח:

יהיו f(n) ו – g(n) פונקציות.

נניח כי $f(n),g(n)\geq 1$ לכל n.

צריך להוכיח: $f(n)+g(n)=O(f(n)*(g(n)))$

כלומר, צריך להוכיח: קיימים C ו – n_0 כך ש- $f(n)+g(n)\leq C*f(n)*g(n)$ לכל $n>n_0$.

נוכיח על פי הגבולות:

$f(n)=O(g(n)) \leftrightarrow \lim_{n\to\infty} \sup |\frac{f(n)}{g(n)}| < \infty$

$$\lim_{n\to\infty} \sup \left|\frac{f(n)+g(n)}{f(n)*g(n)}\right| = \lim_{n\to\infty} \sup \left|\frac{f(n)}{f(n)*g(n)} + \frac{g(n)}{f(n)*g(n)}\right| =$$

$$= \lim_{n\to\infty} \sup |\frac{1}{g(n)} + \frac{1}{f(n)}|$$

מהנחה $g(n),f(n)\geq 1$ ולכן:

$$\lim_{n\to\infty} \sup |\frac{1}{g(n)} + \frac{1}{f(n)}| \leq \lim_{n\to\infty} \sup |\frac{1}{1} + \frac{1}{1}| = 2 < \infty$$

מהנתון כי $g(n),f(n)\geq 1$ נסיק כי הגבול של ה – sup הגדול ביותר שיכול להתקבל הוא כאשר $f(n)=g(n)=1$ ולכן מכך שהגבול הגדול ביותר של הביטוי קטן מאינסוף כל ביטוי הקטן ממנו בוודאי קטן מאינסוף.

כנדרש.

ד. הטענה שגויה.

נראה דוגמא נגדית:

נגדיר:

$f(n)=\frac{1}{n^2}$ ו – $g(n)=1$

$$f\big(g(n)\big) = f(1) = \frac{1}{1^2} = 1$$

$$c * f(n)^{g(n)} = c * \left(\frac{1}{n^2}\right)^1 = \frac{c}{n^2}$$

עבור כל $n_0>\sqrt{c}$

אי השוויון $\frac{c}{n^2} \leq 1$ שגוי!

ולכן הטענה הכללית שגויה.

הראנו דוגמא נגדית.

<u>שאלה 3:</u>

א.

$$T(n) = T\left(n^{\frac{2}{3}}\right) + 17$$

$$T\left(n^{\frac{2}{3}}\right) = T\left(n^{\frac{2}{3}^{\frac{2}{3}}}\right) + 17$$

$$T\left(n^{\frac{4}{9}}\right) = T\left(n^{\frac{8}{27}}\right) + 17$$

.

.

.

.

$$T(n) = T\left(n^{\frac{2}{3}}\right) + 17 = \left(T\left(n^{\frac{2}{3}^{\frac{2}{3}}}\right) + 17\right) + 17 = \left(\left(T\left(n^{\frac{8}{27}}\right) + 17\right) + 17 + 17\right) = \cdots = T\left(\left(n^{\frac{2}{3}^{k}}\right)\right) + 17k$$

כאשר n יגיע ל2 זה יהיה תנאי הבסיס ולכן יש לחשב אחרי כמה פעולות n יגיע ל2:

$$n^{\frac{2}{3}^{k}} = 2$$

$$log(n^{\frac{2}{3}^{k}}) = log2$$

$$\frac{2}{3}^{k}\log(n)=1\Rightarrow k=\frac{\log(1)-\log\log n}{\log\frac{2}{3}}\Rightarrow k=-\frac{\log\log n}{\log\frac{2}{3}}=\frac{\log\log n}{\log\frac{3}{2}}$$

לכן כאשר c2=18,c1=1 :

$$c1\cdot\log\log n\le T(n)=\frac{\log\log n}{\log\frac{3}{2}}+\frac{\log\log n}{\log\frac{3}{2}}\cdot 17$$
$$=\left(\frac{18}{\log\frac{3}{2}}\right)\log\log n\le c2\cdot\log\log n$$

לכן:

$$T(n)=T\left(n^{\frac{2}{3}}\right)+17=\theta(\log\log n)$$

ב.

$$T(n)=7T\left(\frac{n}{2}\right)+n^{4}\log\log n$$

לפי שיטת המאסטר:

$$f(n)=n^{4}\log\log n=\ \Omega\left(n^{\log_2 7+\varepsilon}\right)$$

$$7\cdot\left(\frac{n}{2}\right)^{4}\log\log\frac{n}{2}=\frac{7}{16}n^{4}\log\log\frac{n}{2}<c\cdot n^{4}\log\log n$$

וגם כאשר c=9/16 ולכל n>2:

$$\frac{7}{16}n^{4}\log\log\frac{n}{2}<\frac{9}{16}\cdot n^{4}\log\log n$$

ולכן קיים c<1 ו – n_0 המקיימים:

$$7\cdot\left(\frac{n}{2}\right)^{4}\log\log\frac{n}{2}<9c\cdot n^{4}\log\log n$$

ולכן:

$$T(n)=\theta(n^{4}\log\log n)$$

ג.

$$T(n) = T(cn) + T\big((1-c)n\big) + 1$$

נחלק ל3 מקרים, כאשר $1>c>0.5$, כאשר $c=0.5$ וכאשר $0<c<0.5$.

כאשר $c=0.5$:

$$T(n) = 2T\left(\frac{n}{2}\right) + 1$$

לפי שיטת המסטר :

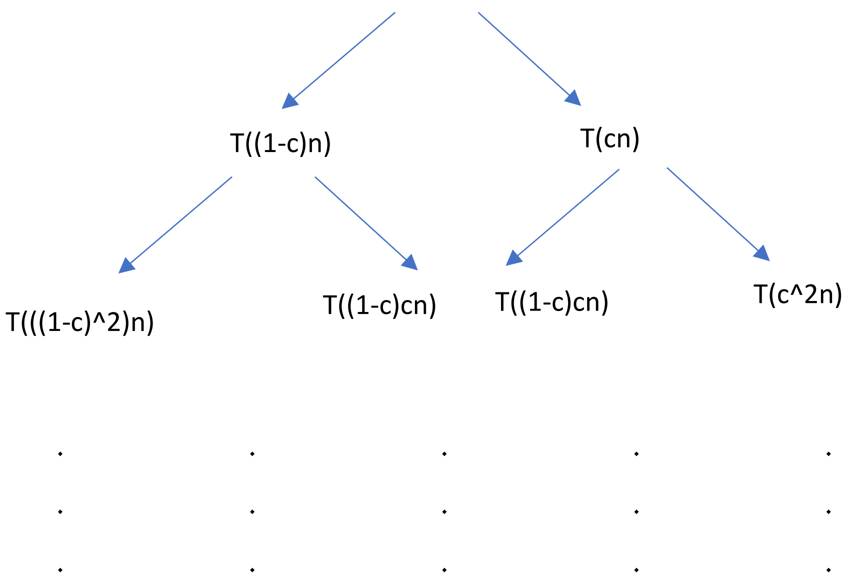
$$f(n) = 1 = O(n^{\log_2 2-\varepsilon})$$

ולכן :

$$\mathrm{T(n)} = \theta\left(n^{\log_2 2}\right) = \theta(\mathrm{n})$$

כאשר $1>c>0.5$ נשים לב שזה סימטרי למקרה כאשר $0<c<0.5$. ולכן מספיק שנחשב את אחד מהם.

נפתח עץ רקורסיה שידמה את הקריאות הרקורסיביות שמתבצעות :



נשים לב שהפיתוח השמאלי ביותר של העץ מתבצע :
$\log_{\frac{1}{1-c}} n$ פעמים.

נשים לב שהפיתוח הימני ביותר של העץ מתבצע :
$\log_{\frac{1}{c}} n$ פעמים.

ננחש ש:

$$T(n) = \theta(n)$$

(מכיוון שראינו שזה המקרה כאשר c=0.5)

נוכיח באינדוקציה $O(n)$ ו- $\Omega(n)$:

מכיוון שהנחנו כי $c > \frac{1}{2}$ נובע כי $\log_{\frac{1}{1-c}} n < \log_{\frac{1}{c}} n$

ולכן מספיק שנוכיח שכאשר עץ הקריאות הרקורסיביות מתבצע $\log_{\frac{1}{1-c}} n$ פעמים וחסום מלמטה ע״י $\Omega(n)$ וכאשר עץ הקריאות מתבצע $\log_{\frac{1}{c}} n$ הוא חסום מלמעלה ע״י $O(n)$ וככה נוכיח שהפונקציה $T(n) = \theta(n)$

נוכיח חסם תחתון:

נשים לב שכאשר עץ הקריאות מתבצע $\log_{\frac{1}{1-c}} n$ פעמים, מספר הפעולות שמתבצעות הן:

$$1 + 2 + \cdots + 2^{\log_{\frac{1}{1-c}} n}$$

נראה כי הביטוי הנ״ל הוא $\Omega(n)$.

מקרה בסיס:

כאשר n=1 :

$1 = \Omega(1)$.

נניח שלכל k<n הטענה מתקיימת כלומר:

$$T(k) = \Omega(k) \geq c \cdot k$$

נוכיח עבור n :

$$T(n) = T(cn) + T\big((1-c)n\big) + 1 \geq c1 \cdot cn + c2 \cdot (1-c)n + 1$$
$$\geq (1-c) \cdot (c1 + c2) \cdot n + 1 = \Omega(n)$$

נוכיח חסם עליון:

נשים לב שכאשר עץ הקריאות מתבצע $\log_{\frac{1}{c}} n$ פעמים, מספר הפעולות שמתבצעות הן:

$$1 + 2 + \cdots + 2^{\log_{\frac{1}{c}} n}$$

נראה כי הביטוי הנ״ל הוא $O(n)$.

מקרה בסיס:

כאשר n=1 :

$.1 = O(1)$

נניח שלכל k<n הטענה מתקיימת כלומר:

$$T(k) = O(k) \leq c \cdot k$$

נוכיח עבור n :

$$T(n) = T(cn) + T\big((1-c)n\big) + 1 \leq c1 \cdot cn + c2 \cdot (1-c)n + 1$$
$$\leq c \cdot (c1 + c2) \cdot n + 1 = O(n)$$

ולכן:

$$T(n) = \theta(n)$$

ד.

$$T(n) = T\left(\frac{2n}{5}\right) + 3T\left(\frac{n}{5}\right) + n$$

זמן הריצה הזה דומה ל*merge sort*, וכידוע זמן הריצה של *merge sort* הוא $\theta(nlogn)$ לכן יהיה סביר לנחש שגם זמן הריצה של האלגוריתם הנ״ל יהיה $\theta(nlogn)$. נוכיח זו באינדוקציה שלמה, תחילה נוכיח O ולאחר מכן נוכיח Ω.

הוכחה ש: $T(n) = O(nlogn)$

מקרה בסיס1212: $T(2) = 1 \leq c \cdot 2log2$ זה נכון כאשר c=2.

נניח שלכל k>0 כך ש: $0 < k < n$ מתקיים: $T(k) = O(klogk)$ צריך להוכיח נכונות עבור n:

$$T(n) = T\left(\frac{2n}{5}\right) + 3T\left(\frac{n}{5}\right) + n \leq c1\frac{2n}{5}\log\frac{2n}{5} + 3 \cdot c2 \cdot \frac{n}{5}\log\frac{n}{5} + n$$

$$\leq (c1 + c2)\frac{5n}{5}\log\frac{2n}{5} + n = (c1 + c2) \cdot n \cdot (\log\frac{2}{5} + logn) + n$$

$$\leq (c1 + c2) \cdot n \cdot \log n + n \leq (c1 + c2) \cdot n \cdot (\log n + 1)$$

$$\leq (c1 + c2) \cdot n \cdot (\log n + \log n) = (2c1 + 2c2)n\log n = O(n\log n)$$

הוכחה ש: $T(n) = \Omega(nlogn)$

מקרה בסיס1313: $T(2) = 1 \geq c \cdot 2log2$ זה נכון כאשר $c = \frac{1}{10}$.

נניח שלכל k>0 כך ש: $n < k < 0$ מתקיים: $T(k) = \Omega(klogk)$ צריך להוכיח נכונות עבור n:

$$T(n) = T\left(\frac{2n}{5}\right) + 3T\left(\frac{n}{5}\right) + n \geq c1\frac{2n}{5}\log\frac{2n}{5} + 3 \cdot c2 \cdot \frac{n}{5}\log\frac{n}{5} + n$$

$$\geq c1\frac{2n}{5}\log\frac{2n}{5} = c1 \cdot \frac{2}{5} \cdot n \cdot \left(\log\frac{2}{5} + \log n\right) \geq c1 \cdot \frac{2}{5} \cdot (n\log n)$$

$$c1 \cdot \frac{2}{5} \cdot (n\log n) = \Omega(\mathrm{n}\log n)$$

לכן הוכחנו כי:

$T(n) = O(n\log n)$ וגם: $T(n) = \Omega(\mathrm{n}\log n)$

ולכן:

$$T(n) = \theta(\mathrm{n}\log n)$$

ה.

$$T(n) = \frac{3}{2}T(n-1) + 1$$

שיטת האיטרציה:

$$T(n-1) = \frac{3}{2}T(n-2) + 1$$

$$T(n-2) = \frac{3}{2}T(n-3) + 1$$

.

.

.

$$T(n) = \frac{3}{2}T(n-1) + 1 = \frac{3}{2}\cdot\left(\frac{3}{2}T(n-2) + 1\right) + 1$$
$$= \frac{3}{2}\cdot\left(\frac{3}{2}\left(\frac{3}{2}T(n-3) + 1\right) + 1\right) + 1 = \cdots$$
$$= (\frac{3}{2})^{n-1}T(1) + 1 + \frac{3}{2} + \cdots + (\frac{3}{2})^{n-2}$$
$$= (\frac{3}{2})^{n-1} + \frac{\frac{3}{2}^{n-1} - 1}{\frac{1}{2}} = 3\cdot\left(\frac{3}{2}\right)^{n-1} - 2$$

נוכיח כי :

$$3\cdot\left(\frac{3}{2}\right)^{n-1} - 2 = \theta((\frac{3}{2})^{n})$$

נוכיח תחילה O ואז Ω

קיים c=3 ולכל n>3 :

$$3\cdot\left(\frac{3}{2}\right)^{n-1} - 2 \le 3\cdot(\frac{3}{2})^{n-1} \le 3\cdot\frac{3}{2}\cdot\left(\frac{3}{2}\right)^{n-1} = 3\cdot(\frac{3}{2})^{n} = c\cdot(\frac{3}{2})^{n}$$

$$3\cdot\left(\frac{3}{2}\right)^{n-1} - 2 = O((\frac{3}{2})^{n})$$

וגם קיים $c = \frac{2}{3}$ ולכל n>3 מתקיים :

$$3\cdot\left(\frac{3}{2}\right)^{n-1} - 2 \ge 3\cdot\left(\frac{3}{2}\right)^{n-1} - \left(\frac{3}{2}\right)^{n-1} = 2\cdot(\frac{3}{2})^{n-1} \ge (\frac{3}{2})^{n-1}$$
$$= \frac{2}{3}\cdot(\frac{3}{2})^{n} = c\cdot(\frac{3}{2})^{n}$$
$$3\cdot\left(\frac{3}{2}\right)^{n-1} - 2 = \Omega((\frac{3}{2})^{n})$$

ולכן :

$$T(n) = \theta((\frac{3}{2})^{n})$$

שאלה 4:

א.

ניתוח זמן ריצה Bubble Sort :

נשים לב, שבמקרה הגרוע ביותר, האיברים ממוינים בסדר הפוך.

Function Bubble Sort(Array[1…n]

1.for i ← 1 down to n-1

2.for j ← n down to i+1

3.if Array[j-1]>Array[j]

4.temp ← Array [j-1]

5.Array [j-1]← Array[j]

6.Array j ← temp

1. **פעולה 1 -** לולאת for הרצה על כל איברים המערך פחות אחד לכן מספר הפעולות הוא $n-1-1+1=n-1$

2. **פעולה 2 -**בכל איטרציה של הלולאת for של שורה **1** מתבצעת איטרציה נוספת על אברי המערך מהאיבר ה – n עד האיבר ה – i+1 כלומר, באיטרציה הראשונה יתבצעו n-1 השוואות כך שהאיבר האחרון במערך יגיע למקומו, באיטרציה השנייה n-2 השוואות כך שהיאבר שלפני האחרון יגיע למקומו במערך וכן הלאה... סך הכל הלולאה תתבצע:

$$(n-1)+(n-2)+(n-3)+\cdots+2+1=\frac{(n-1)n}{2}=\frac{n^2}{2}-\frac{n}{2}$$

למספר הפעמים שהלולאה השניה מתבצעת נוסיף את מספר הפעמים שהלולאה הראשונה מתבצעת ונקבל:

$$\frac{n^2}{2}-\frac{n}{2}+n-1=\frac{n^2}{2}+\frac{n}{2}-1$$

3. **פעולות 3** – **6** – בכל מעבר על איברי המערך בלולאה הפנימית, במקרה הגרוע מתרחשות ארבע פעולות אטומיות. פעולת השוואה אחת ושלוש פעולות השמה. כלומר בכל איטרציה מתבצעות בנוסף 4 פעולות כלומר:
אם במקרה הגרוע נצטרך בכל אחת מהכניסות ללולאות לבצע ארבעת הפעולות הללו סך הכל נצטרך לבצע:

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

4. **סך הכל** זמן הריצה של האלגוריתם הוא סכום של כל הפעולות הנ״ל ולכן הוא $\theta(n^2)$.

ב. נרשום את פונקציית זמן הריצה על פי המקרים של הפונקציה הרקורסיבית. נשים לב, שמקרה הבסיס הוא כש – n=0 או n=1 וכש – n>1 הפונקציה תחשב את הערך של power-1*base כאשר base קבוע. ולכן פונקציית זמן הריצה תהיה:

$$T(n) = \begin{cases} 1, n = 0 \backslash n = 1 \\ T(n-1) + 1, n > 1 \end{cases}$$

$$\mathrm{T}(n) = T(n-1) + 1 = T(n-2) + 2 = T(n-3) + 3 = \cdots$$
$$= T(n-k) + k$$

מתי הפונקציה תגיע למקרה הבסיס שלה ? ← כש – k=n-1 ולכן זמן הריצה של הפונקציה הרקורסיבית הזו הוא ליניארי כלומר:

$$\mathrm{T(n) = T(1) + n - 1 = \theta(n)}$$

ג. פונקציית זמן הריצה של האלגוריתם היא:

$$\mathrm{T}(n) = \begin{cases} 1\,, & n = 1, n = 0 \\ T(n-1) + d2, & n\%2 = 1 \\ T(n/2) + d1, & n\%2 = 0 \end{cases}$$

אילו היה רק המקרה שזמן הריצה מתחלק בכל פעם ב2, זמן הריצה היה שווה ל $logn$, לכן סביר להניח שגם במקרה הגרוע שאחרי כל פעם שאנו מחלקים בשתיים אנו צריכים להוריד אחד, יישמר עדיין זמן הריצה אסימפטוטית $logn$.

נוכיח שיעילות האלגוריתם היא $T(n) = \theta(\log(n))$.

באינדוקציה שלמה, ניקח מקרה בסיס $\mathrm{T}(2) = \mathrm{T}(1) + 1 = 2 = \theta(\log(2)) = \theta(1)$

נניח שלכל מספר טבעי k,כך ש- $n > k \geq 4$ מתקיים: $T(k) = \theta(\log k)$ כלומר קיימים קבועים c1,c2 כך ש:

$$c1 \cdot \log(k) \leq T(k) \leq c2 \cdot \log(k)$$

צל:

$$T(n) = \theta(\log(n))$$

נחלק ל2 מקרים:

n זוגי:

$$T(n) = T(n/2) + d1$$

לפי הנחת האינדוקציה:

$$c1\cdot\log(n/2)\le T(n/2)\le c2\cdot\log(n/2)$$

ולכן:

$$c1\cdot\log\left(\frac{n}{2}\right)\le T\left(\frac{\mathrm{n}}{2}\right)+d1\le(c2+d1)\cdot\log\left(\frac{n}{2}\right)=$$

(מכיוון ש- $(c2+d1)\cdot\log\left(\frac{n}{2}\right)=c2\cdot\log\left(\frac{n}{2}\right)+\mathrm{d1}\log\left(\frac{n}{2}\right)\ge c2\cdot\log\left(\frac{n}{2}\right)+d1\ge T\left(\frac{\mathrm{n}}{2}\right)+d1$ כי k הוא לפחות 4 לכן n/2 גדול מ2.

אי - זוגי:

$$T(n)=T(n-1)+d2$$

לפי הנחת האינדוקציה:

$$c1\cdot\log(n-1)\le T(n-1)\le c2\cdot\log(n-1)$$

ולכן:

$$c1\cdot\log(n-1)\le T(\mathrm{n}-1)+d2\le(c2+d2)\cdot\log(n-1)$$

(מכיוון ש-

$(c2+d2)\cdot\log(n-1)=c2\cdot\log\left(\frac{n}{2}\right)+\mathrm{d2}\log(n-1)\ge c2\cdot\log(n-1)+d2\ge T(n-1)+d2$

כי k הוא לפחות 4 לכן n-1 גדול ממש מ2 ולכן הלוגריתם של n-1 גדול ממש מ1.

שאלה 5:

א. **תיאור האלגוריתם:**

האלגוריתם מקבל מערך arr ממוין ואיבר X בתוך המערך. נחלק את האלגוריתם לשני חלקים ונראה ששני החלקים ביחד הם Olog(d) כאשר d הוא מספר האיברים הקודמים ל – X.

חלק ראשון:

נעבור על תאי המערך החל מהתא שהאינדקס שלו הוא 2^0, בחזקות הולכות וגדלות של 2, על מנת למצוא את התא הראשון בעל אינדקס של חזקה 2 אשר האיבר שהוא מכיל גדול או שווה ל- X .
כלומר נעבור על התאים: arr[1],arr[2],arr[4],arr[8],arr[16] וכן הלאה עד שנמצא את התא שהאינדקס שלו הוא חזקה של 2 שהאיבר בו גדול או שווה ל-X .
מכיוון שהתהליך הוא בחזקות הולכות וגדלות של 2, זמן הריצה המקסימלי של האלגוריתם יהיה O(log(d) +1) משום שבמקרה הגרוע ביותר האינדקס של התא במערך המכיל את X הוא עוקב מידי של תא שהאינדקס שלו הוא חזקה של 2 והאיבר שלו קטן מ – X.

כאמור, במקרה הגרוע ביותר האינדקס של התא במערך המכיל את X הוא עוקב מידי של תא שהאינדקס שלו הוא חזקה של 2 והאיבר שלו קטן מ-X. לכן במקרה (ובכל מקרה אחר) זה נצטרך לחפש את X בין התא שהאינדקס שלו חזקה 2 **שאיברו קטן מ – X** לבין התא בעל אינדקס של חזקה 2 שאיברו גדול מ – X שהוא כאמור במקרה הגרוע התא האחרון במערך (מכיוון שאילו החזקה הבאה של שתיים היתה גדולה מגודל המערך היו לנו פחות איברים לחפש מתוכם את X).

חלק שני:

נבצע חיפוש בינארי על האיברים שנותרו.

נסמן ב – 2^i את התא הראשון במערך שהאיבר בו גדול או שווה ל – X.

לכן X יהיה ממוקם בין התא שהאינדקס שלו 2^{i-1} לבין התא 2^i.
נשים לב, שבמקרה הגרוע ביותר ההפרש בין שני האינדקסים האלו הוא d .
נסביר זאת, מכיוון שבמקרה הגרוע ביותר X נמצא אינדקס אחד אחרי התא שבמיקום ה2^{i-1}, לכן מספר האיברים שנמצאים לפני X הוא 2^{i-1} ולפי ההגדרה הוא שווה d. וההפרש בין 2^i לבין 2^{i-1} הוא 2^{i-1} כלומר, d.
מכך שידוע שזמן הריצה של חיפוש בינארי הוא *log(n)* ובמקרה שלנו *n=d* אזי זמן הריצה של החיפוש הבינארי יהיה *log(d)*.

הערה: בחלק של האלגוריתם בו נבצע חיפוש בינארי על האיברים בטווח המתאים, נטפל גם במקרה שהערך X אינו נמצא במערך ונחזיר 1- , זמן הריצה יישאר אותו הדבר מכיוון שפעולה זו דורשת זמן ריצה של $\theta(1)$.

נוכיח ש:

log(d)+1+log(d)≤O(log(d))

כלומר נוכיח:

log(d)+1+log(d)≤c*log(d)

ואי השוויון הזה מתקיים עבור c=4 . משום ש :

$$2\log d + 1 \le 4 \cdot \log d$$

הוכחנו שסכום שני זמני הריצה שווים ל – Olog(d) ולכן זמן הריצה של כל האלגוריתם הוא לכל היותר Olog(d) במקרה הגרוע.

ב.

המטרה: למצוא את החציון של המערך המורכב מארבעה מערכים ממויינים שרירותיים כלשהם
בזמן ריצה ליניארי ובזיכרון נוסף בגודל O(1) .

תיאור האלגוריתם:

נשים לב שמהגדרת חציון, החציון יהיה ממוקם בתא ה n/2 במידה וסכום התאים של ארבעת המערכים הוא זוגי, אחרת, במידה והסכום אי זוגי הוא יהיה ממוקם באמצע המערך.
לכן תחילה נחשב את סכום התאים של ארבעת המערכים הנתונים ונשמור את המידע במשתנה שנקרא לו sum.
אם sum זוגי החציון יהיה ממוקם בתא ה – sum/2 (ונעדכן את sum בהתאם) ואם sum אי – זוגי החציון יהיה ממוקם בתא ה – (sum+1)/2 (ונעדכן את sum בהתאם) .נגדיר ארבעה משתנים, אחד לכל מערך, מסוג int ונאתחל אותם בערך 0, (האינדקס הראשון של כל אחד מהמערכים) ונפעל בצורה הבאה : נעבור על תאי ארבעת המערכים בלולאת while החל מהתא הראשון sum/2 פעמים או (sum+1)/2 פעמים (בהתאם לזוגיות של הערך sum כפי שתואר לעיל) ובכל מעבר נשמור במשתנה חיצוני (מחוץ ל – scope של הלולאה) את הערך הקטן ביותר מבין ארבעת המערכים, נעדכן את המצביע של המערך בו נמצא איבר המינימום מבין הארבעה באיטרציה הנוכחית לאיבר הבא (במידת האפשר, במידה ואי אפשר משמע שמכיוון שנשארו עוד איטרציות לבצע, החציון אינו נמצא באותו מערך ונפסיק לעבור עליו) ונקטין את הערך של sum ב - 1.
באיטרציה האחרונה האיבר שיישמר במשתנה החיצוני יהיה החציון זאת משום שלקחנו n/2 פעמים את האיבר הקטן ביותר מבין ארבעת המערכים ממש לפי הסדר אילו היו ממוינים ולכן האיבר האחרון שיישמר יהיה בהכרח החציון של המערך המשלב בסדר ממוין את ארבעת המערכים.

<u>ניתוח זמן ריצה:</u>

חישוב הערך של sum ← O(1)

עדכון הערך של sum ← O(1)

לולאת while ← O(sum/2)
פעולות בתוך הלולאה :

חישוב המינימום← O(sum/2)*3

עדכון האינדקס של המערך בו המינימום נמצא ← O(sum/2)

עדכון sum ← $O(sum/2)$

נחבר את כל זמני הריצה ונקבל:
$O((sum/2)*6+2)$
Sum הוא קבוע ליניארי המהווה את סכום גודלם
של ארבעת המערכים והוא קטן גם באופן ליניארי במהלך
האלגוריתם ולכן זמן הריצה הוא ליניארי, חישוב:
נסמן $(sum)=n$

$(n/2)*6+2=O(n/2)$

$n/2+2\leq C*n/2$

עבור $C=2$ ו – $n_0>4$

אי השוויון מתקיים.

ולכן, זמן הריצה של אלגוריתם הוא: $O(n)$ כך ש– n הוא גודל המערכים יחד.

<u>ניתוח זיכרון</u>:

שמירת המשתנה sum ← $O(1)$

שמירת משתנה אינדקס לכל מערך ← $O(1)$

שמירת הערך של המינימום ← $O(1)$

סך כל המשתנים מסתכם במספר קבוע ולכן סך הזיכרון הדרוש הוא $O(1)$.