CS1001.py HW3

שאלה 1:

א'. 1. הטענה **נכונה**. נוכיח:

$$n\log(n) = n(\log(n) - \log(2) + \log(2)) = 2\cdot\frac{n}{2}\left(\log\left(\frac{n}{2}\right) + \log(2)\right) = 2\cdot\frac{n}{2}\log\left(\frac{n}{2}\right) + 2\cdot\frac{n}{2}\log(2) \le$$
$$\le 2\left(\log(n) + \log(n-1) + \cdots + \log\left(\frac{n}{2}+1\right)\right) + 2\left(\log\left(\frac{n}{2}\right) + \log\left(\frac{n}{2}-1\right) + \cdots + \log(3) + \log(2) + \log(2)\right) =$$
$$= 2\log(n\cdot(n-1)\cdot\ldots\cdot 3\cdot 2\cdot 2) = 2\log(n!\cdot 2) = 2\log(n!) + 2\log(2) \le 2\log(n!) + 2\log(n!) = 4\log(n!)$$

לכן קיימים קבועים $c > 0$ ו-$n_0 \in \mathbb{N}$ כך ש-$n\log(n) \le c\cdot\log(n!)$, למשל $c = 4$ ו-$n_0 = 4$.

2. הטענה **נכונה**. נוכיח לפי ההגדרה השקולה שראינו בתרגול:

$$\lim_{n\to\infty}\frac{n^k}{\sum_{i=0}^{k}a_i n^i} = \lim_{n\to\infty}\frac{1}{\sum_{i=0}^{k}\frac{a_i n^i}{n^k}} = \lim_{n\to\infty}\frac{1}{\sum_{i=0}^{k}a_i n^{i-k}} = \frac{1}{a_k} < \infty \implies n^k = O\left(\sum_{i=0}^{k}a_i n^i\right)$$

3. הטענה **אינה נכונה**. דוגמה נגדית: $f_1(n) = n^2 + n, f_2(n) = n^2$, לכן $g_1(n) = n^2, g_2(n) = n^2$ אבל:

$$\frac{f_1(n)}{f_2(n)} = \frac{n^2+n}{n^2} = 1 + \frac{1}{n} = O\left(\frac{1}{n}\right) \ne O(1) = O\left(\frac{n^2}{n^2}\right) = O\left(\frac{g_1(n)}{g_2(n)}\right)$$

4. הטענה **אינה נכונה**. דוגמה נגדית: $f_1(n) = 3^n, f_2(n) = 5n + 2$, לכן $g_1(n) = 3^n, g_2(n) = n$ אבל:

$$f_1\circ f_2(n) = f_1(f_2(n)) = 3^{5n+2} = 9\cdot 243^n = O(243^n) \ne O(3^n) = O(g_1(g_2(n))) = O(g_1\circ g_2(n))$$

5. הטענה **נכונה**. נוכיח:

$$f(n) \le c_1\cdot g(n) \le c_1\cdot c_2\cdot h(n) = O(h(n))$$

הא״ש הראשון נכון כי נתון $f(n) = O(g(n))$ לכן קיימים $c_1 > 0$ ו-$n_1 \in \mathbb{N}$ כך שלכל $n > n1$ האי-שוויון נכון וכנ״ל לגבי g ו-h עבור $c_2 > 0$ ו-$n_2 \in \mathbb{N}$. לכן נבחר את המקסימום מביניהם ואז הא״ש השני נכון לכל n שגדול ממנו. הקבוע במקרה הזה הוא $c_1\cdot c_2$.

6. הטענה **נכונה**. נוכיח:

$$\lim_{n\to\infty}\frac{\log(n)^k}{n^\varepsilon} = \lim_{n\to\infty}\left(\frac{\log(n)}{n^{\frac{\varepsilon}{k}}}\right)^k = \left(\frac{k}{\varepsilon}\right)^k\lim_{n\to\infty}\left(\frac{\varepsilon}{k}\cdot\frac{\log(n)}{n^{\frac{\varepsilon}{k}}}\right)^k = \left(\frac{k}{\varepsilon}\right)^k\lim_{n\to\infty}\left(\frac{\log(n)^{\frac{\varepsilon}{k}}}{n^{\frac{\varepsilon}{k}}}\right)^k = \left(\frac{k}{\varepsilon}\right)^k\lim_{n\to\infty}\left(\frac{\log(n)}{n}\right)^\varepsilon = 0 < \infty$$

כאשר בשוויון האחרון, מכלל לופיטל מה שבתוך הסוגריים מתאפס ופונקציית החזקה עבור מעריך בין 0 ל-1 כאשר הבסיס שואף ל-0 החזקה כולה שואפת ל-0. לכן $\log(n)^k = O(n^\varepsilon)$.

7. הטענה **אינה נכון**. דוגמה נגדית: $f_1(n) = \log(n^3), f_2(n) = \log(n)$, לכן $g_1(n) = \log(n), g_2(n) = \log(n)$ אבל:

$$f_1(n) - f_2(n) = 2\log(n) = O(\log(n)) \ne 0 = O(\log(n) - \log(n)) = 0\ (g_1(n) - g_2(n))$$

ב'. 1. $\boldsymbol{O(n^2)}$. פונקציית ה- len לוקחת זמן קבוע. בלולאה החיצונית מתבצעות $\text{floor}(\log(n)+1) = O(\log(n))$ איטרציות שכן בכל איטרציה מקטינים את n פי 2 ולוקחים ערך תחתון (ראינו בכיתה מקרים דומים). בפנים רק הלולאה הפנימית משפיעה, אך הן תלויות – n קטן פי 2 בכל איטרציה חיצונית ולכן נסכום לקבלת מספר האיטרציות הפנימיות:

$$\frac{n}{2}+\frac{n}{4}+\ldots+1=\frac{\frac{n}{2}*\left(1-\frac{1}{2}^{\log(n)}\right)}{1/2}=n-\frac{n}{2^{\log(n)}}\le n=O(n)$$

בתוך כל איטרציה פנימית מתבצעת c*n עבודה (בשל החיפוש, in) כאשר ה-append מתבצע בזמן קבוע ולא נכלל בחישוב. לכן סה"כ נכפול לקבלת $O(n^2)$.

2. $\boldsymbol{O(log(n)\cdot log(n!))}$. עד הלולאות זמן קבוע. הלולאה החיצונית מבצעת כ-$500-n$ איטרציות והפנימית תלויה בחיצונית, לכן נסכום איבר איבר:

$$\log(500)+\log(501)+\ldots+\log(n)=\log\left(\frac{n!}{500!}\right)=O(\log(n!))$$

בכל איטרציה של הלולאה הפנימית מתבצעות $O(\log(n))$ איטרציות תמיד (log עונה על השאלה כמה פעמים (בערך, אבל אסימפטוטית ההבדל זניח) צריך לחלק את n ב-2 לקבלת 1). לכן סה"כ נכפול לקבלת $O(log(n)\cdot log(n!))$.

3. רשימה היא $mutable$ ולכן ניתן לבצע עליה פעולות $in-place$ ללא יצירת רשימה חדשה. האופרטור "+=" בפייתון מהווה קיצור למתודה $extend$ של $list$ שפועלת $in-place$ בניגוד לאופרטור "+" (רגיל) שיוצר העתק בזיכרון של הרשימה המקורית ומחזיר אותו. לכן עבור הפונקציה הראשונה, בכל איטרציה היא מגדירה מקום חדש בזיכרון ו"מעתיקה" את תוכן הרשימה לרשימה חדשה כולל האיבר הנוסף, ואילו הרשימה המקורית עליה רץ האיטרטור לא משתנה. עבור הפונקציה השנייה לעומת זאת, ההוספה מתבצעת באותו מקום בזיכרון, כלומר מעין דריסה של האובייקט הקודם, ולכן הרשימה המקורית, זאת שהאיטרטור רץ עליה משתנה בכל איטרציה.

<u>שאלה 3</u>:

ב׳. החלק שלפני הלולאה, השמת אורך הרשימה, וידוא הקלט ויצירת רשימה עולים זמן קבוע $O(1)$. בלולאה מתבצעות כ-$\frac{n}{k}$ איטרציות. בכל איטרציה מתבצעת $slicing$ שב-WCT (n מתחלק ב-k ללא שארית) עולה זמן של $O(k)$ ולאחר מכן המיון בחירה מוסיף זמן של $O(k^2)$. כלומר כל איטרציה עולה $O(k^2)$ לסיבוכיות הזמן (לפי המשפט שראינו בכיתה שסכום של פונקציות הוא O של המקסימום מבין הפונקציות). נכפיל בכמות האיטרציות ונקבל $O(kn)$.

ד׳. הפונקציה מבצעת שתי לולאות מקוננות. הלולאה הפנימית דואגת למזג כל זוג רשימות סמוכות באופן הבא: בכל איטרציה פנימית אנו ממזגים שתי רשימות סמוכות ברשימת הקלט כלומר $\frac{m}{2}$ איטרציות פנימיות. ב-WCT כל הרשימות הן בעלות האורך המקסימלי k, ולכן מהידיעה שהסיבוכיות של $merge$ היא $O(m+n)$ (כאשר m ו-n אורכי רשימות הקלט), $merge$ עושה $2k$ איטרציות. לאחר המיזוג הפונקציה שומרת את רשימת הפלט הממוזגת $in-place$ ברשימה המקורית במיקום הקטן יותר מבין שתי הרשימות שמוזגו (זמן קבוע), ואז "מסירה" (שוב $in-place$) את הרשימה במיקום הגדול יותר מבין אלה שמוזגו (זמן קבוע). כלומר סה"כ עבודה של $c\cdot m\cdot k$ באיטרציה הראשונה של הלולאה החיצונית. באיטרציה הבאה של הלולאה החיצונית מתבצע אותו תהליך כאשר הפעם אורך כל רשימה פנימית הוא $2k$ וכמות האברים ב-lst הוא $\frac{m}{2}$, לכן $\frac{m}{4}$ איטרציות פנימיות ו-$4k$ עבודה בכל איטרציה כזאת. כלומר שוב עבודה של $c\cdot m\cdot k$ ולמעשה זאת תהיה העבודה שמתבצעת בכל איטרציה חיצונית (כל פעם הרשימה מתקצרת בחצי ואורך הפנימיות גדל פי 2). תנאי העצירה ללולאה החיצונית הוא שב-lst תהיה רשימה אחת (שתהיה ממוינת וממוזגת). מהידיעה ש-m חזקה שלמה של 2, ישנן בדיוק $\log(m)$ איטרציות חיצוניות (מספר הפעמים שניתן לחלק חזקה שלמה של 2 ב-2 על מנת לקבל 1). לכן סה"כ עבודה של $WCT(m,k)=c\cdot k\cdot m\cdot\log(m)=O(m\cdot k\cdot\log(m))$ כדרוש.

ה׳. בפונקציה זו מבוצעות שתי הפונקציות מהסעיפים הקודמים בזו אחר זו, ולכן העבודה הכוללת היא סכום העבודות:

$$c\cdot k\cdot n+c\cdot m\cdot k\cdot\log(m)=c\cdot\left(k\cdot n+\left\lceil\frac{n}{k}\right\rceil\cdot k\cdot\log\left(\left\lceil\frac{n}{k}\right\rceil\right)\right)=O\left(n\cdot\left(k+\log\left(\frac{n}{k}\right)\right)\right)$$

כאשר הזנחת פונקציית התקרה נובעת מכך שלתוספת קבוע אין משמעות אסימפטוטית.

ו׳. k=1 היה גורם לסיבוכיות זהה, אך נשים לב שגם עבור k=log(n)>1 זה מתקיים:

$$O\left(n\cdot\left(\log(n)+\log\left(\frac{n}{\log(n)}\right)\right)\right)=O(n\log(n)+n\log(n)-n\log(\log(n)))=O(n\log(n))$$

השוויון האחרון נובע מהכללת הזהות $f_1(n)+f_2(n)=O\left(\max\left(g_1(n),g_2(n)\right)\right)$ עבור 3 פונקציות (ראינו בתרגול).

שאלה 4:

א'. b. סיבוכיות הזמן ב-WCT היא $O(logn)$ (n אורך הרשימה), משום שהקוד משתמש באלגוריתם החיפוש הבינארי, עם שינוי קל – במקום לבדוק איבר אחד באמצע בכל פעם, אנו בודקים גם את שני האיברים השכנים לו ומקצרים את טווח החיפוש בהתאם. עבור רשימות גדולות טווח החיפוש קטן בערך פי 2 בכל איטרציה (האיבר הנוסף שמקזזים מאותו צד של ההקטנה לא משפיע במונחים אסימפטוטיים).

ב'. b. האלגוריתם עובר בין כל שני איברים שכנים ברשימה ובודק אם האיבר שנמצא באינדקס הגדול יותר קטן יותר מזה שלפניו ואם כן מחליף ביניהם $in-place$. כך אנו מנצלים את העובדה שהסטייה של כל איבר ממיקומו ברשימה הממוינת היא לכל היותר בגודל של 1. כל הפעולות שנעשות בתוך הלולאה לוקחות זמן קבוע $O(1)$, ולכן מספר האיטרציות קובע את סיבוכיות הזמן – יש $n-1$ איטרציות, כלומר $O(n)$.

ג'. a. הטענה נכונה. עבור $n=1$ (n אורך הרשימה) מתקיים $i=n-1 \ \wedge \ i=0$. עבור $n=2$ או ששני האיברים שווים ואז שניהם מינימום מקומי או שהם שונים ואז הקטן יותר הוא מינימום מקומי לפי הגדרה. נניח בשלילה שקיימת רשימה בת $n \geq 3$ מספרים שאין בה מינימום מקומי. אזי היא בהכרח מקיימת $L[n-2] < L[n-1]$ וגם $L[0] > L[1]$ (אחרת לפחות אחד מאיברי הקצה הם מינימום מקומי). כעת מתקיים $L[n-3] < L[n-2]$ בהכרח (אחרת $L[n-2]$ מינימום מקומי) וכן $L[1] > L[2]$ (אחרת $L[1]$ מינימום מקומי) וכן הלאה. באיזשהו שלב מגיעים לאיבר האמצעי, $L[n//2]$, או לשני איברי האמצע $L[(n-1)//2]$ ו-$L[(n-1)//2+1]$ (אם n אי-זוגי או זוגי בהתאמה).במקרה הראשון, אם $(L[n//2] > L[n//2+1]) \vee (L[n//2] > L[n//2-1])$, אזי $L[n//2+1]$ או $L[n//2-1]$ מינימום מקומי בהתאמה. אם האיבר האמצעי שווה לפחות לאחד משניהם, אזי אותו איבר שהאמצעי שווה לו הוא מינימום מקומי. אם האיבר האמצעי קטן בדיוק מאחד מהם אזי השכן השני של האמצעי הוא מינימום מקומי. ואם האמצעי קטן משניהם, אזי הוא בעצמו מינימום מקומי. במקרה השני, אם שני האמצעיים לא שווים, אזי זה שקטן יותר הוא מינימום מקומי, ואם שווים, אזי שניהם מינימום מקומי. בכל מקרה קיבלנו שקיים מינימום מקומי – סתירה, ולכן הטענה נכונה לכל n.

c. הפעולות שלפני הלולאה מתבצעות בזמן קבוע $O(1)$ (אורך הרשימה בפייתון מחושב טרם ריצת הקוד ביעילות ונשמר עם הרשימה) ואם $n=1$ (n גודל הקלט) אזי הסיבוכיות היא $O(1)$. עבור $n>1$ התוכנית נכנסת ללולאה. ב-WCT (המינימום המקומי באמצע) ישנן כ-$\frac{n}{2}$ איטרציות ובפנים כל הפעולות מתבצעות בזמן קבוע (גישה לאיבר ברשימה לפי אינדקס והשוואה של שני מספרים). לכן סה"כ $WCT(n)=O(n)$ עבור $n \geq 2$.

שאלה 5:

ד'. הגדרת הרשימה הריקה תורמת זמן קבוע לכן נזניחה. הגדרת רשימת העזר נעשית באמצעות 5^k איטרציות של $O(1)$. לאחר מכן, עוברים על רשימת הקלט (n פעמים) ובכל איטרציה נעזרים בפונקציה מסעיף א' שסיבוכיותה $O(k)$. לאחר מכן, עוברים על רשימת העזר (5^k איטרציות), כאשר את האיטרציה הפנימית נבצע n פעמים (כך התאמנו את רשימת העזר לספור מופעים במקומות המתאימים). בכל איטרציה פנימית נעזרים בפונקציה מסעיף א' שסיבוכיותה $O(k)$. בשאר האיטרציות החיצוניות, כאשר התנאי לא מתקיים, הבדיקה של התנאי בלבד עולה זמן קבוע. נחשב לכן את סך העבודה:

$$c_1 \cdot 5^k + c_2 \cdot n \cdot k + c_3 \cdot n \cdot k + c_4 \cdot (5^k - n) \leq c_5 \cdot n \cdot k + c_6 \cdot 5^k = O(kn + 5^k)$$

השוויון האחרון נובע מהזהות $f_1(n) + f_2(n) = O(g_1 + g_2)$.

ו'. בפונקציה מסעיף ה' ישנן 2 לולאות מקוננות ב"ת ולכן סה"כ $5^k \cdot n$ איטרציות. בכל איטרציה מתבצעת קריאה לפונקציה מסעיף א' במה שהערכנו כ-$O(k)$. כעת, ב-WCT (בכל איטרציה התנאי מתקיים) בכל איטרציה מתבצעת גם קריאה לפונקציה מסעיף ב', שגם אותה הערכנו כ-$O(k)$ ולכן סה"כ $O(kn)$ פר איטרציה. סה"כ קיבלנו עבודה כוללת ב-WCT של $O(5^k \cdot kn)$ כדרוש. ישנה גם העבודה הנעשית לפני הלולאה ופעולות נוספות (השוואה, הוספה לרשימה וכו') אבל מניחים שהיא קבועה.

שאלה 6 :

יחס הזהב הוא הפתרון החיובי של המשוואה $x^2 - x - 1 = 0$. נוכל להשתמש באלגוריתם שמחפש שורש של פונקציה בשיטת החצייה שראינו בכיתה, על הפונקציה $f(x) = x^2 - x - 1$, פונקציה רציפה (פולינום). ראינו כי הפרמטר שמאפשר לנו לקרב את M (ערך האמצע) לשורש עד לרמת דיוק מסוימת הוא TOL (החסם למספר האיטרציות): לאחר ביצוע N איטרציות M רחוק בלכל היותר $\frac{U-L}{2^{N+1}}$ מהשורש (כי מבצעים דגימה בינארית – בכל איטרציה האינטרוול קטן בחצי). נעריך כי $\sqrt{5}$ הוא בין 2 ל-3 ולכן $\frac{1+\sqrt{5}}{2}$ הוא בין 1.5 ל-2 ולכן נבחר $L = 1.5, U = 2$ (השורש השני של הפונקציה שלילי ולכן לא יפריע בתחום). נדרוש דיוק של 3 ספרות אחרי הנקודה, כלומר עד 4 ספרות כדי להיות בטוחים (כך קטן הסיכוי בו הערך שנמצא יהיה רחוק בפחות מ-10^{-4} אבל עם ספרה שלישית שונה, בכל מקרה תמיד ניתן להקטין):

$$\frac{U-L}{2^{N+1}} < 10^{-4} \Rightarrow \frac{1}{2^{N+2}} < 10^{-4} \Rightarrow 2^{N+2} > 10^4 \Rightarrow N > \log(10^4) - 2 \Rightarrow N = \lfloor \log(10^4) - 1 \rfloor = 12$$

לגבי EPS, חשוב שיהיה קטן מספיק כדי לא לעצור את הריצה בטרם ביצוע כל N האיטרציות – נשאיר כ-$default$.
נקבל:

```
print(root(lambda x: x ** 2 - x - 1, 1.5, 2, TOL=12))
```

```
Itertion 0 L = 1.5 M = 1.75 U = 2 f(m) = 0.3125
Itertion 1 L = 1.5 M = 1.625 U = 1.75 f(m) = 0.015625
Itertion 2 L = 1.5 M = 1.5625 U = 1.625 f(m) = -0.12109375
Itertion 3 L = 1.5625 M = 1.59375 U = 1.625 f(m) = -0.0537109375
Itertion 4 L = 1.59375 M = 1.609375 U = 1.625 f(m) = -0.019287109375
Itertion 5 L = 1.609375 M = 1.6171875 U = 1.625 f(m) = -0.00189208984375
Itertion 6 L = 1.6171875 M = 1.62109375 U = 1.625 f(m) = 0.0068511962890625
Itertion 7 L = 1.6171875 M = 1.619140625 U = 1.62109375 f(m) = 0.002475738525390625
Itertion 8 L = 1.6171875 M = 1.6181640625 U = 1.619140625 f(m) = 0.0002908706665039062
Itertion 9 L = 1.6171875 M = 1.61767578125 U = 1.6181640625 f(m) = -0.0008008480072021484
Itertion 10 L = 1.61767578125 M = 1.617919921875 U = 1.6181640625 f(m) = -0.0002550482749938965
Itertion 11 L = 1.617919921875 M = 1.6180419921875 U = 1.6181640625 f(m) = 1.7896294593811035e-05
No root found in 12 iterations
None
```

אמנם הפונקציה לא הכריזה על שורש אבל המטרה הייתה לקרב אותו, ואכן קיבלנו ערך מקורב עד הספרה השלישית לערך התאורטי 1.6180339887499.