תרגיל בית 3

שאלה 1

1. **הוכחה**

מחוקי לוגריתמים :

$$\log n! \geq \sum_{k=\frac{n}{2}+1}^{n} \log(k) \geq \log\left(\left(\frac{n}{2}\right)^{\frac{n}{2}}\right) = \frac{n}{2}\log\left(\frac{n}{2}\right) = \frac{n}{2}(\log(n) - \log(2)) = \frac{n}{2}(\log(n) - 1)$$

לכל $n > 4$ מתקיים :

$$\log(n) > 2 \Rightarrow \frac{n}{2}(\log(n) - 1) \geq \frac{n}{4}\log(n)$$

לכן, נבחר $n_0 = 4, c = 4$, לפי הגדרת $O(\cdot)$:

$$\forall n > n_0. \ \ 4 \cdot \log n! \geq 4 \cdot \frac{n}{4}\log n \geq n \cdot \log n$$

ולכן :

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

■

2. **הוכחה**

$$\sum_{i=0}^{k} a_i n^i = a_k n^k + \sum_{i=0}^{k-1} a_i n^i$$

וכן

$$n^k < a_k n^k$$

מכיוון שידוע כי

$$a_k \in \mathbb{R}^+$$

ולכן

$$n^k = O\left(\sum_{i=0}^{k} a_i n^i\right)$$

■

3. **הוכחה**

לפי הגדרת $O(\cdot)$:

$$\exists N \in \mathbb{N}. \forall n > N. \exists c. |f_1(n)| < c \cdot |g_1(n)|$$

ובנוסף :

$$\exists N' \in \mathbb{N}. \forall n' > N'. \exists c'. |f_2(n')| < c' \cdot |g_2(n')|$$

נחלק את אי-השוויונות ונקבל:

$$\frac{|f_1(n)|}{|f_2(n')|} < \frac{c}{c'} \cdot \frac{|g_{1(n)}|}{|g_{2(n')}|}$$

מכיוון ש $f(n), g(n) > 0$ נוריד את הערך המוחלט ונבחר את $n = \max\{n, n'\}$ כך שעבור $\frac{c}{c'}$ מתקיים:

$$\frac{f_{1(n)}}{f_{2(n)}} = O(\frac{g_{1(n)}}{g_{2(n)}})$$

■

4. __הוכחה__

לפי הגדרת $O(\cdot)$:

$$\exists N \in \mathbb{N}. \forall n > N. \exists c. |f_1(n)| < c \cdot |g_1(n)|$$

ובנוסף:

$$\exists N' \in \mathbb{N}. \forall n' > N'. \exists c'. |f_2(n')| < c' \cdot |g_2(n')|$$

מכיוון ש $f(n), g(n) > 0$ נכפיל את אי השוויונות ונקבל:

$$|f_1(n)| \cdot |f_2(n')| < c \cdot c' \cdot |g_1(n)| \cdot |g_2(n')|$$

ונבחר את $n = \max\{n, n'\}$ כך שעבור $c \cdot c'$ מתקיים:

$$f_1 \cdot f_2(n) = O(g_1 \cdot g_2(n))$$

■

6. __הפרכה__

יהיו $0 < \epsilon < 1, k \geq 1$. נניח בשלילה שמתקיים:

$$\exists N \in \mathbb{N}. \forall n > N. \exists c. \left|\left(logn(n)\right)^k\right| < c \cdot |n^\epsilon|$$

מכיוון ש$(\log(n))^k > 0, n^\epsilon > 0$ ניתן להוריד את הערך המוחלט. נחלק את אי השוויון בn^ϵ ונקבל:

$$\frac{\left(logn(n)\right)^k}{n^\epsilon} < c$$

וזו סתירה כיוון ש - c קבוע ו - $n \to \infty$.

■

7. __הוכחה__

נניח:

$$f_1(n) = O\left(g_1(n)\right) \wedge f_2(n) = O(g_2(n))$$

לפי הגדרת $O(\cdot)$:

$$\exists N \in \mathbb{N}. \forall n > N. \exists c. |f_1(n)| < c \cdot |g_1(n)|$$

ובנוסף:

$$\exists N' \in \mathbb{N}. \forall n' > N'. \exists c'. |f_2(n')| < c' \cdot |g_2(n')|$$

מכיוון ש - $f(n), g(n) > 0$ נחסר את המשוואות ונקבל:

$$f_1(n) - f_2(n') = c \cdot |g_1(n)| - c' \cdot |g_2(n')|$$

נבחר $n_0 = n + n'$

$$f_1(n_0) - f_2(n_0) = (c - c') \cdot \left(g_1(n_0) - g_2(n_0)\right)$$

ולכן מתקיים:

$$f_1(n) - f_2(n) = O\left(g_1(n) - g_2(n)\right)$$

■

<u>שאלה 2</u>

ב.

1. הפונקציה len היא מסיבוכיות $O(1)$ (זניח)

   בלולאת while זו הסיבוכיות היא כאורך הרשימה – $O(n)$

   לולאת ה - if גם מבצעת כמות איטרציות כגודל הטווח שהיא מקבלת- במקרה הגרוע כל איבר שנכנס לwhile ייכנס ל if.

   כיוון שהif בתוך הwhile הסיבוכיות של הפונקציה כולה היא כפל בין הסיבוכיות של שתי הלולאות - $O(n^2)$

2. len וכל שאר הפעולות המובנות של פייתון שבפונקציה הן מסיבוכיות $O(1)$ וזניחות.

   לולאת הfor (הגדולה) - רצה כגודל הטווח - $O(n)$.

   לולאת הfor (הבינונית) - רצה כגודל הטווח של הלולאה הגדולה - $O(n)$.

   לולאת while (הקטנה) - רצה רק עבור $k < n$. (k ערכו ההתחלתי של k הוא 1 והוא מוכל ב2 כל איטרציה ולכן החל מהאיטרציה השנייה ערכו הוא חזקה של 2).

   לכן: $i > \log(n) \leftarrow 2^i < m$ לכן, סיבוכיות לולאה זו היא $O(\log(n))$.

   כיוון שמדובר ב3 לולאות אחת בתוך השנייה הסיבוכיות הכוללת היא מכפלה של שלושתן (שאר הפעולות בסיבוכיות $O(1)$ זניחות) והיא $O(n^2 \log(n))$.

ג.

הפונקציה add_to_list_1 יוצרת משתנה חדש במרחב הזיכרון של פייתון שמעתיק אליו את הרשימה המקורית + התוספת. לאחר יצירת הרשימה החדשה ומיקום המשתנים בתאיה הפונקציה סיימה את פעולתה.

לעומת זאת, בadd_to_list_2 הרשימה אינה מועתקת למשתנה חדש והתוספת מוספת אליה. כיוון שלרשימה אותו השם ואין פקודת עצירה הפונקציה למעשה לא תפסיק להוסיף את התוספת לרשימה הקיימת.

<u>שאלה 3</u>

ב.

סיבוכיות הזמן של selection sort היא $O(k^2)$ כפי שנלמד בשיעור.
סיבוכיות הזמן של generate_sorted_blocks נגזרת מכמות האיברים שנכנסים ללולאת ה - for : ערך עליון של אורך הרשימה (N) חלקי הקבוע K. לכן סיבוכיות הזמן של generate_sorted_blocks היא $O\left(\frac{n}{k}\right)$.
כיוון שselection sort נמצא בתוך generate_sorted_blocks הסיבוכיות הכוללת היא מכפלה של שניהם :

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

<u>שאלה 4</u>

א.

b. סיבוכיות הזמן היא $O(\log(n))$. ישנם שני if, אולם הם מתייחסים רק למקרה יחיד ולכן הסיבוכיות שלהם זניחה והיא $O(1)$.

שאר הפונקציה מממשת חיפוש בינארי, וכפי שנלמד בהרצאה הסיבוכיות של חיפוש בינארי היא $O(\log(n))$.

ב.

b. הפונקציה עוברת על האינדקסים של הרשימה הכמעט ממוינת, ומשווה כל פעם בין שני אינדקסים. במידה והאיבר באינדקס ה - i גדול מהאיבר באינדקס ה - i+1 היא תחליף ביניהם׳ במידה ולא תמשיך. כך תעבור על כל האינדקסים ברשימה.
יש בפונקציה שתי לולאות אחת בתוך השנייה שכל אחת מהן מבצעת איטרציות כגודל הטווח (אורך הרשימה פחות 1). כיוון שהלולאות אחת בתוך השנייה הסיבוכיות היא מכפלה של הסיבוכיות של שתיהן ולכן היא $O((n-1)^2)$

<u>שאלה 5</u>

א. הפונקציה מבצעת ראשית $O(nk)$ איטרציות עבור שינוי האיבר הרלוונטי לספרה 1 : n פעמים מתבצע string_to_int שהיא פונקציה המבצעת k איטרציות בעצמה. לאחר מכן, הפונקציה עוברת איבר-איבר ברשימה באורך k^5 איברים ולכן סיבוכיות הזמן שלה היא $O(nk + k^5)$.

ב. הפונקציה מבצעת ראשית k^5 איטרציות בלולאה הראשונה, בה היא מבצעת שוב k איטרציות עבור הפונקציה int_to_string. לאחר מכן, באותה הלולאה, מבצעת הפונקציה שימוש באופרטור המובנה של פייתון in שהוא בסיבוכיות זמן של $O(n)$. לכן, בסיכום כל השלבים הפונקציה רצה בסיבוכיות זמן של $O(k^5 \cdot nk)$.

<u>שאלה 6</u>

```python
def golden_ratio (L=1.6 , U=1.65 , EPS=10**-5, TOL=200):

    for i in range(TOL):
        M = (L+U)/2
        fM= (lambda x: x**2-x-1)(M)
        if abs(fM) < EPS:
            return M
        elif not L < M < U:
            return None
        elif fM < 0:
            L = M
        else:
            U = M
    return None
print (golden_ratio())
```

```
C:\Anaconda\envs\HomeWork1\python.exe C:/Users/USER/PycharmProjects/HomeWork1/6.py
Itertion 0 L = 1.6 M = 1.625 U = 1.65 f(m) = 0.015625
Itertion 1 L = 1.6 M = 1.6125 U = 1.625 f(m) = -0.012343750000000098
Itertion 2 L = 1.6125 M = 1.61875 U = 1.625 f(m) = 0.0016015624999998757
Itertion 3 L = 1.6125 M = 1.615625 U = 1.61875 f(m) = -0.005380859374999769
Itertion 4 L = 1.615625 M = 1.6171875 U = 1.61875 f(m) = -0.00189208984375
Itertion 5 L = 1.6171875 M = 1.61796875 U = 1.61875 f(m) = -0.0001458740234376865 2
Itertion 6 L = 1.61796875 M = 1.6183593749999998 U = 1.61875 f(m) = 0.0007276916503902164
Itertion 7 L = 1.61796875 M = 1.6181640625 U = 1.6183593749999998 f(m) = 0.0002908706665039062 5
Itertion 8 L = 1.61796875 M = 1.61806640625 U = 1.6181640625 f(m) = 7.248878479026999e-05
Itertion 9 L = 1.61796875 M = 1.618017578125 U = 1.61806640625 f(m) = -3.669500350955701e-05
Itertion 10 L = 1.618017578125 M = 1.6180419921875 U = 1.61806640625 f(m) = 1.7896294593811035e-05
Itertion 11 L = 1.618017578125 M = 1.61802978515625 U = 1.6180419921875 f(m) = -9.39950346956486 3e-06
Found an approximated root
1.61802978515625
```

הטעות חסומה כנדרש כיוון שבפייתון המציג מסי׳ בשיטת floating point מספרים מוצגים כחזקות של 2. יחס הזהב אינו מסי׳ שניתן להציג כחזקה כלשהי של 2 (בהצגה בינארית ע״י floating point) ולכן ניתן להתקרב אליו עד כדי שגיאה.

בחרנו ערך TOL גדול מן הנדרש כפי שניתן לראות בהרצה - למרות שה TOL הוגדר 200, נדרשו רק 11 איטרציות על מנת להגיע לרמת הדיוק הנדרשת.