CS1001.py HW1

Question 1:

שלוש פונקציות הקיימות במחלקה str שאינן קיימות במחלקה list:

```
>>> 'abcd'.replace('b', 'p')
'apcd'
>>> str.replace('abcd','b', 'p')
'apcd'
>>> 'abcd'.split()
['abcd']
>>> str.split('abcd')
['abcd']
>>> 'abcd'.capitalize()
'Abcd'
>>> str.capitalize('abcd')
'Abcd'
```

שלוש פונקציות הקיימות במחלקה list שאינן קיימות במחלקה str:

```
>>> ['a', 'b', 'c', 'd'].pop(2)
'c'
>>> list.pop(['a', 'b', 'c', 'd'], 2)
'c'
>>> lst = ['a', 'b', 'c', 'd']
>>> lst.append('e')
>>> lst
['a', 'b', 'c', 'd', 'e']
>>> lst = ['a', 'b', 'c', 'd']
>>> list.append(lst, 'e')
>>> lst
['a', 'b', 'c', 'd', 'e']
>>> lst = ['a', 'b', 'c', 'd']
>>> lst.remove('b')
>>> lst
['a', 'c', 'd']
>>> lst = ['a', 'b', 'c', 'd']
>>> list.remove(lst, 'b')
>>> lst
['a', 'c', 'd']
```

Question 2:

1. "12345678":

Iteration	i	ID[i]	val	total
1	0	'1'	1	1
2	1	'2'	2	5
3	2	'3'	3	8
4	3	'4'	4	16
5	4	'5'	5	21
6	5	'6'	6	24
7	6	'7'	7	31
8	7	'8'	8	38

2. "31857489":

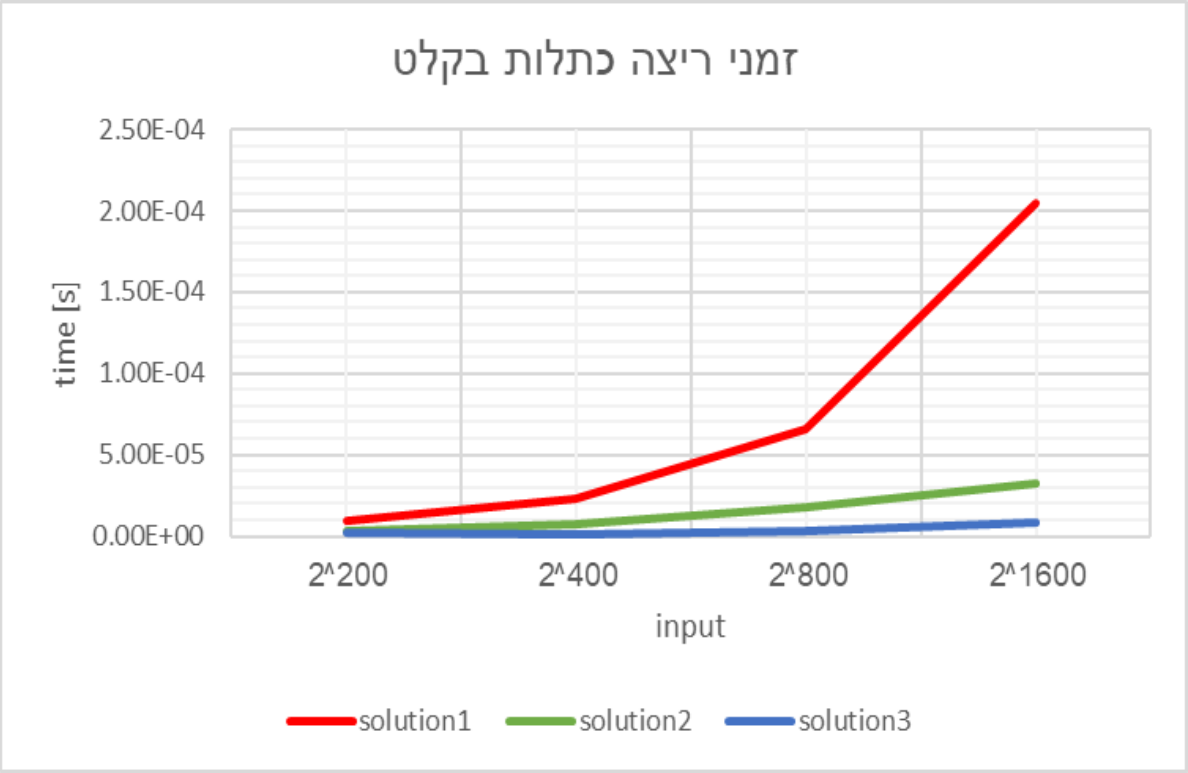
Iteration	i	ID[i]	val	total
1	0	'3'	3	3
2	1	'1'	1	5
3	2	'8'	8	13
4	3	'5'	5	14
5	4	'7'	7	21
6	5	'4'	4	29
7	6	'8'	8	37
8	7	'9'	9	46

Question 3:

הערה: כל הזמנים המופיעים בטבלאות מהווים ממוצע של 1000 ריצות על כל קלט, ליציבות הזמנים ולהפחתת התלות בגורמים אקראיים שיכולים להשפיע בריצה מסוימת.

א'.

solution	2**200	2**400	2**800	2**1600
1	0.9800000 00001979 3e-05 sec	2.3042400 00001556 3e-05 sec	6.5895799 99998563 e-05 sec	20.479410 00001003 5e-05 sec
2	0.3492500 00001255 5e-05 sec	0.7375900 00003685 e-05 sec	1.7891400 00004252 4e-05 sec	3.2842199 99997439 5e-05 sec



ניתן לראות בפתרון הראשון כי עם הגדלת ערכו של הקלט, זמן הריצה גדל גם הוא, ואף בקצב הולך וגדל. בפתרון השני לעומת זאת, זמן הריצה גדל בקצב איטי יותר עבור הגדלת הקלט. ניתן לשער כי זמן הריצה, שתלוי בעיקר במספר האיטרציות שבלולאה, הולך וגדל באופן משמעותי יותר בפתרון הראשון יחסית לפתרון השני, כי מלבד מספר הספרות (מהן מורכב הקלט) שקובע את מספר האיטרציות, בפתרון זה ישנן שתי פעולות מתמטיות שנעשות על הקלט, חלוקה שלמה (רצפה) ושארית. פעולות אלה ככל הנראה תלויות בגודל הקלט. בפתרון השני עוברים על הספרות כמחרוזת, ובכל איטרציה בודקים תו בודד – הבדיקה בכל איטרציה אינה תלויה בגודל הקלט, ולכן זמן הריצה לא גדל באופן משמעותי.

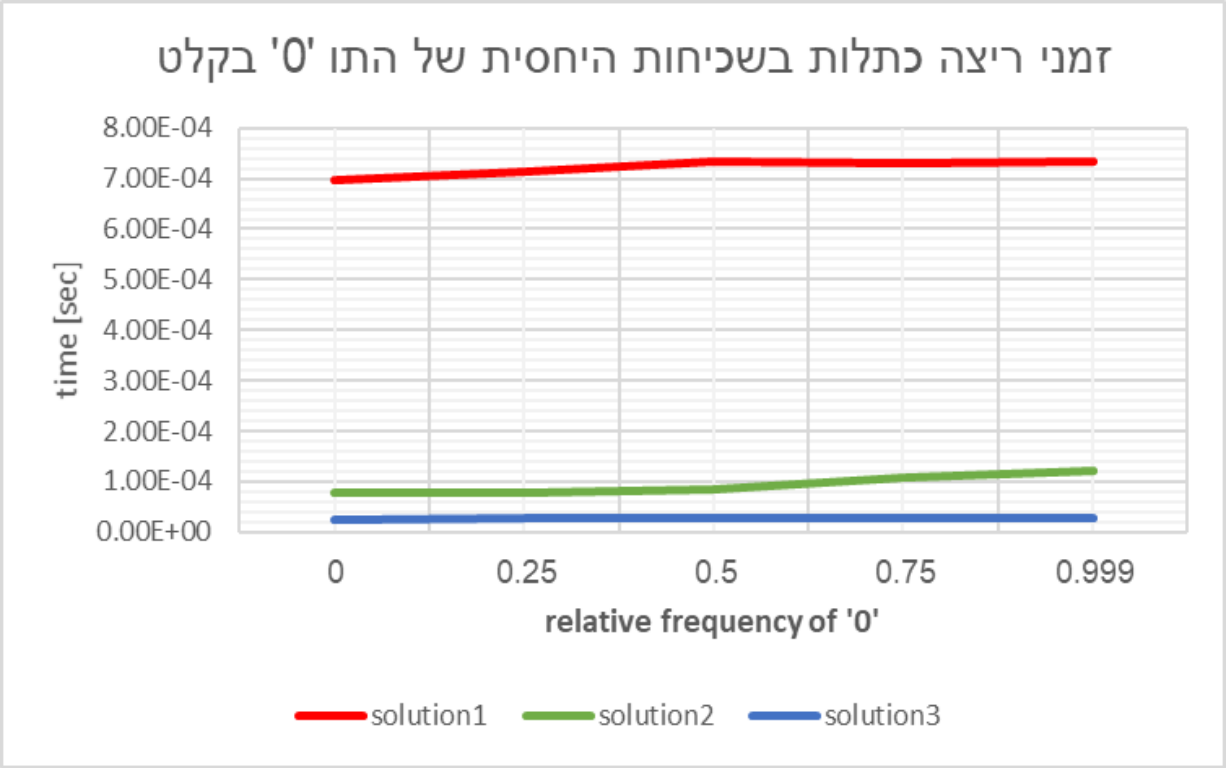
ב'.

solution	2**200	2**400	2**800	2**1600
3	0.17782999999998012e-05 sec	0.139059999998281 12e-05 sec	0.281499999999823 58e-05 sec	0.801030000000668 7e-05 sec

כפי שניתן לראות, הפתרון השלישי אכן יעיל יותר מבחינת זמן ריצה בהשוואה לשני הפתרונות הראשונים – זמני הריצה לקלטים השונים הם הנמוכים ביותר בפתרון זה. כמו בפתרון השני, גם כאן זמני הריצה לא משתנים הרבה יחסית לשינוי בפתרון הראשון עם הגדלת הקלט.

ג'. נבדוק מקרים עבור 1000 ספרות כדוגמה:

sol	int('7' *1000)	int('7077' *250)	int('70' *500)	int('7000' *250)	int('7'+'0' *999)
1	69.76264000003312e-05 sec	71.271080000030 9e-05 sec	73.23324999999444e-05 sec	73.039780000004 84e-05 sec	73.450759999997732e-05 sec
2	7.918309999953977e-05 sec	7.871600000497 39e-05 sec	8.329060000011168e-05 sec	10.641890000027 843e-05 sec	12.021579999978372e-05 sec
3	2.687559999975519 5e-05 sec	2.714050000029 987e-05 sec	2.723499999979 14e-05 sec	2.797449999980 0173e-05 sec	2.779040000032750 8e-05 sec



ניתן לראות כי עבור כל שיטת פתרון שונה, זמן הריצה לא משתנה באופן משמעותי יחסית לכמות האפסים, ולא באופן מגמתי מובהק כתלות במספר האפסים (לא מונוטוני עולה ולא מונוטוני יורד על גבי התחום הכולל, כי בכל אחת משיטות הפתרון יש קלט ש"שובר" את העלייה בזמן הריצה כתלות במספר האפסים, למשל בפתרון הראשון בקלט הרביעי משמאל יש יותר אפסים מאשר בקלט השלישי, ואילו בקלט השלישי זמן הריצה מעט ארוך יותר, וזאת בשונה לשאר הקלטים בהם הזמן גדל במקצת ככל שמספר האפסים גדל. נסיק (אף על פי שנבדקו קלטים באורך מסוים) כי למספר האפסים בקלט בעל מספר ספרות קבוע, אין השפעה מוחלטת על זמן הריצה של כל אחד מהפתרונות.

ד'.

```
num = 2**100
cnt = 0
for i in range(num):
    cnt = cnt + 1
```

עבור קטע קוד זה, נצפה לזמן ריצה ארוך באופן משמעותי מאשר זמן הריצה שלקח לכל הפתרונות בסעיפים הקודמים על גבי הקלטים שעסקנו בהם. זאת משום ששלושת הפתרונות הקודמים עסקו בחישובים בהם מספר האיטרציות הוא באורך של כמות הספרות שיש במספר, אך הפעם אנחנו מספר האיטרציות הוא 2^{100}, שזה משמעותית יותר פעמים מאשר 482 (כמות הספרות שיש ב-2^{1600}, הקלט הגדול ביותר לו נדרשנו בסעיף א').
לפתרון 2, שהוא הכי דומה לקוד שלעיל מבחינת מבנה, לקח כ-3.28×10^{-5} שניות לבצע 482 איטרציות. ננסה להעריך את זמן הריצה של הקוד שלעיל, כאשר ההערכה תתבסס על ההשערה לפיה זמן הריצה פרופורציוני לכמות האיטרציות. נקבל:

$$\frac{2^{100}}{482} * 3.2842199999974395 \times 10^{-5}\ sec \approx \mathbf{8.637434552444627} \times 10^{22}\ \boldsymbol{sec}$$

כלומר זמן ריצה מסדר גודל של כמעט 10^{23} (כמספר החלקיקים הנמצאים במול אחד של חומר נתון) שניות, או 10^{16} שנים.