מבוא מורחב למדעי המחשב ~ תרגיל בית מספר 1

שחר פרץ

13 ביוני 2024

2

סעיף א'

פירוט והסבר השגיאות שהתקבלו:

1. אם נריץ על טיפוס שאינו מחרוזת, נקבל את השגיאה:

```
Exception has occurred: AssertionError
exception: no description
```

זאת מכיוון שהביטוי `ininstance(id_num,str)` בעל ערך `False` בעבור `id_num` מטיפוס שאינו `str`, על כן ה־keyword "`assert`" תקפיץ שגירה.

2. אם נריץ על מחרוזת באורך קצר מ־8, נקבל את השגיאה:

```
Exception has occurred: AssertionError
exception: no description
```

זאת מכיוון שהביטוי `len(id_num) == 8` בעל ערך `False` בעבור `id_num` באורך שאינו 8, על כן ה־keyword "`assert`" תקפיץ שגיאה.

3. אם צריך על מחרוזת באורך 8 המכילה תווים שאינם ספרות, נקבל את השגיאה:

```
Exception has occurred: ValueError
invalid literal for int() with base 10: 'a'
```

משום שבשורה `val = int(id_num[i])` נעשה נסיון להמיר את התו במיקום ה־i (ונעשה מעבר על כל המיקומים) ממחרוזת למספר, אך התו "a" לא מסמל שום מספר בבסיס 10 (ערך ברירת־המחדל של המתודה `int`).

סעיף ב'

אם נסיר את הפקודות המתחילות במילה `assert`, אז הפונקציה לא תבדוק יותר את תקינות הקלט, ותקיץ שגיאות אחרות שנובעות מאופן המימוש עצמו. לדוגמה, עבור טיפוסים שאינם מחרוזת נקבל (ככל הנראה, כתלות בקלט) `TypeError` כשנרצה לגשת לאינדקס של טיפוס שלא תומך בזאת, ועבור מחרוזות מתחת לאורך של 8 תווים נקבל `IndexError` כשפייתון ינסה לגשת לאינדקס גבוהה מדי שלא קיים במחרוזת.

סעיף ג'

טבלת מעקב עבור `control_digit("87654321")`:

Iteration	i	id_num[i]	val	total
1	0	"8"	8	8
2	1	"7"	7	13
3	2	"6"	6	19
4	3	"5"	5	20
5	4	"4"	4	24
6	5	"3"	3	30
7	6	"2"	2	32
8	7	"1"	1	34

טבלת מעקב עבור control_digit("33455896") (ת.ז. שלי):

Iteration	i	id_num[i]	val	total
1	0	"3"	3	3
2	1	"3"	3	9
3	2	"4"	4	13
4	3	"5"	5	14
5	4	"5"	5	19
6	5	"8"	8	26
7	6	"9"	9	35
8	7	"6"	6	38

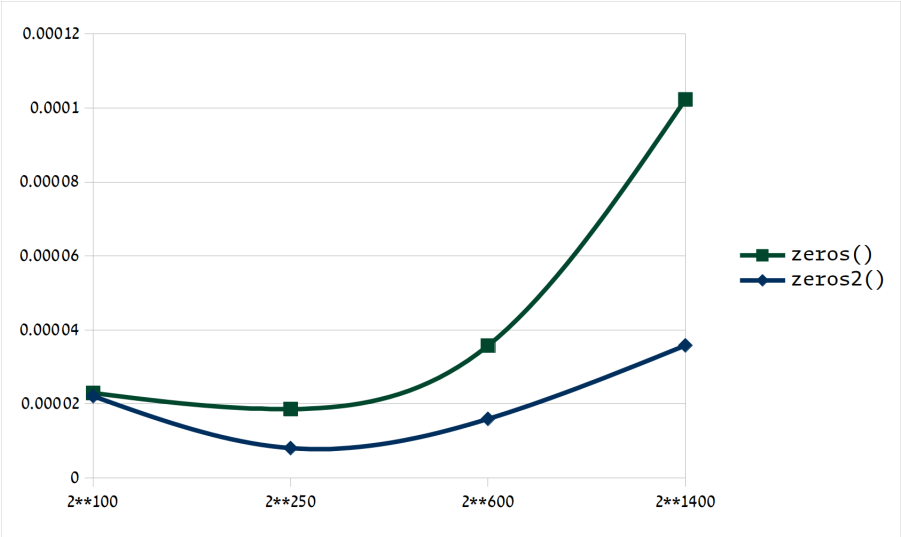
המשך בעמוד הבא

סעיף א'

מדידות הזמן בעבור שני הפתרונות הראשונים:

input	zeros()	zeros2()	output
2^{100}	$2.297 \cdot 10^{-5}s$	$2.211 \cdot 10^{-5}s$	6
2^{250}	$1.864 \cdot 10^{-5}s$	$8.073 \cdot 10^{-6}$	11
2^{600}	$3.577 \cdot 10^{-5}s$	$1.596 \cdot 10^{-5}s$	16
2^{1400}	$1.023 \cdot 10^{-4s}$	$3.582 \cdot 10^{-5}s$	31

נשים לב, שפרט עבור החישוב הראשון והשני (בהם משום מה המהירות קטנה עבור קלט גדול יותר), כמות הזמן תגדל כתלות בקלט. נתבונן בגרף המתאר את החישובים:



כלומר, עבור קלט שגדל באופן מעריכי, סיבוכיות הפונקציה zeros() תדמה להיות מעריכית, ו־zeros2() ליניארית (אם כי, בהתבוננות בקוד, נבין שאין זו ההתנהגות האמיתית בשאיפה לאינסוף).

סעיף ב'

להלן התוצאות בעבור הפתרון השלישי:

input	zeros3()	output
2^{100}	$1.404 \cdot 10^{-5}s$	6
2^{250}	$1.892 \cdot 10^{-6}s$	11
2^{600}	$2.301 \cdot 10^{-6}s$	16
2^{1400}	$3.56 \cdot 10^{-6s}$	31

כלומר, הפונקציה תחזיר בקביעות פתרונות מהירים בערך פי 10 מאלו של
כי הם ממומשים עם הגיון דומה) – כלומר, היא מהירה פי קבוע, אך לא אסימפטוטית, מ־zeros2().

סעיף ג'

לדוגמה, בעבור הקלטים $10^{100000} - 1, 10^{99999}$ (שניהם עם 999 ספרות) נקבל:

input	zeros()	zeros2()	zeros3()	output
$10^{100000} - 1$	$2.5975s$	$0.0325s$	$0.0251s$	0
10^{99999}	$2.6043s$	$0.0345s$	$0.0237s$	99999

נסיק, שהשינוי בזמן זניח, וייתכן שנגרם בגלל שגיאת מדידה. הסבר אפשרי לעליה מזערית בזמן במימושים של zeros() ו־zeros2(), הוא שייתכן ונגרם מהפעולה הנוספת של cnt=cnt+1 במספרים עם יותר אפסים, זאת תחת ההנחה שהשינוי לא נגרם משגיאת מדידה (כנראה שלא כי התוצאות הורצו מספר פעמים).

סעיף ד'

מסעיף (א), תוך הזנחת שאר הפעולות פרט ל־`cnt+=1` (מתוך הנחה שזה לא משפיע על סדר הגודל), ידוע שכמות הזמן שלוקח לחשב $\log_{10}(2^{1400})$ את השורה הזו, היא $3.58 \cdot 10^{-5}$ שניות. הלולאה שניתנה בסעיף הזה, תחזור על השורה הזו 2^{1000} פעמים, כלומר נחשב את היחס בין כמות הפעולות, ונמצא:

$$\frac{2^{1000}}{\log_{10}(2^{1400})} \cdot 3.58 \cdot 10^{-5} \approx 10^{294} s$$

(אך זהו רק חסם עליון, אם כי די הדוק בסדר הגודל). כאן, נוכל להבחין בהפרש הזמנים בין סיבוכיות לוגוריתמית (כמו סעיף (א)) לליניארית (הלולאה הזו) עבור מספרים בסדר גודל גבוה. זוהי, הסיבה שהלולאה בסעיף (א) רצה בזמן קצר משמעותית.

המשך בעמוד הבא

סעיף ב'

נשאלנו, מה השגיאה בפתרון שהוצע. הבעיה היא, שהפונקציה בודקת האם לכל תו ב־st1, כמות המופעים שלו שווה לזו של כמות המופעים ב־st2, אך לא להיפך – כלומר, אם קיים $a \in st2$ כך ש־$a \notin st1$, הפונקציה תחזיר True חרף העובדה שאין st1 אנגרמה ל־st2. לדוגמה, בעבור הקלט הבא:

```
>>> is_anagram_v2("", "a")
True
```

שהוא שגוי.

סעיף ג'

השוני בין המימוש בסעיף א' לבין הנוכחי, הוא שהמימוש בסעיף א' עובר על st1 ומשווה תווים בינו לבין st2, בעוד המימוש השני ישווה את סידורם. אסימפטוטית, המימוש הראשון ירוץ מהר יותר – עבור $n = \max\{len(st1), len(st2)\}$, סיבוכיות סעיף א' תהיה $O(n)$ ושל סעיף ג' $O(n \log n)$ (תחת ההנחה של שימוש בתנהגות אסימפטוטית של merge-sort את המימוש של הפונקציה sort).

הערה: המציאות היא, שמחיקת אינדקס מ־list, וכן חיפוש בתוך list, אינן פעולות שלוקחות $O(1)$, אלא $O(n)$, ומטור חשבוני נקבל ש־$O(n^2)$ היא הסיבוכיות לפתרון שלי בסעיף (א). אך, ניתן לממש אותו מחדש בסיבוכיות $O(n)$ (בממוצע) ע"י שימוש ב־hash table, לדוגמה. אך או אחרת לא הגענו לניתוח כזה בשלב הזה בקורס אז נתתי הסבר שונה קמעה.

שחר פרץ, 2024

נוצר באמצעות תוכנה חופשית בלבד