

Programování 2

4. cvičení, 10-3-2022

tags: Programovani 2, čtvrtek 1, čtvrtek 2

Farní oznamy

1. Tento text a kódy ke cvičení najdete v repozitáři cvičení na <https://github.com/PKvasnick/Programovani-2>.
2. **Domácí úkoly:** Zadáno minulý týden,
 - 2 povinné a 1 volitelná úloha - jenom kvůli počítání bodového limitu, můžete si vybrat libovolné 2 nebo všechny 3.
 - 2 termíny, 10. 3. pro 10 bodů, 17. 3. pro 5
 - dnes jen jedna poznámka později

Dnešní program:

- Kvíz
- Domácí úkoly
- Opakování: Načítání a zpracování posloupností
- Třídění

Na zahřátí

"Code is like humor. When you have to explain it, it's bad." – Cory House

Dobrý kód nepotřebuje mnoho komentářů, ale někdy potřebuje.

Co dělá tento kód

```
1 d = dict.fromkeys(range(10), [])
2 for k in d:
3     d[k].append(k)
4 d
```

`dict.fromkeys` je někdy dobrá náhrada `defaultdict` nebo `Counter`, ale takto inicializovat slovník není dobrý nápad.

Počítání věcí

- `defaultdict`
- `Counter`

Někdy ale takovéto speciální struktury nepotřebujeme.

Domácí úkoly

Dělicí bod posloupnosti

U této úlohy se nevyžaduje jednorůchodový algoritmus!

Pro danou posloupnost reálných čísel a_0, a_1, \dots, a_{n-1} najděte index j takový, že pro všechny indexy i $a_i < a_j$ pokud $i < j$, a $a_i > a_j$, pokud $i > j$. Jestliže takový index neexistuje, vrátíte -1 . Pokud má posloupnost více dělicích bodů, vraťte první zleva (nejmenší i). Jako řešení budou akceptovány pouze algoritmy s lineární náročností.

Vstupní posloupnost načtete z konzoly číslo po čísel, každé číslo na novém řádku. Posloupnost je ukončená řádkou s -1 , která nepatří do posloupnosti. Výsledek zapíšete na standardní výstup jako celé číslo následované (jedním) znakem nového řádku.

Příklad 1

Vstup:

-1 (prázdná posloupnost)

Výstup:

-1 (nenalezeno)

Příklad 2

Vstup:

1
-1

Výstup:

0

Příklad 3

Vstup:

1
1
1
2
3
3
3
-1

Výstup:

3

Příklad 4

Vstup:

3
3
2
1

1
-1

Výstup:

-1

Testy

Pro tuto úlohu je 7 testů: posloupnosti délky 0 a 1, malá posloupnost s dělícím bodem, posloupnosti s dělícím bodem na levém a pravém konci, posloupnost bez dělícího bodu, a *dlouhá* posloupnost s dělícím bodem.

Poznámky

Abyste ušetřili porovnání, uvědomte si, že posloupnost s dělícím bodem musí být částečně setříděná.

Opakování a pokračování:

Jednoduché algoritmy pro posloupnosti

Tady si procvičíme úplně jednoduché věci, zčásti také proto, abychom si zopakovali některé postupy, které využijete pro domácí úkoly.

```
1 from functools import reduce
2
3 def read_from_console():
4     while "-1" not in (line := input()):
5         yield float(line)
6     return
7
8 maximum = reduce(max, read_from_console, float("-inf"))
9 print(maximum)
```

Takovýto kód bude rychlý, protože cyklus se vykonává uvnitř funkce, a tedy běží v C a ne v Pythonu.

Složitější případy

- rozhodnout, zda je posloupnost čísel monotonní a jak (konstantní, rostoucí, neklesající, klesající, nerostoucí)
- v posloupnosti čísel nalézt druhou největší hodnotu a počet jejích výskytů
- v posloupnosti čísel určit délku nejdelšího souvislého rostoucího úseku
- v posloupnosti čísel určit počet různých hodnot
- v posloupnosti čísel nalézt souvislý úsek se součtem K (pro zadanou hodnotu K)
- v posloupnosti kladných čísel nalézt souvislý úsek se součtem K (pro zadanou hodnotu K)
- v posloupnosti čísel nalézt souvislý úsek s maximálním součtem.

Řešení

- Pro část úloh stačí implementovat funkci pro `reduce`.
- Implementovat stav

- V některých případech může být jednorůchodový algoritmus velice složitý

Průběžný výpočet střední hodnoty a standardní odchýlky

Chceme

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$$

spočítat průběžně, jak nám přicházejí nová data.

```
1  # Mean and variance on the fly
2  from math import sqrt
3  from random import gauss
4  from functools import reduce
5
6
7  def data_generator():
8      data = [gauss(0,1) for _ in range(10)]
9      yield from data # <---
10
11
12  class State:
13      def __init__(self, n, mean, sd2):
14          self.n = n
15          self.mean = mean
16          self.var = sd2
17
18
19  def update(s: State, x: float) -> State:
20      s.n += 1
21      diff = x - s.mean
22      s.mean += diff / s.n
23      s.var += (x - s.mean) * diff
24      return s
25
26
27  def report(s: State) -> tuple[int, float, float]:
28      return s.n, s.mean, sqrt(s.var/(s.n-1))
29
30
31  def main() -> None:
32      s = State(0, 0, 0)
33      s = reduce(update, data_generator(), s)
34      n, mu, sd = report(s)
35      print(f"{n} {mu:.5f} {sd:.5f}\n")
36      return
37
38
39  if __name__ == '__main__':
40      main()
41
```

Třídění

Insertion sort

Začínáme třídit z kraje seznamu, následující číslo vždy zařadíme na správné místo do již utříděné části.

```
1 def insertion_sort(b):
2     for i in range(1, len(b)):
3         up = b[i]
4         j = i - 1
5         while j >= 0 and b[j] > up:
6             b[j + 1] = b[j]
7             j -= 1
8         b[j + 1] = up
9     return b
```

Bucket sort

- Nahrubo si setřídíme čísla do přihrádek
- Setřídíme obsah přihrádek
- Spojíme do výsledného seznamu

```
1 def bucketSort(x):
2     arr = []
3     slot_num = 10 # 10 means 10 slots, each
4                   # slot's size is 0.1
5     for i in range(slot_num):
6         arr.append([])
7
8     # Put array elements in different buckets
9     for j in x:
10        index_b = int(slot_num * j)
11        arr[index_b].append(j)
12
13    # Sort individual buckets
14    for i in range(slot_num):
15        arr[i] = insertionSort(arr[i])
16
17    # concatenate the result
18    k = 0
19    for i in range(slot_num):
20        for j in range(len(arr[i])):
21            x[k] = arr[i][j]
22            k += 1
23    return x
```

To je docela ošklivý kód, uměli bychom ho vylepšit?
