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 včera,
    - 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

Dnešní program:

- Kvíz
- Domácí úkoly
- Opakování: Načítání a zpracování posloupností , matice
- BInární vyhledávání

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

```
d = dict.fromkeys(range(10), [])
for k in d:
    d[k] = k
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

Dělící bod posloupnosti

U této úlohy se nevyžaduje jednoprůchodový algoritmus!

Pro danou posloupnost reálných čísel $a_0, a_1, \ldots a_{n-1}$ 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 dělících 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:

Jednoprůchodové algoritmy pro posloupností

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

```
from functools import reduce

def read_from_console():
    while "-1" not in (line := input()):
        yield float(line)
    return

maximum = reduce(max, read_from_console, float("-inf"))
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.

Řešeni

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

- V některých případech může být jednoprůchodový algoritmus velice složitý

Průběžný výpočet střední hodnoty a standardní odchýlky

Chceme

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

spočítat průběžně, jak nám přicházejí nová data.

```python
# Mean and variance on the fly
from math import sqrt
from random import gauss
from functools import reduce


def data_generator():
    data = [gauss(0,1) for _ in range(10)]
    yield from data # <---


class State:
    def __init__(self, n, mean, sd):
        self.n = 0
        self.mean = 0
        self.var = 0


def update(s: State, x: float) -> State:
    s.n += 1
    diff = x - s.mean
    s.mean += diff / s.n
    s.var += (x - s.mean) * diff
    return s


def report(s: State) -> tuple[int, float, float]:
    return s.n, s.mean, sqrt(s.var/(s.n-1))


def main() -> None:
    s = State(0, 0, 0)
    s = reduce(update, data_generator(), s)
    n, mu, sd = report(s)
    print(f"{n} {mu:.5f} {sd:.5f}\n")
    return


if __name__ == '__main__':
    main()

```

Třídění

Insertion sort

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

```
def insertion_sort(b):
    for i in range(1, len(b)):
        up = b[i]
        j = i - 1
        while j >= 0 and b[j] > up:
            b[j + 1] = b[j]
            j -= 1
        b[j + 1] = up
    return b
```

Bucket sort

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

```
def bucketSort(x):
    arr = []
    slot_num = 10 # 10 means 10 slots, each
                  # slot's size is 0.1
    for i in range(slot_num):
        arr.append([])

    # Put array elements in different buckets
    for j in x:
        index_b = int(slot_num * j)
        arr[index_b].append(j)

    # Sort individual buckets
    for i in range(slot_num):
        arr[i] = insertionSort(arr[i])

    # concatenate the result
    k = 0
    for i in range(slot_num):
        for j in range(len(arr[i])):
            x[k] = arr[i][j]
            k += 1
    return x
```

To je docela ošklivý kód, uměli bychom ho vylepšit?
