Řešení domácích úkolů - 5. týden

Dělící bod posloupnosti

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.

Analýza

Podle definice ze zadání hledáme index i v posloupnosti, pro který platí

$$max(a_{j|j<i}) < a_i < min(a_{j|j>i})$$

Takže bychom měli dělat zhruba (až na okrajové případy) toto:

```
...
n = len(a)
for i in range(n):
    if max(a[:i]) < a[i] < min(a[i+1:]):
        print(i)
        break
```

Toto má ale složitost $O(n^2)$, a určitě to jde zlepšit, stačí si uvědomit, že počítáme takovéto věci:

$$\max(a_0)$$
$$\max(a_0, a_1)$$
$$\max(a_0, a_1, a_2)$$
$$\dots$$

přičemž zjevně platí

$$\max(a_0, a_1, \dots, a_k, a_{k+1}) = \max(\max(a_0, a_1, \dots, a_k), a_{k+1})$$

a tedy můžeme všechny tyto veličiny spočíst v čase $O(n)$. Podobně to platí z druhé strany,

$$\min(a_{n-k-1}, a_{n-k}, \dots, a_{n-2}, a_{n-1}) = \min(a_{n-k-1}, \min a_{n-k}, \dots, a_{n-2}, a_{n-1})$$

Vidíme tedy, že najít dělící bod posloupnosti lze v čase $O(n)$. Protože na začátku potřebujeme spočítat kumulativní maxima a minima, potřebujeme celou posloupnost v paměti.

Můžeme také zkusit sekvenční řešení - tedy řešení bez načtení celé posloupnosti do paměti. Stav, který potřebujeme aktualizovat v době načítání, je:

- **seznam kandidátů** - ten musíme propagovat až do konce načítání, protože nemůžeme prohlásit nějakou pozici v posloupnosti za dělící bod předtím, než jsme viděli celou posloupnost. Seznam proto, že cestou můžeme nalézt několik dělících bodů, i když vracet budeme pouze první. Posloupnost kandidátů bude rostoucí, následující dělící bod musí mít hodnotu větší než předchozí. Do seznamu kandidátů zapíšeme načtenou hodnotu, pokud je větší než průběžné maximum. Pokud tomu tak není, odstraníme ze seznamu kandidátů všechny položky, jejichž hodnota není menší než právě načtená hodnota.
- **průběžné maximum** - aktualizujeme obvyklým způsobem při načtení hodnoty, větší než aktuální maximum.

Vzorové řešení

Varianta s načtením posloupnosti

- pole s (levými) kumulativními maximy a (pravými) kumulativními minimy definujeme "s převisem", tedy `cum_max[i] = max(a[0], a[1], ... a[i-1])` a `cum_min[i] = min(a[i+1], a[i+2], ... a[n-1])`:

```
cum_max[0] = a[0]
cum_max[i] = max(cum_max[i-1], a[i-1])
cum_min[-1] = a[-1]
cum_min[i] = min(cum_min[i+1], a[i+1])
```

- Výhoda takovéhoto uspořádání je v tom, že se nám do vyhledávání nepletou okrajové případy. Abychom dostali jednoduchý kód, po vyřešení okrajových případů voláme `sys.exit()`.

```
import sys

a = []
while (x := int(input())) != -1:
    a.append(x)

n = len(a)

if n == 0:
    print(-1)
    sys.exit()

if n == 1:
    print(0)
    sys.exit()

cum_max = [a[0]] * n
cum_min = [a[-1]] * n

for i in range(1,n):
```

```
    cum_max[i] = max(cum_max[i-1], a[i-1])

for i in range(n-2, -1, -1):
    cum_min[i] = min(cum_min[i+1], a[i+1])

if a[0] < cum_min[0]:
    print(0)
    sys.exit()

for i in range(1, n-2):
    if cum_max[i] < a[i] < cum_min[i]:
        print(i)
        sys.exit()

if a[-1] > cum_max[-1]:
    print(n-1)
    sys.exit()

print(-1)
```

Průběžná varianta

Protože si musíme pamatovat nejen hodnoty kandidátů ale i jejich pozici, bude seznam kandidátů tvořen dvojicemi (index, hodnota). Abychom si zjednodušili kód, do seznamu zapíšeme inicializační položku (-1, -1.0e-10) a maximum nastavíme také na -1.0e-10. Pro praktické účely by takováto inicializace měla postačovat, pokud hrozí, že naše posloupnost bude obsahovat obrovská záporná čísla, je potřeba inicializaci upravit.

Kód řešení je pak překvapivě jednoduchý:

```
MIN_VALUE = -1.0e-10
candidates = [[-1, MIN_VALUE]]
maximum = MIN_VALUE

index = 0
while (n := int(input())) != -1:
    if n > maximum:
        maximum = n
        candidates.append([index, n])
    else:
        while candidates[-1][1] >= n:
            candidates.pop()
    index += 1

if len(candidates) == 1:
    print(-1)
else:
    print(candidates[1][0])
```

Obvyklé problémy

Tato úloha bývá vnímána jako těžká. Většina problémů pochází z nepochopení zadání. Je proto potřeba dobře si zadání přečíst, a pak začít s řešením, vycházejícím důsledně z definice. To bude $O(n^2)$, a alespoň máte něco v ruce, než začnete přemýšlet, jak řešení zefektivnit. Asi nejdůležitější je uvědomit si, že musíte vidět celou posloupnosti, než můžete o některé položce říct, že je dělícím bodem.