Řešení domácích úkolů - 5. týden

Medián

Pro posloupnost n čísel, načtených obvyklým způsobem ze standardního vstupu (jedno číslo na řádek, zakončeno -1), vypočítejte medián posloupnosti a vypište ho na standardní výstup.

Medián množiny čísel je takové číslo, že náhodně vybrané číslo z množiny má stejnou pravděpodobnost, že bude větší a že bude menší než medián.

Víc technicky, máme-li setříděnou množinu n čísel $x_0, x_2, \ldots, x_{n-1}$, medián bude $x_{n//2}$ pro liché n a $\frac{1}{2} \times (x_{n//2-1} + x_{n//2})$ pro sudé n - je to tedy prostřední hodnota pro liché n a průměr dvou prostředních hodnot pro n sudé.

Příklad 1:

Vstup:

1\

2\

3\

-1

Výstup:

2 (nebo 2.0)

Příklad 2:

Vstup:

1\

2\

3\

4\

-1

Výstup:

2.5

Poznámky:

- Nemusíte se starat co váš kód vypočte a vypíše pro prázdnou množinu. Toto chování nebude testováno. Váš kód se ale musí chovat slušně pro množinu velikosti 1.
- Pro úlohu máme 3 testy s následujícími vstupy:
    - 31 celých čísel 0 až 9 v náhodném pořadí
    - 1 číslo
    - 200 celých čísel od 100 do 999.
- Toto je čeština a ne Python, takže od a do b znamená *včetně* a i b.

Řešení

Analýza Načítáme sérii čísel, která můžou být libovolná, kladná, záporná i 0. Pouze číslo -1 je výjimečné, protože při jeho objevení na vstupu načítání posloupnosti končí.

Pro nalezení mediánu potřebujeme uspořádanou posloupnost, tedy potřebujeme celou posloupnost uložit do paměti. Výpočet je pak snadný a po seřazení posloupnosti stačí vybrat příslušné prostřední členy.

Vzorové řešení

```
values = []

while (v := int(input())) != -1:
    values.append(v)

values.sort()
mid = len(values) // 2
if len(values) % 2 == 0:
    print(f"{0.5 * (values[mid-1] + values[mid])}")
else:
    print(f"{values[mid]}")
```

Alternativní řešení

Alternativní řešení se omezovala na různě chytré způsoby spočtení mediánu pro případ sudého a lichého počtu členů. Zde verze, která dostává kytičku za eleganci:

```
values = []

while (v := int(input())) != -1:
    values.append(v)

values.sort()
mid = len(values) // 2
median = 0.5 * (values[mid] + values[~mid])
print(median)
```

Magie tady spočívá ve využití operátora bitové negace `~`, který pro celé `x` dává `~x == -x - 1` a to je právě to, co potřebujeme. Tu samou službu by nám prokázalo prostší `median = 0.5 * (values[mid] + values[-mid-1])`, akorát to nevypadá tak svůdně.

Skutečně alternativní řešení u této úlohy se zakládá na pozorování, že pro nalezení mediánu nepotřebujeme mít setříděnou celou posloupnost: u hodnot, které jsou větší, resp. menší než medián nemusíme znát jejich přesné pořadí. Díky tomu můžeme získat významnou úsporu - snížit náročnost z $O(n \log n)$ na $O(n)$. Jak to udělat si ukážeme ve cvičení.