resitev

January 28, 2024

0.1 Day 09: Encoding Error

(Povezava na nalogo)

Končno se začenjajo spodobne naloge, pri katerih moraš malo pomisliti, preden začneš programirati!

0.2 Prvi del

Če preskočimo zgodbico: imamo zaporedje števil. Zanima nas, katero je prvo število, ki ga ne moremo dobiti kot vsoto dveh izmed zadnjih 25 števil pred njim. Preverjati je potrebno le števila od petindvajsetega naprej.

Branje podatkov je končno trivialno.

[1]:
```
s = [int(x) for x in open("input.txt")]
```

0.2.1 Naivna rešitev

Naivna rešitev je, da napišemo funkcijo `valid(n, t)`, ki preveri, ali je število `n` vsota kakega para števil iz seznama `t`. Potem pokličemo to funkcijo za vsa števila od 25-ega naprej.

Če že pišemo naivno rešitev, jo napišimo spodobno.

[2]:
```
def valid(n, t):
    for i, x in enumerate(t):
        for y in t[:i]:
            if x + y == n:
                return True
    return False


for i, x in enumerate(s[25:]):
    if not valid(x, s[i:i + 25]):
        first_invalid = x
        break

print(first_invalid)
```

```
85848519
```

Funkcija `valid` gre prek seznama `t`. Potrebujemo vrednost (`x`) in indeks (`i`), zato da gremo z `y` le prek prvih `i` elementov `t`, torej le do `x`-a. Če je vsota `x + y` enaka `n`, je število veljavno. Če takega para ni, ni.

V zanki, ki sledi, gre `x` od elementa z indeksom 25 naprej, `i` pa od `0`. Veljati mora, da je `x` vsota dveh elementov od `i`-tega do `i+25`-ega (`s[i + 25]` pa je ravno `x`).

Vse skupaj se da skrajšati.

[3]:
```
s = [int(x) for x in open("input.txt")]

def valid(n, t):
    return any(x + y == n for i, x in enumerate(t) for y in t[:i])

first_invalid = next(x for i, x in enumerate(s[25:]) if not valid(x, s[i:i +␣
↪25]))
```

Kdor hoče, lahko to očitno stlači v eno samo vrstico. Jaz ne bi.

0.2.2 Boljša rešitev

Najprej je potrebno opozoriti, da ta rešitev precejkrat kopira seznam `s` - namreč vsakič, ko naredimo rezino. Temu bi se izognili tako, da bi uporabili `numpy`.

A to ni glavni problem. Najbolj moteča je funkcija `valid`: pri 25 predhodnikih, se obrne 25*24/2 = 300-krat. Za vsako število. Gre boljše? Se lahko domislimo rešitve, pri kateri čas reševanja ne bi naraščal s kvadratom velikost okna?

Takole: štejemo, na koliko načinov lahko dobimo vsako število. Vsakič, ko dodamo število, dobeležimo `1` pri vseh vsotah tega števila in števil iz prejšnjih 24. Istočasno pa zaradi tega eno število izpade iz okna in odštejemo `1` pri vseh vsotah izpadlega števila in taistih 24.

[4]:
```
from collections import defaultdict

def get_invalid(s, width):
    sum_count = defaultdict(int)

    def update_counts(x, d, args):
        for y in args:
            sum_count[x + y] += d

    for i in range(width):
        update_counts(s[i], 1, s[:i])

    for i in range(width, len(s)):
        x = s[i]
        if not sum_count[x]:
            return x
        window = s[i - 24:i]
        update_counts(s[i - 25], -1, window)
```

```
        update_counts(x, 1, window)

first_invalid = get_invalid(s, 25)

print(first_invalid)
```

```
85848519
```

0.3 Drugi del

V drugem delu je potrebno poiskati podzaporedje znotraj podanega zaporedja, pri katerem je vsota elementov enaka prvemu neveljavnemu številu. Rezultat, ki ga je potrebno vrniti, je vsota najmanjšega in največjega števila v tem oknu.

To bomo rešili tako, da se bomo vlekli okno spremenljive širine prek seznama. Kadar je vsota števil prevelika, povečamo spodnjo mejo, da se vsota zmanjša. Kadar je premajhna, povečamo zgornjo, da se vsota zmanjša.

[5]:
```
fr = to = 0
v = 0
while v != first_invalid:
    if v > first_invalid:
        v -= s[fr]
        fr += 1
    else:
        v += s[to]
        to += 1

print(min(s[fr:to]) + max(s[fr:to]))
```

```
13414198
```