# resitev

January 28, 2024

## 1 Day 6: Tuning Trouble

(besedilo naloge)

Dobimo nek niz, recimo `mjqjpqmgbljsphdztnvjfqwrcgsmlb`. Iščemo zaporedje štirih znakov, med katerimi ni nobenega ponovljenega znaka. Zanima nas, koliko znakov tega niza moramo prebrati, da naletimo na takšno četvorko.

Drugi del je podoben, le da namesto četvorke iščemo štirinajsterico, ki ne vsebuje ponovljenega znaka.

[3]:
```
data = open("example.txt").read().strip()
```

Gremo čez možne indekse, `for i in range(n, len(data))`, in opazujemo nize `data[i - n:i]`, kjer je `n` v prvem delu `4` in v drugem `14`. Vsi njegovi elementi so različni, če množica njegovih elementov vsebuje točno `n` elementov. Če je tako, izpišemo `i`. Rešitev naloge (obeh delov) je torej

[17]:
```
for n in (4, 14):
    for i in range(n, len(data)):
        if len(set(data[i - n:i])) == n:
            break
    print(i)
```

```
7
19
```

Bolj zanimiva različica iste ideje sestavi generator, ki bi - če bi ga pustili - zgeneriral vse `i`-je, pri katerih imamo `n` različnih znakov, `(i for i in range(len(data)) if len(set(data[i-n:i])) == n)`. Zanima nas samo prvi: dobimo ga z `next`.

[18]:
```
for n in (4, 14):
    print(next(i for i in range(len(data)) if len(set(data[i-n:i])) == n))
```

```
7
19
```

Ta rešitev je sicer manj pregledna, časovno pa nič slabša od gornje, saj v resnici ne zgenerira vseh takšnih `n` temveč samo prvega.