

## Day 15: Rambunctious Recitation

(Povezava na nalogo)

Naloga ni nek programerski presežek, zahteva pa, da se spomnimo, česa ne smemo početi s seznamei, temveč zahteva slovarje.

Zaporedje se začne s številkami

```
initial = 0, 8, 15, 2, 12, 1, 4
```

Vsak naslednji člen dobimo iz prejšnjega:

- če se je prejšnji člen pojavil prvič, je naslednji člen 0,
- sicer pa je prejšnji člen enak številu korakov, ki so minili od prejšnje ponovitve tega člena.

Prvi del naloge sprašuje po 2020-tem členu, drugi pa po 30-milijontem.

### Rešitev

Nevarna skušnjava, ki pa ji lahko podležejo le manj izkušeni, je shranjevati zaporedje in takrat, ko nas zanima, koliko časa je minilo od zadnje pojavitve števila, le-to v resnici iskati po zaporedju.

Zaporedja v resnici ni potrebno shranjevati, pač pa si moramo za vsako število zapomniti, kdaj se je zadnjič pojavilo.

Vsa stvar je nekoliko zavozlana: prejšnje število smemo dodati šele po tem, ko že izračunamo naslednje. Ko razvozlamo, pa je rešitev kar kratka.

```
last_spoken = {}
```

```
next = None
```

```
for turn in range(2021):
    number = next
    next = initial[turn] if turn < len(initial) else turn - last_spoken.get(number, turn)
    if number is not None:
        last_spoken[number] = turn
```

```
number
```

```
289
```

V drugem delu nadaljujemo odtod naprej.

```
for turn in range(2021, 30000001):
    number = next
    next = turn - last_spoken.get(number, turn)
    last_spoken[number] = turn
```

```
number
```

1505722

Trik je, očitno, da nam v drugem delu ni več potrebno preverjati, ali vzeti naslednji člen iz začetnega zaporedja. Zaradi tega je program za kako tretjino hitrejši.

Seveda bi lahko že začetni del razdelili v dve zanki, vendar ... vse skupaj je potem še bolj sitno zavozlano. Tako je bilo lažje.