

Day 5: Binary Boarding

(Povezava na nalogo)

Naloga je res lahka. Zanimiva pa je zaradi lepega trika v drugem delu.

Sedež na letalski karti je opisan z zaporedjem črk F, B, L in R. Prvi del opisuje vrsto:

For example, consider just the first seven characters of FBFBFFRLR:

Start by considering the whole range, rows 0 through 127.

F means to take the lower half, keeping rows 0 through 63.

B means to take the upper half, keeping rows 32 through 63.

F means to take the lower half, keeping rows 32 through 47.

B means to take the upper half, keeping rows 40 through 47.

B keeps rows 44 through 47.

F keeps rows 44 through 45.

The final F keeps the lower of the two, row 44.

drugi pa sedež znotraj te vrste

Start by considering the whole range, columns 0 through 7.

R means to take the upper half, keeping columns 4 through 7.

L means to take the lower half, keeping columns 4 through 5.

The final R keeps the upper of the two, column 5.

Vrniti je potrebno vsoto številko vrste, pomnožene z 8, in številke sedeža.

Pretvarjanje v število

Ni potrebno biti posebej pameten, da uvidimo, da dobimo številko vrste tako, da zaporedje FBFBFF spremenimo v 0101100 (F spremenimo v 0 in B v 1) in ga interpretiramo kot dvojiško število. Pythonov `int` poleg niza prijazno sprejme še en argument, namreč osnovno številskega sistema,

```
int("0101100", 2)
```

```
44
```

Drugi del je enako dolgočasen, le da L postane 0 in R 1. Še več, ker je potrebno v končnem rezultatu pomnožiti vrstico z 8, je to isto, kot če bi v prvotnem nizu pretvorili vse črke, kot je treba, in kar vse skupaj pretvorili v `int`.

```
x = "FBFBFFRLR"
```

```
int(x.replace("F", "0").replace("B", "1").replace("R", "1").replace("L", "0"), 2)
```

```
357
```

Podatke preberemo tako, da to storimo z vsako vrstico datoteke.

```
seats = [int(x.replace("F", "0").replace("B", "1").replace("R", "1").replace("L", "0"), 2)
         for x in open("input.txt")]
```

Prvi del: največja številka sedeža

Brez komentarja:

```
max(seats)
```

885

Drugi del: manjkajoči sedež

Zgodba pravi, da prvih toliko in toliko števil ne obstaja. Naloge je odkriti naš sedež (to je, sedež, ki je še prost) med najmanjšo in največjo številko sedeža.

```
mi, ma = min(seats), max(seats)
```

Ena ideja so množice: od množice vseh sedežev, `set(range(mi, ma + 1))` odštejemo množico zasedenih sedežev, `set(seats)`. Dobljena množica bo (predvidoma) imela le en element; ven ga izvlečemo s `pop`.

```
(set(range(mi, ma + 1)) - set(seats)).pop()
```

623

Zvitejša je tale: od vsote števil vseh možnih sedežev odštejemo vsoto števil vseh zasedenih sedežev.

Vsota števil vseh sedežev do (vključno) `ma` je `ma * (ma + 1) // 2`. Ker prvih `mi` (brez `mi`) sedežev manjka, moramo od tega odšteti vsoto sedežev do `mi`-tega (brez `mi`-tega), `mi * (mi - 1) // 2`. Od tega odštejemo vsoto vseh sedežev.

```
(ma * (ma + 1) - mi * (mi - 1)) // 2 - sum(seats)
```

623

Dodatek: ročno pretvarjanje

Če Pythonov `int` ne bi znal pretvarjati števil v dvojiškem zapisu, bi morali pretvarjati sami.

Recimo tako

```
x = "FBFBFBFRLR"
```

```
a = 0
for c in x:
    a *= 2
    if c in "BR":
        a += 1
```

```
a
```

```
357
```

Če vemo, da je `True` enak 1 in `False` enak 0, to skrajšamo v

```
a = 0
for c in x:
    a = 2 * a + (c in "BR")
```

```
a
```

```
357
```

In če znamo uporabljati `reduce` in `lambda`, iz tega naredimo

```
from functools import reduce
```

```
reduce(lambda a, c: 2 * a + (c in "BR"), x, 0)
```

```
357
```

Celotno datoteko tako preberemo s

```
seats = [reduce(lambda a, c: 2 * a + (c in "BR"), x.strip(), 0) for x in open("input.txt")]
```

`strip` je potreben, ker je na koncu vrstice še `\n`, ki bi nam vsa števila še enkrat pomnožil z 2.