resitev-2

January 28, 2024

0.1 Rešitev

Tole imamo še od prej.

[1]:
```
import numpy as np
import re

instr = np.array([[line.startswith("on")] +
                  [int(x) for x in re.findall("-?\d+", line)]
                  for line in open("example2.txt")])

instr[:, [2, 4, 6]] += 1
```

Zdaj zberimo vse možne koordinate `x`, vse možne koordinate `y` in vse možne koordinate `z`. Koordinate `x` najdemo v stolpcih 1 in 2. Vzemimo takšno matriko, sploščimo jo in sestavimo množico njenih elementov. Podobno za `y` in `z`.

[2]:
```
xs = set(instr[:, 1:3].flatten())
ys = set(instr[:, 3:5].flatten())
zs = set(instr[:, 5:7].flatten())
```

Os `x` bo razrezana na vseh številkah, ki se pojavijo v `xs`. Uredimo te meje in jih zložimo v tabele. Da so v tabelah, nam bo prišlo prav malo kasneje.

[3]:
```
xa = np.array(sorted(xs))
ya = np.array(sorted(ys))
za = np.array(sorted(zs))
```

Zdaj pa pripravimo slovarje, s katerimi bomo za vsako mejo izvedeli njihovo zaporedno številko.

[4]:
```
xs = {x: i for i, x in enumerate(xa)}
ys = {x: i for i, x in enumerate(ya)}
zs = {x: i for i, x in enumerate(za)}
```

Dimenzije tabele, ki predstavlja stanje reaktorja bodo enake številu blokov. `1` nam ni potrebno prišteti, ker zadnje število vedno predstavlja le gornjo mejo; bloka s to številko nikoli ne bomo prižgali ali ugasnili.

[5]:
```
reactor = np.zeros((len(xs), len(ys), len(zs)), dtype=bool)
```

Zdaj pa gremo čez navodila ter prižigamo in ugašamo bloke v podanih intervalih (številk).

[6]:
```
for on, x0, x1, y0, y1, z0, z1 in instr:
    reactor[xs[x0]:xs[x1], ys[y0]:ys[y1], zs[z0]:zs[z1]] = on
```

Koordinate iz navodil preslikujemo v številke blokov prek slovarjev `xs`, `ys` in `zs`.

Stanje reaktorja zdaj poznamo: vemo, kateri bloki so prižgani. Da ugotovimo število prižganih celic, pa potrebujemo število celic v vsakem bloku. To dobimo takole:

[7]:
```
volumes = \
    (xa[1:] - xa[:-1])[:, None, None] \
    * (ya[1:] - ya[:-1])[:, None] \
    * (za[1:] - za[:-1])
```

`xa[1:] - xa[:-1]` so razmiki med zaporednimi koordinatami v `xs` - to so širine blokov po smeri `x`. Podobno za ostale dimenzije. Dimenziji `x` dodamo še dve osi, `y` še eno in to zmnožimo skupaj. Zdaj bo `volumes[n][m][k]` dimenzija bloka, ki se začne na `n`-ti koordinati `x`, `m`-ti koordinati `y` in `k`-ti koordinati `z`. Torej tistega bloka, katerega stanje opisuje `reactor[n][m][k]`. Prostornine vseh prižganih blokov dobimo z `volumes[reactor[:-1, :-1, :-1]]`. `-1` moramo odšteti zaradi gornjih mej (spet!). Očitno je `xa[1:] - xa[:-1]`, kolikor je dolg `volumes` za 1 krajši od `len(xs)`, kolikor je dolg reaktor.

Rezultat, ki ga iščemo, je torej

[8]:
```
np.sum(volumes[reactor[:-1, :-1, :-1]])
```

[8]:
```
2758514936282235
```

Za preglednost in občutek je tu še celotna rešitev.

[9]:
```
xs = set(instr[:, 1:3].flatten())
ys = set(instr[:, 3:5].flatten())
zs = set(instr[:, 5:7].flatten())

xa = np.array(sorted(xs))
ya = np.array(sorted(ys))
za = np.array(sorted(zs))

xs = {x: i for i, x in enumerate(xa)}
ys = {x: i for i, x in enumerate(ya)}
zs = {x: i for i, x in enumerate(za)}

reactor = np.zeros((len(xs), len(ys), len(zs)), dtype=bool)
for on, x0, x1, y0, y1, z0, z1 in instr:
    reactor[xs[x0]:xs[x1], ys[y0]:ys[y1], zs[z0]:zs[z1]] = on

volumes = \
    (xa[1:] - xa[:-1])[:, None, None] \
```

```
    * (ya[1:] - ya[:-1])[:, None] \
    * (za[1:] - za[:-1])

np.sum(volumes[reactor[:-1, :-1, :-1]])
```

[9]: 2758514936282235

Razen imenitnega zaključka z računanjem prostornin je to bolj vaja iz razmišljanja in Pythona kot iz `numpy`-ja.