resitev-checkpoint

January 28, 2024

1 Day 22: Crab Cups

(Povezava na nalogo)

Imamo oštevilčene kozarce, postavljene v krog. Razpored števil in kozarec, pri katerem začnemo, sta v podatkih. V primeru na sliki je določeno, da začnemo pri kozarcu 3.

V vsakem koraku naredimo tole.

- Vzamemo tri kozarce, ki sledijo trenutnemu v smeri urinega kazalca (če je trenutni kozarec 3, so to kozarci 8, 9 in 1).
- Poiščemo kozarec, katerega številka je za 1 manjša od trenutnega. V tem primeru je to kozarec 2. Če bi bil to ravno kateri izmed treh kozarcev, zmanjšamo številko še za 1 (in po potrebi še in po potrebi še, tako da pridemo do enega izmed kozarcev, ki so še v krogu). Če izberemo kozarec 0 (ki ne obstaja), vzamemo kozarec z najvišjo številko (v tem primeru 9).
- Odvzete tri kozarce postavimo za ta kozarec. V gornjem primeru postavimo 8, 9 in 1 za 2, torej med 2 in 5.
- Naslednji trenutni kozarec je kozarec, ki sledi trenutnemu. V gornjem primeru je to 2 (ker smo 8, 9 in 1 odstranili).

1.1 Prvi del

Gornji recept ponovimo stokrat. Kakšen je vrstni red kozarcev od 1 naprej?

Da se ne hecamo brez potrebe s tem, ali se trije kozarci, ki sledijo trenutnemu, že ovijajo okrog, bomo vrteli seznam: trenutni element bo vedno ničti. Ko gremo na naslednji element, bomo v resnici dali trenutni element na konec.

Dobimo takšen program.

[ ]:
```
initial = "389125467"
state = list(map(int, initial))

for round in range(100):
    current, removed, state = state[0], state[1:4], state[4:]
    place = current - 1 or 9
    while place in removed:
        place = place - 1 or 9
    pi = state.index(place) + 1
    state[pi:pi] = removed
    state.append(current)
```

```
pos = state.index(1)
state = state[pos + 1:] + state[:pos]
print("".join(map(str, state)))
```

V `state` prepišemo števila, pretvorjena v `int`-e. Nato v zanki vzamemo prvo število, naslednja tri in ostala. Poiščemo mesto za vstavljanje; `place` bo trenutni - 1, oziroma `9`, če bi trenutni s tem postal 0. V zanki nato poskrbimo za primer, ko je `place` eden od odstranjenih kozarcev.

Nato odkrijemo, kjer v seznamu je kozarec, za katerim morajo biti trije kozarci, ki jih premikamo (`pi`). V mesto med `pi` in `pi` - torej točno za `pi`, damo odstranjene kozarce, na konec pa še trenutnega. Ne spreglejmo, da `state` pred temu prirejani vsebuje samo preostale kozarce, brez prvega (trenutnega) in tistih treh, ki mu sledijo.

Po stotih potezah poiščemo kozarec 1. Seznam prevrtimo tako, da imamo najprej kozarca 1 do konca in nato kozarce od začetka do tega kozarce. Števila pretvorimo v števke, združimo v niz in izpišemo.

1.2 Drugi del

Slaba novica: kozarcu 9 sledijo še kozarci od 10 do milijon. Poleg tega ne naredimo sto korakov temveč deset milijonov korakov.

Gornji program nima šans. Lahko ga prepišemo v numpy, pa se bo izvajal uro ali dve. Dodamo trik ali dva, pa bo morda dvakrat ali trikrat hitrejši.

Tu je potrebno uporabiti drugačen pristop. Sprogramirali ga bomo na tem, krajšem primeru, potem pa ga dopolnili do milijonov.

Osnovni problem gornjega programa je, da kopira sezname v nov seznam. Tudi če se temu izognemo in premikamo števila znotraj nekega seznama, jih je še vedno potrebno premikati. In s premikanjem milijona števil je veliko dela. Tudi če ne vrtimo seznama, tako da je trenutni element vedno na začetku, ne bomo nič na boljšem, saj je treba premikati elemente seznama, ko odstranjujemo in vstavljamo te tri elemente.

Potrebujemo nekaj, kar Python, kolikor vem, ni vdelano v Python: dvojno povezani seznami.

Devet kozarcev bomo predstavili s seznamom dolžine 10; prvi element bo za okras. No, za praktičnost. Element 5, recimo, vsebuje terko (2, 4), ki pove, da se kozarec s številko 5 nahaja med kozarcema 2 in 4.

To nam poenostavi prestavljanje kozarcev: ko prestavimo tri kozarce, je potrebno le prevezati vse puščice, ki kažejo na njih in iz njih.

Najprej sestavimo tabelico, kot je gornja.

[ ]:
```
initial = "389125467"
state = list(map(int, initial))

circle = [None] * (len(state) + 1)
for c, d, e in zip(state[-1:] + state, state, state[1:] + state[:1]):
    circle[d] = (c, e)
```

[ ]:
```
circle
```

[ ]:
```

```