resitev

January 28, 2024

Obvezni del naloge bo zelo lahek za tiste, ki se je lotijo pametno. In težak za tiste, ki ne. Dodatni del pa bo zelo lahek samo za tiste, ki znajo kar zelo dobro programirati.

0.1 Obvezni del

Spet imamo isto sliko marsovskih ladij. Zadnjič smo napisali funkcijo, ki vrne seznam zunanjih krogov. Zdaj pa bi radi množico zunanjih krogov. Potem pa nas zanima tole: če odstranimo te kroge, kateri krogi so zunanji potem? Tako dobimo novo množico zunanjih krogov. Stvar ponovimo: odstranimo kroge iz drugega nivoja in zdaj so zunanji neki tretji krogi.

Napiši funkcijo `cebula(krogi)`, ki prejme takšen seznam krogov, kot smo jih imeli prejšnji teden, in vrne seznam množic krogov po nivojih.

Za kroge s slike vrne seznam

```
[
 {(428, 89, 63.2), (164.4, 136.8, 50.8), (282.8, 71.5, 45.6),
  (59.2, 182.8, 50.8), (259.9, 186, 47.6), (391, 229.4, 58.4),
  (150.3, 245.5, 50.8), (88.6, 44.3, 37.5)},

 {(99.6, 43.1, 17.2), (371.6, 233.6, 10.6), (242.8, 187.3, 7.5),
  (293.7, 187.3, 7.5), (455.4, 66.5, 12.4), (449.5, 99.6, 13.6),
  (139.6, 138, 10.6), (69.8, 46.5, 10.6), (398.1, 95.5, 13),
  (408.7, 210.5, 8.9), (259.8, 187.3, 7.5), (185, 138, 10.6),
  (225.8, 187.3, 7.5), (267.4, 51.7, 17.2), (276.7, 187.3, 7.5),
  (144.3, 243.6, 38.8)},

 {(267.4, 51.7, 10.6), (127.3, 245.5, 7.5), (161.3, 245.5, 7.5),
  (99.6, 43.1, 10.6)}],
```

Zunanjih krogov je 8. Ko odstranimo te, imamo 16 zunanjih krogov (pretežno zaradi avionskih oken). Ko odstranimo te, ostanejo še 4 zunanji krogi.

0.1.1 Rešitev

Vzemimo funkcijo `vsebovani`, ki smo jo pisali za prejšnjo domačo nalogo. Odstranimo slovar `notranji`, saj ga ne potrebujemo. Tako dobimo:

[1]:
```
from collections import defaultdict

def vsebovanost(krogi):
```

```
    zunanji = defaultdict(list)
    for krog0 in krogi:
        for krog1 in krogi:
            x0, y0, r0 = krog0
            x1, y1, r1 = krog1
            if r0 > r1 and (x1 - x0) ** 2 + (y1 - y0) ** 2 < r0 ** 2:
                zunanji[krog1].append(krog0)
    return zunanji
```

Funkcija je nerodna zaradi poimenovanja: moramo si ponavljati, da se slovar `zunanji` imenuje `zunanji`, ker nam `zunanji[krog]` pove, kateri krog je zunanji krog kroga `krog`. Ključi tega slovarja torej niso zunanji temveč notranji krogi.

Prejšnjič smo napisali tudi funkcijo `zunanji`, ki vrne vse kroge, ki niso vsebovani v nobenem drugem krogu. Z drugimi besedami, tiste kroge iz `krogi`, ki se ne pojavijo kot ključi v slovarju, ki ga vrne `vsebovani`.

[2]:
```
def zunanji(krogi):
    vsi_zunanji = []
    for krog in krogi:
        if not je_vsebovan(krog, krogi):
            vsi_zunanji.append(krog)
    return vsi_zunanji
```

Odkar znamo množice, poznamo zunanje kroge poiskati preprosteje: od množice vseh krogov odštejemo množico ključev slovarja, ki ga vrne `vsebovani`.

[3]:
```
def zunanji(krogi):
    return set(krogi) - set(vsebovani(krogi))
```

Funkcija `cebula` je potem takšna:

[4]:
```
def cebula(krogi):
    krogi = set(krogi)
    nivoji = []
    while krogi:
        zunaj = zunanji(krogi)
        nivoji.append(zunaj)
        krogi -= zunaj
    return nivoji
```

V vsakem krogu izvemo, kateri so zunanji krogi. Dodamo jih v seznam in odstranimo iz množice krogov.

Tule pa je še malo drugačna - pravzaprav lepša - rešitev.

[5]:
```
def cebula(krogi):
    nivoji = []
    while krogi:
```

```
        notranji = set(vsebovanost(krogi))
        nivoji.append(set(krogi) - notranji)
        krogi = notranji
    return nivoji
```

`notranji` so notranji krogi (ker nastopajo kot ključi). V `nivoji` dodamo vse kroge, ki niso notranji, `krogi` pa bodo notranji krogi.

Če pogledate, kaj počne funkcija `zunanji`, boste videli, da gre pravzaprav za enako rešitev, le da pri prejšnji dvakrat odštevamo. Razmislite, vredno je.

0.2 Dodatna naloga: hierarhija

Napiši funkcijo `hierarhija(krogi)`, ki prejme seznam krogov, vrne pa slovar, v katerega ključi so vsi krogi, pripadajoče vrednosti pa krog, ki vsebuje ta krog. Če ključ predstavlja zunanji krog, je pripadajoča vrednost `None`.

Za kroge na sliki vrne slovar, ki, med drugim, vsebuje tole:

```
{(88.6, 44.3, 37.5): None,
 (69.8, 46.5, 10.6): (88.6, 44.3, 37.5),
 (99.6, 43.1, 17.2): (88.6, 44.3, 37.5),
 (99.6, 43.1, 10.6): (99.6, 43.1, 17.2),
...
```

Tole opisuje štiri kroge zgoraj levo. Imamo zunanji krog s središčem na x-koordinati 88.6. Ta vsebuje dva kroga s koordinatamo 69.8 in 99.6; tadva kroga sta ključa, pripadajoča vrednost pa je oni, zunanji krog (koordinata 88.6). Potem pa imamo še četrti krog, katerega koordinata x je prav tako 99.6 in se nahaja znotraj večjega kroga z enako koordinato x.

Na podoben način so opisani še vsi drugi krogi.

Ta naloga je za vas najbrž kar težka. Če vas zaintrigira, pišite in bom z veseljem pomagal.

0.2.1 Rešitev

Tale gre pa tako:

[6]:
```
def hierarhija(krogi):
    krogi = set(krogi)
    lastniki = dict.fromkeys(krogi, None)
    while krogi:
        notranji = vsebovanost(krogi)
        for krog, zunaj in notranji.items():
            lastniki[krog] = zunaj[0]
        krogi = set(notranji)
    return lastniki
```

Funkcija je zelo podobna prejšnji - v osnovi imamo enako zanko. Vse, kar smo spremenili je, da nimamo seznama `nivoji` temveč `lastniki` in da imamo namesto `append`-a zanko.

Najprej tisti čudni `lastniki = dict.from_keys(krogi, None)`. Ta je isto kot

```
lastniki = {}
for key in krogi:
    lastniki[key] = None
```

Naredi torej slovar s podanimi ključi, vrednosti pa so `None`. `None` bi lahko tudi izpustili in napisali kar `lastniki = dict.from_keys(krogi)`.

S tem povemo, da za začetek, noben krog nima lastnika.

Znotraj zanke pa počnemo tole: imamo slovar `notranji`, ki za vsak ključ pove, kdo so krogi okrog njega. Naredimo nekaj zelo neumnega. Vsak krog ima lahko več zunanjih krogov, ampak mi si preprosto izmislimo, da je lastnik prvi krog iz tega seznama. To je morda narobe… Ampak nič hudega. Če je okrog nekega kroga še več krogov, bo ta krog prišel na vrsto kasneje.

Vsakemu krogu torej nastavljamo "lastnika", dokler kroga ne odstranimo iz `krogi` (ker je zunanji) in se z njim ne ukvarjamo več.

Pred `lastniki[krog] = zunaj[0]` bi lahko postavili `if len(lastniki[krog]) == 1`. Na ta način bi preverili, da ima ta krog samo še en zunanji krog in bi ga tako nastavili samo enkrat. Vendar … pravzaprav ni potrebe.