resitev

January 28, 2024

0.1 Iskanje Marsovcev

V zadnjem času smo na nabu nad Slovenijo zaznali večje število letečih predmetov, med katerimi so nekateri tudi neznani. Najbrž marsovske ladje.

Tule je posnetek s kamere. (Za lažje delo smo označili x-koordinate središč zaznanih krogov).

- O marsovskih ladjah sicer ne vemo ničesar. Predpostavljali pa bomo, da (a) nimajo dobrih namenov (zihr je zihr, boljše bo, da streljamo prvi :) in (b) da je bolj nevarna tista ladja, ki vsebuje več krogov, karkoli že ti krogi so.

  Napiši funkcijo `najnevarnejsa(krogi)`, ki prejme seznam trojk, ki predstavljajo središča in polmere krogov. Vrniti mora koordinati središča tistega kroga, ki vsebuje največ krogov. Če je takšnih več. naj vrne središče enega izmed njih.

  Upoštevaj tako posredno kot neposredno vsebovane kroge: krog levo zgoraj vsebuje tri kroge in ne le dveh.

  Krogi se lahko dotikajo, nikoli pa se ne sekajo.

- Napiši funkcijo `je_vsebovan(krog, krogi)`, ki za podani krog pove, ali je vsebovan v katerem drugem krogu.

- Napiši funkcijo `zunanji(krogi)`, ki vrne koordinate središč zunanjih krogov, to je, krogov, ki niso vsebovani v nobenem drugem krogu.

- Nekateri od teh predmetov niso tako zelo neznani. Krog, ki vsebuje dva kroga (na sliki sta narisana rdeče) je v resnici ptič z izbuljenimi očmi. In tista strašno nevarna marsovska ladja je najbrž letalo s petimi okni. Pravilo je torej takšno

  – Krog, ki vsebuje dva kroga, tadva pa ne vsebujeta drugih krogov, je ptič.
  – Krog, ki vsebuje več kot dva kroga, ki ne vsebujejo drugih krogov, je letalo.
  – Vsi ostali krogi so marsovci.

  Ta pravila se seveda nanašajo na zunanje kroge, torej kroge, ki niso vsebovani v nobenem drugem krogu.

  Napiši funkcije `ptici(krogi)`, `letala(krogi)` in `marsovci(krogi)`, ki vrnejo sezname koordinate središč vseh ptičev, letal in marsovskih ladij.

Nekateri testi bodo sestavili velike sisteme krogov. Funkcije je potrebno napisati tako, da se vsak test konča v 15 sekundah. Zato bo potrebno uporabljati slovarje - vsaj pri nekaterih.

0.2 Nasveti

V nekem trenutku se ti bo najbrž splačalo napisati pomožno funkcijo, ki prejme seznam krogov in vrne slovar, katerega ključi so trojke (x, y, r), pripadajoče vrednosti pa seznami vseh krogov, ki so neposredno ali posredno znotraj tega kroga.

Poleg tega zna biti koristna funkcija, ki vrne obrnjeni slovar - za vsak krog pove, znotraj katerih krogov se posredno ali neposredno nahaja.

Lahko pa napišeš tudi funkcijo, ki hkrati sestavi in vrne oba slovarja.

0.3 Rešitev

Začnimo s funkcijo, ki, kot svetuje nasvet, vrne slovarja s seznamov notranjih oz. zunanjih krogov za vsak krog.

[1]:
```
def vsebovanost(krogi):
    notranji = defaultdict(list)
    zunanji = defaultdict(list)
    for krog0 in krogi:
        x0, y0, r0 = krog0
        for krog1 in krogi:
            x1, y1, r1 = krog1
            if r0 > r1 and (x1 - x0) ** 2 + (y1 - y0) ** 2 < r0 ** 2:
                notranji[krog0].append(krog1)
                zunanji[krog1].append(krog0)
    return notranji, zunanji
```

Oba slovarja bosta slovarja s privzetimi vrednostmi: ključi bodo trojke, ki predstavljajo kroge, pripadajoče vrednosti pa seznami krogov, torej imamo `defaultdict(list)`.

Nato potrebujemo zanko čez vse pare krogov. To ne bo ena, temveč dve gnezdeni zanki. Čeprav seznam `krogi` vsebuje trojke, jih ne bomo razpakirali takoj, v glavi zanke, temveč šele znotraj. To pa zato, ker bomo potrebovali tako celotno trojko kot posamezne elemente.

Vsak krog torej razpakiramo. Pogoj sestavimo tako, da previmo, ali je `krog0` večji kot `krog1` (da bomo vedeli, kdo je v kom) in ali je razdalja med središčema manjša od polmera večjega kroga. Če je tako, da dodamo `krog1` v seznam krogov, ki so znotraj `krog0`, `krog0` pa v seznam krogov znotraj `krog1`.

Na koncu vrnemo oba slovarja.

To končavši se lotimo ostalih funkcij.

0.3.1 najnevarnejsa

Funkcija dobi seznam krogov. Z njimi brž pokličemo funkcijo `vsebovanost`, zanimal pa nas bo le prvi slovar, seznam notranjih krogov za vsak krog. Gremo prek ključev in vrednosti tega slovarja ter si zapomnimo tistega z najdaljšim seznamom. Ker naloga zahteva, da vrnemo koordinati središča, brez polmera, vrnemo le prva dva elementa trojke.

[1]:
```
def najnevarnejsa(krogi):
    notranji, _ = vsebovanost(krogi)
    naj_krogov = 0
    for sredisce, krogov in notranji.items():
        if len(krogov) > naj_krogov:
            naj_sredisce = sredisce
            naj_krogov = len(krogov)
    return naj_sredisce[:2]
```

0.3.2 je_vsebovan

Pokličemo `vsebovanost`. Zanima nas drugi slovar, tisti, katerega ključi so krogi, pripadajoče vrednosti pa krogi, znotraj katerega se nahaja krog, shranjen kot ključ. Če se krog ne nahaja znotraj nobenega drugega kroga, potem se ne pojavi kot ključ. Krog je torej vsebovan, če nastopa kot ključ v slovarju.

[2]:
```
def je_vsebovan(krog, krogi):
    return krog in vsebovanost(krogi)[1]
```

0.3.3 zunanji

Lahko bi naredili takole:

[3]:
```
def zunanji(krogi):
    vsi_zunanji = []
    for krog in krogi:
        if not je_vsebovan(krog, krogi):
            vsi_zunanji.append(krog)
    return vsi_zunanji
```

Vendar ne bomo. To bi lahko bilo zelo počasno. Funkcija `je_vsebovan` gre čez vse pare krogov, da sestavi slovar(ja), na koncu koncev pa nas zanima samo en ključ (v samo enem od teh dveh slovarjev).

Še huje. Recimo, da imamo 1000 krogov. Potem `zunanji` 1000-krat pokliče `je_vsebovan`, ta pa pogleda 1000-krat 1000 parov. Skupaj je to milijarda operacij.

Boljše je tako:

[4]:
```
def zunanji(krogi):
    _, zunanji = vsebovanost(krogi)
    vsi_zunanji = []
    for krog in krogi:
        if krog not in zunanji:
            vsi_zunanji.append(krog[:2])
    return vsi_zunanji
```

Tu le enkrat pokličemo `vsebovanost`. Nato gremo čez vse kroge. Za vsakega preverimo, ali je zunanji (kar prepoznamo po tem, da ga ni v slovarju `zunanji`; ime je tule žal malo zavajajoče, a

v okviru funkcije `vsebovanost` je bilo smiselno…) in ga, če je res tako, dodamo v seznam zunanjih krogov.

0.3.4 ptici

Tu postanejo stvari zabavne. Ptiče lahko prepoznamo, recimo, tako.

[5]:
```
def ptici(krogi):
    vsebuje, vsebovani = vsebovanost(krogi)
    pticji = []
    for krog in krogi:
        if krog in vsebovani or len(vsebuje[krog]) != 2:
            continue
        for vkrog in vsebuje[krog]:
            if vsebuje[vkrog]:
                break
        else:
            pticji.append(krog[:2])
    return pticji
```

Sestavimo slovar, kot vedno. Gremo čez vse kroge. Če je nek krog vsebovan (torej nezunanji) ali pa ne vsebuje dve krogov, s `continue` skočimo na naslednji krog.

Sicer gremo prek vseh notranjih krogov. Če kateri izmed njih vsebuje druge kroge (seznam, krogov `vsebuje[vkrog]` je neprazen), prekinemo zanko. (`continue` bi to nadaljeval z naslednjim krogom te, notranje, ne zunanje zanke.)

Če se notranja zanka ni prekinila, gre za ptiča.

Če se trika z `else` po `for` ne domislimo, nam lahko pomaga, da vemo, da sta notranja kroga natančno dva.

[6]:
```
def ptici(krogi):
    vsebuje, notranji = vsebovanost(krogi)
    pticji = []
    for krog, vkrogu in vsebuje.items():
        if (krog not in notranji
                and len(vkrogu) == 2 \
                and vkrogu[0] not in vsebuje and vkrogu[1] not in vsebuje):
            pticji.append(krog[:2])
    return pticji
```

0.3.5 letala

Iskanje letal je podobno, le pogoj `!= 2` spremenimo v `== 2`. Da bo različen od `0` pa vemo, ker gremo čez slovar `vsebuje`, ki opisuje le kroge, ki vsebujejo vsaj en krog.

[7]:
```
def letala(krogi):
    vsebuje, vsebovani = vsebovanost(krogi)
    letala = []
```

```
    for krog, vkrogu in vsebuje.items():
        if krog in vsebovani or len(vkrogu) == 2:
            continue
        for vkrog in vkrogu:
            if vkrog in vsebuje:
                break
        else:
            letala.append(krog[:2])
    return letala
```

0.3.6 marsovci

Pri marsovcih moramo logiko še malo preobrniti. Najprej: iti moramo čez vse krogi `for krog in krogi` ne le čez `vsebuje.items()`, ker ne smemo preskočiti krogov, ki ne vsebujejo nobenega kroga.

Če je krog notranji, nadaljujemo z naslednjim krogom.

Če krog ne vsebuje nobenega kroga, je marsovec: dodamo v seznam.

Če vsebuje kak krog, gremo čez te, notranje kroge. Če kateri od njih tudi sam vsebuje kroge, gre za marsovca: dodamo v seznam in prekinemo zanko.

[8]:
```
def marsovci(krogi):
    vsebuje, vsebovani = vsebovanost(krogi)
    nlp = []
    for krog in krogi:
        if krog in vsebovani:
            continue
        if krog not in vsebuje:
            nlp.append(krog[:2])
        else:
            for vkrog in vsebuje[krog]:
                if vkrog in vsebuje:
                    nlp.append(krog[:2])
                    break
    return nlp
```