resitev

January 28, 2024

0.1 Rekurzivne ovire

V tej domači nalogi vadimo pisanje rekurzivnih funkcij. Obe funkciji vam bi bilo lažje napisati iterativno; tako bi bilo za te nalogi tudi bolj naravno. Vendar potrebujemo takšne, preproste primere kot pripravo na težje, pri katerih brez rekurzije (skoraj) ne bo šlo.

Tako kot na predavanjih je smiselno najprej ločeno obravnavati prazen seznam potem pa prvi element in ostanek (ali pa ostanek in prvi element, kakor bo naneslo).

1. Napiši *rekurzivno funkcijo* `stolpec3(ovire)`, ki dobi seznam trojk, ki opisujejo ovire (`(x0, x1, y)`) in vrne `True`, če katera ovira pokriva (tudi) stolpec 3 in `False`, če ne.
2. Napiši *rekurzivno funkcijo* `najsirsa(ovire)`, ki vrne najširšo oviro. Če je enako širokih najširših ovir več, naj vrne prvo med njimi.

   Klic `[(1, 1, 4), (3, 5, 3), (2, 8, 1), (8, 10, 2)]` vrne `(2, 8, 1)`.

0.1.1 Rešitev

`stolpec3(ovire)`

- Če ovir ni, vrnemo `False`, saj očitno ni ovire, ki bi zakrivala tretji stolpec.
- Če ga prva ovira zakriva, vrnemo `True`.
- Sicer ponovimo vajo na ostalih ovirah.

[1]:
```
def stolpec3(ovire):
    if ovire == []:
        return False
    prva = ovire[0]
    if prva[0] <= 3 <= prva[1]:
        return True
    else:
        return stolpec3(ovire[1:])
```

Z drugimi besedami: funkcija mora povedati ali imamo kakšno oviro in zakriva tretji stolpec bodisi prva ovira bodisi katera od preostalih.

[2]:
```
def stolpec3(ovire):
    return ovire and (ovire[0][0] <= 3 <= ovire[0][1] or stolpec3(ovire[1:]))
```

Ne spreglej oklepaja (tistega, za `and`): `and` ima prednost pred `or`, zato je del pogoja z `or` v oklepajih.

`najsirsa(ovire)`

- Če ovir ni, vrnimo oviro širine `-1`. Naloga nekako predpostavlja, da ovire bodo; da jih ni, se bo zgodilo na koncu rekurzije in takrat želimo vrniti "oviro", ožjo od vseh ostalih, tako da bo že prva ovira, ki jo bomo srečali na poti iz rekurzije, širša.
- Sicer izvemo, katera je najširša ovira iz ostanka. Primerjamo jo s prvo in če je prva širša (ali enako široka!) vrnemo prvo, sicer najširšo izmed ostalih.

[3]:
```
def najsirsa(ovire):
    if not ovire:
        return (0, -1, 0)

    prva = ovire[0]
    naj_ostanek = najsirsa(ovire[1:])
    if prva[1] - prva[0] >= naj_ostanek[1] - naj_ostanek[0]:
        return prva
    else:
        return naj_ostanek
```