resitev

January 28, 2024

1 Ovire v eni vrsti

Vračamo se k prvi nalogi Zemljevid ovir. Tokrat jo bomo reševali s pomočjo izpeljanih seznamov. Če vam kaj pomaga (najbrž vam vsaj malo) imate poleg testov že napisane funkcije, ki rešijo naloge … le predolge so. :)

1.1 Obvezna naloga

Napišite funkcije

- `stevilo_ovir(ovire)`,
- `dolzina_ovir(ovire)`,
- `sirina(ovire)`,
- `globina(ovire, x)`,
- `senca(ovire)`

tako, da bodo vsebovale samo stavek `return` in v njem to, kar je pač potrebno, da izračunate, kar je treba izračunati.

1.1.1 Rešitev

Število ovir je že napisano v vrstici. Daljše skoraj ne gre. :)

[1]:
```
def stevilo_ovir(ovire):
    return len(ovire)
```

`dolzina_ovir(ovire)` mora vrniti vsoto dolžin vseh ovir. Dolžina ovire `(x0, x1, y)` je `x1 - x0 + 1`, vsota tega je `sum(x1 - x0 + 1 ... za vsako oviro`.

[2]:
```
def dolzina_ovir(ovire):
    return sum(x1 - x0 + 1 for x0, x1, _ in ovire)
```

Ker `y` ne potrebujemo, tretjo vrednost iz trojke poimenujemo kar `_`.

`sirina(ovire)` je enaka največjemu (po angleško: maksimalnemu) `x1`.

[3]:
```
def sirina(ovire):
    return max(x1 for _, x1, _ in ovire)
```

`globina(ovire, x)` mora vrniti najmanjši `y` med vsemi `x0, x1, y`, za katere velja, da je `x` med `x0` in `x1`, torej `x0 <= x <= x1`.

To bi lahko bilo tole:

[6]:
```
def globina(ovire, x):
    return min(y for x0, x1, y in ovire if x0 <= x <= x1)
```

vendar ne deluje v primeru, da v stolpcu ni nobene ovire. Tedaj `(y for x0, x1, y in ovire if x0 <= x <= x1)` ne zgenerira ničesar in `max` ne ve, kaj vrniti. Pogledši dokumentacijo funkcije `min` izvemo, da ji lahko podamo še argument `default`, s katero predpišemo vrednost, ki naj jo vrne v takšnem primeru. V naši nalogi želimo, da tedaj vrne `None`, torej

[7]:
```
def globina(ovire, x):
    return min((y for x0, x1, y in ovire if x0 <= x <= x1), default=None)
```

`senca(ovire)` mora sestaviti seznam s toliko elementi, kolikor je stolpcev in vsak vsebuje `True`, če `globina` za ta stolpec vrne `None` in `False`, če ne.

[8]:
```
def senca(ovire):
    return [globina(ovire, x) is None for x in range(1, sirina(ovire) + 1)]
```

1.2 Dodatna naloga

Napišite funkcijo `indeksi(s, subs)`, ki prejme niza `s` in `subs` ter vrne seznam indeksov znotraj `s`, na katerih se pojavi `subs`. Klic `indeksi("pepelka peče prepelice", "pe")` vrne `[0, 2, 8, 16]`, saj se pe pojavi na indeksih 0, 2, 8 in 16. Tudi ta funkcija sme seveda obsegati samo `return`.

Potem napišite v eni vrstici funkcijo

- `pretvori_vrstico(vrstica)`.

1.2.1 Rešitev

Reševanje dodatne naloge tokrat ni bilo popularno. Prvo funkcijo, `indeksi` je že še kdo napisal, druga, `pretvori_vrstico` pa je zahtevala preveč trikov. Nekatere smo sicer že pokazali pri rešitvi prve domače naloge.

V funkciji `indeksi` bi lahko navidez uporabljali metodo `index`. Vendar bi bilo to zoprno že v več vrsticah - v eni pa bi bilo res zahtevno.

Veliko preprosteje je pomisliti drugače: kaj hoče funkcija? Kaj moramo vrniti? Vse tiste indekse, za katere velja, da se na njih začenja podniz `subs`. Kako torej preverimo, ali se na `i`-tem mestu `s`-ja začenja `subs`? `s[i:i + len(subs)] == subs`. Lahko pa uporabimo tudi metodo `startswith`, ki pove, ali se niz začne s podanim nizom: `s[i:].startswith(subs)`.

Funkcija `indeksi` je torej:

[9]:
```
def indeksi(s, subs):
    return [i for i in range(len(s)) if s[i:].startswith(subs)]
```

Zdaj pa pretvorimo vrstico. Zanima nas, kje so začetki ovir. Ovire za začnejo tam, kjer najdemo `.#`. Da bomo našli tudi ovire, ki bi bile čisto na začetku, na začetek prištejemo piko. Ovire se torej začenjajo na `indeksi("." + s, ".#")`. Vendar bodo ti indeksi žal za 1 premajhni: če imamo `s = "#"`, se ovira začne na `1` (ker v teh nalogah pač štejemo od 1). Klic `indeksi("." + s, ".#")`, ali,

konkretneje, `indeksi(".#", ".#")` bo vrnil `[0]`. Nič hudega: indeksom bi sicer lahko prištevali 1 kdaj kasneje, a preprosteje, če nizu `s` namesto ene same pike prištejemo dve, pa se bodo vsi indeksi povečali za `1`. Stolpci, v katerih se začenjajo ovire, so torej `indeksi(".." + s, ".#")`.

Pa konci? Zdaj iščemo `"#."`. Da pravilno zaznamo tudi oviro, ki bi bila čisto na koncu niza, nizu prištejemo še `"."`. Da povečamo indekse za `1`, dodamo piko na začetku (namesto pike bi lahko na začetek dodali tudi karkoli drugega). Konci ovir so torej na `indeksi("." + s  + ".", subs)`.

Funkcija mora vrniti pare začetkov in koncev.

[10]:
```
def pretvori_vrstico(vrstica):
    return list(zip(indeksi(".." + vrstica, ".#"), indeksi("." + vrstica + ".",␣
 ↪"#.")))
```