resitev

January 28, 2024

Izvedši, da je njihov zemljevid s pikami in lojtrami neuporaben in živcira študente računalništva, kognitivne znanosti, upravne informatike in digitalnega jezikoslovja, so začali na Oddelku za gospodarstvo in motorizirani promet Mestne občine Ljubljana pripravljati podatke v drugačni obliki. Zato bo potrebno napisati nove funkcije za njihovo branje.

Ovire so zapisane v večvrstičnem nizu:

```
4: 5-6
13: 90-100, 5-8, 9-11
5: 9-11, 19-20, 30-34
4: 9-11
13:  22-25, 17-19
```

To pomeni, da je v 4. vrstici ovira od 5 do 6. V 13. vrstici so ovire od 90 do 100, od 5 do 8 in od 9 do 11. V 5. so od … In potem so se spomnili, da je v 4. vrstici še ena ovira. In v 13. še dve.

Oblika vsake posamične vrstice je torej takšna: imamo številko vrstice, ki ji sledi dvopičje, temu pa pari koordinat `(x0, x1)`. Med koordinatama je `-`, med pari pa vejice. Kot vidimo zgoraj.

0.1 Kako iz/do večvrstičnih nizov

Če je `s` večvrstični niz, ga razbijemo v seznam posamičnih vrstic tako, da pokličemo `s.splitlines()`.

Če imamo niza `prva = "Prva vrstica"` in `druga = "Druga vrstica"`, ga zložimo v dvovrstični niz, tako da seštejemo `prva + "\n" + druga`. Kombinacija `\n` pomeni prehod v novo vrstico.

0.2 Obvezna naloga

Napiši naslednje funkcije.

- `vrstica(s)` prejme niz z eno vrstico in vrne seznam trojk `(x0, x1, y)`, ki predstavljajo ovire v tej vrstici.

  Klic `vrstica("4: 1-3, 5-11, 15-33")` vrne seznam `[(1, 3, 4), (5, 11, 4), (15, 33, 4)]`.

- `preberi(s)` celoten, večvrstični niz z ovirami in vrne seznam ovir. Ovire naj bodo shranjene v takšnem vrstnem redu, v kakršnem se pojavljajo.

  Če jo pokličemo z gornjim nizom, vrne

  ```
  [(5, 6, 4),
   (90, 100, 13), (5, 8, 13), (9, 11, 13),
  ```

```
(9, 11, 5), (19, 20, 5), (30, 34, 5),
(9, 11, 4),
(22, 25, 13), (17, 19, 13)]
```

- `intervali(xs)` prejme seznam parov `(x0, x1)` in vrne seznam nizov, ki opisujejo te intervale.

  Klic `intervali([(6, 10), (12, 16), (20, 22), (98, 102)])` vrne `["6-10", "12-16", "20-22", "98-102"]`.

- `zapisi_vrstico(y, xs)` prejme številko vrstice in seznam parov `(x0, x1)`. Vrniti mora opis ene vrstice.

  Klic `zapisi_vrstico(5, [(6, 10), (12, 16)]))` vrne niz `"5: 6-10, 12-16"`. Pazi: nobenih odvečnih ali manjkajočih presledkov ali vejic!

0.2.1 Rešitev

0.2.2 `vrstica`

Vrstico moramo najprej razdeliti glede na `":"`. Ker vemo, da bo samo eden, bomo rezultat funkcije takoj razpakirali na dva dela, niz s številko vrstice (`ys` - `s` kot "string") in niz s koordinatami x (`xs`). `ys` nato razbijemo glede na `", "` in čez dobljene kose poženemo zanko `for`; vsak kos, končno, razbijemo še po `-` da dobimo začetek in konec ovire. Trojke z začetkom in koncem ovire ter koordinato `y` vestno zlagamo v seznam.

[1]:
```
def vrstica(s):
    ovire = []
    ys, xs = s.split(":")
    y = int(ys)
    for x0x1 in xs.split(", "):
        x0s, x1s = x0x1.split("-")
        ovire.append((int(x0s), int(x1s), y))
    return ovire
```

Dve opazki. `ys` pretvorimo v `int` kar pred zanko, namesto da bi to počeli vsakič posebej. Program je zaradi tega za vrstico daljši, je pa za (pri tako malo podatkih neopazno) malenkost hitrejši.

Druga: trojko s koordinatami sestavimo kar ob klicu funkcije `append`.

0.2.3 `preberi`

Tu ni kaj: celoten niz razbijemo v vrstice s `splitlines`, čez dobljeni seznam poženemo zanko `for` in za vsako vrstico pokličemo `vrstica` ter njene rezultate zlagamo v seznam.

[2]:
```
def preberi(s):
    ovire = []
    for v in s.splitlines():
        ovire += vrstica(v)
    return ovire
```

0.2.4 `intervali`

Pripravimo prazen seznam. Gremo čez ovire in za vsako v seznam dodamo niz. Sestavljanje niza je nekoliko okorno - `str(x0) + "-" + str(x1)`. Nekoč se bomo naučili, da gre tudi lažje.

[3]:
```
def intervali(xs):
    intv = []
    for x0, x1 in xs:
        intv.append(str(x0) + "-" + str(x1))
    return intv
```

0.2.5 `zapisi_vrstico`

Naloga preverja, ali znamo poklicati `join`. :) Potrebno je vzeti številko vrstice in dvopičje, k temu pa dodati z vejico združeni seznam, ki nam ga pripravi prejšnja funkcija.

[4]:
```
def zapisi_vrstico(y, xs):
    return str(y) + ": " + ", ".join(intervali(xs))
```

0.3 Dodatna naloga

Napiši funkcijo `zapisi(ovire)`, ki prejme seznam ovir in vrne niz, ki vsebuje opis ovir v novi obliki.

Klic

```
zapisi([(5, 6, 4), (9, 11, 4),
        (9, 11, 5), (19, 20, 5), (30, 34, 5),
        (5, 8, 13), (9, 11, 13), (17, 19, 13), (90, 100, 13)])
```

vrne niz

```
4: 5-6, 9-11
5: 9-11, 19-20, 30-34
13: 5-8, 9-11, 17-19, 90-100
```

Spet: niz mora biti v točno takšni obliki, brez odvečnih ali manjkajočih presledkov ali vejic.

0.3.1 Rešitev

Tole gre na kup načinov. Eden, ki ne uporablja prejšnjih funkcij, je tale:

[5]:
```
def zapisi(ovire):
    yvrste = None
    zemljevid = ""
    for x0, x1, y in ovire:
        if y != yvrste:
            if yvrste != None:
                zemljevid += vrsta + "\n"
            vrsta = str(y) + ": " + str(x0) + "-" + str(x1)
            yvrste = y
        else:
```

```
            vrsta += ", " + str(x0) + "-" + str(x1)
    zemljevid += vrsta
    return zemljevid
```

Osnovni problem je, da ne vemo, koliko ovir je v posamezni vrstici.

V `vrsta` bomo nabirali vsebino trenutne vrstice. V `yvrste` bo shranjena številka trenutne vrstice.

Zato za vsako oviro pogledamo, ali je v isti vrstici kot te, ki jih trenutno sestavljamo. Če ni, se začenja nova vrstica: trenutno vrstico dodamo v `zemljevid` (razen, če je `yvrste` enak `None`, kar pomeni, da je to šele prva vrstica. Nato v spremenljivko `vrsta` zapišemo začetek te vrstice (koordinato y in trenutni par koordinat) in v `yvrste` shranimo številko trenutne vrstice.

Če pa je ta ovira v vrstici, ki jo trenutno sestavljamo (`else`), pa le dodamo oviro v vrstico.

Na koncu, po zanki, dodamo v zemljevid še zadnjo vrstico, saj ni bilo nobene ovire, ki bi jo "zaključila".

Če hočemo, lahko pišemo tudi naravnost v `zemljevid`:

[6]:
```
def zapisi(ovire):
    prejy = None
    zemljevid = ""
    for x0, x1, y in ovire:
        if y != prejy:
            if prejy != None:
                zemljevid += "\n"
            zemljevid += str(y) + ": " + str(x0) + "-" + str(x1)
            prejy = y
        else:
            zemljevid += ", " + str(x0) + "-" + str(x1)
    return zemljevid
```

Lepše pa je, seveda, če uporabimo funkcije, ki smo si jih pripravili zgoraj. Recimo tako, da je `vrsta` seznam ovir v neki vrsti. Ko je vrste konec, jih dodamo z `zapisi_vrstico`:

[7]:
```
def zapisi(ovire):
    prejy = None
    zemljevid = ""
    for x0, x1, y in ovire:
        if y != prejy:
            if prejy != None:
                zemljevid += zapisi_vrstico(prejy, vrsta) + "\n"
            vrsta = []
            prejy = y
        vrsta.append((x0, x1))
    zemljevid += zapisi_vrstico(prejy, vrsta) + "\n"
    return zemljevid
```

Pa če ovire ne bi bile podane po vrsti? Ubogi študenti na FRI rešujejo problem, kjer so ovire v

seznamu poljubno pomešane, izpisati pa jih morajo po vrsti…

Takole: pripravimo seznam seznamov, ki ima za elemente toliko praznih seznamov, kolikor je vrstic. Vsak, očitno, pripada eni vrstici. Nato gremo čez seznam ovir in vsako oviro vržemo v pripadajoči seznam. Na koncu se zapeljemo čez te sezname in vsakega, ki ni prazen, izpišemo v zemljevid.

[8]:
```
def zapisi(ovire):
    # Poišči število vrstic
    visina = 0
    for _, _, y in ovire:
        if y > visina:
            visina = y

    # Pripravi seznam seznamov; vrstice[i] bo seznam ovir v i-ti vrstici
    # Na začetek pa damo še en prazen seznam, da se bodo indeksi začeli z 1.
    vrstice = [[]]
    for _ in range(visina):
        vrstice.append([])

    # Vsako oviro v pripadajoči seznam
    for x0, x1, y in ovire:
        vrstice[y].append((x0, x1))

    # In zdaj le še izpišemo sezname v zemljevid
    zemljevid = ""
    for y, vrstica in enumerate(vrstice):
        if vrstica:
            zemljevid += zapisi_vrstico(y, vrstica) + "\n"
    return zemljevid
```

Ni tako grozno, ne?

Še lepše gre s slovarji. Konkretno, z `defaultdict`-om. Te smo, ah, smola, spoznali šele na naslednjih predavanjih, po roku za oddajo naloge. Vseeno poglejmo rešitev.

[9]:
```
def zapisi(ovire):
    vrstice = defaultdict(list)
    for x0, x1, y in ovire:
        vrstice[y].append((x0, x1))

    zapis = ""
    for y, vrstica in sorted(vrstice.items()):
        zapis += zapisi_vrstico(y, sorted(vrstica)) + "\n"
    return zapis
```

Vzeli smo `defaultdict(list)`: če uporabimo nek ključ, ki ga ni v slovarju, se ta sam od sebe pojavi, pripadajoča vrednost pa je prazen slovar.

V prvem delu funkcije gremo čez ovire in dodajamo pare `(x0, x1)` v pripadajoče vrstice `y`.

V drugem delu gremo čez slovar. Vzamemo vse pare (ključ, vrednost), ki jih dobimo z `items`. S `sorted` jih uredimo; ker gre za terke, jih bo `sorted` uredil po prvem elementu, če sta prva elementa enaka, pa po drugem. (Da bi bila prva elementa enaka, se tu ne more zgoditi. Razmisli, zakaj!). Za vsako vrstico dodamo v `zapis` zapis te vrstice, ki ga dobimo tako, da pokličemo funkcijo `zapisi_vrstico` z urejenim seznamom ovir v tej vrstici. Preverjanje, ali je vrstica morda brez ovir, zdaj ni več potrebno, saj takih vrstic v slovarju ne bo.