resitev

January 28, 2024

1 Prvi trije

Vse funkcije, ki jih je potrebno napisati, dobijo podatke o rodbini v globalnem slovarju `otroci`, katerega ključi so imena osebe, pripadajoče vrednosti pa seznami imen njihovih otrok. Rodbina je takšna kot na predavanjih (glej objavljeno drevo).

1.1 Obvezni del

Napiši funkcijo `prvi(oseba)`, ki vrne tisto ime v rodbini podane osebe, ki je prvo po abecedi. **Funkcija mora prestati teste v doglednem času (največ nekaj sekund)! Pol minute ni ugleden čas.**

```
>>> prvi("Elizabeta")
"Aleksander"
>>> prvi("Hans")
"Erik"
>>> prvi("Adam")
"Adam"
```

Napiši funkcijo `prvi_trije(oseba)`, ki vrne seznam prvih treh imen po abecedi - spet znotraj rodbine podane osebe. Seznam naj bo urejen po abecedi.

```
>>> prvi_trije("Elizabeta")
["Aleksander", "Alenka", "Barbara"]
>>> prvi_trije("Herman")
["Herman", "Margareta"]
>>> prvi_trije("Adam")
["Adam", "Aleksander", "Alenka"]
```

Pomoč: prve tri med potomci določene osebe velja iskati med prvimi tremi med potomci vsakega otroka.

Še pomoči: prve tri elemente seznama najpreprosteje dobiš tako, da seznam urediš in nato vzameš prvi tri elemente.

1.1.1 Rešitev

Vsaka oseba predpostavi, da je prva po abecedi ona. Nato vsakega otroka vpraša, kdo je prvi v njegovi rodbini. Če otrok pove za kakšno ime, ki je po abecedi prej, si to zapomnimo.

[1]:
```
def prvi(oseba):
    naj = oseba
    for otrok in otroci[oseba]:
        najotroka = prvi(otrok)
        if najotroka < naj:
            naj = najotroka
    return naj
```

Napaka, ki je ne smemo narediti, je:

[2]:
```
def prvi(oseba):
    naj = oseba
    for otrok in otroci[oseba]:
        if najotroka < prvi(otrok):
            naj = prvi(otrok)
    return naj
```

Kot smo povedali na predavanjih, se tu zgodi naslednje: če dvakrat vprašamo vsakega otroka, bo ta dvakrat vprašal vsakega od svojih, torej bodo vnuki vprašani štirikrat, pravnuki osemkrat, prapravnuki šestnajstkrat in tako naprej.

Prve tri lahko dobimo takole:

[3]:
```
def prvi_trije(oseba):
    naj = [oseba]
    for otrok in otroci[oseba]:
        naj += prvi_trije(otrok)
    return sorted(naj)[:3]
```

Vsaka oseba da v seznam svoje ime, nato pa še prve tri, ki jih predlaga vsak otrok. To uredi in vrne prve tri.

1.1.2 Nerešitev

Nekaj študentov je odkrilo tole rešitev, ki mi žal ni prišla na misel že ob sestavljanju naloge (in tudi tedaj bi jo težko preprečil): uporabili so funkcijo `rodbina`, ki smo jo napisali na predavanjih. Ta jim da seznam vseh imen, potem pa le vrnejo prvo ali prva tri.

[4]:
```
def rodbina(oseba):
    clani = [oseba]
    for otrok in otroci[oseba]:
        clani += rodbina(otrok)
    return clani

def prvi(oseba):
    return min(rodbina(oseba))

def prvi_trije(oseba):
    return sorted(rodbina(oseba))[:3]
```

To deluje, zato sem bil to prisiljen to priznati kot pravilno rešitev. Žal se ob njej naučimo le, kako poklicati funkcij (kar, upam, že znamo) in kako dobiti najmanjše tri elemente seznama (kar tudi ni umetnost). O rekurziji pa se na ta način ne naučimo ničesar.

1.2 Dodatna naloga (neobvezno)

Napiši funkcijo `prvi_trije_brez(oseba)`, ki je podobna kot `prvi_trije`, le da ne upošteva podane osebe.

```
>>> prvi_trije("Adam")
["Aleksander", "Alenka", "Barbara"]
>>> prvi_trije("Herman")
["Margareta"]
>>> prvi_trije("Margareta")
[]
```

Ta naloga najbrž ni težja od obvezne.

1.2.1 Rešitev

Vsaka oseba da v seznam imena svojih otrok. Nato od otrok pridobi prve tri za vsakega od njih (in noben od njih ne bo dodal v seznam sebe!). Nato uredimo po abecedi in vrnemo prve tri.

[5]:
```
def prvi_trije_brez(oseba):
    naj = otroci[oseba][:]
    for otrok in otroci[oseba]:
        naj += prvi_trije_brez(otrok)
    return sorted(naj)[:3]
```

Nekaj študentov me je presenetilo z izvirno rešitvijo (morda se je je domislil le eden in se je razširila):

[6]:
```
def prvi_trije_brez(oseba):
    naj = []
    for otrok in otroci[oseba]:
        naj += prvi_trije(otrok)
    return sorted(naj)[:3]
```

Prepisali so torej kar funkcijo `prvi_trije`, le da so spremenili `naj = [oseba]` v `naj = []`. V zanki pa še vedno kličejo `prvi_trije` in ne `prvi_trije_brez`. Duhovito.

1.3 Še bolj dodatna naloga

Napiši gornje funkcije še brez uporabe rekurzije. Ta naloga pa je nekoliko težja.

1.3.1 Rešitev

Najprej s finto, ki zahteva, da vemo, kako delujejo zanke v Pythonu: `pregledati` bo seznam vseh oseb, ki jih je potrebno pogledati. V začetku je to samo `oseba`. Nato gremo čez seznam vseh oseb, ki jih je potrebno pogledati. Za vsako od njih preverimo, če je po abecedi prej od prve doslej, nato pa zabeležimo, da je potrebno pregledati še njene otroke.

[7]:
```
def prvi(oseba):
    pregledati = [oseba]
    naj = oseba
    for oseba in pregledati:
        if oseba < naj:
            naj = oseba
        pregledati += otroci[oseba]
    return naj
```

Tole ima dve težavi. Prva, manj pomembna, je, da zahteva, da Pythonov `for` obvlada sezname, ki se tekom zanke podaljšujejo. To pač deluje in v večini drugih jezikov bi bodisi delovalo bodisi bi bilo lahko dogoljufati. Drugi problem je, da je seznam vedno daljši in daljši in daljši, tako da se tudi testi izvajajo kar dolgo.

Namesto zanke `for` lahko uporabimo `while`. Dokler obstaja kakšna oseba, ki jo je potrebno pogledati, jo vzamemo s seznama, pogledamo in dodamo njene otroke. S seznama jo vzamemo s `pop` - ta vrne in pobriše zadnji element.

[8]:
```
def prvi(oseba):
    pregledati = [oseba]
    naj = oseba
    while pregledati:
        oseba = pregledati.pop()
        if oseba < naj:
            naj = oseba
        pregledati += otroci[oseba]
    return naj
```

Če koga zanima kaj več, naj v zanko doda `print(oseba)` in opazuje vrstni red, v katerem se izpisujejo imena.

Drugo funkcijo bi lahko spremenili tako, da bi osebe obravnavala v enakem vrstnem redu kot prva, vendar bi jo to nekoliko upočasnilo … Več o tem v tretjem letniku.

Funkcija `prvi_trije` je podobna.

[9]:
```
def prvi_trije(oseba):
    pogledati = [oseba]
    naj = []
    for oseba in pogledati:
        pogledati += otroci[oseba]
        naj = sorted(naj + [oseba])[:3]
    return naj
```

V vsakem koraku k `naj` dodamo `oseba`, uredimo in obdržimo prve tri. V tretjem letniku bomo izvedeli, kako to početi malo hitrejše.

Da preskočimo trenutno osebo, je ne dodamo v `naj`, pač pa v `naj` dodajamo otroke.

[10]:
```python
def prvi_trije_brez(oseba):
    pogledati = [oseba]
    naj = []
    for oseba in pogledati:
        pogledati += otroci[oseba]
        naj = sorted(naj + otroci[oseba])[:3]
    return naj
```