resitev

January 28, 2024

Na predavanju smo delali z rodbino, ki se je začela z Adamom ter končala z Aleksandrom.

Par popravkov.

1. Starost, ki je zapisana pri Eriku, je napačna. Pravilna je 38 (s 83 leti bi bil starejši od svojega očeta).
2. Te številke v resnici niso letnice, temveč cene. Drevo pa ne kaže rodbine, temveč gre za drevo podizvajalcev cenitev za dražbo.

Pravilno je torej tako.

[1]:
```
podizvajalci = {
    "Adam": ["Matjaž", "Cilka", "Daniel"],
    "Aleksander": [],
    "Alenka": [],
    "Barbara": [],
    "Cilka": [],
    "Daniel": ["Elizabeta", "Hans"],
    "Erik": [],
    "Elizabeta": ["Ludvik", "Jurij", "Barbara"],
    "Franc": [],
    "Herman": ["Margareta"],
    "Hans": ["Herman", "Erik"],
    "Jožef": ["Alenka", "Aleksander", "Petra"],
    "Jurij": ["Franc", "Jožef"],
    "Ludvik": [],
    "Margareta": [],
    "Matjaž": ["Viljem"],
    "Petra": [],
    "Tadeja": [],
    "Viljem": ["Tadeja"],
}


cene = {"Adam": 111,
        "Matjaž": 90, "Cilka": 88, "Daniel": 85,
        "Viljem": 58, "Elizabeta": 67, "Hans": 64,
        "Tadeja": 20, "Ludvik": 50, "Jurij": 49, "Barbara": 45, "Herman": 39,␣
 ↪"Erik": 38,
```

```
        "Franc": 30, "Jožef": 29, "Margareta": 3,
        "Alenka": 9, "Petra": 7, "Aleksander": 5}
```

0.1 Obvezna naloga

Cenitev naročimo pri eni od oseb v drevesu. V resnici pa dela ne opravi ta oseba, temveč cenitev delegira osebam, pod sabo. Pa tudi te ne opravijo dela same, temveč ga delegirajo. Šele oseba, ki nima podizvajalcev, dejansko opravi delo.

Kakšna je cena? - Če delo naročimo pri Alenki, bo cena 9 - kolikor piše. - Če delo naročimo pri Jožefu, ga bo delegiral Alenki, Petri in Aleksandru. Ti bodo zahtevali 9 + 7 + 5 cekinov. Jožef si vzame še 10 % provizije, torej bo skupna cena (9 + 7 + 5) * 1.1 = 23,1 cekinov. - Če delo naročimo pri Juriju, ga bo delegiral Francu in Jožefu. Franc zahteva 30 cekinov, Jurij pa, kot smo pravkar izračunali, vemo 23,1. Jurij na to vzame 10 %, torej je cena (30 + 23.1) * 1.1 = 58,41 cekinov. - Če delo naročimo pri Elizabeti, bo to stalo 50 (Ludvik) in 58,41 (Jurij) in 45 cekinov + provizija, torej (50 + 58,41 + 45) * 1.1 = 168,751 cekinov.

Napišite naslednje funkcije.

- `velikost_ekipe(oseba)` prejme ime osebe in vrne število vseh oseb, ki jih ta oseba delegira, vključno z njo.
- `koncni_izvajalci(oseba)` prejme ime osebe in vrne seznam vseh oseb, ki bodo dejansko opravile delo. To so tiste osebe, ki nimajo podizvajalcev.
- `cena(oseba)` prejme ime osebe in vrne ceno, ki jo bo ta oseba zaračunala za delo (tako kot smo računali v gornjem primeru).

0.1.1 Rešitev

Prva naloga je seveda prvo, kar smo s to rodbino delali na predavanjih. Zanima nas "velikost rodbine".

[2]:
```
def velikost_ekipe(oseba):
    velikost = 1
    for podizvajalec in podizvajalci[oseba]:
        velikost += velikost_ekipe(podizvajalec)
    return velikost
```

Ali, krajše,

[3]:
```
def velikost_ekipe(oseba):
    return 1 + sum(velikost_ekipe(otrok) for otrok in podizvajalci[oseba])
```

Druga naloga sprašuje po članih rodbine, ki nimajo potomcev. To je podobno, kot da bi nabirali člane rodbine, vendar je pri nabiranju članov vsak dal v množico tudi sebe, tu pa se dodajo v množico le tisti, ki nimajo potomcev. Oziroma, v tej zgodbici, podizvajalcev.

Skratka, tisti, ki nimajo podizvajalcev, vrnejo množico, ki vsebuje njih same. Tisti, ki jih imajo, vrnejo unijo množic, ki jih vrnejo njihovi podizvajalci.

[4]:
```
def koncni_izvajalci(oseba):
    if podizvajalci[oseba] == []:
        return {oseba}

    pod = set()
    for podizvajalec in podizvajalci[oseba]:
        pod |= koncni_izvajalci(podizvajalec)
    return pod
```

Tretja naloga je na nek način podobna. Tisti, ki nimajo podizvajalcev, vrnejo svojo ceno. Tisti, ki jih imajo, vrnejo vsoto cen podizvajalcev, pomnoženo z 1.1.

[5]:
```
def cena(oseba):
    if podizvajalci[oseba] == []:
        return cene[oseba]

    skupna = 0
    for podizvajalec in podizvajalci[oseba]:
        skupna += cena(podizvajalec)
    return 1.1 * skupna
```

Ali, krajše,

[6]:
```
def cena(oseba):
    if podizvajalci[oseba] == []:
        return cene[oseba]
    return 1.1 * sum((cena(otrok) for otrok in podizvajalci[oseba]))
```

0.2 Dodatna naloga

EU ugotovi, da tole s podizvajalci ne gre nikamor več. Zato sprejme uredbo, po kateri je za določen tip dela dovoljeno najemati podizvajalce le do določene globine. Če je globina 0, to pomeni, da mora delo opraviti ta, ki smo ga najeli. Če je globina 1, to pomeni, da sme najemati le neposredne podizvajalce. Če je globina 2, smejo neposredni podizvajalci najeti podizvajalce na naslednjem nivoju, ti pa ne smejo najemati svojih podizvajalcev. In tako naprej.

Če za delo najamemo Adama in je globina omejena na 2, bodo delo opravili Viljem, Cilka, Elizabeta in Hans. (Pazi: Cilka je sicer takoj pod Adamom, vendar nima podizvajalcev, zato dela sama.)

Vsem funkcijam, ki si jih sprogramiral(a) za obvezno nalogo, dodaj še argument `globina` in poskrbi, da bodo vračale rezultate, skladne z uredbo.

0.2.1 Rešitev

Vse bistvo naloge je v tem, da uvidimo, da je dodatni argument v vsakem klicu za 1 manjši. Če nekomu dovolimo `n` nivojem podizvajalcev, smejo imeti njegovi neposredni podizvajalci `n - 1` nivojev. Ko pride `n` do `0`, se vedemo, kot da nekdo nima podizvajalcev.

[7]:
```python
def velikost_ekipe(oseba, n):
    velikost = 1
    if n > 0:
        for podizvajalec in podizvajalci[oseba]:
            velikost += velikost_ekipe(podizvajalec, n - 1)
    return velikost


def koncni_izvajalci(oseba, n):
    if n == 0 or podizvajalci[oseba] == []:
        return {oseba}
    pod = set()
    for otrok in podizvajalci[oseba]:
        pod |= koncni_izvajalci(otrok, n - 1)
    return pod


def cena(oseba, n):
    if n == 0 or podizvajalci[oseba] == []:
        return cene[oseba]
    return 1.1 * sum((cena(otrok, n - 1) for otrok in podizvajalci[oseba]))
```