resitev

January 28, 2024

0.1 Obvezna

Napiši rekurzivno funkcijo `lihih(s)`, ki prejme seznam in vrne število lihih števil v njem.

0.1.1 Rešitev

Najprej dolgočasno korektna rešitev.

- Če je seznam prazen, je število lihih elementov `0`.
- Če je prvi element lih, je lihih toliko, kot jih je v ostanku in še ta zraven.
- Sicer je lihih toliko, kolikor jih je v ostanku.

[1]:
```
def lihih(s):
    if s == []:
        return 0
    if s[0] % 2 == 1:
        return lihih(s[1:]) + 1
    else:
        return lihih(s[1:])
```

[2]:
```
lihih([5, 7, 4, 2, 5, 1, 0])
```

[2]:
```
4
```

Zdaj pa nekaj bolj inspiriranega. Namesto `s == []` lahko pišemo `not s`. To je resnično (`True`), če je `s` prazen. Vse prazne stvari so namreč neresnične (`False`), neprazne pa resnične (`True`). Kaj pa vrne `bool(s)`? Pretvori seznam v `bool`, tako da rne `True`, če je seznam "resničen" (neprazen) in `False` sicer.

[3]:
```
bool([2, 45, 6])
```

[3]:
```
True
```

[4]:
```
bool([])
```

[4]:
```
False
```

[5]:
```
bool([False])
```

[5]:
```
True
```

Zadnji primer služi temu, da nas pouči, da rezultat ni povezan z vsebino seznama, pomembno je le, ali je ta prazen ali ne.

Naprej. Tole je nepotrebno zapletanje:

```
if s[0] % 2 == 1:
    return lihih(s[1:]) + 1
else:
    return lihih(s[1:])
```

Napišemo lahko preprosto

```
return lihih(s[1:]) + s[0] % 2
```

Zloživši to skupaj, dobimo krajšo rešitev.

[6]:
```
def lihih(s):
    return bool(s) and (s[0] % 2 + lihih(s[1:]))
```

0.2 Dodatna

Napiši rekurzivno funkcijo `vsaj_n_lihih(s, n)`, ki vrne `True`, če seznam `s` vsebuje vsaj `n` lihih števil.

Funkcija naj seveda ne kliče funkcije `lihih`.

Namig: da boš lepo sprogramirala (ali sprogramiral), razmisli, koliko lihih elementov mora vsebovati ostanek seznama (to je, `s[1:]`).

0.2.1 Rešitev

Če je `n` enak `0`, je elementov dovolj. Sicer pa mora biti seznam neprazen in ostanek mora vsebovati toliko lihih elementov, kolikor je `n` minus `1`, če je prvi lih.

[7]:
```
def vsaj_n_lihih(s, n):
    if not n:
        return True
    if not s:
        return False
    return vsaj_n_lihih(s[1:], n - s[0] % 2)
```

Zdaj pa samo še skrajšajmo:

[8]:
```
def vsaj_n_lihih(s, n):
    return not n or s and vsaj_n_lihih(s[1:], n - s[0] % 2)
```