Uvod v Računalništvo

Vaje 8

Luka Šveigl

1. DOMAČA NALOGA:

V sedmi vrstici algoritma za iskanje največjega števila (naloga 8.5) imamo operacijo:

Če A_i > doSedajNajvecji

Razloži, kaj bi se zgodilo, če bi zgornjo operacijo spremenili v:

- Če $A_i \geq$ doSedajNajvecji: V tem primeru bi algoritem vrnil zadnjo pojavitev največjega števila v seznamu, saj bi vrednost in položaj največjega števila nadomestil tudi v primeru, da sta največje št. in št. ki ga z njim primerjamo enaka.
- Če $A_i <$ doSedajNajvecji: V tem primeru bi algoritem vrnil prvo pojavitev najmanjšega števila v seznamu, saj bi vrednost in položaj "največjega števila" spremenil le v primeru, če je vrednost, ki jo primerjamo manjša od doSedajNajvečji.

Kaj lahko poveš o pomembnosti uporabe pravilnih primerjalnih operacij ($<$, $>$, $\leq$, $\geq$, $=$, $\neq$) pri pisanju pogojnih in iterativnih informacij:

Pravilna uporaba primerjalnih operatorjev je zelo pomembna, saj lahko najmanjša napaka povzroči, da algoritem vrača popolnoma drugačen, najvjretneje nepričakovan rezultat. Kot smo videli pri prvem zgornjem primeru, sprememba iz $>$ v $\geq$ povzroči, da algoritem vrne zadnjo pojavitev največjega št. v seznamu, namesto prvega. Takšne napake oz. spremembe lahko popolnoma spremenijo delovanje algoritma, zato moramo zelo dobro razumeti, kakšen rezultat hočemo od algoritma in kakšen operator moramo uporabiti, da bomo tak rezultat dobili.

2. DOMAČA NALOGA:

Spodaj je predstavljen Evklidov algoritem za iskanje največjega skupnega delitelja dveh pozitivnih celih števil I in J:

Korak 1. Kot vhod dobiš dve pozitivni celi števili. Večje izmed njiju označi z *I*, manjše pa z *J*.

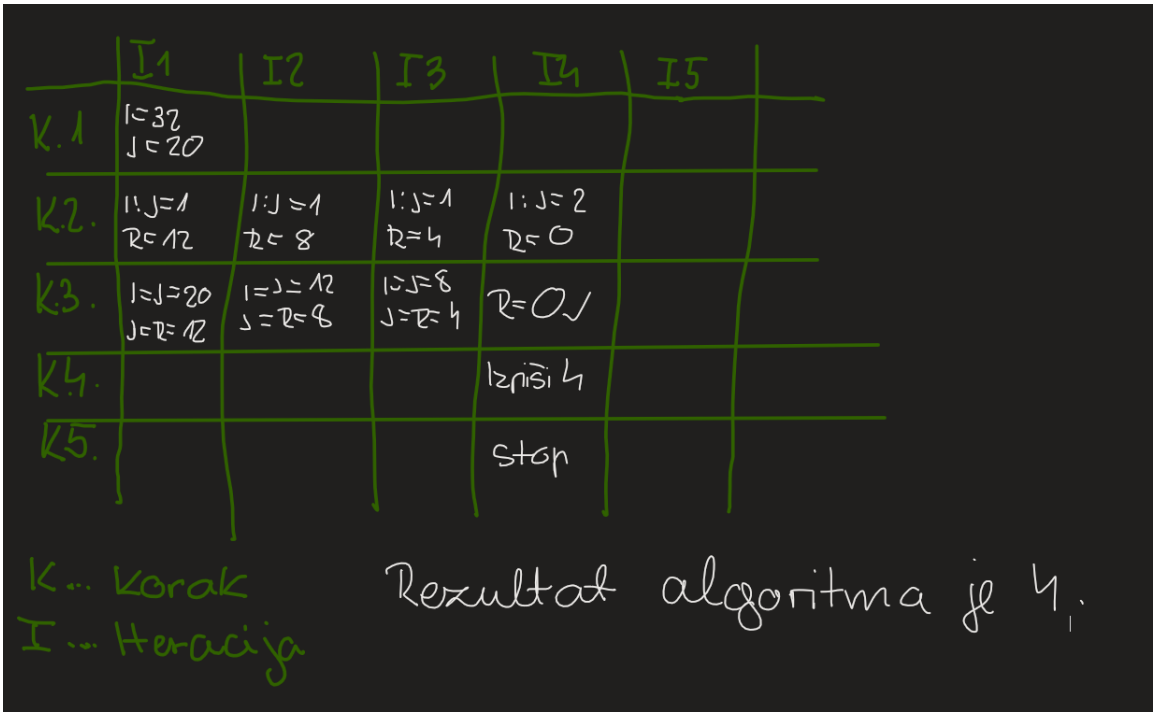
Korak 2. Deli *I* z *J*, ostanek označi z *R*.

Korak 3. Če *R* ni enak 0, nastavi *I* na vrednost enako *J*, *J* nastavi na vrednost enako *R* in pojdi na korak 2.

Korak 4. Izpiši odgovor, ki je kar enak vrednosti *J*-ja.

Korak 5. Ustavi se.

a) Pojdi skozi algoritem z vhodnima številoma 20 in 32. Po koncu vsakega koraka zabeleži vrednosti I, J in R. Poišči končni odgovor algoritma.



b) Ali algoritem deluje pravilno pri vhodu 0 in 32? Opiši, kaj točno se zgodi in spremeni algoritem tako, da vrne primerno sporočilo o napaki.

Pri vhodu 0 in 32 algoritem ne deluje pravilno, saj se v J (delitelj) zapiše število 0, s čimer pa ni možno deliti. Zaradi tega algoritem pri koraku 2 naleti na izjemo.

Popravljen alogritem:

Korak 1. Kot vhod dobiš dve pozitivni celi števili. Večje izmed njiju označi z I, manjše pa z J.

Korak 2. Če je J enako 0, vrni sporočilo “Deljenje z 0 ni mogoče” in ustavi program.

Korak 3. Deli I z J, ostanek zapiši v R.

Korak 4. Če R ni enak 0, nastavi I na vrednost enako J, J nastavi na vrednost enako R in pojdi na korak 2.

Korak 5. Izpiši odgovor, ki je kar enak vrednosti J.

Korak 6. Ustavi se

Algoritem popravimo tako, da pred korak z deljenjem vstavimo nov korak, ki preveri če je J enako 0 in v primeru da je, vrne sporočilo o napaki in ustavi delovanje programa.

3. DOMAČA NALOGA:

Napiši algoritem za izračun tedenskega honorarnega plačila za delo. Kot vhod algoritem dobi število opravljenih delovnih ur v tednu in urno postavko. Končno plačilo naj se določi tako, da se za vse ure do dopolnjene 40. ure upošteva osnovno (podano) urno postavko, za ure od dopolnjene 40. do dopolnjene 54. količnik 1.50 urne postavke in za ure od dopolnjene 54. dalje količnik 2.00 urne postavke. Algoritem naj izračuna in izpiše višino plačila. Po izpisu plačilca naj uporabnika tudi vpraša, ali želi opraviti naslednji izračun. Celotni postopek naj se ponavlja, dokler uporabnik ne odgovori z "NE".

Korak 1. Kot vhod dobiš dve pozitivni števili, prvo označi z opravljeneUre, drugo pa z urnaPostavka.

Korak 2. Spremenljivke placa, ureNad54 in ureNad40 postavi na 0.

Korak 3. Če je vrednost opravljeneUre večja od 54, odštej 54 od opravljeneUre. Razliko shrani v ureNad54, v opravljeneUre pa zapiši vrednost 54.

Korak 4. Če je vrednost opravljeneUre večja od 40, odštej 40 od opravljeneUre. Razliko shrani v ureNad40, v opravljeneUre pa zapiši vrednost 40.

Korak 5. Vrednost ureNad54 pomnoži z zmnožkom 2.00 in urnaPostavka in rezultat prištej oznaki placa.

Korak 6. Vrednost ureNad40 pomnoži z zmnožkom 1.5 in urnaPostavka in rezultat prištej oznaki placa.

Korak 7. Vrednost opravljeneUre pomnoži z urnaPostavka in rezultat prištej oznaki placa.

Korak 8. Izpiši odgovor, ki je enak vrednosti placa.

Korak 9. Izpiši sporočilo "Želite opraviti naslednji izračun?".

Korak 10. Sprejmi vhod uporabnika in ga označi z odgovor.

Korak 11. Če odgovor ni enak "NE" pojdi na korak 1.

Korak 12. Ustavi se.

Algoritem se seveda da tudi optimizirati, in odstraniti potencialno nepotrebne spremenljivke ureNad54 in ureNad40. Na 0 jih nastavimo zato, da v korakih 5 in 6 ne zabrede v nedefinirano delovanje, ker slučajno teh spremenljivk nebi bilo. Tudi če uporabnik vnese vrednost opravljeneUre, ki bo manjša kot npr. 54, se s spremenljivko ureNad54 ne bo zgodilo nič, zato pa bo rezultat pri množenju z urno postavko 0 in ne bo vplival na končni rezultat.

4. DOMAČA NALOGA:

Podana je funkcija:

```
(define (mystery input-list)
  (cond ((null? input-list) 0)
    (else ( + 1 (mystery (cdr  input-list))))))
```

a) Kaj bo rezultat ukaza **(mystery(list 3 4 5))**?

Rezultat tega ukaza bo 3.

b) Razložite kaj na splošnem dela funkcija.

Ta funkcija vrača velikost seznama, in sicer tako, da najprej pregleda če je seznam prazen in v tem primeru vrne 0, drugače pa se premika po seznamu in inkrementira števec velikosti, katerega nato vrne.