Algoritmi

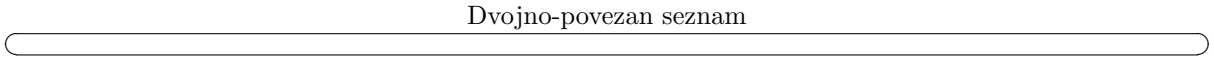
18. november 2011

Kazalo

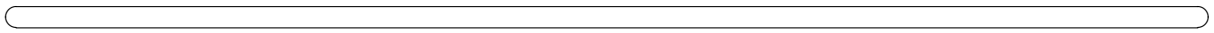
Poglavje 1

Podatkovne strukture

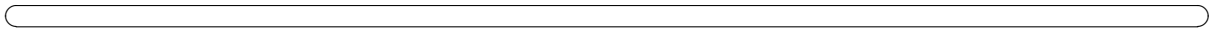
1.1 Dvojno-povezan seznam



1.2 Binarno drevo



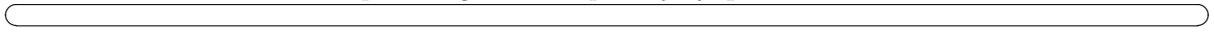
1.3 Ulomki



Poglavje 2

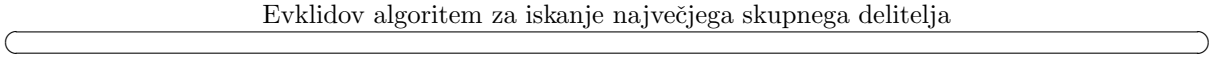
Teorija stevil

2.1 Praštevila



2.2 Največji skupni delitelj in najmanjši skupni večkratnik

2.2.1 Evklidov algoritem



2.2.2 Najmanjši skupni večkratnik

$$\mathrm{lcm(a,b)} = \frac{a * b}{\gcd(a,b)}$$

2.3 Fibonaccijevo zaporedje

Algoritem za izračun n-tega števila Fibonaccijevega zaporedja reda k

```
#red 2
memo = {0:0, 1:1}
def fib(n):
    if not n in memo:
        memo[n] = fib(n-1) + fib(n-2)
    return memo[n]

#red k
memok = {}
def fib(n,k):
    if not k in memok:
        memok[k] = dict(zip(range(k), [0 for i in range(k-1)]+[1]))
    if not n in memok[k]:
        memok[k][n] = sum([fib(n-i-1, k) for i in range(k)])
    return memok[k][n]
```

Poglavje 3

Iskalni algoritmi

3.1 Binarno iskanje

Algoritem binarnega iskanja oz. bisekcije v urejeni tabeli

Poglavje 4

Grafi

Graf formalno podamo kot dvojček vozlišč V in povezav E med vozlišči: $G = \langle V, E \rangle$.
Navadno uporabimo enega od naslednjih zapisov grafov:

- **Seznam sosednosti** – za vsako vozlišče hranimo seznam povezanih vozlišč (omogoča hrambo dodatnih podatkov v vozliščih)
- **Seznam pojavnosti** – za vsako vozlišče hranimo seznam njegovih povezav in za vsako povezavo njen seznam vozlišč (omogoča hrambo dodatnih podatkov v vozliščih in na povezavah)

- **Matrika sosednosti** – kvadratna matrika vozlišč, kjer $M_{i,j} \neq 0$ pomeni povezavo med vozliščema v_i in v_j. (omogoča hrambo ene vrednosti na povezavah)
- **Matrika pojavnosti** – matrika vozlišč, kjer $M_{i,j} \neq 0$ pomeni da ima vozlišče v_i povezavo e_j. (omogoča hrambo ene vrednosti na obeh koncih povezave)

4.1 Eulerjev in Hamiltonov graf

4.1.1 Eulerjev graf

Graf je Eulerjev, kadar vsebuje Eulerjev cikel – tak sprehod, ki vsako povezavo uporabi natanko enkrat in se na koncu vrne v izhodišče.

Psevdokoda Fleuryjevega algoritma za iskanje Eulerjevega cikla

```
izberi začetno vozlišče
prečkaj poljubno povezavo in jo odstrani (tu most prečkamo le če ni boljše izbire)
če je bil odstranjen most, odstrani vse točke ki so ostale izolirane
če je še kaka povezava, pojdi na 1
sicer zaključi algoritem.
```

4.1.2 Hamiltonov graf

Graf je Hamiltonov, kadar vsebuje Hamiltonov cikel – tak sprehod, ki vsako vozlišče obišče natanko enkrat in se na koncu vrne v izhodišče.

4.2 Iskanje maksimalnega pretoka skozi graf

Psevdokoda Ford-Fulkersonovega algoritma za iskanje maksimalnega pretoka skozi graf

```
vse pretoke nastavi na 0
označi izvorno vozliče
iz označenih vzemi poljubno vozlišče in ga zaznamuj kot obiskanega
označi vse neobiskane sosede do katerih obstaja nezasičena povezava
če je povečanje(ali zmanjšanje) toka po povezavi po kateri smo prišli v trenutno vozlišče ↩
    manjše od shranjenega, ga shrani
če obstajajo označena vozlišča in ponor še ni označen, pojdi na 2
če je ponor označen, pojdi po poti od ponora nazaj in popravi pretoke s shranjenim in pojdi ↩
    na 1
če ni več označenih vozlišč, zaključi algoritem.
```

4.3 Topološko urejanje

Psevdokoda algoritma za topološko urejanje grafa

```
naredi delovno kopijo grafa
poišči vozlišče brez vhodnih povezav
če takega vozlišča ni, zaključi algoritem z napako. (odkrili smo cikel)
sicer vozlišče označi z naslednjo oznako in ga odstrani iz grafa
če obstaja še kako vozlišče, pojdi na 1
sicer prenesi oznake na prvotni graf in zaključi algoritem.
```

4.4 Iskanje najcenejših poti

4.4.1 Dijsktrov algoritem

Psevdokoda Dijkstovega algoritma za iskanje najcenejših poti

```
izberi začetno vozlišče, označi ga kot obiskanjega in nastavi njegovo trenutno razdaljo na 0
ostala vozlišča označi kot neobiskana, nastavi trenutne razdalje na neskončno in jih dodaj v ↩
    vrsto neobiskanih
izračunaj dolžine poti od izbranega vozlišča do neobiskanih sosedov
če je dobljena dolžina do soseda kje manjša od že znane, jo shrani
izbrano vozlišče označi kot obiskano in ga odstrani iz vrste neobiskanih (razdalja do tu je ↩
    že minimalna)
če vrsta neobiskanih ni prazna, izberi vozlišče z najmanjšo trenutno razdaljo in pojdi na 3
sicer zaključi algoritem.
```

Poglavje 5

Računska geometrija

5.1 Najbližji par v množici točk

Preprost algoritem za iskanje najbližjega para točk v množici točk s pometanjem

Poglavje 6

Random stuff

6.1 Reševanje enačb

Algoritem za reševanje linearnih enačb

```
1  def solve(eq,var='x'):
2      eq1 = eq.replace("=","-(")+")"
3      c = eval(eq1,{var:1j})
4      return -c.real/c.imag
```

6.2 Možne vsote podmnožic

Algoritem za izračun vseh možnih vsot podmnožic neke množice

6.3 Hanojski stolpi

Rekurzivni algoritem za reševanje igre Hanojski stolpi

6.4 CRC varnostno kodiranje

Algoritem za izračun CRC varnostnih bitov

6.5 Levenshteinova razdalja med nizi

Vrste urejanj so:

- brisanje znaka: $d_L(\text{"}aaa\text{"},\text{"}aa\text{"}) = 1$
- vstavljanje znaka: $d_L(\text{"}aaa\text{"},\text{"}aaaa\text{"}) = 1$
- spremembo znaka: $d_L(\text{"}aaa\text{"},\text{"}aba\text{"}) = 1$

Algoritem za izračun Levensteinove razdalje med dvema nizoma