

Algoritmi

16. november 2011



This work is licensed under a Creative Commons
Attribution-NonCommercial-ShareAlike 3.0
Unported License

Kazalo

| | | |
|----------|---|----------|
| 1 | Podatkovne strukture | 2 |
| 1.1 | Dvojno-povezan seznam | 2 |
| 1.2 | Binarno drevo | 2 |
| 1.3 | Ulomki | 2 |
| 2 | Teorija števil | 2 |
| 2.1 | Praštevila | 2 |
| 2.2 | Največji skupni delitelj in najmanjši skupni večkratnik | 2 |
| 2.3 | Fibonaccijevo zaporedje | 3 |
| 3 | Iskalni algoritmi | 3 |
| 3.1 | Binarno iskanje | 3 |
| 4 | Grafi | 3 |
| 4.1 | Eulerjev in Hamiltonov graf | 4 |
| 4.2 | Iskanje maksimalnega pretoka skozi graf | 4 |
| 4.3 | Topološko urejanje | 4 |
| 4.4 | Iskanje najcenejših poti | 4 |
| 5 | Računska geometrija | 5 |
| 5.1 | Najbližji par v množici točk | 5 |
| 6 | Random stuff | 5 |
| 6.1 | Možne vsote podmnožic | 5 |
| 6.2 | Hanojski stolpi | 5 |
| 6.3 | CRC varnostno kodiranje | 5 |
| 6.4 | Levenshteinova razdalja med nizi | 5 |

Poglavje 1

Podatkovne strukture

1.1 Dvojno-povezan seznam

Dvojno-povezan seznam

1.2 Binarno drevo

Binarno drevo

1.3 Ulomki

Ulomki

Poglavje 2

Teorija števil

2.1 Praštevila

Preprosti algoritem za preverjanje praštevilstosti

2.2 Največji skupni delitelj in najmanjši skupni večkratnik

2.2.1 Evklidov algoritem

Evklidov algoritem za iskanje največjega skupnega delitelja

2.2.2 Najmanjši skupni večkratnik

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

2.3 Fibonaccijevo zaporedje

Algoritem za izračun n -tega števila Fibonaccijevega zaporedja reda k

```
1 #red 2
2 memo = {0:0, 1:1}
3 def fib(n):
4     if not n in memo:
5         memo[n] = fib(n-1) + fib(n-2)
6     return memo[n]
7
8 #red k
9 memok = {}
10 def fib(n,k):
11     if not k in memok:
12         memok[k] = dict(zip(range(k), [0 for i in range(k-1)]+[1]))
13     if not n in memok[k]:
14         memok[k][n] = sum([fib(n-i-1, k) for i in range(k)])
15     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

```

0 izberi začetno vozlišče
1 prečkaj poljubno povezavo in jo odstrani (tu most prečkamo le če ni boljše izbire)
2 če je bil odstranjen most, odstrani vse točke ki so ostale izolirane
3 če je še kaka povezava, pojdi na 1
4 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

```

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

4.3 Topološko urejanje

Psevdokoda algoritma za topološko urejanje grafa

```

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

4.4 Iskanje najcenejših poti

4.4.1 Dijkstraov algoritem

Psevdokoda Dijkstovega algoritma za iskanje najcenejših poti

```

0 izberi začetno vozlišče, označi ga kot obiskanega in nastavi njegovo trenutno razdaljo na 0
1 ostala vozlišča označi kot neobiskana, nastavi trenutne razdalje na neskončno in jih dodaj v  $\leftrightarrow$ 
    vrsto neobiskanih
2 izračunaj dolžine poti od izbranega vozlišča do neobiskanih sosedov
3 če je dobljena dolžina do sosedu kje manjša od že znane, jo shrani
4 izbrano vozlišče označi kot obiskano in ga odstrani iz vrste neobiskanih (razdalja do tu je  $\leftrightarrow$ 
    že minimalna)
5 če vrsta neobiskanih ni prazna, izberi vozlišče z najmanjšo trenutno razdaljo in pojdi na 3
6 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 Možne vsote podmnožic

Algoritem za izračun vseh možnih vsot podmnožic neke množice

6.2 Hanojski stolpi

Rekurzivni algoritem za reševanje igre Hanojski stolpi

6.3 CRC varnostno kodiranje

Algoritem za izračun CRC varnostnih bitov

6.4 Levenshteinova razdalja med nizi

Vrste urejanj so:

- brisanje znaka: $d_L("aaa", "aa") = 1$

- vstavljanje znaka: $d_L("aaa", "aaaa") = 1$
- spremembo znaka: $d_L("aaa", "aba") = 1$

Algoritem za izračun Levensteinove razdalje med dvema nizoma
