Vadnica visokih strokovnih algoritmov in podatkovnih struktur 1

(delovni osnutek, samo za interno uporabo)

Jurij Mihelič

11. november 2019

Začenjate s prebiranjem zbirke nalog za predmet *Algoritmi in podatkovne strukture* za visokošolski strokovni študijski program. Gre za delovni osnutek, kar pomeni, da naloge pogosto niso dovolj dobro izpiljene in preverjene. Prav tako besedilo verjetno vsebuje še veliko slovničnih in drugih napak. Da študentom olajšam učenje snovi in opravljanje izpita, sem osnutek vseeno javno objavil. Želim vam mnogo prijetnih algoritmičnih uric.

Naslov: Vadnica visokih strokovnih algoritmov in podatkovnih struktur 1
Različica: 0
Datum: 11. november 2019

Avtor: Jurij Mihelič
Elektronska pošta: jurij.mihelic@fri.uni-lj.si

Laboratorij za algoritmiko
Fakulteta za računalništvo in informatiko
Univerza v Ljubljani
Večna pot 113, 1000 Ljubljana

Kazalo

1 Problemi in algoritmi

1.1 Osnovni pojmi

1.1 Zakaj je *mestni* (arabski oz. indijski) zapis števil tako pomemben (tudi za algoritmiko)? Namig: predstavljajte si algoritem za seštevanje (ali pa množenje) rimskih številk.

1.2 Kaj je *število* (angl. number), *številka* (angl. numeral) in *števka* (angl. digit)? In kaj je *cifra* in kaj *mož*?

1.3 Kaj je več 2^{10} ali 10^2?

1.4 Kaj je *bit* in kaj je *bajt*? Kaj je več 42 kB ali 42 KiB? Koliko bitov je v 42 MiB?

1.5 Katere osnove navadno uporabljajo logaritmske funkcije v algoritmiki: $\log n$, $\lg n$ in $\ln n$?

1.6 Od kje oz. od koga pride izraz *algoritem*? S kakšnimi algoritmi se je ukvarjala dotična oseba?

1.7 Opredeli (intuitivno, vendar natančno) pojem *algoritma*. Obrazloži pomembne dele definicije.

1.8 Sošolki povej en *dvoumen* in en *nejasen* stavek (vendar ne oboje hkrati). Kaj je zahtevnejše: iskanje sošolke ali stavka?

1.9 Kaj je *računski problem*? Podaj primer računskega problema, ki ni povezan z računanjem.

1.10 Pojasni razliko med *problemom* (kadar v algoritmiki rečemo problem imamo v mislih računski problem), *nalogo* in *rešitvijo*.

1.11 Naštej in obrazloži vrste *računskih problemov*:

- iskalni,
- odločitveni,
- preštevalni,
- naštevalni in
- optimizacijski.

Za vsako vrsto podaj primer ali dva.

1.12 Preveri veljavnost trditve:

a) Seštevanje, odštevanje, množenje dve števil so računski problemi, iskanje najmanjšega elementa v seznamu števil pa ni.

b) Za dana števila x, y, z je vprašanje ali je $x + y = z$ odločitiveni problem.

c) Ali v danem seznamu elementov obstaja dani element je iskalni problem.

d) Urejanje seznama $5, 2, 9, 3$ je računski problem.

e) Naloga problema poišči najbližjo točko koordinatnemu središču je seznam točk $(3, 2), (1, 5), (4, -2)$

1.13 Zakaj je Turingov stroj pomemben za algoritmiko? Oglej si poljuben film o Alanu Turingu.

1.2 Snovanje in implementacija algoritmov

1.14 Kaj je *predpogoj* za dobro snovanje algoritmov?

1.15 Obrazloži nekaj kriterijev po katerih ocenjujemo kakovost algoritmov. Kateri kriterij je najpomembnejši?

1.16 Naštej in primerjaj načine (*opisni jeziki*) za opis algoritmov. Kateri načini so primernejši za ljudi in kateri za računalnike?

1.17 Sošolki v naravnem jeziku obrazloži algoritem za iskanje največjega elementa v tabeli? Nato skupaj narišita diagram poteka za ta algoritem.

1.18 Obrazloži faze razvoja algoritma od idejene zasnove do njegovega izvajanja. Obrazloži posamezne stopnje in semantične vrzeli med njimi. Kateri del je bolj abstrakten in kateri manj?

1.19 Naštej nekaj pristopov oz. *metod za snovanje* algoritmov. Več zabave s tem bo v sledečih poglavjih.

1.20 Kaj je *sintaktična* in kaj *semantična* napaka v programu? Kaj je *programski hrošč?*

1.21 Naštej nekaj načinov za *razhroščevanje* kode?

1.22 Kaj je *profiliranje* in kaj *instrumentacija* kode?

1.23 Kaj je *sled* algoritma?

1.24 Ali za izvajanje algoritma vedno potrebujemo računalnik? Obrazloži.

1.3 Algoritmi od vsepovsod

1.25 *Največji in najmanjši element* Zasnuj algoritme za iskanje največjega in najmanjšega elementa ter oboje hkrati. Kateri izmed algoritmov naredi manj primerjav elementov?

1.26 *Zaporedno iskanje* Zasnuj algoritem za iskanje danega elementa v dani tabeli. V čem je razlika v nalogi tega problema v primerjavi s problemom "največji in najmanjši element" iz predhodne naloge?

1.27 *Ugani število* S sošolko igrajta igro "ugani število": zamisli si število med 1 in 128, ona pa naj ugiba, možni odgovori so manjše, enako, večje. Koliko ugibanj potrebuje v najslabšem primeru v različnih pristopih, npr. zaporedno iskanje, razpolavljanje (bisekcija).

1.28 Načelo razpolavljanja (bisekcija) je eno izmed najbolj uporabnih načel v algoritmiki (in življenju nasploh). Kje se še uporablja?

1.29 *Dvojiško iskanje* Zasnuj algoritem *dvojiško iskanje*, ki uporablja načelo razpolavljanja, za iskanje števila v urejenem zaporedju. V čem je razlika med nalogo tega problema in nalogo problema "zaporedno iskanje" iz naloge →26? Zapiši tako rekurzivno kot iterativno obliko algoritma.

1.30 *Množenje s prištevanjem* Zasnuj algoritem za množenje dveh števil preko prištevanja. Namig: pomagaj si z definicijo množenja $a \cdot b = \underbrace{b + b + \cdots + b}_{a-\text{krat}}$.

1.31 Kdo je bil Evklid iz Aleksandrije? S čim se je še ukvarjal poleg algoritmov?

1.32 Opiši *Evklidov algoritem* za iskanje *največjega skupnega delitelja*. Zapiši tako rekurzivno kot iterativno obliko algoritma.

1.33 Opiši še en algoritem za iskanje *največjega skupnega delitelja*, ki deluje preko faktorizacije števil.

1.34 Prikaži sled Evklidovega algoritma za števili a) 123 in 456, b) 321 in 654 ter b) 59 in 61.

1.35 Kaj se zgodi po prvem koraku Evklidovega algoritma, če je prvo število manjše od drugega?

1.36 S pomočjo *Eratostenovega sita* izračunaj praštevila manjša od $N = 42$.

1.37 *Faktoriela* Zapiši rekurzivni algoritem za izračun faktoriele glede na formulo $n! = n \cdot (n-1)!$ in $0! = 1$. Ali zapisani algoritem vsebuje repno rekurzijo? Če ne, ga spremeni, da jo bo, nato pa vse skupaj spremeni v iteracijo. Opazuj spremembe!

1.4 Preverjanje pravilnosti

1.38 Na svetovnem spletu poišči nekaj primerov znanih programskih hroščev.

1.39 Kaj je poglavitno vprašanje (intuitivno), ki si ga postavimo, ko preverjamo pravilnost nekega algoritma?

1.40 Naštej (štiri) načine s katerimi lahko preverjamo pravilnost algoritmov.

1.41 Utemelji pravilnost algoritma "množenje s prištevanjem" (glej nalogo →30) preko *intuitivnega razumevanja*. Ali algoritem deluje za negativna števila?

1.42 Zakaj se pri razvoju programov zelo pogosto uporablja testiranje s testnimi primeri? Zakaj za testiranje algoritmov to pogosto ni zadostno?

1.43 Koliko različnih vhodov je možnih za Evklidov algoritem, če privzamemo 32-bitna števila? Koliko let bi trajalo popolno testiranje algoritma, če imamo na voljo testni sistem, ki vsako sekundo preizkusi miljardo (10^9) vhodov?

1.44 Algoritem za kvadriranje kvadratnih matrik velikosti $n \times n$ želimo *popolno testirati* s testnimi primeri. Če so vsi elementi matrik 8 bitna števila, zapišite (v odvisnosti od n)

a) število različnih vhodov in

b) število različnih vhodov, če namesto kvadriranja vzamemo potenciranje.

1.45 Dano je polje dolžine 200. Tina Sredinec je implementirala algoritem, ki izračuna sredinsko pozicijo m v polju med pozicijama l (leva meja) in r (desna meja), po formuli $m = (l + r)/2$. Vse spremenljivke vsebujejo 8 bitna nepredznačena števila. Kje se skriva programski hrošč? Kako bi program popravil?

1.46 Ugotoviti želimo pravilnost nekega algoritma za urejanje seznama. Kateri dve lastnosti moramo preveriti?

1.47 V čem je prednost *formalnega dokazovanja pravilnosti* algoritmov. Na katerem matematičnem načelu sloni dokazovanje pravilnosti algoritmov, ki vsebujejo zanke?

1.48 Formalni dokaz pravilnosti algoritma pogosto temelji na indukciji. Kaj je *matematična indukcija*, *hipoteza*, *osnovni primer*, *induktivna predpostavka* in *induktivni korak*? V algoritmiki pa za dokazovanje zank uporabljamo tudi *zančne invariante*.

1.49 S pomočjo matematične indukcije dokaži $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

1.50 S pomočjo indukcije dokaži pravilnost algoritma za iskanje maksimuma v tabeli števil.

```
m = a[0]
for i = 1 to n−1 do
    if a[i] > m then m = a[i]
```

1.51 S pomočjo indukcije dokaži pravilnost "množenja s prištevanjem".

1.52 Dokaži pravilnost Evklidovega algoritma. Uporabite znani izrek v zvezi s tem.

1.5 Namigi in rešitve izbranih nalog

1.5 Dogovor je naslednji: $\log n = \log_{10} n$, $\lg n = \log_2 n$ in $\ln n = \log_e n$.

1.14 Dobro razumevanje problema preko natančne (matematične) definicije.

1.15 Pravilnost, učinkovitost, prilagodljivost, enostavnost, implementabilnost. Pravilnost je temelj vsakega algoritma.

1.34 a)

#	a	b	q	r
0	123	456 0	123	
1	456	123	3	87
2	123	87	1	36
3	87	36	2	15
4	36	15	2	6
5	15	6	2	3
6	3	2	0	
7	3	0		

1.37 Ne-repna rekurzija: fac, repna rekurzija: factail

```
fun fac(n) is
    if n == 0 then 1 else n * fac(n − 1)
```

```
fun factail (r, n) is
    if n == 0 then r else factail (r * n, n − 1)
```

1.39 Ali program deluje, tako kot mislimo, da bi moral delovati?

1.43 Št. vhodov: $2^{64} \approx 1,8 \cdot 10^{19}$, čas testiranja: $2^{64}/10^9/60/60/24/365 = 585$ let.

1.44 a) 256^{n^2} - za vsak element v matriki je 2^8 različnih števil, vseh elementov je n^2, b) 256^{n^2+1} poleg matrike je vhodni podatke tudi potenca

1.45 Za $l = 150$ in $r = 190$ dobimo $l + r = 340 \pmod{256} = 84$ in torej $m = (l + r)/2 = 42$, kar očitno ni sredina med 150 in 180. Popravek: $m = l + (r - l)/2$.

1.46 Da rezultat vsebuje enake elemente kot vhodno zaporedje in da je rezultat urejen seznam.

1.47 Zanesljivost pravilnosti, indukcija.

2 Zahtevnost algoritmov

2.1 Splošna vprašanja

2.53 Kaj je zahtevnost algoritma?

2.54 Naštej nekaj virov, ki jih algoritem lahko potrebuje za svoje izvajanje.

2.55 Kateri viri ustrezajo meri *časa* in kateri *prostora*?

2.56 Kaj je *Von Neumannov* model računalniške arhitekture?

2.57 Kaj je RAM model računanja? Zakaj ga uporabljamo v algoritmiki?

2.58 Kaj je natančna zahtevnost in kaj asiptotična zahtevnost algoritma?

2.59 Od česa je lahko odvisna zahtevnost algoritma? Prikaži s primerom.

2.60 Izberi nek problem, nato naštej nekaj primerov nalog zanj, katerih težavnost je različna:

a) glede na velikost naloge

b) glede na podatke v sami nalogi.

2.61 Glede na (vhodne) podatke, katere vrste določanja zahtevnosti poznamo?

2.62 Zakaj najpogosteje uporabljamo zahtevnost v najslabšem primeru?

2.63 Kdo je bil John von Neumann? S čim vsem se je ukvarjal? Katere izmed algoritmov za urejanje je napravil?

2.2 Natančna zahtevnost

2.64 Koliko korakov zahteva algoritem ”množenje s prištevanjem“ (glej nalogo 30)?

2.65 Koliko korakov zahteva Eratostenovo sito za poljuben N? En korak je mišljen kot odstranjevanje večkratnikov nekega števila X.

2.66 Določi natančno zahtevnost v številu primerjav elementov glede na najboljši, najslabši in povprečni primer za algoritem (zaporednega iskanje, glej →26). Pri tem je n velikost polja a.

```
for i = 0 to n−1 do
    if a[i] == key then return i
return −1
```

2.67 Zakaj se kot pomembna operacija, s katero ocenjujemo zahtevnost, pogosto pojavi primerjava elementov, primerjavo indeksov pa navadno zanemarimo?

2.68 Določi natančno zahtevnost v smislu realnega časa na poljubnem RAM modelu računanja za algoritem "zaporedno iskanje" iz naloge →26

2.69 Kolikšna je globina rekurzije pri algoritmu "dvojiškega iskanja"?

2.70 Kolikokrat natančno se izvede primerjava indeksov v zanki for, katere spodnja meja je A in zgornja B? Kolikokrat pa se izvede telo zanke?

2.71 Kolikokrat se izvede telo notranje zanke v spodnjih algoritmih? Kakšna je vrednost spremenljivke s? Podaj ugotovitve glede na parametre funkcije.

```
fun a(n) is
    s = 0
    for (i = 0; i < 7 * n; i++) do
        s++
```

```
fun b(n) is
    s = 0
    for (i = 0; i < 8 * n; i += 2) do
        s++
```

```
fun c(n, p) is
    s = 0
    for (i = 0; i < n; i += p) do
        s++
```

```
fun d(m, n, p) is
    s = 0
    for (i = m; i < n; i += p) do
        s++
```

```
fun e(n) is
    s = 0
    for (i = 1; i < n; i *= 2) do
        s++
```

```
fun f(n, p) is
    s = 0
    for (i = 1; i < n; i *= p) do
        s++
```

```
fun g(n, m, p) is
    s = 0
    for (i = 0; i < 3 * n; i += p) do
        for (j = 0; j < m; j += 2 * p) do
            s++
```

```
fun h(n) is
    s = 0
    for (i = 1; i <= n; i++) do
        for (j = 1; j <= i; j++) do
            s++
```

2.3 Asimptotična zahtevnost

2.72 Zapiši formalne definicije za O, Ω in Θ notacijo. Kaj pravzaprav s tem definiramo: relacijo, funkcijo ali množico?

2.73 Zapiši formalne definicije za o, ω notacijo. V čem se o notacija razlikuje od O

notacije in ω od Ω?

2.74 Obrazloži podobnosti in razlike med naslednjimi pojmi:

- zgornja asimptotična meja,
- tesna zgornja asimptotična meja,
- spodnja asimptotična meja,
- tesna spodnja asimptotična meja in
- tesna asimptotična meja.

2.75 Kako v praksi *zlorabljamo* asimptotično notacijo in relacijo $\in$?

2.76 Kako si razlagamo *veriženje* asimptotične notacije? Npr.

$$5n^4 + 3n^2 + n = 5n^4 + O(n^2) = O(n^4).$$

2.77 Dana je časovna zahtevnost $T(n) = 5n^3 + 3n^2 + 2$. Kaj od naštetega velja?

a) $T(n) = O(n \lg n)$	d) $T(n) = \Theta(n \lg n)$	g) $T(n) = \Omega(n \lg n)$
b) $T(n) = O(n^3)$	e) $T(n) = \Theta(n^3)$	h) $T(n) = \Omega(n^3)$
c) $T(n) = O(n^9)$	f) $T(n) = \Theta(n^9)$	i) $T(n) = \Omega(n^9)$

2.78 Za dani funkciji obkroži veljavne trditve:

$$f(n) = 9999n^2 + 999n \log n + 99 \quad \text{in} \quad g(n) = n^3 + 2n^2 + 3.$$

a) $f(n) = O(n^{99})$	b) $g(n) = \Omega(1)$	c) $f(n)$ narašča hitreje kot $g(n)$
d) $f(n) = O(n^2)$	e) $g(n) = \Omega(n^3)$	f) $f(n) + g(n) = O(g(n))$
g) $O(f(n)) = n^2$	h) $g(n) = \Omega(2^n)$	i) $f(n) \cdot g(n) = \Theta(9999n^2)$
j) $f(n) = O(n^{1.618})$	k) $g(n) = O((1 + 1 + 1)^n)$	l) $g(n)/f(n) = O(n \lg n)$

2.79 Katere izmed spodnjih trditev so resnične?

a) $\sqrt{4}\, n^4 \log^4 n + 4n^4 = \Omega(n^3 \log^3 n)$	b) $3n^2 + 2n + 1 = \Omega(n \log^{12} n)$
c) $2^{3456} = O(\log n)$	d) $n^{2/3} = O(n^{0{,}666})$
e) $\Omega(n^{\lg 4}) = (\sqrt{16})^{\lg n}$	f) $27^{\log_3 n} = \Theta(n^3)$
g) $\sum_{i=1}^{n} O(n) = \Theta(n^2)$	h) $(n + \lg n)^{42} - n^{42} = \Omega(n^{42})$

2.80 Pokaži po definiciji in, da velja:

a) $5n^2 - 12n = O(n^2)$

b) $5n^2 + 12n = O(n^2)$

c) $3n^3 + 5n^2 - 6n = \Theta(n^3)$.

2.81 Reši predhodno nalogo še z uporabo limit.

2.82 Dokaži, da za asimptotično notacijo velja tranzitivnost:

$$f(n) = O(g(n)) \land g(n) = O(h(n)) \implies f(n) = O(h(n)).$$

Za katere vrste notacij ($\Omega, \Theta, o, \omega$) še velja podobno?

2.83 Za različne asimptotske zahtevnosti izračunaj zahtevnost pri dvakrat večji nalogi. Podaj rešitev glede na zahtevnost pri prvotni velikosti.

a) $\Theta(n)$

b) $\Theta(n^2)$

c) $\Theta(n^3)$

d) $\Theta(\lg n)$

e) $\Theta(n \lg n)$

f) $\Theta(2^n)$

2.84 (Moorov zakon) Dan je nek algoritem za nek problem. Kako veliko nalogo lahko rešimo na dvakrat hitrejšem računalniku v enakem času? Če je časovna zahtevnost algoritma:

a) logaritemska

b) polinomska

c) eksponentna.

Še nekaj težjih nalog

2.85 Dokaži $f(n) = O(g(n)) \iff g(n) = \Omega(f(n))$

2.86 Dokaži $f(n) = O(g(n)) \land f(n) = \Omega(g(n)) \iff f(n) = \Theta(g(n))$

2.87 Dokaži $f(n) = o(g(n)) \implies f(n) = O(g(n))$

2.88 Dokaži $f(n) = \omega(g(n)) \implies f(n) = \Omega(g(n))$

2.89 Kako se definicije asimptotičnih notacij z limitami povezujejo s formalnimi definicijami? Pokaži, da

$$f(n) = O(g(n)) \iff \lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$

Podobno razmisli še za ostale notacije.

2.90 Pokaži, da so potence logaritma od zgoraj omejene s polinomi, torej $O(\log^a n) = O(n^b)$ za poljubni pozitivni konstanti a in b. Namig: limite in l'Hospitalovo pravilo.

2.91 Pokaži, da so polinomi od zgoraj omejeni z eksponentnimi funkcijami, torej: $O(n^b) = O(c^n)$, za poljubni pozitivni konstanti b in c.

2.4 Mojstrov izrek

2.92 Nek algoritem smo zasnovali po metodi *deli in vladaj*, pri čemer problem razdelimo na sedem podproblemov (iste vrste), od katerih je vsak polovične velikosti. Priprava podproblemov terja $5n^2 + 3n$ korakov in sestavljanje terja $15n$ korakov.

a) Zapiši časovno zahtevnost $T(n)$ algoritma z rekurenčno enačbo.

b) Reši enačbo s pomočjo znanega izreka.

c) Ali velja $T(n) = O(n^{2.807})$? Zakaj?

2.5 Namigi in rešitve izbranih nalog

2.53 Zahtevnost algoritma pove katere in koliko virov potrebuje algoritem za svoje izvajanje (v nekem modelu računanja).

2.59 Od algoritma, modela računanja in od velikosti vhoda in samih podatkov v vhodu.

2.65 Odstranjevanje večkratnikov X je potrebno do: hitro vidimo, da za $X < N$ ali tudi $X < N/2$. Z malo razmislega pa pridemo do $X \leq \sqrt{N}$.

2.66 Najboljši primer: 1, najslabši primer: n in povprečni primer $(n + 1)/2$.

2.68 Najboljši primer: $c_1 + c_2 + c_3$, najslabši primer: $c_1 \cdot (n + 1) + c_2 \cdot n + c_3$, povprečni primer: $\frac{c_1+c_2}{2}n + \frac{c_1+c_2}{2} + c_3$, kjer je c_1 zahtevnost preverjanja pogoja v odločitvenem stavku, c_2 cena primerjave elementov in c_3 cena stavka return.

2.70 Primerjava indexov: $B - A + 2$, telo zanke $B - A + 1$

2.71 a) $s = 7n$, b) $s = 4n$, c) $s = \lceil n/p \rceil$, d) $s = \lceil \frac{n-m}{p} \rceil$, e) $s = \lceil \lg n \rceil$, f) $s = \lceil \log_p n \rceil$, g) $s = 3/2nm/p^2$, h) $s = n(n + 1)/2 \sim n^2/2$

2.77 b) c) e) g) h)

2.78 a) b) d) e) f) k) l)

2.83 a) $2cn$, $2\times$, b) $4cn^2$, $4\times$, c) $7cn^3$, $8\times$, d) $c \lg(2n) = c + c \lg n$, $+c$, e) $c2n \lg(2n) = 2cn + 2cn \lg n$, $+cn$, $2\times$, f) $c2^{2n}$, $\times 2^n$

2.84 Izhajamo iz: $T_2(n) = 1/2T_1(n)$ (2x hitrejši) in $T_2(n_2) = T_1(n_1)$ (na voljo imamo enako časa). Torej $T_1(n_2) = 2T_1(n_1)$.

a) log: $T(n) = O(\lg n)$: $c_1 \lg n_2 = 2c_1 \lg n_1$, torej $n_2 = n_1^2$.

b) poly: $T(n) = O(n^k)$: $c_1 n_2^k = 2c_1 n_1^k$ oz. $n_2 = \sqrt[k]{2} n_1$. Razmišljaj: $k = 1$ torej 2x večjo nalogo, $k = 2$ torej $\sqrt{2}$ večjo nalogo.

c) exp: $T(n) = O(2^n)$: $c_1 2^{n_2} = 2c_1 2^{n_1} = 2^{n_1+1}$, torej $n_2 = n_1 + 1$

2.90

$$a, b \in \mathbb{N} : \lim_{n\to\infty} \frac{\ln^a n}{n^b} = \lim_{n\to\infty} \frac{a/n \ln^{a-1} n}{bn^{b-1}} = \lim_{n\to\infty} \frac{a/n \ln^{a-1} n}{bn^b} = \lim_{n\to\infty} \frac{a!}{b^a n^b} = 0.$$

2.91

$$a, b \in \mathbb{N} : \lim_{n\to\infty} \frac{n^a}{b^n} = \lim_{n\to\infty} \frac{an^{a-1}}{b^n \ln b} = \lim_{n\to\infty} \frac{a!}{b^n \ln^a b} = 0.$$

2.92 a) $T(n) = 7T(n/2) + 5n^2 + 18n$ oz. $T(n) = 7T(n/2) + O(n^2)$, b) $a = 7, b = 2, d = 2$, master theorem: $a > b^d$, torej $T(n) = O(n^{\log_2 7}) = O(n^{2.808})$ c) ne, ker $\log_2 7 > 2.807$

3 Osnovne podatkovne strukture

3.1 Sklad

3.93 Profesor Mate Matik želi preveriti ali je dano zaporedje oklepajev pravilno gnezdeno. Na primer npr. ()([]) in ()[{(()[])}] sta pravilno gnezdeni zaporedji, (()[]} in (([()]) pa nista. Pomagaj mu in zasnuj algoritem za ta problem.

3.94 Kvantni fiziki iz Palindromije so izdelali novo vrsto naprave, ki generira številčne nize naključnih palindormov. Na voljo je le operacija next(), ki vrne naslednji znak palindorma oz. -1, če je palindroma konec. Na voljo imaš le en sklad in vrsto, zasnuj algoritem, ki preveri ali naprava res generira palindrom.

3.2 Povezani seznami

3.95 Enojno povezani seznam zaporedoma vsebuje elemente 3, 0, 4, 1, 5.

a) Seznam uporabimo kot vrsto in nad njim izvedemo naslednje operacije: dequeue, dequeue, enqueue(9), enqueue(2), dequeue, enqueue(6). Kakšen seznam dobimo?

b) Seznam uporabimo kot sklad in nad njim izvedemo naslednje operacije: pop, pop, push(9), push(2), pop, push(6). Kakšen seznam dobimo?

c) Uporabite predstavitev seznama s poljem kapacitete 8, pri čemer naj bo element i hranjen na indeksu i. Namig: zapišite polji item in next ter vrednosti first in free.

d) Zapišite funkcijo podvoji(), ki podvoji kapaciteto seznama (predstavljenega s poljem).

3.3 Drevesa

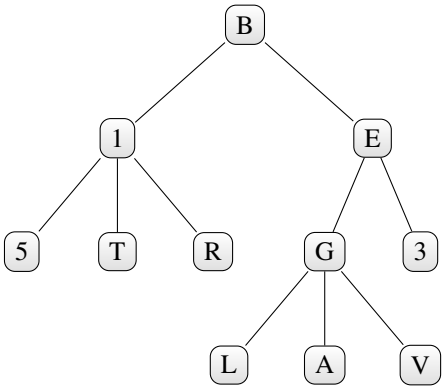
3.96 Celovito drevo stopnje tri je implicitno predstavljeno s poljem 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5.

a) Nariši drevo.

b) Zapiši enačbe za indekse vseh otrok vozlišča z indeksom i.

c) Zapiši zaporedje vozlišč drevesa, če izvedemo premi obhod drevesa.

3.97 Dano je naslednje drevo.



a) Zapiši stopnjo drevesa.

b) Najmanj koliko vozlišč bi moral dodati, da bi drevo postalo polno?

c) Zapiši vrstni red vozlišč pri obratnem obhodu drevesa.

3.98 Gradnja min kopice z zaporednim vstavljanjem.

a) Zgradi min kopico z zaporednim vstavljanjem števil iz naslednjega zaporedja

$$9, 3, 1, 9, 7, 6, 2, 7, 5.$$

Izriši drevo vsakič, ko vstaviš element v kopico. Jasno označi kopice.

b) Koliko (natančno) zamenjav se opravi v **najslabšem** primeru in koliko v **najboljšem** primeru pri vstavljanju i-tega elementa? Namig: elemente začni šteti z 1.

c) Zapiši primer zaporedij dolžine pet, kjer pride do najslabšega in do najboljšega primera.

d) Zapiši asimptotično časovno zahtevnost v najslabšem primeru za takšen način gradnje.

e) Zapiši algoritem, ki v konstantnem času vrne **drugi** najmanjši element kopice.

3.4 Namigi in rešitve izbranih nalog

3.95 a) 1,5,9,2,6 b) 6,9,4,1,5 pri obeh a) in b) smo upoštevali tudi, če ste dodajali/odvzemali z nasprotnega konca c) item: 0,1,-,3,4,5,-,- next: 4,5,6,0,1,-1,7,-1, first=3, free=2, d) Zanimivi del programa je del, ki popravi seznam prostih celic. Samo kopiranje elementov ni tako zanimivo.

3.96 a) Drevo narišemo po nivojih. b) $3i + 1, 3i + 2, 3i + 3$, c) 3,1,5,9,2,4,6,5,3,1,5

3.97 a) 3, b) 2, c) 5TR1LAVG3EB

3.98 a) 3 b) 3 c) 2 d) 2

4 Urejanje in izbiranje

4.1 Navadna urejanja

Pod navadna urejanja sodijo navadno izbiranje *(angl. selection sort),* navadne zamenjave*, imenovano tudi urejanje z mehurčki (angl. bubble sort), in* navadno vstavljanjem *(angl. insertion sort).*

4.99 *Navadno izbiranje* Zapiši sled urejanja z navadnim izbiranjem v nepadajočem vrstnem redu za vhodno zaporedje $3, 2, 8, 9, 1, 5, 4, 6, 0, 7$.

4.100 Zapiši sled urejanja z navadnim izbiranjem v nenaraščajočem vrstnem redu za vhodno zaporedje $3, 2, 8, 9, 1, 5, 4, 6, 0, 7$.

4.101 Koliko primerjav in zamenjav naredi navadno izbiranje na vhodnem zaporedju a) $0, 1, 2, 3, 4, 5, 6, 7, 8, 9$ in koliko na zaporedju b) $9, 8, 7, 6, 5, 4, 3, 2, 1, 0$?

4.102 Koliko natančno primerjav $C(n)$ naredi navadno izbiranje na zaporedju velikosti n?

4.103 Koliko asimptotično primerjav $C(n)$ naredi navadno izbiranje na zaporedju velikosti n? Odgovor zapiši tako s pomočjo tilda kot Θ notacije.

4.104 Koliko (natančno in asimptotično) zamenjav $S(n)$ naredi navadno izbiranje na zaporedju velikosti n?

4.105 Navadno izbiranje izboljšamo tako, da na vsakem koraku hkrati poiščemo najmanjši in največji element v še neurejenem delu zaporedja. Nato oba elementa postavimo (zamenjava) na ustrezno mesto. Zapiši sled urejanja za zaporedje

$$3, 2, 8, 9, 1, 5, 4, 6, 0, 7.$$

4.106 Na voljo imate algoritem za hkratno iskanje najmanjšega in največjega elementa, ki v zaporedju dolžine n porabi $2n - 2$ primerjav. Koliko natančno primerjav porabi s tem algoritmom izboljšano navadno izbiranje?

4.107 Na voljo imate algoritem za hkratno iskanje najmanjšega in največjega elementa, ki v zaporedju dolžine n porabi $3/2n - 2$ primerjav. Koliko natančno primerjav porabi s tem algoritmom izboljšano navadno izbiranje?

4.108 Navadno izbiranje želimo implementirati na enojno povezanem seznamu? Kolikšna je asimptotična časovna zahtevnost takega algoritma?

4.109 *Navadne zamenjave* Zapiši sled urejanja z navadnimi zamenjavami v nepadajočem vrstnem redu za vhodno zaporedje 3, 2, 8, 9, 1, 5, 4, 6, 0, 7.

4.110 Zapiši sled urejanja z navadnimi zamenjavami v nenaraščajočem vrstnem redu za naslednje vhodno zaporedje 3, 2, 8, 9, 1, 5, 4, 6, 0, 7.

4.111 Urejanje z navadnimi izmenjavami izboljšamo tako, da v postopek končamo, če v zadnji iteraciji ni prišlo do nobene zamenjave. Takšen postopek pravilno uredi poljubno zaporedje? Utemelji.

4.112 Urejanje z navadnimi izmenjavami izboljšamo tako, da v naslednji iteraciji delamo primerjave le do indeksa zadnje zamenjave na predhodni iteraciji.

4.113 *Navadno stresanje* TODO: Shaker sort - navadno stresanje - sled

4.114 *Navadno vstavljanje* Zapiši sled urejanja z navadnimim vstavljanjem v nepadajočem vrstnem redu za vhodno zaporedje 3, 2, 8, 9, 1, 5, 4, 6, 0, 7.

4.115 Katera izmed osnovnih navadnih urejanj v tem razdelku so stabilna? Utemelji!

4.116 *Črno beli diski* TODO: Levitin p.102. Danih je $2n$ diskov dveh barv: n črnih in n belih. Bele želimo spravit na levi konec in črna na desni konec. Dovoljena operacija je edino zamenjava dveh sosednjih diskov. Zasnuj algoritem za reševanje tega problema in določi število potrebnih zamenjav.

4.2 Napredna urejana

V tem razdelku se lotimo algortimov za urejanje, katerih časovna zahtevnost je boljša od kvadratne.

4.117 Izračunaj delilno zaporedje za *Shellovo urejanje* zaporedja 3141 števil, če je število zunanjih iteracij algoritma enako $t = \lfloor \log_2 n \rfloor - 1$ in $h_t = 1$ ter $h_{k-1} = 2h_k + 1$.

4.118 Uredi zaporedje 3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8, 9, 7, 9, 3, 2, 3, 8, 4, 6 s Shellovim urejanjem v naraščajočem vrstnem redu.

4.3 Namigi in rešitve izbranih nalog

4.99

	3	2	8	9	1	5	4	6	**0**	7
0	0	2	8	9	**1**	5	4	6	3	7
1	0	1	8	9	**2**	5	4	6	3	7
2	0	1	2	9	8	5	4	6	**3**	7
3	0	1	2	3	8	5	**4**	6	9	7
4	0	1	2	3	4	**5**	8	6	9	7
5	0	1	2	3	4	5	8	**6**	9	7
6	0	1	2	3	4	5	6	8	9	**7**
7	0	1	2	3	4	5	6	7	9	**8**
8	0	1	2	3	4	5	6	7	8	9

4.102

$$C(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} 1 = \sum_{i=0}^{n-2} (n-i-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}.$$

4.103

$$C(n) \sim \frac{n^2}{2} = \Theta(n^2).$$

4.104

$$S(n) = n - 1 \sim n = \Theta(n).$$

4.105

	3	2	8	**9**	1	5	4	6	**0**	7
0	0	2	**8**	7	**1**	5	4	6	3	9
1	0	1	3	**7**	**2**	5	4	6	8	9
2	0	1	2	**6**	**3**	5	4	7	8	9
3	0	1	2	3	**4**	**5**	6	7	8	9
4	0	1	2	3	4	5	6	7	8	9

4.106

$$C(n) = \sum_{i=0}^{\frac{n-2}{2}} (2(n-2i)-2) = \frac{n(n+2)}{2} - 2.$$

4.107

$$C(n) = \sum_{i=0}^{\frac{n-2}{2}} (n-2i) = \sum_{i=2}^{n} 2i$$

.

4.108 $\Theta(n^2)$. Najmanjši elementi si zapomnimo v kazalcu min. Zamenjavo izvedemo tako, da zamenjamo elementa (ne prevezujemo vozlišč).

4.109

	3	2	8	9	1	5	4	6	0	7
1	0	3	2	8	9	1	5	4	6	7
2	0	1	3	2	8	9	4	5	6	7
3	0	1	2	3	4	8	9	5	6	7
3	0	1	2	3	4	5	8	9	6	7
4	0	1	2	3	4	5	6	8	9	7
5	0	1	2	3	4	5	6	7	8	9
6	0	1	2	3	4	5	6	7	8	9
7	0	1	2	3	4	5	6	7	8	9
8	0	1	2	3	4	5	6	7	8	9

4.114

i										
	3	2	8	9	1	5	4	6	0	7
1	2	3	8	9	1	5	4	6	0	7
2	2	3	8	9	1	5	4	6	0	7
3	2	3	8	9	1	5	4	6	0	7
4	1	2	3	8	9	5	4	6	0	7
5	1	2	3	5	8	9	4	6	0	7
6	1	2	3	4	5	8	9	6	0	7
7	1	2	3	4	5	6	8	9	0	7
8	0	1	2	3	4	5	6	8	9	7
9	0	1	2	3	4	5	6	7	8	9

4.115

- Navadno izbiranje: ni stabilno, protiprimer 2, 2, 1;
- navadne zamenjave: je stabilno, enaki elementi se ne zamenjajo;
- navadno vstavljanje: je stabilno, vstavljamo kvečjemju do enakega elementa.

4.116 Uporabi navadne zamenjave ali navadno vstavljanje.

4.117 Število iteracij $t = 10$ in delilno zaporedje je $(1023, 511, 255, 127, 63, 31, 15, 7, 3, 1)$.

4.118 $t = 3$, delilno zaporedje $(7, 3, 1)$.

h	;3	1	4	1	5	9	2	6	5	3	5	8	9	7	9	3	2	3	8	4	6
7	;3	1	2	1	5	4	2	6	3	3	3	8	9	6	9	5	4	5	8	9	7
3	;1	1	2	2	3	3	3	4	4	3	5	5	5	6	7	8	6	8	9	9	9
1	;1	1	2	2	3	3	3	3	4	4	5	5	5	6	6	7	8	8	9	9	9

5 Osnovne metode snovanja algoritmov

5.1 Metode snovanja

5.119 Naštej nekaj metod snovanja algoritmov.

5.120 Pri katerih metodah snovanja algoritmov se osredotočamo na reševanje podproblemov.

5.2 Groba aritmetika

V tem razdelku najdemo nekaj nalog, ki temeljijo na uporabi metode grobe sile oz. uporabe definicije problema, za razvoj aritmetičnih algoritmov za nekatere osnovne aritmetične operacije, kot sta seštevanje in množenje. V nadaljevanju predpostavimo, da naravna števila vključujejo število 0.

5.121 *Seštevanje po bitih* Za dani n-bitni naravni števili a in b zasnuj algoritemi za izračun njune vsote, pri čemer kot osnovno operacijo uporabi seštevanje bitov.

5.122 Določi asimptotično časovno zahtevnost za algoritem iz predhodne naloge. Se da hitreje?

5.123 Algoritem iz predhodne naloge spremeni, da deluje za števili a in b v desetiškem zapisu.

5.124 Algoritem seštevanja iz predhodne naloge temelji na seštevanju kvečjemu treh števk (števki števili a, b in prenos). Pokaži, da je vsota treh desetiških števk kvečjemu dvomestna. Ali velja enako za števke v poljubni številski osnovi?

5.125 *Seštevanje preko operacij predhodnik in naslednik* Za dani n-bitni naravni števili a in b zasnuj algoritemi za izračun njune vsote, pri čemer kot osnovni operaciji privzemi $\text{pred}(i) = i - 1$, ki vrne prednika števila i, in $\text{succ}(i) = i + 1$, ki vrne naslednika števila i.

5.126 Določi asimptotično časovno zahtevnost za algoritem iz predhodne naloge.

5.127 *Množenje z zaporednim prištevanjem* Za dani n-bitni naravni števili a in b zasnuj algoritem za izračun njunega zmnožka $a \cdot b$ z uporabo seštevanja. Pri tem uporabi metodo grobe sile in definicijo zmnožka $a \cdot b = \underbrace{b + b \cdots + b}_{a \text{ seštevancev}}$.

5.128 Algoritem iz predhodne naloge razširi, da bo deloval pravilno za poljubni celi števili a in b.

5.129 Določi časovno zahtevnost množenja s prištevanjem glede na a) število a in glede na b) velikost števil (t.j. število bitov, ki jih potrebujemo za dvojiški zapis števil).

5.130 *Potenciranje z zaporednim množenjem* Za dani n-bitni naravni števili a in b zasnuj algoritem za izračun potence a^b z uporabo množenja. Uporabi definicijo $a^b = \underbrace{a \cdot a \cdot \cdots \cdot a}_{b \text{ množencev}}$.

5.131 Naj bo a n-bitno naravno število. Koliko bitov potrebujemo za zapis a^a?

5.132 Določi asimptotično časovno zahtevnost za algoritem iz naloge 130, če imaš na voljo algoritem za množenje dveh n-bitnih števil s časovno zahtevnostjo $O(n^2)$.

5.3 Groba sila

5.133 Razvij algoritem za izračun vrednosti polinoma $p(x)$ v točki x po naslednji formuli

$$p(x) = \sum_{i=0}^{n} a_i x^i.$$

5.134 Koliko množenj je potrebnih v algoritmu iz predhodne naloge?

5.135 Izboljšaj algoritem iz predhodne naloge, da bo potreboval $\sim 2n$ množenj.

5.136 *Hornerjev algoritem* Izboljšaj algoritem iz predhodne naloge, da bo potreboval $\sim n$ množenj.

5.4 Namigi in rešitve izbranih nalog

5.122 $\Theta(n)$. Ne da se hitreje, ker je potrebno upoštevati vseh n-bitov.

5.124 Desetiško: $9 + 9 + 9 = 27 \leq 99$, šestnajstiško $F + F + F = 2D \leq FF$. Za poljubno osnovo r pa zapišemo $(r-1)+(r-1)+(r-1) = 3(r-1) \leq (r-1)r+(r-1)$, torej $r^2-3r+2 \geq 0$ oz. $(r-2)(r-1) \geq 0$. Trditev torej velja za $r \geq 2$ oz. za vse neunarne zapise števil.

5.125 add(a,0)=a, add(a,b)=add(succ(a),pred(b))

5.126 $\Theta(b) = \Theta(2^n)$.

5.127 Uporabi zanko, ki v $a - 1$ korakih izračuna zmnožek.

5.128 Upoštevaj vse možne primere pozitivnosti in negativnosti števil a in b.

5.129 Upoštevati moramo tudi časovno zahtevnost seštevanja: a) $\Theta(a \lg a)$, b) $\Theta(n2^n)$, kjer $n = \lg a$.

5.131 $\lg a^a = a \lg a = n2^n$.

5.132 $O(aN^2)$, kjer je $N \leq n2^n$ (glej predhodno nalogo). Torej $O(an^2 4^n) = O(n^2 8^n)$. Glej tudi rešitev predhodne naloge.

5.134

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2).$$

5.135 Potenco x^n računamo sproti po formuli $x^n = x^{n-1} \cdot x$.

5.136 V formuli za $p(x)$ zaporedoma izpostavljaj x, nato algoritem zasnuj po tako dobljeni formuli.