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

1.4 Od kje pride izraz *algoritem*? S kakšnimi algoritmi se je ukvarjala dotična oseba?

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

1.6 Sošolki povej en *dvoumen* in en *nejasen* stavek (vendar ne oboje). Kaj je težje sošolka ali stavek?

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

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

1.9 Naštej in obrazloži vrste *računskih problemov*: iskalni, odločitveni, preštevalni, naštevalni in optimizacijski. Za vsako vrsto podaj primer.

1.10 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 $(3, 2), (1, 5), (4, -2)$

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

1.2 Snovanje in implementacija algoritmov

1.12 Kaj je *predpogoj* za dobro snovanje algoritmov?
1.12 Dobro razumevanje problema preko natančne (matematične) definicije.

1.13 Obrazloži nekaj kriterijev po katerih ocenjujemo kakovost algoritmov. Kateri kriterij je najpomembnejši?
1.13 Pravilnost, učinkovitost, prilagodljivost, enostavnost, implementabilnost. Pravilnost je temelj vsakega algoritma.

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

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

1.16 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?

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

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

1.19 Naštej nekaj načinov za *razhroščevanje* kode?

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

1.21 Kaj je sled algoritma?

1.22 Ali za izvajanje algoritma vedno potrebujemo računalnik? Obrazloži.

1.3 Algoritmi od vsepovsod

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

1.24 *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?

1.25 *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).

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

1.27 *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 →24? Zapiši tako rekurzivno kot iterativno obliko algoritma.

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

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

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

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

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

1.32 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		

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

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

1.35 *Faktoriela* Zapiši rekurzivni algoritem za izračun faktoriele glede na formulo $n! = n \cdot (n-1)!$ in $0! = 1$. Ali je algoritem vsebuje repno rekurzijo? Če ne, ga

spremeni, da jo bo, nato pa vse skupaj spremeni v iteracijo. Opazuj spremembe!
1.35 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).

1.4 Preverjanje pravilnosti

1.36 Na svetovnem spletu poišči nekaj primerov znanih programskih hroščev.

1.37 Kaj je poglavitno vprašanje (intuitivno), ki si ga postavimo, ko preverjamo pravilnost nekega algoritma?
1.37 Ali program deluje, kot mislimo, da bi moralo delovati?

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

1.39 Utemelji pravilnost algoritma "množenje s prištevanjem" (glej nalogo →28) preko *intuitivnega razumevanja*. Ali algoritem deluje za negativna števila?

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

1.41 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?
1.41 Št. vhodov: $2^{64} \approx 1,8 \cdot 10^{19}$, čas testiranja: $2^{64}/10^9/60/60/24/365 = 585$ let.

1.42 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?
1.42 Za $l = 150$ in $r = 190$ dobimo $l + r = 340 (\mathrm{mod} 256) = 84$ in torej $m = (l + r)/2 = 42$, kar očitno ni sredina med 150 in 180. Popravek: $m = l + (r - l)/2$.

1.43 Ugotoviti želimo pravilnost nekega algoritma za urejanje seznama. Kateri dve lastnosti moramo preveriti?
1.43 Da rezultat vsebuje enake elemente kot vhodno zaporedje in da je rezultat urejen seznam.

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

1.45 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*.

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

1.47 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]
```

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

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

2 Zahtevnost algoritmov

2.1 Splošna vprašanja

2.50 Kaj je zahtevnost algoritma?
2.50 Zahtevnost algoritma pove katere in koliko virov potrebuje algoritem za svoje izvajanje (v nekem modelu računanja).

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

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

2.53 Kaj je *Von Neumannov* model računalniške arhitekture?

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

2.55 Kaj je natančna zahtevnost in kaj asiptotična zahtevnost algoritma?

2.56 Od česa je lahko odvisna zahtevnost algoritma? Prikaži s primerom.
2.56 Od algoritma, modela računanja in od velikosti vhoda in samih podatkov v vhodu.

2.57 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.

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

2.59 Zakaj najpogosteje uporabljamo zahtevnost v najslabšem primeru?

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

2.2 Natančna zahtevnost

2.61 Koliko korakov zahteva algoritem "množenje s prištevanjem" (glej nalogo 28)?

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

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

2.63 Določi natančno zahtevnost v številu primerjav elementov glede na najboljši, najslabši in povprečni primer za algoritem (zaporednega iskanje, glej →24). Pri tem je n velikost polja a.

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

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

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

2.65 Določi natančno zahtevnost v smislu realnega časa na poljubnem RAM modelu računanja za algoritem "zaporedno iskanje" iz naloge →24

2.65 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.

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

2.3 Asimptotična zahtevnost

2.4 Namigi in rešitve izbranih nalog

3 Osnovne podatkovne strukture

3.1

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).*

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

4.67

	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

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

4.69 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) $9, 8, 7, 6, 5, 4, 3, 2, 1, 0$?

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

4.70

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

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

4.71

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

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

4.72

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

4.73 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

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

4.73

	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

4.74 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?

4.74

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

4.75 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?

4.75

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

.

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

4.76 $\Theta(n^2)$. Najmanjši elementi si zapomnimo v kazalcu min. Zamenjavo izvedemo tako, da zamenjamo elementa (ne prevezujemo vozlišč).

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

4.77

0	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

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

4.79 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.

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

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

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

4.82

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 |

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

4.83

- 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.

4.84 *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.

4.84 Uporabi navadne zamenjave ali navadno vstavljanje.

4.2 Napredna urejana

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

4.85 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 $h_{k-1} = 2h_k + 1$.

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

4.86 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.

4.86 $t = 3$, 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

4.3 Namigi in rešitve izbranih nalog

5 Drevesa in kopica

5.0.1 Dvojiška drevesa

5.0.2 Večsmerna drevesa

5.0.3 Kopica

6 Grafi in grafni algoritmi

7 Metode snovanja algoritmov

8 Osnovne metode snovanja algoritmov

8.1 Metode snovanja

8.87 Naštej nekaj metod snovanja algoritmov.

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

8.2 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.

8.89 *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.

8.90 Določi asimptotično časovno zahtevnost za algoritem iz predhodne naloge. Se da hitreje?
8.90 $\Theta(n)$. Ne da se hitreje, ker je potrebno upoštevati vseh n-bitov.

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

8.92 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?
8.92 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.

8.93 *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.
8.93 add(a,0)=a, add(a,b)=add(succ(a),pred(b))

8.94 Določi asimptotično časovno zahtevnost za algoritem iz predhodne naloge.
8.94 $\Theta(b) = \Theta(2^n)$.

8.95 *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 $a \cdot b = \underbrace{b + b \cdots + b}_{a \text{ seštevancev}}$.

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

8.96 Algoritem iz predhodne naloge razširi, da bo deloval pravilno za poljubni celi števili a in b.
8.96 Upoštevaj vse možne primere pozitivnosti in negativnosti števil a in b.

8.97 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).
8.97 Upoštevati moramo tudi časovno zahtevnost seštevanja: a) $\Theta(a \lg a)$, b) $\Theta(n2^n)$, kjer $n = \lg a$.

8.98 *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 $a^b = \underbrace{a \cdot a \cdot \cdots \cdot a}_{b \text{ množencev}}$.

8.99 Naj bo a n-bitno naravno število. Koliko bitov potrebujemo za zapis a^a?
8.99 $\lg a^a = a \lg a = n2^n$.

8.100 Določi asimptotično časovno zahtevnost za algoritem iz naloge 98, če imaš na voljo algoritem za množenje dveh n-bitnih števil s časovno zahtevnostjo $O(n^2)$.
8.100 $O(aN^2)$, 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.

8.3 Groba sila

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

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

8.102 Koliko množenj je potrebnih v algoritmu iz predhodne naloge?
8.102

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

8.103 Izboljšaj algoritem iz predhodne naloge, da bo potreboval $\sim 2n$ množenj.
8.103 Potenco x^n računamo sproti po formuli $x^n = x^{n-1} \cdot x$.

8.104 *Hornerjev algoritem* Izboljšaj algoritem iz predhodne naloge, da bo potreboval $\sim n$ množenj.
8.104 V formuli za $p(x)$ zaporedoma izpostavljaj x, nato algoritem zasnuj po tako dobljeni formuli.

8.4 Namigi in rešitve izbranih nalog

9 Deli in vladaj

9.1 Strassenov algoritem

9.105 Izvedi en korak Strassenovega algoritma. Matriki sta

$$A = \begin{bmatrix} 1 & 4 & 1 & 4 \\ 2 & 3 & 2 & 3 \\ 3 & 2 & 3 & 2 \\ 4 & 1 & 4 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

9.105 TODO: Glej zgled pri APS2.

9.106 Zapiši rekurzivno formulo za časovno zahtevnost Strassenovga algoritma za matrike velikosti $n \times n$ glede na parameter n. Reši formulo s pomočjo mojstrovega izreka.

9.106

$$T(n) = 7T(n/2) + O(n^2) = O(n^{\log_2 7})$$

9.107 Zapiši rekurzivno formulo za časovno zahtevnost Strassenovga algoritma za matrike velikosti $n \times n$ glede na parameter velikost matrike n^2. Reši formulo s pomočjo mojstrovega izreka.

9.107

$$T(n^2) = 7T(n^2/4) + O(n^2)$$

$$T(x) = 7T(x/4) + O(x) = O(x^{\log_4 7}) = O((n^2)^{\log_4 7}) = O(n^{2\log_4 7}) = O(n^{\log_2 7})$$

9.2 Namigi in rešitve izbranih nalog