Vadnica APS1

Friday, January 15, 2021 5:34 PM

1 Problemi in algoritmi

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.

- Arabski zapis števil je pomemben zato, da je algoritem jasen oz. Enostavno razumljiv, saj arabski zapis števil predstavlja dobro reprezentacijo samih števil, saj seštevanje 2 arabskih št. Poteka, kot nek algoritem in je bol preprosto, kot pa seštevanje rimskih števil.

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

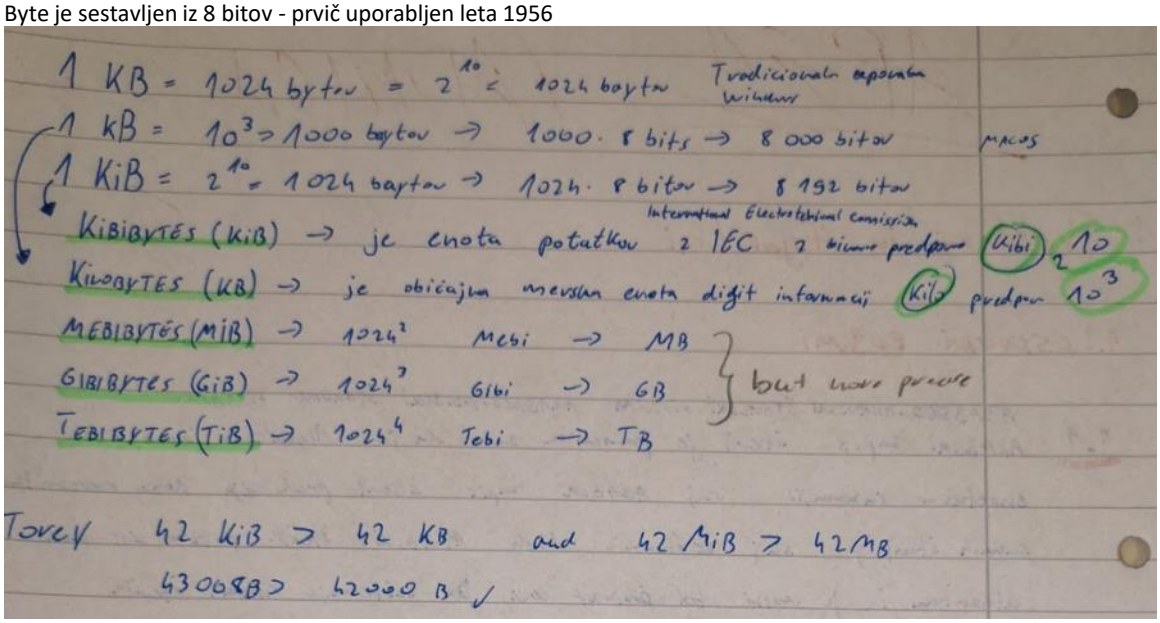
- To je matematični objekt za štetje, merjenje in za predstavitev - primer: naravna števila(1,2,3,4,5,..), napisan simbol, kot npr. "5" imenujemo številka(NUMERAC)
- Števka (DIGIT) - gre za en simbol, ki predstavlja število, uporabljen sam ali kot kombinacija števk "55"
- Cifra je reprezentacija števila na kovancu
- Mož je reprezentacija slikice na drugi strani kovanca

1.3 Kaj je vec 2^10 ali 10^2?

- Več je 2^10 saj -> 2*2*2*2*2*2*2*2*2*2 = 1.024 1024, 10^2 = 100

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

- Bit je najmanjša predstavitev podatka - manjša predstavitev byte, informacija je predstavljeno z 0||1
- Byte je sestavljen iz 8 bitov - prvič uporabljen leta 1956



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

- $\log_n$ -> $\log_{10}$ n
- $\lg_n$ -> $\log_2$ n
- $\ln_n$ -> $\log_e$ n

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

- Algoritem je prvi definiral Al-Khuarizmi -> perzijski matematik iz 9. stol., opisal algoritem za seštevanje, deljenje, množenje, kvadratni koren,... V arabskem (indijskem) zapisu.

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

- ALGORITEM -> algoritem je jasen, nedvoumen in mehaničen postopek za računajne računskega problema
- Jasen: pomeni, da je enostaven za razumet, skladen z okoljem
- Nedvoumen: razumljiv samo na en način, enoličen
- Mehaničen: uporablja elementarne operacije, slepo izvaja ukaze

1.9 Kaj je racunski problem? Podaj primer racunskega problema, ki ni povezan z racunanjem.

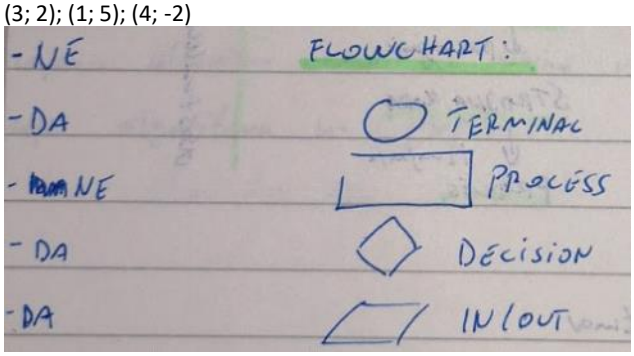
- Seštevanje števil do 100 (primer problema)

1.10 Pojasni razliko med problemom (kadar v algoritmiki recemo problem imamo v mislih racunski problem), nalogo in rešitvijo.
- Računski problem: je problem v algoritmiki (seštevanje števil do 100) -> natančno opisuje želeni odnos med nalogami in njihovimi rešitvami
- Naloga je konkreten primer problema (12+13)
- Rešitev se nanaša na neko nalogo (=25)

1.11 Naštej in obrazloži vrste racunskih problemov:
- ISKALNI: za dani št. Izračunamo vsoto, iskanje poti
  - Rešitev je objekt, ki zadošča danim omejitvam
- ODLOČITVENI: ali obstaja pot, ali velja za števila x,y,z -> x+y = z?
  - Rešitev je oblike T/F
- PREŠTEVALNI: na koliko načinov...?, koliko je...?
  - Rešitev je število objektov, ki zadošča podanim omejitvam
- NAŠTEVALNI: naštejemo vse poti do menze
  - Rešitev je množica, ki zadošča danim omejitvam, podobno kot preštevalni le da jih vse naštejemo
- OPTIMIZACIJSKI: najkrajša pot do menze
  - Iskanje najboljše rešitve izmed vseh možnih

1.12 Preveri veljavnost trditve:

a) Seštevanje, odštevanje, množenje dve števil so racunski problemi, iskanje najmanjšega elementa v seznamu števil pa ni.
b) Za dana števila x; y; z je vprašanje ali je x + y = z odlocitiveni problem.
c) Ali v danem seznamu elementov obstaja dani element je iskalni problem.
d) Urejanje seznama 5; 2; 9; 3 je racunski problem.
e) Naloga problema poišci najbližjo tocko koordinatnemu središcu je seznam tock (3; 2); (1; 5); (4; -2)



1.13 Zakaj je Turingov stroj pomemben za algoritmiko? Oglej si poljuben film o Alanu Turingu. -> NE BOM!
- Zato, ker se s Turingovim strojem da izračunati vse, kar se sploh da izračunati

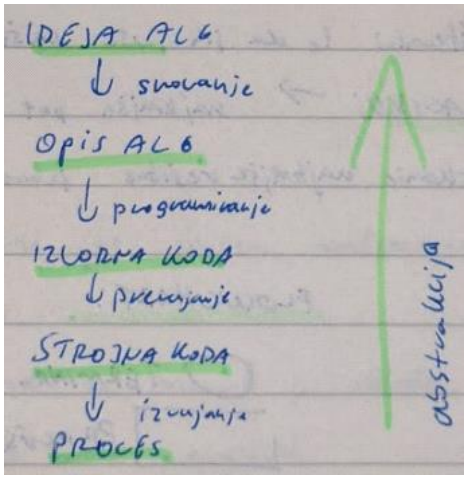
Snovanje in implementacija algoritmov

1.14 Kaj je predpogoj za dobro snovanje algoritmov?
- Predpogoj je dobro razumevanje problema

1.15 Obrazloži nekaj kriterijev po katerih ocenjujemo kakovost algoritmov. Kateri kriterij je najpomembnejši?
- Pravilnost - mora delovati (najpomembnejši)
- Učinkovitost - kako učinkovit je alg.
- Preprostost - preprosto razumljiv in preprost za branje
- Implementabilnost

1.16 Naštej in primerjaj nacine (opisni jeziki) za opis algoritmov. Kateri nacini so primernejši za ljudi in kateri za racunalnike?
- Naravni jezik
- Diagram poteka
- Psevdokoda
- Programski jezik
- Strojni jezik

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



1.19 Naštej nekaj pristopov oz. metod za snovanje algoritmov.
- Brute force
- Gredy
- Backtracking
- Transform & conquer
- Reduce & conquer
- Divide & conquer

1.20 Kaj je sintakticna in kaj semanticna napaka v programu? Kaj je programski hrošc?
- Sintaktična napaka: napaka med pisanjem kode - neupoštevanje pravil programskega jezika npr. pozabljeno ; v Javi
- Semantična napaka: logična napaka v nekem programu (program ne deluje po pričakovanjih)
- Programski hrošč: pomankljivost ali napaka v programski kodi, ki se izrazi z napačnim delovanjem oz odpovedjo programa

1.21 Naštej nekaj nacinov za razhroščevanje kode?
- Printf metode
- Trace (sledenje programu)
- Breakpoints (preko določenih točk v programu)
- Watch (opazovanje)

1.22 Kaj je profiliranje in kaj instrumentacija kode?
- Profiliranje in instrumentacija kode je ugotavljanje, koliko časa/pomnilnika, porabijo posamezni deli programa, programu dodamo ukaze za merjenje

1.23 Kaj je sled algoritma?
- Izpis vrednosti glavnih spremenljivk v izvanjanju kode

1.24 Ali za izvajanje algoritma vedno potrebujemo racunalnik?
- Ne, lahko se naredi kakšna simulacija algoritma tudi na papir

Algoritmi od vsepovsod

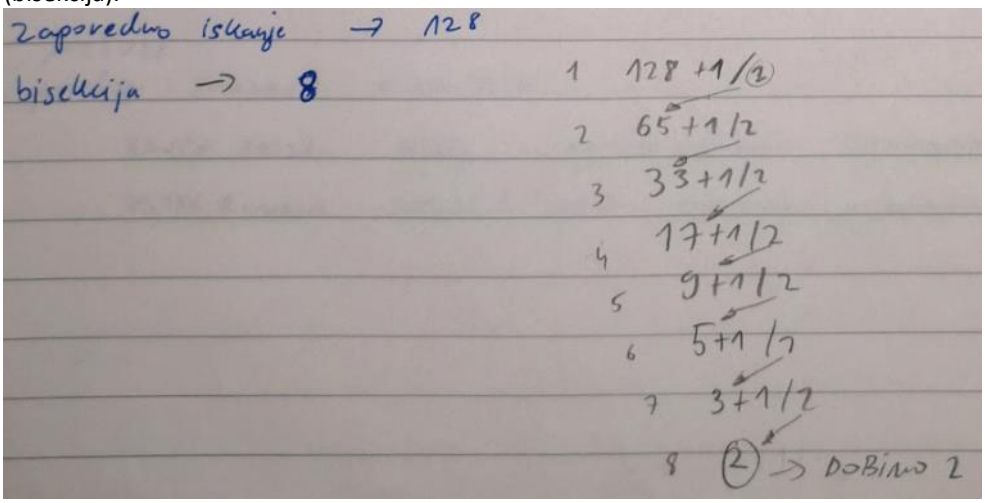
1.25 Najvecji in najmanjši element Zasnuj algoritme za iskanje najvecjega in najmanjšega elementa ter oboje hkrati. Kateri izmed algoritmov naredi manj primerjav elementov?

```
int min = a[0]
for (int i = 1; i < = a.length - 1, i++) {
    if (min > a[i]) {
        min = a[i];
    }
}
int max = a[0]
for (int i = 1; i <= a.length - 1, i++) {
    if (msx < a[i]) {
        max = a[i];
    }
}
```

1.26 Zaporedno iskanje Zasnuj algoritem za iskanje danega elementa v dani tabeli. V cem je razlika v nalogi tega problema v primerjavi s problemom ”najvecji in najmanjši element“ iz predhodne naloge?

```
int x = 42;
for (int i = 0; i<= a.length - 1; i++) {
    if (x >= a[i]) {
        sout("1 najden");
    }
}
```

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, vecje. Koliko ugibanj potrebuje v najslabšem primeru v razlicnih pristopih, npr. zaporedno iskanje, razpolavljanje (bisekcija).



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

- Bisekcija se uporablja za iskanje 0 zveznih funkcij.

1.29 Dvojiško iskanje Zasnuj algoritem dvojiško iskanje, ki uporablja nacelo razpolavljanja, za iskanje števila v urejenem zaporedju. V cem je razlika med nalogo tega problema in nalogo problema ”zaporedno iskanje“ iz naloge !26? Zapiši tako rekurzivno kot iterativno obliko algoritma.

- Rekurzivno

```
public static int findBinary(int[] a, int l, int r, int v) {
    if (r < l){
        return -1;
    }
    int mediana = l + (r - l) / 2;
    if (v < a[mediana]) {
        return findBinary(a, l, mediana - 1, v);
    }

    if (v > a[mediana]) {
        return findBinary(a,mediana + 1, r, v);
    }
    return mediana;
}
```

1.30 Množenje s prištevanjem Zasnuj algoritem za množenje dveh števil preko prištevanja.
Namig: pomagaj si z definicijo množenja

$$a \cdot b = \underbrace{b + b + \cdots + b}_{a\text{–krat}}.$$

```
int sum = 0;
for (int i = 0; i < a.length; i++) {
    sum += b;
}
```

Tuki ni a.length (zmotu sem se), a je samo število in ne tabela

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

- Starogrški matematik, ki se je ukvarjal z geometrijo in algoritmi.

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

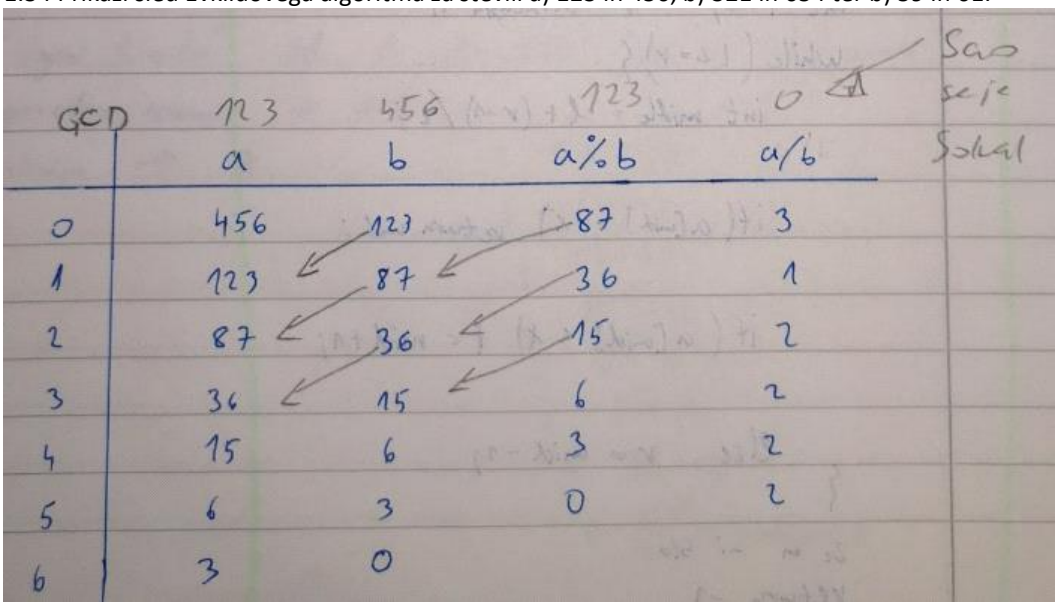
```
int gcd(int a, int b) {
    if ( b == 0) return a;
    return gcd(b, a%b);
}
```

```
int gcd(int a, int b) {
    for (int i = 0; i <= a; i++) {
        if (a%i == 0 && b%i == 0) {
            gcd = i;
        }
    }
    return gcd;
}
```

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

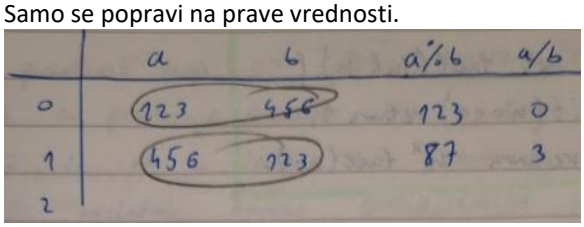
- ///

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, ce je prvo število manjše od drugega?

- Samo se popravi na prave vrednosti.



1.36 S pomocjo Eratostenovega sita izracunaj praštevila manjša od N = 42.

```
booleon prime[] = new boolean[n + 1];
for (int i = 0; i < n; i++) {
    for (int p = 2M p*p <= n; p++) {
        if (prime[p]) {
            for (int i = p*p; i <= n; i += p) {
                prime[i] = false;
            }
        }
    }
}
```

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

```
public int fact(int n) {
    if (n == 0) return 1;
    return n * fact(n - 1);
}
```

Preverjanje pravilnosti

1.38 Na svetovnem spletu poišci nekaj primerov znanih programskih hroščev.
- The mother
- The Y2K

1.39 Kaj je poglavitno vprašanje (intuitivno), ki si ga postavimo, ko preverjamo pravilnost nekega algoritma?
- Ali algoritem dela, kar bi moral delati?

1.40 Naštej (štiri) nacine s katerimi lahko preverjamo pravilnost algoritmov.
- POPOLNOST: da bi alg. bil popolen, morajo vse njegove akcije biti natančno deginirane
- NEDVOUMNOST: množica inštrukcij je nedvoumna, če obstaja le en način predstavitve
- DETERMINIZEM: če slediš ukazom se mora alg. Izvesti
- KONČNOST: alg. se mora po določenem času končati

1.46 Ugotoviti želimo pravilnost nekega algoritma za urejanje seznama. Kateri dve lastnosti moramo preveriti?
- Ali rezultat vsebujje enake elemente, kot vhodno zaporedje
- Ali je rezultat urejen seznam

1.47 V cem je prednost formalnega dokazovanja pravilnosti algoritmov. Na katerem matematicnem nacelu sloni dokazovanje pravilnosti algoritmov, ki vsebujejo zanke?
- Dokaz je osnovan predvsem na matematiki/statistiki. Dokazi lahko pri kompleksnih alg. postanejo ekstremno dolgi
- Dokazi z:
  - INDUKCIJO: metoda za dokazovanje neke trditve
  - OSNOVNI PRIMER: dokažemo pravilnost za majhen primer
  - INDUKTIVNA PREDPOSTAVKA: neka trditev velja za..
  - INDUKTIVNI KORAK: če neka trditev velja za n, velja tudi za n + 1

1.48 Formalni dokaz pravilnosti algoritma pogosto temelji na indukciji. Kaj je matematicna indukcija, hipoteza, osnovni primer, induktivna predpostavka in induktivni korak? V algoritmiki pa za dokazovanje zank uporabljamo tudi zancne invariante.
- ZANČNA INVARIANTA:
  - Postopek za preverjanje pravilnosti zank, podoben matematični indukciji le, da se izvaja do terminiranja zanke, ima pa naslednje korake:
    - INICIALIZACIJA, VZDRŽEVANJE, TERMINACIJA

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

2 Zahtevnost algoritmov

Osnovni pojmi

2.53 Kaj je zahtevnost algoritma?
- Zahtevnost pomeni, katere in koliko virov potrebuje alg. za svoje izvajanje v nekem modelu računanja

2.54 Naštej nekaj virov, ki jih algoritem lahko potrebuje za svoje izvajanje.
- Čas, prostor, komunikacija, št. Operacij, električna energija

2.55 Kateri viri ustrezajo meri casa in kateri prostora?
DOPOLNI

2.56 Kaj je Von Neumannov model racunalniške arhitekture?
DOPOLNI

2.57 Kaj je RAM model racunanja? Zakaj ga uporabljamo v algoritmiki?
- Random Access Machine: gre za zaporedno izvajanje operacij
- RAM predstavlja model računanja
- Gre za predstavitev števil in kazalcev

2.58 Kaj je natancna zahtevnost in kaj asiptoticna zahtevnost algoritma?
- Natančna zahtevnost algoritma: $T(n) = x^{12} + x + 3$
- Asimptotična zahtevnost algoritma: $O(n^2)$

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

- Od algoritma, modela računanja, ter od velikosti in vrste podatkov
- VELIKOSTI NALOGE: 2*3 ali pa 1234*56789
- PODATKOV V NALOGI: 1234*1000 ali pa 1234* 5678

2.60 Izberi nek problem, nato naštej nekaj primerov nalog zanj, katerih težavnost je razlicna:
a) glede na velikost naloge
b) glede na podatke v sami nalogi.

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

- Best case (BC)
- Worst case (WC)
- Average case (AC)

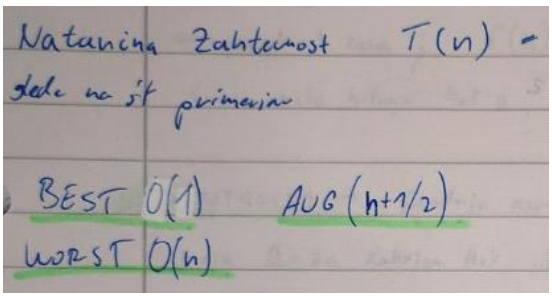
2.62 Zakaj najpogosteje uporabljamo zahtevnost v najslabšem primeru?

- Zato, ker nas pri algoritmiki zanimajo rezultati v najslabšem primeru

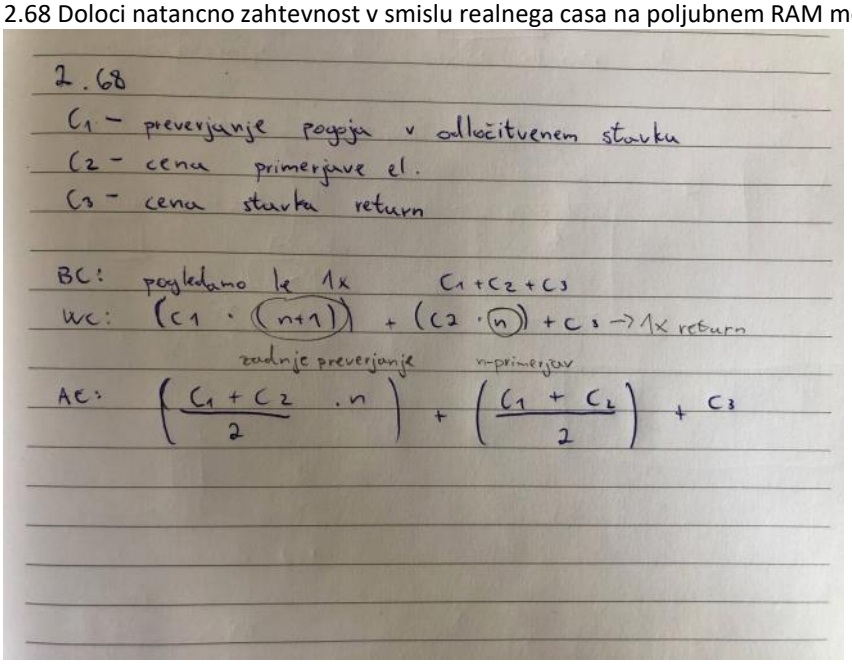
Natančna zahtevnost

2.66 Doloci natancno zahtevnost v številu primerjav elementov glede na najboljši, najslabši in povprecni 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
```



2.68 Doloci natancno zahtevnost v smislu realnega casa na poljubnem RAM modelu racunanja za algoritem ”zaporedno iskanje“



2.69 Kolikšna je globina rekurzije pri algoritmu ”dvojiškega iskanja“?
BC: 1
WC: [lg n] + 1
AC: ??

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

telo zanke?
For i=A to B DO....

- Primerjava indeksov: (B - A) + 2
- Telo zanke: (B - A) + 1

Asimptotična zahtevnost

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

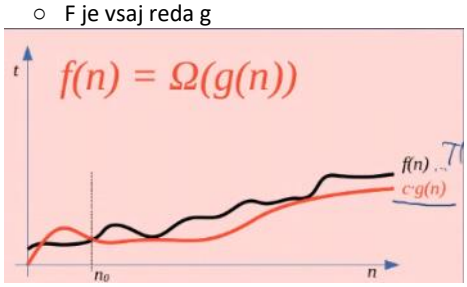
O-notacija -> zgornja asimptotična meja

- F je od zgoraj omejena z g
- F ne raste hitreje, kot g
- F je kvečjemu reda g

f(n) <= c * g(n)

Ω-notacija -> spodnja asimptotična meja

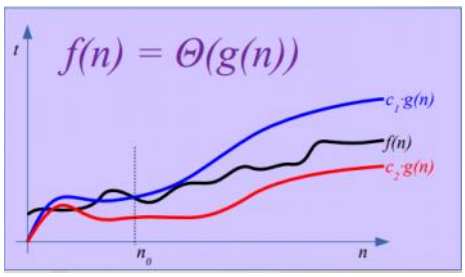
- F je od spodaj omejena z g
- F ne raste počasneje, kot g
- F je vsaj reda g



f(n) >= c * g(n)

Θ-notacija -> tesna asimptotična meja

- F je od zgoraj in od spodaj omejena z g
- F je reda g



$$\leq \quad O(g(n)) = \{f(n) \mid \exists c, n_0 > 0 \, \forall n \geq n_0 : 0 \leq f(n) \leq cg(n)\}$$

$$= \quad \Theta(g(n)) = \{f(n) \mid \exists c_1, c_2, n_0 > 0 \, \forall n \geq n_0 : 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

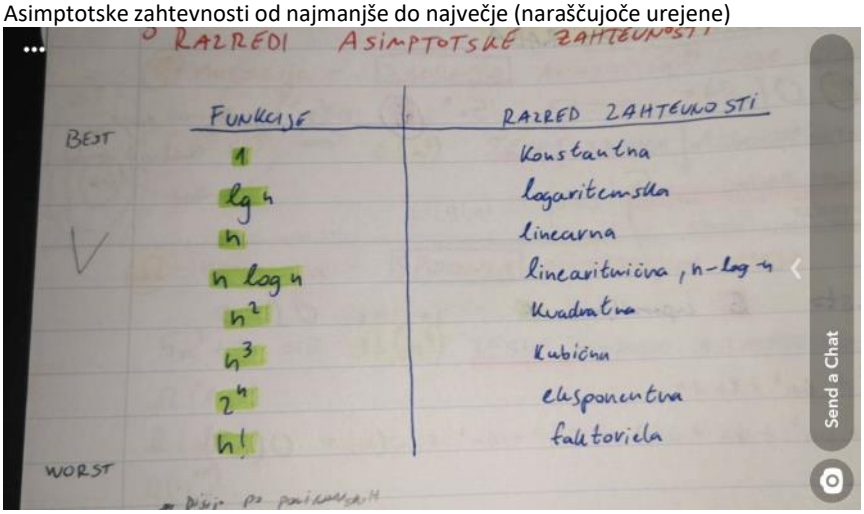
$$\geq \quad \Omega(g(n)) = \{f(n) \mid \exists c, n_0 > 0 \, \forall n \geq n_0 : 0 \leq cg(n) \leq f(n)\}$$

2.75 Kako v praksi zlorabljamo asimptoticno notacijo in relacijo 2?

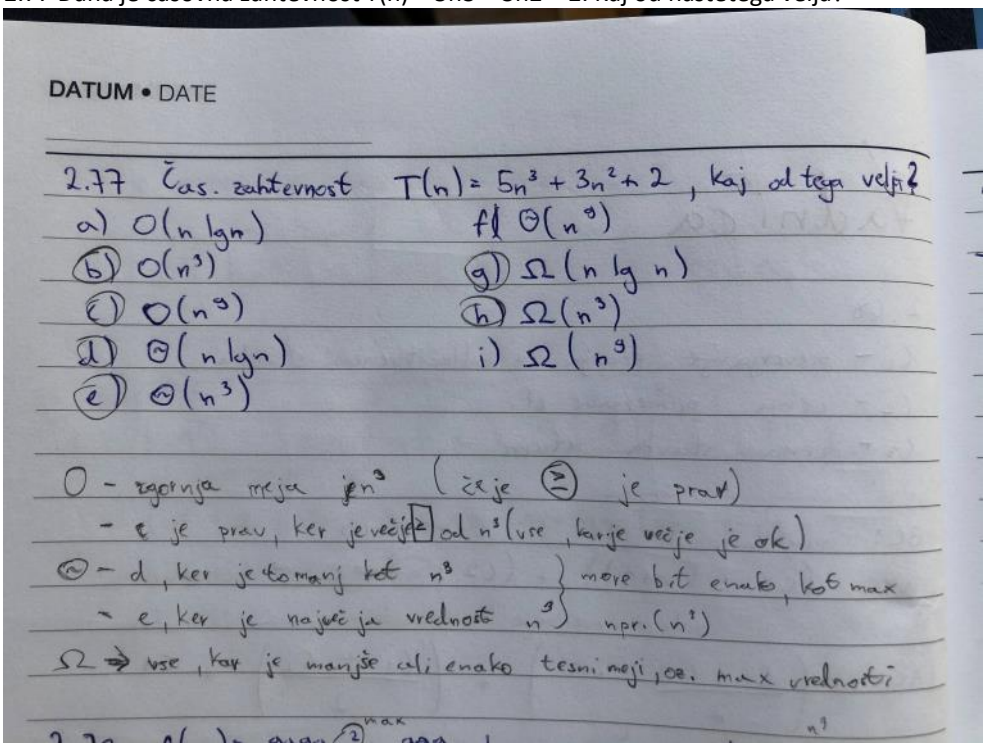
- Zloraba zato, ker je v praksi ne uporabljamo z znakom € ampak uporabimo kar =
- namesto ∈ uporabljamo =
  - za vse O, Ω in Θ
    f (n)=O(g(n)) ≡ f (n)∈O(g (n))

- Leva za vse / desna za enega
  2 n 2+3 n+1=2 n 2+Θ(n)=Θ(n)

Asimptotske zahtevnosti od najmanjše do največje (naraščujoče urejene)



2.77 Dana je casovna zahtevnost T(n) = 5n3 + 3n2 + 2. Kaj od naštetega velja?



2.78 Za dani funkciji obkroži veljavne trditve:

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

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) $g(n)/f(n) = O(n \lg n)$

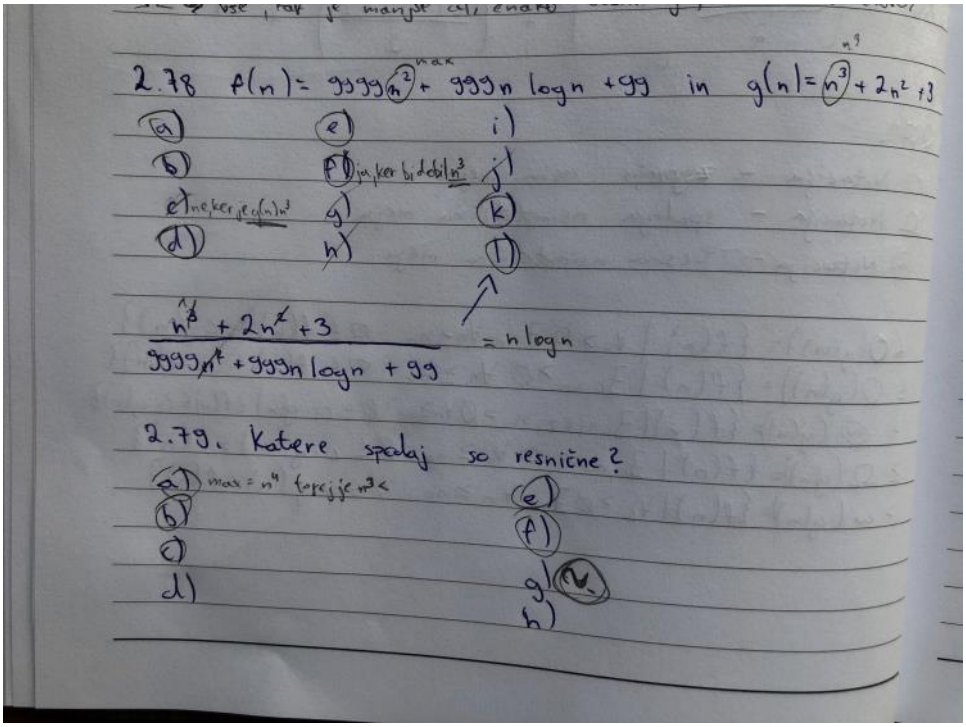
2.79 Katere izmed spodnjih trditev so resnicne?

a) $\sqrt{4}\, n^4 \log^4 n + 4n^4 = \Omega(n^3 \log^3 n)$ b) $3n^2 + 2n + 1 = \Omega(n \log^{12} n)$

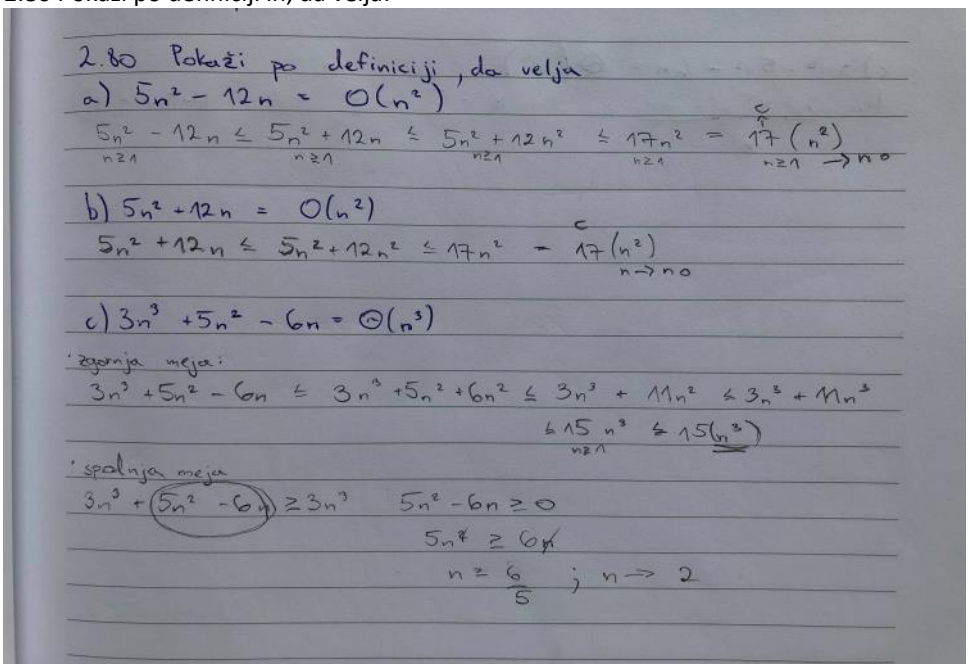
c) $2^{3456} = O(\log n)$ d) $n^{2/3} = O(n^{0,666})$

e) $\Omega(n^{\lg 4}) = (\sqrt{16})^{\lg n}$ f) $27^{\log_3 n} = \Theta(n^3)$

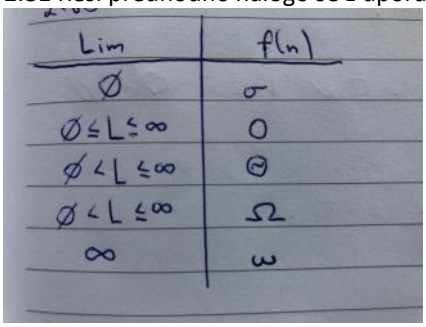
g) $\sum_{i=1}^{n} O(n) = \Theta(n^2)$ h) $(n + \lg n)^{42} - n^{42} = \Omega(n^{42})$

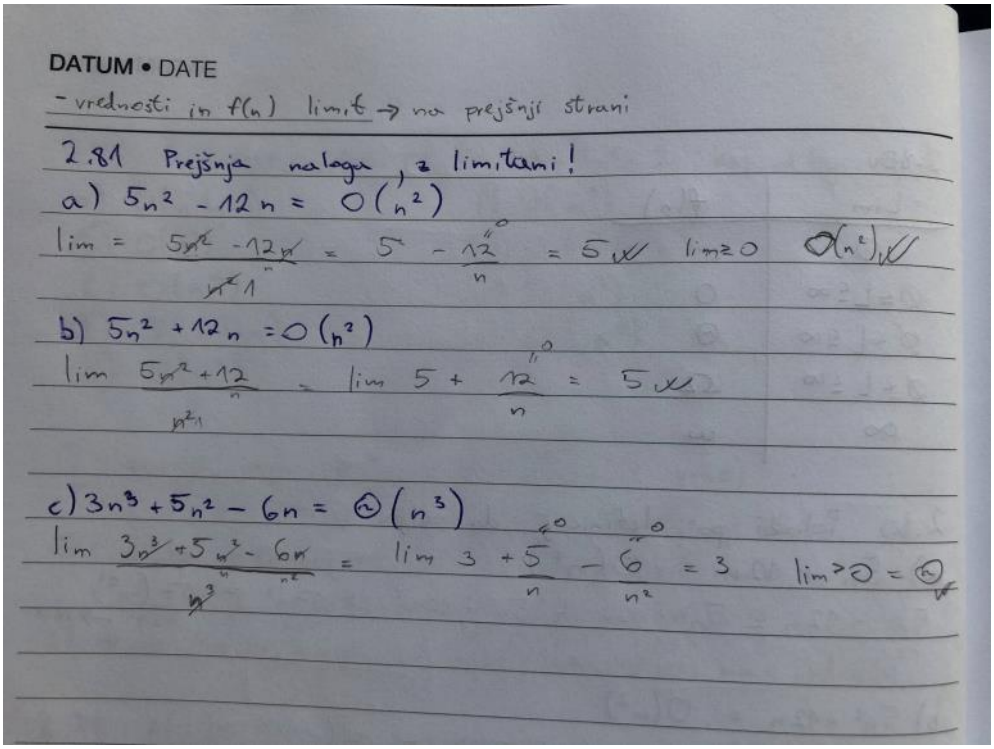


2.80 Pokaži po definiciji in, da velja:



2.81 Reši predhodno nalogo še z uporabo limit.





3 Osnovne podatkovne strukture

Sklad

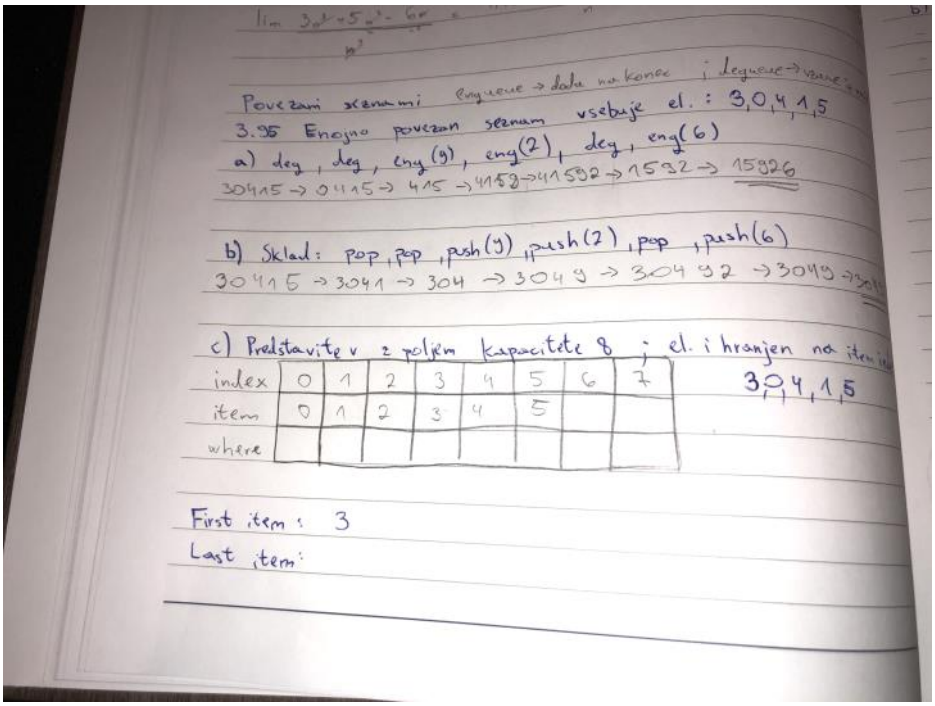
3.93 Profesor Mate Matik želi preveriti ali je dano zaporedje oklepajev pravilno gnezdeno. Na primer npr. ()([]) in ()[{(()[])}] sta pravilno gnezdeni zaporedji, (()[]} in (([()]) pa nista. Pomagaj mu in zasnuj algoritem za ta problem.

```
public static boolean isBalancedWithBrackets(String expr) {
    if (expr.length() % 2 == 1) { return false; }
    else {
        Stack<Character> s = new Stack<>();
        for (char bracket : expr.toCharArray()) {
            switch(bracket) {
                case "{": s.push("}");
                    break;
                case "(": s.push(")");
                    break;
                case "[": s.push("]");
                    break;
                default:
                    if (s.isEmpty() ||bracket != s.peek()) {
                        return false;
                    }
                    s.pop();
            }
        }
        return s.isEmpty();
    }
}
```

Povezani seznami

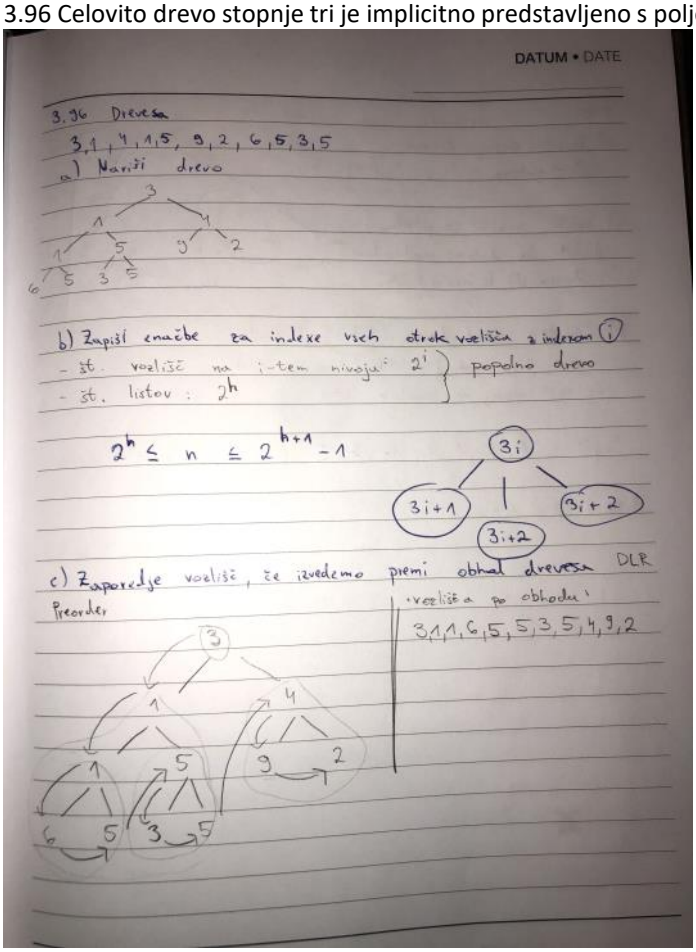
3.95 Enojno povezani seznam zaporedoma vsebuje elemente 3; 0; 4; 1; 5.
a) Seznam uporabimo kot vrsto in nad njim izvedemo naslednje operacije: dequeue, dequeue, enqueue(9), enqueue(2), dequeue, enqueue(6). Kakšen seznam dobimo?
b) Seznam uporabimo kot sklad in nad njim izvedemo naslednje operacije: pop, pop, push(9), push(2), pop, push(6). Kakšen seznam dobimo?
c) Uporabite predstavitev seznama s poljem kapacitete 8, pri cemer 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).

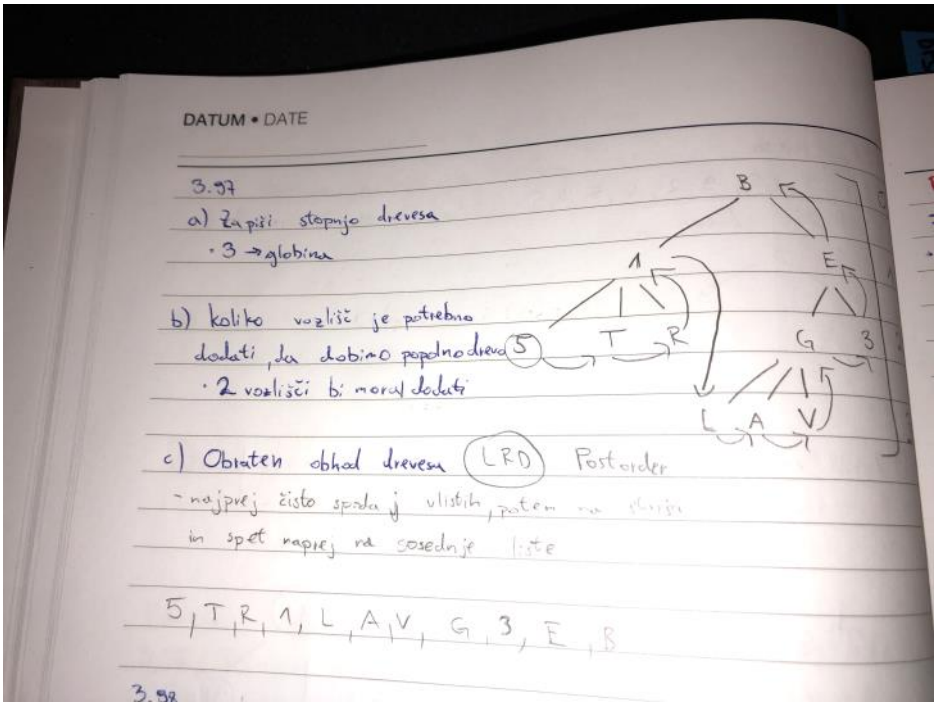


Drevesa

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



3.97 Dano je naslednje drevo.


DATUM • DATE
3.97
a) Zapiši stopnjo drevesa
· 3 → globina
b) koliko vozlišč je potrebno
dodati, da dobimo popolno drevo
· 2 vozlišči bi morali dodati
c) Obraten obhod drevesa (LRD) Postorder
- najprej čisto spodaj vlistih, potem na sosednje liste
in spet naprej na sosednje liste
5, T, R, 1, L, A, V, G, 3, E, B
3.98
B
1
E
5
T
R
G
3
L
A
V