

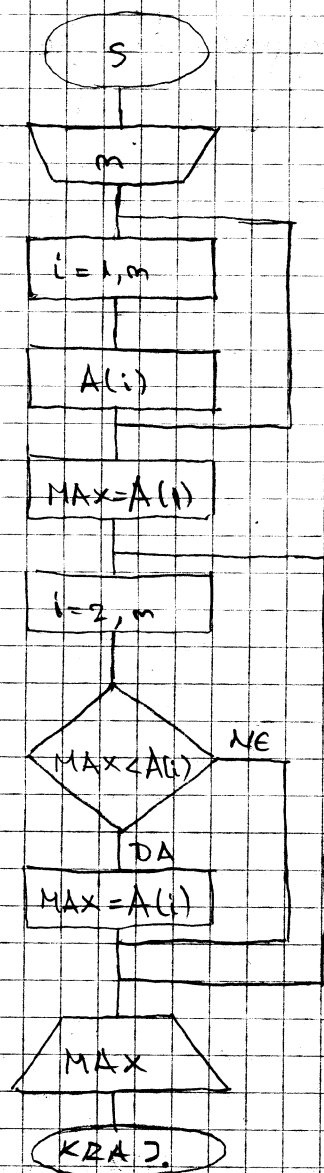
Forma zapisivanja algoritama:

1. PROBLEM: Pronaći najveći broj u nizu.

ULAZ: Niz brojeva.

IZLAZ: Broj

OPIS:



PSEUDOKOD (izmišljeni jezik):

NPR.

RADI DOK $i < 10$

$\{ i := i + 1; i++ \quad i \leftarrow i + 1$

ili

$i := i + 1;$

}

START

INPUT $m(\text{int})$

FOR $i = 1, m$

INPUT $A(i)$

NEXT i

$MAX = A(1)$

FOR $i = 2, m$

IF $MAX < A(i)$ THEN

$MAX = A(i)$

NEXT i

PRINT MAX

END

ili: (prof. način):

niz $A \leftarrow$ niz od m brojeva

najveci \leftarrow niz $A[i]$

for $i = 2$ to m

if najveci $<$ niz $A[i]$ then
najveci \leftarrow niz $A[i]$

end for

2. PROBLEM: Određivanje NZD

OPIS: Za dva zadana cijela broja odrediti NZD

ULAZ: Dva cijela broja

IZLAZ: Broj

ALGORITAM:

TEOREMA: Abo su a, b i q, r cijeli brojevi. Abo je $a = bq + r$ skup zajedničkih djelilaca brojeva a i b jednak je skupu zajedničkih djelilaca b i r .
 $\Rightarrow \{m \mid m \mid a \wedge m \mid b\} = \{m \mid m \mid b \wedge m \mid r\}$

21, 35

$$21 : 35 = 0$$

21

$$35 : 21 = 1$$

14

$$21 : 14 = 1$$

7

$$14 : 7 = 2$$

0

očitavamo zadnji br. s kojim smo
djelili

START

INPUT a, b

$r := 1$

WHILE $r \neq 0$

$q := a : b$

$r := a - \text{int}(q) \cdot b$

$a := b$

NZD := b

$b := r$

WEND

PRINT NZD

END

u C++:

int a, b, r ;

cin >> a ;

cin >> b ;

while (true) {

$r = a \% b$ ostatak pri dijeljenju

if ($r == 0$) break;

else { $a = b$; $b = r$;} }

cout << "NZD(' << a << b ;', '<< b <') = '
' << b ;

[3] Dat je prirodan br. m koji predstavlja visinu stuba u cm. Puž se penje preko dave 7 cm, a preko moči spušta 1 cm. Napisati algoritam koji za zadani br. m računa br. dave koji je potreban pužu da se popne na vrh stuba.

INPUT m (idemo od pretpostavke da je se puž popeo
 $i=0, j=0$ pa se spušta)

$x(0) = m$

[20]

$i = i + 1$

$x(i) = x(i-1) - 7$

$j = j + 1$

IF $x(i) \leq 0$ THEN GOTO [10]

$x(i+1) = x(i) + 3$

$i = i + 1$

GOTO [20]

[10]

PRINT j

Primer: $m=30$

$i=0, j=0$

$x_0=30$

① $i=1$

$x_1=23$

$j=1$

$x_2=26$

$i=2$

② $i=3$

ALGORITAMSKA REKURZIVNE (SAMOPOZIVAJUĆE)

PROCEDURE / FUNKCIJE

1. Stub dužine N cm. Preko dانا puž se popne 7 cm, a preko moći spusti h cm. Odrediti broj dانا (algoritamski) za koji će se puž popeti na vrh stuba (koristiti rekursivnu fun.).

NEREKURZIVNO RJEŠENJE:

(1) INPUT N

(2) $DAN := 1$

(3) WHILE $N - 7 > 0$

(4) $N := N - 7 + h$

(5) $DAN := DAN + 1$

(6) END WHILE

(7) PRINT DAN

(1) $N = 15$

(2) $DAN = 1$

(3) $15 - 7 > 0$ ✓

(4) $N = 15 - 7 + h = 12$

(5) $DAN = 1 + 1 = 2$

(6) → (3) $12 - 7 > 0$ ✓

(4) $N = 12 - 7 + h = 9$

(5) $DAN = 2 + 1 = 3$

(3) $9 - 7 > 0$ ✓

(4) $N = 9 - 7 + h = 6$

(5) $DAN = 4$

(3) $6 - 7 > 0$ ✗

(7) ISPIS (4) ✓

REKURZIVNO RJEŠENJE: (fija i pomoćni pozivi te fije u njenom tijelu)

(1) INPUT N

(2) TEKUCI-DAN := 1

(3) PRINT Dam(N, TEKUCI-DAN)

// Rekurzivna fija

(4) Dam(N, TEKUCI-DAN) {

(5) IF $N - 7 \leq 0$

(6) VRATI TEKUCI-DAN

(7) ELSE

(8) VRATI Dam(N-7+1, TEKUCI-DAN)

(9) }

umjesto ovoga:

D := Dam(N, ...)

PRINT D

Primjer:

(1) N = 15

(2) TEKUCI-DAN = 1

(3) → (4) Dam(15, 1)

(5) $15 - 7 \leq 0$ ↓

(7) → (8) Dam(12, 2)

(4) → (5) $12 - 7 \leq 0$ ↓

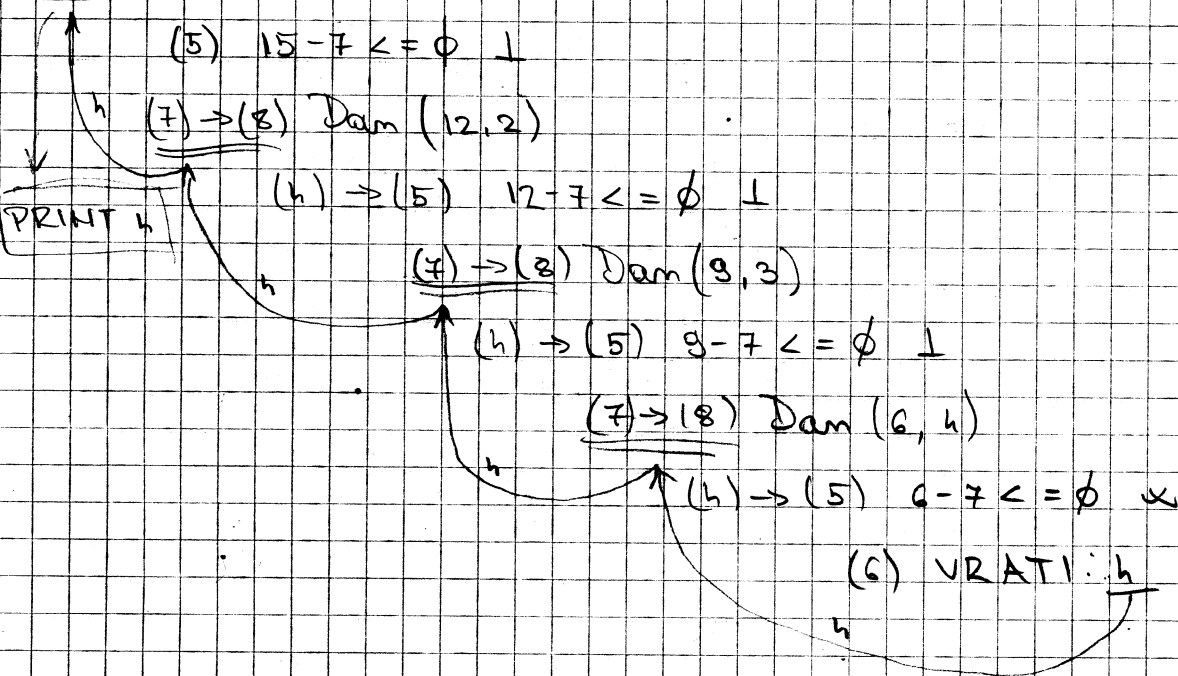
(7) → (8) Dam(9, 3)

(4) → (5) $9 - 7 \leq 0$ ↓

(7) → (8) Dam(6, 4)

(4) → (5) $6 - 7 \leq 0$ ×

(6) VRATI: 4

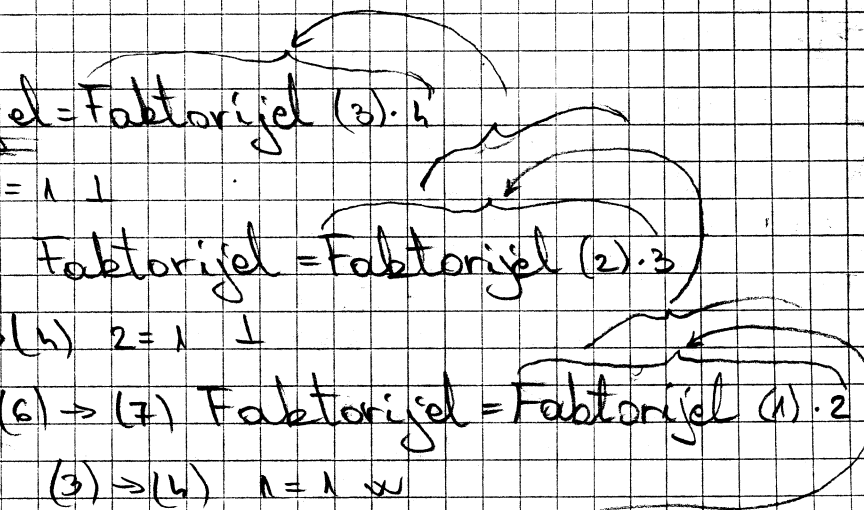


2. Za zadanu prirodan broj N izračunati $N!$ koristeći rekursivnu fkn.

(1) INPUT N $N! = N(N-1)!$
(2) PRINT Faktoriyel (N) $\text{Faktoriyel}(N-1) / 1! = 1$
(3) FUNCTION Faktoriyel (N)
(4) IF $N=1$ THEN
(5) Faktoriyel = 1
(6) ELSE
(7) Faktoriyel = Faktoriyel ($N-1$) * N
(8) END IF
(9) END FUNCTION

Primjer:

(1) $N=4$
(2) PRINT Faktoriyel (N)
(3) \rightarrow (4) $h=1 \downarrow$
(6) \rightarrow (7) Faktoriyel = Faktoriyel (3) * 4
(3) \rightarrow (4) $3=1 \downarrow$
(6) \rightarrow (7) Faktoriyel = Faktoriyel (2) * 3
(3) \rightarrow (4) $2=1 \downarrow$
(6) \rightarrow (7) Faktoriyel = Faktoriyel (1) * 2
(3) \rightarrow (4) $1=1 \downarrow$
(5) Faktoriyel = 1



3. Za zadane prirodne br. N i k koristeci rekursivnu fku izracunati $\binom{N}{k}$.

$$\binom{N}{k} = \binom{N-1}{k} + \binom{N-1}{k-1}$$

$$\binom{N}{k} = \frac{N!}{k!(N-k)!}$$

(1) INPUT N

(2) INPUT k

(3) PRINT $b_k(N, k)$

(4) SUBROUTINE $b_k(N, k)$

(5) IF $k=0$ OR $N=k$ THEN

(6) $b_k = 1$

(7) ELSE

(8) $b_k = b_k(N-1, k) + b_k(N-1, k-1)$

(9) END IF

(10) END SUBROUTINE

Primjer:

$b_k(3, 2)$

$b_k = b_k(3, 2) + b_k(3, 1)$

$b_k(3, 2) = b_k(2, 2) + b_k(2, 1)$

$b_k(2, 2) = 1$

$b_k(2, 1) = b_k(1, 1) + b_k(1, 0) = 2 \quad (N=k, k=0)$

$b_k(3, 2) = 1 + 2 = \underline{3}$

$b_k(3, 1) = b_k(2, 1) + b_k(2, 0) = 2 + 1 = \underline{3}$

$b_k = 3 + 3 = 6$

4) Napisati algoritam za niz Fibonačijevih br.

(1, 1, 2, 3, 5, 8, 13, ...)

$$a_m = a_{m-1} + a_{m-2}, \quad a_1 = a_2 = 1$$

5) Napisati algoritam za izvlačenje (na slučaj) svih brojeva iz skupa $\{1, 2, \dots, 90\}$, bez ponavljanja.

koristiti niz: `int a[90]`

pravimo niz $\left\{ \begin{array}{l} \text{for (int i=1; i < 91; i++)} \\ a[i] = i; \end{array} \right.$

Dobijemo $a[1] = 1, a[2] = 2, \dots, a[90] = 90$

Pravimo novi niz od 1 do 89 članova.

SORTIRANJE (selekcijom)

1. Za dati niz brojeva a_1, a_2, \dots, a_m odrediti niz a'_1, a'_2, \dots, a'_m istih tih brojeva tako da $a'_1 \leq a'_2 \leq \dots \leq a'_m$.

INPUT m

FOR i=1 TO m

INPUT A(i)

B(i) = A(i)

NEXT i

FOR i=1 TO m-1

FOR j=i+1 TO m

IF B(i) > B(j) THEN

m = B(i)

B(i) = B(j)

B(j) = m

END IF

NEXT j

NEXT i

FOR i=1 TO m

PRINT B(i)

NEXT i

Primjer: 17, 4, 19, 1

(Sortiranje selekcijom)

1° i=1

2° i=1

j=2

j=3

B₁ > B₂ ✓

4, 17, 19, 1

m=4

B₁ = 17

B₂ = 4

3° i=1

4° i=2

j=4

j=3

1, 17, 19, 4

1, 17, 19, 4

5° i=2

6° i=3

j=4

j=4

1, 4, 19, 17

1, 4, 17, 19

2. Ilustrirati primjer selekcijom tako da ćemo prikazati niz u opadajućem redoslijedu 7, 17, 13, 14, 5
(ilustrirati sortiranje selekcijom tako da prikazete stanje niza nakon svake iteracije u vanjskoj petlji.)

vanjska FOR i ...
unijetna petlja FOR j ...
NEXT j
NEXT i

FOR i=1 TO m-1
FOR j=i+1 TO m

kao u zadatku

{17, 7, 13, 14, 5}

sad tražimo najveći od ovih

{17, 14, 13, 7, 5}

{17, 14, 13, 7, 5}

{17, 14, 13, 7, 5}

ne idemo dalje jer bada je ovo masas ostaje 5 kao zadnji.

7, 17, 13, 14, 5

17, 7, 13, 14, 5

17, 14, 13, 7, 5

17, 14, 13, 7, 5

17, 14, 13, 7, 5

i=1 j=2
17, 7, 13, 14, 5
i=2 j=3
17, 14, 13, 7, 5
i=3 j=4
17, 14, 13, 7, 5
i=4 j=5
17, 14, 13, 7, 5
i=5 j=6
17, 14, 13, 7, 5
i=6 j=7
17, 14, 13, 7, 5
i=7 j=8
17, 14, 13, 7, 5
i=8 j=9
17, 14, 13, 7, 5
i=9 j=10
17, 14, 13, 7, 5
i=10 j=11
17, 14, 13, 7, 5
i=11 j=12
17, 14, 13, 7, 5
i=12 j=13
17, 14, 13, 7, 5
i=13 j=14
17, 14, 13, 7, 5
i=14 j=15
17, 14, 13, 7, 5
i=15 j=16
17, 14, 13, 7, 5
i=16 j=17
17, 14, 13, 7, 5
i=17 j=18
17, 14, 13, 7, 5
i=18 j=19
17, 14, 13, 7, 5
i=19 j=20
17, 14, 13, 7, 5
i=20 j=21
17, 14, 13, 7, 5
i=21 j=22
17, 14, 13, 7, 5
i=22 j=23
17, 14, 13, 7, 5
i=23 j=24
17, 14, 13, 7, 5
i=24 j=25
17, 14, 13, 7, 5
i=25 j=26
17, 14, 13, 7, 5
i=26 j=27
17, 14, 13, 7, 5
i=27 j=28
17, 14, 13, 7, 5
i=28 j=29
17, 14, 13, 7, 5
i=29 j=30
17, 14, 13, 7, 5
i=30 j=31
17, 14, 13, 7, 5
i=31 j=32
17, 14, 13, 7, 5
i=32 j=33
17, 14, 13, 7, 5
i=33 j=34
17, 14, 13, 7, 5
i=34 j=35
17, 14, 13, 7, 5
i=35 j=36
17, 14, 13, 7, 5
i=36 j=37
17, 14, 13, 7, 5
i=37 j=38
17, 14, 13, 7, 5
i=38 j=39
17, 14, 13, 7, 5
i=39 j=40
17, 14, 13, 7, 5
i=40 j=41
17, 14, 13, 7, 5
i=41 j=42
17, 14, 13, 7, 5
i=42 j=43
17, 14, 13, 7, 5
i=43 j=44
17, 14, 13, 7, 5
i=44 j=45
17, 14, 13, 7, 5
i=45 j=46
17, 14, 13, 7, 5
i=46 j=47
17, 14, 13, 7, 5
i=47 j=48
17, 14, 13, 7, 5
i=48 j=49
17, 14, 13, 7, 5
i=49 j=50
17, 14, 13, 7, 5
i=50 j=51
17, 14, 13, 7, 5
i=51 j=52
17, 14, 13, 7, 5
i=52 j=53
17, 14, 13, 7, 5
i=53 j=54
17, 14, 13, 7, 5
i=54 j=55
17, 14, 13, 7, 5
i=55 j=56
17, 14, 13, 7, 5
i=56 j=57
17, 14, 13, 7, 5
i=57 j=58
17, 14, 13, 7, 5
i=58 j=59
17, 14, 13, 7, 5
i=59 j=60
17, 14, 13, 7, 5
i=60 j=61
17, 14, 13, 7, 5
i=61 j=62
17, 14, 13, 7, 5
i=62 j=63
17, 14, 13, 7, 5
i=63 j=64
17, 14, 13, 7, 5
i=64 j=65
17, 14, 13, 7, 5
i=65 j=66
17, 14, 13, 7, 5
i=66 j=67
17, 14, 13, 7, 5
i=67 j=68
17, 14, 13, 7, 5
i=68 j=69
17, 14, 13, 7, 5
i=69 j=70
17, 14, 13, 7, 5
i=70 j=71
17, 14, 13, 7, 5
i=71 j=72
17, 14, 13, 7, 5
i=72 j=73
17, 14, 13, 7, 5
i=73 j=74
17, 14, 13, 7, 5
i=74 j=75
17, 14, 13, 7, 5
i=75 j=76
17, 14, 13, 7, 5
i=76 j=77
17, 14, 13, 7, 5
i=77 j=78
17, 14, 13, 7, 5
i=78 j=79
17, 14, 13, 7, 5
i=79 j=80
17, 14, 13, 7, 5
i=80 j=81
17, 14, 13, 7, 5
i=81 j=82
17, 14, 13, 7, 5
i=82 j=83
17, 14, 13, 7, 5
i=83 j=84
17, 14, 13, 7, 5
i=84 j=85
17, 14, 13, 7, 5
i=85 j=86
17, 14, 13, 7, 5
i=86 j=87
17, 14, 13, 7, 5
i=87 j=88
17, 14, 13, 7, 5
i=88 j=89
17, 14, 13, 7, 5
i=89 j=90
17, 14, 13, 7, 5
i=90 j=91
17, 14, 13, 7, 5
i=91 j=92
17, 14, 13, 7, 5
i=92 j=93
17, 14, 13, 7, 5
i=93 j=94
17, 14, 13, 7, 5
i=94 j=95
17, 14, 13, 7, 5
i=95 j=96
17, 14, 13, 7, 5
i=96 j=97
17, 14, 13, 7, 5
i=97 j=98
17, 14, 13, 7, 5
i=98 j=99
17, 14, 13, 7, 5
i=99 j=100
17, 14, 13, 7, 5
i=100 j=101
17, 14, 13, 7, 5
i=101 j=102
17, 14, 13, 7, 5
i=102 j=103
17, 14, 13, 7, 5
i=103 j=104
17, 14, 13, 7, 5
i=104 j=105
17, 14, 13, 7, 5
i=105 j=106
17, 14, 13, 7, 5
i=106 j=107
17, 14, 13, 7, 5
i=107 j=108
17, 14, 13, 7, 5
i=108 j=109
17, 14, 13, 7, 5
i=109 j=110
17, 14, 13, 7, 5
i=110 j=111
17, 14, 13, 7, 5
i=111 j=112
17, 14, 13, 7, 5
i=112 j=113
17, 14, 13, 7, 5
i=113 j=114
17, 14, 13, 7, 5
i=114 j=115
17, 14, 13, 7, 5
i=115 j=116
17, 14, 13, 7, 5
i=116 j=117
17, 14, 13, 7, 5
i=117 j=118
17, 14, 13, 7, 5
i=118 j=119
17, 14, 13, 7, 5
i=119 j=120
17, 14, 13, 7, 5
i=120 j=121
17, 14, 13, 7, 5
i=121 j=122
17, 14, 13, 7, 5
i=122 j=123
17, 14, 13, 7, 5
i=123 j=124
17, 14, 13, 7, 5
i=124 j=125
17, 14, 13, 7, 5
i=125 j=126
17, 14, 13, 7, 5
i=126 j=127
17, 14, 13, 7, 5
i=127 j=128
17, 14, 13, 7, 5
i=128 j=129
17, 14, 13, 7, 5
i=129 j=130
17, 14, 13, 7, 5
i=130 j=131
17, 14, 13, 7, 5
i=131 j=132
17, 14, 13, 7, 5
i=132 j=133
17, 14, 13, 7, 5
i=133 j=134
17, 14, 13, 7, 5
i=134 j=135
17, 14, 13, 7, 5
i=135 j=136
17, 14, 13, 7, 5
i=136 j=137
17, 14, 13, 7, 5
i=137 j=138
17, 14, 13, 7, 5
i=138 j=139
17, 14, 13, 7, 5
i=139 j=140
17, 14, 13, 7, 5
i=140 j=141
17, 14, 13, 7, 5
i=141 j=142
17, 14, 13, 7, 5
i=142 j=143
17, 14, 13, 7, 5
i=143 j=144
17, 14, 13, 7, 5
i=144 j=145
17, 14, 13, 7, 5
i=145 j=146
17, 14, 13, 7, 5
i=146 j=147
17, 14, 13, 7, 5
i=147 j=148
17, 14, 13, 7, 5
i=148 j=149
17, 14, 13, 7, 5
i=149 j=150
17, 14, 13, 7, 5
i=150 j=151
17, 14, 13, 7, 5
i=151 j=152
17, 14, 13, 7, 5
i=152 j=153
17, 14, 13, 7, 5
i=153 j=154
17, 14, 13, 7, 5
i=154 j=155
17, 14, 13, 7, 5
i=155 j=156
17, 14, 13, 7, 5
i=156 j=157
17, 14, 13, 7, 5
i=157 j=158
17, 14, 13, 7, 5
i=158 j=159
17, 14, 13, 7, 5
i=159 j=160
17, 14, 13, 7, 5
i=160 j=161
17, 14, 13, 7, 5
i=161 j=162
17, 14, 13, 7, 5
i=162 j=163
17, 14, 13, 7, 5
i=163 j=164
17, 14, 13, 7, 5
i=164 j=165
17, 14, 13, 7, 5
i=165 j=166
17, 14, 13, 7, 5
i=166 j=167
17, 14, 13, 7, 5
i=167 j=168
17, 14, 13, 7, 5
i=168 j=169
17, 14, 13, 7, 5
i=169 j=170
17, 14, 13, 7, 5
i=170 j=171
17, 14, 13, 7, 5
i=171 j=172
17, 14, 13, 7, 5
i=172 j=173
17, 14, 13, 7, 5
i=173 j=174
17, 14, 13, 7, 5
i=174 j=175
17, 14, 13, 7, 5
i=175 j=176
17, 14, 13, 7, 5
i=176 j=177
17, 14, 13, 7, 5
i=177 j=178
17, 14, 13, 7, 5
i=178 j=179
17, 14, 13, 7, 5
i=179 j=180
17, 14, 13, 7, 5
i=180 j=181
17, 14, 13, 7, 5
i=181 j=182
17, 14, 13, 7, 5
i=182 j=183
17, 14, 13, 7, 5
i=183 j=184
17, 14, 13, 7, 5
i=184 j=185
17, 14, 13, 7, 5
i=185 j=186
17, 14, 13, 7, 5
i=186 j=187
17, 14, 13, 7, 5
i=187 j=188
17, 14, 13, 7, 5
i=188 j=189
17, 14, 13, 7, 5
i=189 j=190
17, 14, 13, 7, 5
i=190 j=191
17, 14, 13, 7, 5
i=191 j=192
17, 14, 13, 7, 5
i=192 j=193
17, 14, 13, 7, 5
i=193 j=194
17, 14, 13, 7, 5
i=194 j=195
17, 14, 13, 7, 5
i=195 j=196
17, 14, 13, 7, 5
i=196 j=197
17, 14, 13, 7, 5
i=197 j=198
17, 14, 13, 7, 5
i=198 j=199
17, 14, 13, 7, 5
i=199 j=200
17, 14, 13, 7, 5
i=200 j=201
17, 14, 13, 7, 5
i=201 j=202
17, 14, 13, 7, 5
i=202 j=203
17, 14, 13, 7, 5
i=203 j=204
17, 14, 13, 7, 5
i=204 j=205
17, 14, 13, 7, 5
i=205 j=206
17, 14, 13, 7, 5
i=206 j=207
17, 14, 13, 7, 5
i=207 j=208
17, 14, 13, 7, 5
i=208 j=209
17, 14, 13, 7, 5
i=209 j=210
17, 14, 13, 7, 5
i=210 j=211
17, 14, 13, 7, 5
i=211 j=212
17, 14, 13, 7, 5
i=212 j=213
17, 14, 13, 7, 5
i=213 j=214
17, 14, 13, 7, 5
i=214 j=215
17, 14, 13, 7, 5
i=215 j=216
17, 14, 13, 7, 5
i=216 j=217
17, 14, 13, 7, 5
i=217 j=218
17, 14, 13, 7, 5
i=218 j=219
17, 14, 13, 7, 5
i=219 j=220
17, 14, 13, 7, 5
i=220 j=221
17, 14, 13, 7, 5
i=221 j=222
17, 14, 13, 7, 5
i=222 j=223
17, 14, 13, 7, 5
i=223 j=224
17, 14, 13, 7, 5
i=224 j=225
17, 14, 13, 7, 5
i=225 j=226
17, 14, 13, 7, 5
i=226 j=227
17, 14, 13, 7, 5
i=227 j=228
17, 14, 13, 7, 5
i=228 j=229
17, 14, 13, 7, 5
i=229 j=230
17, 14, 13, 7, 5
i=230 j=231
17, 14, 13, 7, 5
i=231 j=232
17, 14, 13, 7, 5
i=232 j=233
17, 14, 13, 7, 5
i=233 j=234
17, 14, 13, 7, 5
i=234 j=235
17, 14, 13, 7, 5
i=235 j=236
17, 14, 13, 7, 5
i=236 j=237
17, 14, 13, 7, 5
i=237 j=238
17, 14, 13, 7, 5
i=238 j=239
17, 14, 13, 7, 5
i=239 j=240
17, 14, 13, 7, 5
i=240 j=241
17, 14, 13, 7, 5
i=241 j=242
17, 14, 13, 7, 5
i=242 j=243
17, 14, 13, 7, 5
i=243 j=244
17, 14, 13, 7, 5
i=244 j=245
17, 14, 13, 7, 5
i=245 j=246
17, 14, 13, 7, 5
i=246 j=247
17, 14, 13, 7, 5
i=247 j=248
17, 14, 13, 7, 5
i=248 j=249
17, 14, 13, 7, 5
i=249 j=250
17, 14, 13, 7, 5
i=250 j=251
17, 14, 13, 7, 5
i=251 j=252
17, 14, 13, 7, 5
i=252 j=253
17, 14, 13, 7, 5
i=253 j=254
17, 14, 13, 7, 5
i=254 j=255
17, 14, 13, 7, 5
i=255 j=256
17, 14, 13, 7, 5
i=256 j=257
17, 14, 13, 7, 5
i=257 j=258
17, 14, 13, 7, 5
i=258 j=259
17, 14, 13, 7, 5
i=259 j=260
17, 14, 13, 7, 5
i=260 j=261
17, 14, 13, 7, 5
i=261 j=262
17, 14, 13, 7, 5
i=262 j=263
17, 14, 13, 7, 5
i=263 j=264
17, 14, 13, 7, 5
i=264 j=265
17, 14, 13, 7, 5
i=265 j=266
17, 14, 13, 7, 5
i=266 j=267
17, 14, 13, 7, 5
i=267 j=268
17, 14, 13, 7, 5
i=268 j=269
17, 14, 13, 7, 5
i=269 j=270
17, 14, 13, 7, 5
i=270 j=271
17, 14, 13, 7, 5
i=271 j=272
17, 14, 13, 7, 5
i=272 j=273
17, 14, 13, 7, 5
i=273 j=274
17, 14, 13, 7, 5
i=274 j=275
17, 14, 13, 7, 5
i=275 j=276
17, 14, 13, 7, 5
i=276 j=277
17, 14, 13, 7, 5
i=277 j=278
17, 14, 13, 7, 5
i=278 j=279
17, 14, 13, 7, 5
i=279 j=280
17, 14, 13, 7, 5
i=280 j=281
17, 14, 13, 7, 5
i=281 j=282
17, 14, 13, 7, 5
i=282 j=283
17, 14, 13, 7, 5
i=283 j=284
17, 14, 13, 7, 5
i=284 j=285
17, 14, 13, 7, 5
i=285 j=286
17, 14, 13, 7, 5
i=286 j=287
17, 14, 13, 7, 5
i=287 j=288
17, 14, 13, 7, 5
i=288 j=289
17, 14, 13, 7, 5
i=289 j=290
17, 14, 13, 7, 5
i=290 j=291
17, 14, 13, 7, 5
i=291 j=292
17, 14, 13, 7, 5
i=292 j=293
17, 14, 13, 7, 5
i=293 j=294
17, 14, 13, 7, 5
i=294 j=295
17, 14, 13, 7, 5
i=295 j=296
17, 14, 13, 7, 5
i=296 j=297
17, 14, 13, 7, 5
i=297 j=298
17, 14, 13, 7, 5
i=298 j=299
17, 14, 13, 7, 5
i=299 j=300
17, 14, 13, 7, 5
i=300 j=301
17, 14, 13, 7, 5
i=301 j=302
17, 14, 13, 7, 5
i=302 j=303
17, 14, 13, 7, 5
i=303 j=304
17, 14, 13, 7, 5
i=304 j=305
17, 14, 13, 7, 5
i=305 j=306
17, 14, 13, 7, 5
i=306 j=307
17, 14, 13, 7, 5
i=307 j=308
17, 14, 13, 7, 5
i=308 j=309
17, 14, 13, 7, 5
i=309 j=310
17, 14, 13, 7, 5
i=310 j=311
17, 14, 13, 7, 5
i=311 j=312
17, 14, 13, 7, 5
i=312 j=313
17, 14, 13, 7, 5
i=313 j=314
17, 14, 13, 7, 5
i=314 j=315
17, 14, 13, 7, 5
i=315 j=316
17, 14, 13, 7, 5
i=316 j=317
17, 14, 13, 7, 5
i=317 j=318
17, 14, 13, 7, 5
i=318 j=319
17, 14, 13, 7, 5
i=319 j=320
17, 14, 13, 7, 5
i=320 j=321
17, 14, 13, 7, 5
i=321 j=322
17, 14, 13, 7, 5
i=322 j=323
17, 14, 13, 7, 5
i=323 j=324
17, 14, 13, 7, 5
i=324 j=325
17, 14, 13, 7, 5
i=325 j=326
17, 14, 13, 7, 5
i=326 j=327
17, 14, 13, 7, 5
i=327 j=328
17, 14, 13, 7, 5
i=328 j=329
17, 14, 13, 7, 5
i=329 j=330
17, 14, 13, 7, 5
i=330 j=331
17, 14, 13, 7, 5
i=331 j=332
17, 14, 13, 7, 5
i=332 j=333
17, 14, 13, 7, 5
i=333 j=334
17, 14, 13, 7, 5
i=334 j=335
17, 14, 13, 7, 5
i=335 j=336
17, 14, 13, 7, 5
i=336 j=337
17, 14, 13, 7, 5
i=337 j=338
17, 14, 13, 7, 5
i=338 j=339
17, 14, 13, 7, 5
i=339 j=340
17, 14, 13, 7, 5
i=340 j=341
17, 14, 13, 7, 5
i=341 j=342
17, 14, 13, 7, 5
i=342 j=343
17, 14, 13, 7, 5
i=343 j=344
17, 14, 13, 7, 5
i=344 j=345
17, 14, 13, 7, 5
i=345 j=346
17, 14, 13, 7, 5
i=346 j=347
17, 14, 13, 7, 5
i=347 j=348
17, 14, 13, 7, 5
i=348 j=349
17, 14, 13, 7, 5
i=349 j=350
17, 14, 13, 7, 5
i=350 j=351
17, 14, 13, 7, 5
i=351 j=352
17, 14, 13, 7, 5
i=352 j=353
17, 14, 13, 7, 5
i=353 j=354
17, 14, 13, 7, 5
i=354 j=355
17, 14, 13, 7, 5
i=355 j=356
17, 14, 13, 7, 5
i=356 j=357
17, 14, 13, 7, 5
i=357 j=358
17, 14, 13, 7, 5
i=358 j=359
17, 14, 13, 7, 5
i=359 j=360
17, 14, 13, 7, 5
i=360 j=361
17, 14, 13, 7, 5
i=361 j=362
17, 14, 13, 7, 5
i=362 j=363
17, 14, 13, 7, 5
i=363 j=364
17, 14, 13, 7, 5
i=364 j=365
17, 14, 13, 7, 5
i=365 j=366
17, 14, 13, 7, 5
i=366 j=367
17, 14, 13, 7, 5
i=367 j=368
17, 14, 13, 7, 5
i=368 j=369
17, 14, 13, 7, 5
i=369 j=370
17, 14, 13, 7, 5
i=370 j=371
17, 14, 13, 7, 5
i=371 j=372
17, 14, 13, 7, 5
i=372 j=373
17, 14, 13, 7, 5
i=373 j=374
17, 14, 13, 7, 5
i=374 j=375
17, 14, 13, 7, 5
i=375 j=376
17, 14, 13, 7, 5
i=376 j=377
17, 14, 13, 7, 5
i=377 j=378
17, 14, 13, 7, 5
i=378 j=379
17, 14, 13, 7, 5
i=379 j=380
17, 14, 13, 7, 5
i=380 j=381
17, 14, 13, 7, 5
i=381 j=382
17, 14, 13, 7, 5
i=382 j=383
17, 14, 13, 7, 5
i=383 j=384
17, 14, 13, 7, 5
i=384 j=385
17, 14, 13, 7, 5
i=385 j=386
17, 14, 13, 7, 5
i=386 j=387
17, 14, 13, 7, 5
i=387 j=388
17, 14, 13, 7, 5
i=388 j=389
17, 14, 13, 7, 5
i=389 j=390
17, 14, 13, 7, 5
i=390 j=391
17, 14, 13, 7, 5
i=391 j

SORTIRANJE UMETANJEM

(INSERTION SORT)

1. Algoritam; za sortiranje od manjeg ka većem

- 1) $a[m] \leftarrow$ unijeti
- 2) FOR $i := 2$ TO m
- 3) $k := a[i]$ // izvršava se $m-1$ puta.
- 4) $j := i-1$ // $m-1$
- 5) WHILE $(j > 0)$ AND $(a[j] > k)$ DO
- 6) $a[j+1] := a[j]$
- 7) $j := j-1$
- 8) END WHILE
- 9) $a[j+1] := k$
- 10) END FOR ima značenje NEXT i

Primer:

$$a[5] = \{a[1], a[2], a[3], a[4], a[5]\} = \{7, 17, 13, 14, 5\}$$

$$1+2+\dots+m-1 = \frac{m(m-1)}{2}$$

$$(2) i = 2$$

$$m=4 \text{ imamo } 1+2+3 = \frac{4 \cdot 3}{2} = 6$$

$$k := a[2] = 17$$

$$j = i-1 = 1$$

$$(5) j > 0 \wedge 7 > 17 \quad \perp$$

$$(9) a[2] = 17$$

$$(2) i = 3$$

$$k := a[3] = 13$$

$$j = 2$$

$$(5) 2 > 0 \wedge 17 > 13 \quad \text{w}$$

$$(6) a[3] = a[2] = 17$$

$$(5) j=1 \quad 1 > 0 \wedge 7 > 13 \quad \perp \Rightarrow (9) a[2] = 13$$

slično na ispitu (+ 1. zad.)

2. Ilustrirati sortiranje umetanjem tako da ćemo prikazati niz u opadajućem nizu kao 2. iz selekcije

$\{7, 17, 13, 14, 5\}$

$\{17, 7, 13, 14, 5\}$

$\{17, 13, 7, 14, 5\}$

$\{17, 14, 13, 7, 5\}$

s desna na lijevo tražimo odgovarajuće mjesto

i ostaje još 5 ali se mjesto ne mijenja

nedolazi na ispitu

3. Dat je niz $\{11, 5, 10\}$. Prikazati stanje niza nakon svake iteracije u ugnježdenoj petlji pri sortiranju umetanjem (u rastućem redoslijedu)

ne treba algoritam nego samo stanje niza kao u 2. zad., a algoritam je dat u 1. zad.

RIJEŠENJE:

$\{11, 11, 10\}$
// $\{5, 11, 10\}$
 $\{5, 11, 11\}$
// $\{5, 10, 11\}$

nakon ugnježdene p. ide pridruživanje ali nećemo vidjeti baš taj prikaz pa se tu javlja problem

7, 17, 13, 14, 5

17, 7, 13, 14, 5

17, 13, 7, 14, 5

17, 14, 13, 7, 5

METODA DIREKTNE ZAMJENE (BUBBLE SORT)

najsporniji

1. Algoritam:

POS := N

DO {

BOUND := POS

POS := 0

FOR i = 1 TO BOUND - 1 DO ako je radi

IF ($a[i] > a[i+1]$) THEN $a[i] \leftrightarrow a[i+1]$

POS := i

END IF

ISPIS NIZA *

END FOR

} WHILE (POS != 0) uslov dale radimo DO-WHILE

17, 5, 13, 12

5, 17, 13, 12

5, 13, 17, 12

5, 13, 12, 17

pa je 17 na svom mjestu

5, 13, 12, 17

5 i 13 su ok, ali 13 i 12 nisu =>

5, 12, 13, 17

5, 12, 13, 17

2. Prikazati stanja niza ~~pri~~ na kraju svake iteracije u ugnježdjenoj petlji pri sortiranju "bubble" u opadajućem redoslijedu. {7, 17, 13, 14, 5}

7, 17, 13, 14, 5

17, 13, 14, 7, 5

17, 7, 13, 14, 5

17, 14, 13, 7, 5

ima 8 karaka

17, 13, 7, 14, 5

17, 14, 13, 7, 5

17, 13, 14, 7, 5

17, 13, 14, 7, 5

POS = 5, $i = 1, 2, 3, 4, 5$

BOUND = 5

POS = 0

$i = 1, 4$

1° $i = 1$

IF $a_1 < a_2$ T

2° POS = 1

2° $i = 2$

IF $a_2 < a_3$ T

POS = 2

3° $i = 3$

IF $a_3 < a_4$ T

POS = 3

4° $i = 4$

IF $a_4 < a_5$ F

BOUND = 3

POS = 0

$i = 1, 2$

5° $i = 1$

IF $a_1 < a_2$

6° $i = 2$

IF $a_2 < a_3$ T

POS = 2

BOUND = 2

POS = 0

$i = 1, 1$

7° $i = 1$

IF $a_1 < a_2$

$(7, 17, 13, 14, 5)$

POS = 5

BOUND = 5

POS = 0

$i = 1, 4$

1° $i = 1$

IF $7 < 17$ T

2° $i = 2$

IF $17 < 13$ T

POS = 2

3° $i = 3$

IF $7 < 14$ T

POS = 3

4° $i = 4$

IF $7 < 5$ F

BOUND = 3

POS = 0

$i = 1, 2$

5° $i = 1$

IF $17 < 13$ F

6° $i = 2$

IF $13 < 14$ T

POS = 2

BOUND = 2

POS = 0

$i = 1, 1$

7° $i = 1$

IF $17 < 14$ F

$(7, 17, 13, 14, 5)$

$(17, 13, 14, 5)$

$(17, 13, 7, 14, 5)$

$(17, 13, 14, 7, 5)$

$(17, 13, 14, 7, 5)$

$(17, 14, 13, 7, 5)$

$(17, 14, 13, 7, 5)$

PARTIČJSKO SORTIRANJE (QUICK SORT)

Algoritam:

```
(1) QUICKSORT (a, low, high) {  
(2)   IF (low < high) THEN  
(3)       j = PARTITION (a, low, high)  
(4)       QUICKSORT (a, low, j-1)  
(5)       QUICKSORT (a, j+1, high)  
(6)   END IF  
}
```

Imamo rekurziju - QUICKSORT poziva sam sebe
čak 2 puta.

```
(7) PARTITION (a, down, up) {  
(8)   i = down  
(9)   j = up  
(10)  pivot = a[down]  
(11)  WHILE (i < j) DO  
(12)      WHILE (a[i] ≤ pivot) AND (i < j) DO  
          i = i + 1  
      END WHILE  
(13)      WHILE (a[j] > pivot) DO  
          j = j - 1  
      END WHILE  
(14)      IF (i < j) THEN  
          a[i] ↔ a[j]  
      END IF  
(15)  END WHILE  
(16)  a[down] = a[j]  
(17)  a[j] = pivot  
(18)  RETURN j  
}
```

1. Prema datom algoritmu Quicksort-a ilustrirati sortiranje niza $\{1, 17, 13, 2\}$ ispisivanjem stanja niza nakon linije.

$a \leftarrow \{1, 17, 13, 2\}$ prvi i zadnji indeks

QUICKSORT($a, 1, 4$)

$1 < 4 \Rightarrow j = \text{PARTITION}(a, 1, 4) \Rightarrow i = 1; j = 4 \Rightarrow \text{pivot} = a[i] = 1 \Rightarrow$

$j = 1$

WHILE $i < j$

WHILE $(i \leq 1) \wedge (i < 4) \quad i \leftarrow 2$

WHILE $(a[j] > 1) \quad j \leftarrow 3$

$a[3] = 13$

$a[3] = 13 \quad j \leftarrow 2$

$a[2] = 17 \quad j \leftarrow 1$

$i < j \quad 1 < 1$

END WHILE

$a[1] = a[j]$

$a[j] = 1$

RETURN j

$a_1 \quad a_2 \quad a_3 \quad a_4$
 $1 \quad 17 \quad 13 \quad 2$

QUICKSORT($a, 1, 0$) $1 < 0 \perp \text{KRAJ}$

QUICKSORT($a, 2, 4$) $\Rightarrow 2 < 4 \perp \Rightarrow j = \text{PARTITION}(a, 2, 4)$

$i = 2; j = 4$

pivot = 17

$a_2 \leq 17 \quad 2 < 4$

$i = 3$

$a_4 > 17$

$2 < 4$

$a[2] = 2$

$a[2] = 2$

$a[4] = 17$

$i = 4$ (efekat linije (11))

$j = 4$ (efekat linije (12))

$a[2] = 2$

$a[4] = 17 \quad \{1, 2, 13, 17\}$

RETURN j

$(j = 4)$

QUICKSORT($a, 2, 3$) $\Rightarrow 2 < 3 \Rightarrow$

$\Rightarrow j = \text{PARTITION}(a, 2, 3)$

Primer: $\{17, 5, 13, \underline{19}, 8, \underline{14}\}$

17 - pivot

Tražim prvi veći od njega (malesmo)

to je 19 i on je u indeksu i ($i=3$)

Zatim tražim s desna na lijevo prvi manji ili jednak pivotu (17) i to je 14 (s indeksom j), ako je $i < j$ onda zamijeni ($14 \leq 6$ indeksi i, j)

(*) $\{17, 5, 13, 14, 8, 19\}$

sada traži dalje prvi veći i sada i staje kod 19, a j kod 8, ali ovdje nije $i < j$ pa ne mijenja pa imamo

$16 < 5$

$\{8, 5, 13, 14, 17, 19\}$

sada je 8 - pivot i traži prvi veći s desne str. i s lijeve

$\{5, 8, 13, 14, 17, 19\}$

gleda podniz sa 5; i tu nema šta raditi
gleda drugi podniz $8, 13, 14$

u gl. programu je
QUICKSORT (a, l, r)

{particija

QUICKSORT (a, l, n)^{high} ←

{partition...

QUICKSORT (a, l, l) } (ništa se ne dešava)

QUICKSORT (a, l, n)^{high}

(ovo je unutar a)

a gdje je 2? njega presbače jer je bio pivot, on je na svom mjestu

2. $\{11, 8, 5, 20, 3, 1\}$

QUICKSORT(a, 1, 6)

$1 < 6$ T

$i \leftarrow 1, j \leftarrow 6$

pivot $\leftarrow 11$

$(11, 8, 5, 1, 3, 20)$

- traži veći od "Pivota" pa je $20 > 11$
pa je i kod 20

- sada su i i j jer je sada
pivot = 3 (zbog 15, 16, 17)

$(3, 8, 5, 1, 11, 20)$ $j \leftarrow 5$

QUICKSORT(a, 1, 5)

pivot $\leftarrow 3$

$i < j$ pa

$(3, 1, 5, 8, 11, 20)$

sada opet traži veći s desna pa
je i kod 5, a j kod 1

$(i < j \perp)$ pa

1 i 3 mijenja mjesta

$(1, 3, 5, 8, 11, 20)$

sada opet gleda da vidi
dokle stoji i ima li promjena

a) ispisati sve pozive fje QUICKSORT (redoslijedom izvršavanja)

QUICKSORT(a, 1, 6)

QUICKSORT(a, 1, 5)

QUICKSORT(a, 1, 1) odmah završava

QUICKSORT(a, 3, 5)

QUICKSORT(a, 3, 2) odmah završava

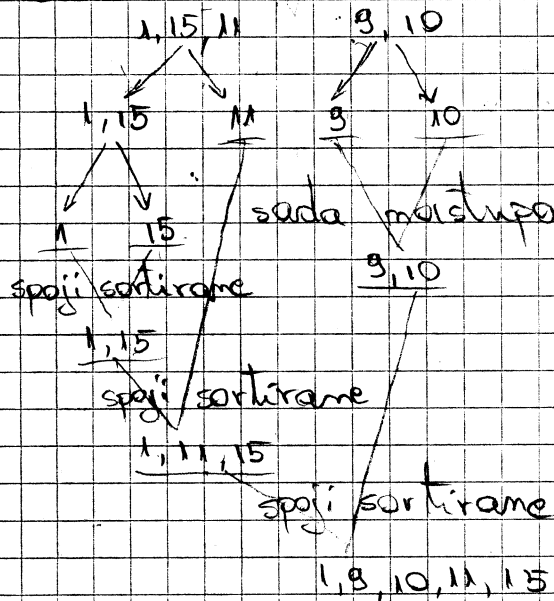
QUICKSORT(a, 5, 5)

- 11 -

QUICKSORT(a, 6, 6)

MERGE-SORT (Sortiranje spajanjem)

Primjer: 1, 15, 11, 9, 10 (sortiramo u neopadajućem redoslijedu)
(ideja)



sada nastupa procedura spajanja:

podvučeni su
sortirani nizovi

Spoji Sortirane ($\{1\}$ $\{15\}$)
 Spoji Sortirane ($\{1, 15\}$ $\{11\}$)
 Spoji Sortirane ($\{9\}$ $\{10\}$)
 Spoji Sortirane ($\{1, 11, 15\}$ $\{9, 10\}$)

5 elemenata

n podjele

$\lceil \frac{n}{2} \rceil$ $O(n)$ za podjele

$n \rightarrow$ broj elemenata: T -vrijeme

$O(n) \cdot T_{\text{spajanja}} \sum_{i=1}^{\lceil \frac{n}{2} \rceil} T_{\text{spajanje}}$

1. Napisati algoritam za spajanje dva ~~razdvojena~~ ^{sortirana} niza
u sortirani niz.

ALGORITAM: Sort Spajanjem (A, p, q)

ako je ($p < q$) onda {

$r \leftarrow \text{CijeliDio}((p+q)/2)$

Sort Spajanjem (A, p, r)

Sort Spajanjem ($A, r+1, q$)

Spaji Sortiranje (A, p, r, q)

}

2. Dati rimski broj pretvoriti u decimalni (algoritam)
3. Algoritam za malaženje n-tog člana F. niza.
4. Poređivanje (faktORIZACIJA) prirodnog broja
 - a) faktORIZACIJA prirodnog br. (rastavljanje na prirodne faktore)
 - b) za ispitivanje da li je dati prirodan br. prost

Za Fibonačijev niz

$$\begin{bmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$$

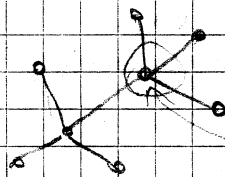
$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^2 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}^3 = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \dots$$

ima neka formula
po kojoj se računa
F. niz (ovo nije siguran
ratio)

Elementarne strukture podataka

- stek
- red
- vezna lista
- stablo (ukorijenjeno)
- implementacije ovih struktura podataka vršićemo pomoću nizova.



ako uzmemo
ovo kao root
onda ga
predstavljamo
kao korijen
stabla

Stog - sastoji se od raznih tipova pod.

STEK:

Karakteristike su:

- dinamički skup (u njemu se mijenja broj elemenata; skup se mijenja)
- operacije: PUSH (umetanje, dodavanje elemenata)
POP (brisanje, uzimanje elemenata)
(odstranjivanje)
- redoslijed uzimanja elemenata obrnut je redoslijedu dodavanja

Implementacije možemo izvršiti nizom S sa najviše n elemenata.

Postoji svojstvo niza $top[S]$ koje daje indeks posljednjeg umetnutog elementa

Dakle, stek se sastoji od $S[1 \dots top[S]]$ elemenata.
 $S[i]$ predstavlja element na i -m steka.



limit steka
(me razmotrano)

S

1	2	3	4	5	6	7
15	6	2	9			1

 \rightarrow implementacije sa 7 elementa

Slika 1. \nearrow $top[S] = h$

Implementacija operacija:

STACK_EMPTY(S)

if $top[S] = \emptyset$
 then return TRUE (tačno je da je prazan)
 else return FALSE

PUSH(S, x)

$top[S] \leftarrow top[S] + 1$

$S[top[S]] \leftarrow x$

povećava indeks poslednjeg

POP(S)

if STACK_EMPTY(S)

then error "under flow"

else $top[S] \leftarrow top[S] - 1$

return $S[top[S] + 1]$

1. Koristeći sliku 1. kao model predstaviti rezultat uzastopnih izvršavanja sledećih operacija

da doda h
 dodavanje 1
 dodavanje 3

(1) PUSH(S, h)

(2) PUSH(S, 1)

(3) PUSH(S, 3)

(4) POP(S)

(5) PUSH(S, 8)

(6) POP(S)

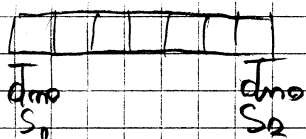
na početnom praznom steku S namještenom u niz S[1..n]

1	2	3	4	5	6	
						- S
h						(1)
h	1					(2)
h	1	3				(3)
h	1					(4)
h	1	8				(5)
h	1					(6)

2. Objasniti kako implementirati dva steka u jednom nizu $A[1..m]$ na takav način da se ne desava preobracenje nijednog od stekova suc dok je ukupan broj elemenata u oba steka manji od m .

S_1, S_2 - stekovi

S - niz



- dobijemo dupli stek prezentirano nižu

- imamo više ideja: npr. mijetiti 1 stek, pa drugi ili možda ništa, ali mora se paziti na ograničenje niža, ne smije biti > od n
- dno S₁ je na m₁, a dno S₂ je na m₁+1

Kod steka se ne mora gledati konvencija redoslijeda računarskih operacija npr. $A+B \times C$

$A+B \Rightarrow +AB \Rightarrow AB+$
infixni izraz prefiksni postfixni

3. Konvertovati infixni izraz u postfixni
 $A+B \times (C-D) + (E-F) \times G/H$ A možemo staviti najprije, ne mora biti na početku

Zg: $((A+(B \times (C-D))) + ((E-F) \times G)/H)$

Rješenje: $ABCD - * + EF - G * H / +$

(Prvo računamo $C-D$ a to je I_1 , onda je $B \times I_1$ pa $A + I_2$) sortirano je jer oduzima e i f (stijeva na desno).

PRINCIP:

Operatori se premještaju na mjesto desnih zagrada a lijeva zagrada se uklanja

4. Napisati u postfixnom zapisu izraz:

$A - B \times C / (D + E \times F)$

- operatori se premještaju na mjesto desnih zagrada, a lijeva zagrada se uklanja

$(A - ((B \times C) / (D + (E \times F))))$

$A((B \times C) / (D + (E \times F))) -$

$A((B \times C))(D + (E \times F)) / -$

$A(BC \times) / (D + (E \times F) +) / -$

$A(BC \times)(D(EF \times) +) / -$

na 1. zagradu ide *
a lijevu zagradu uklanjamo
na 2. zagradu ide +
na 3. ide /
na 4. ide -

$I_2 = I_1 / I_3$
 $I_1 = A$

Rj: $ABC \times DEF \times + / -$

$I_1 = B \times C$

$I_2 = D + I_1$

$I_5 = A - I_4 = A - I_1 / I_3 = A - ((B \times C) / (D + I_2)) = A - ((B \times C) / (D + (E \times F)))$

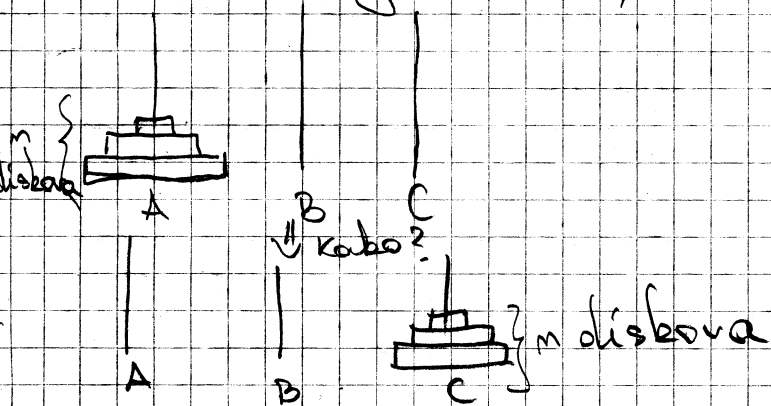
$I_2 = E \times F$

$I_4 = I_1 / I_2$

PRINCIP STAKA

Problem tornjeva Hanoja

1. Postoje 3 štapa A, B, C, a na štapu A se nalazi ~~različnih~~ različitih diskova poredanih po veličini tako da je najveći na dnu, a najmanji na vrhu. Potrebno je da se diskovi prebace sa štapa A na štap C koristeći štap B kao pomoćni. Pri tome se sa štapa može uklanjati samo disk sa vrha, a ni u jednom trenutku se ne dozvoljava da ~~manji~~ manji disk bude ispod većeg diska na bilo kojem štapu.



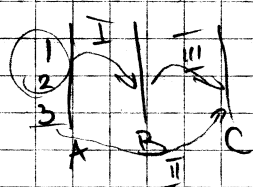
$$x_1 = A \rightarrow C$$

$$x_2 = C \rightarrow B, x_1(2), B \rightarrow C$$

$$n=1 \quad \text{ide} \quad A \rightarrow C$$

$$n=2 \quad A \rightarrow B \quad A \rightarrow C \quad B \rightarrow C$$

$$n=3$$



polazni
pomoćni
ciljni

$$n=n$$

$$A \rightarrow B$$

$$A \rightarrow C$$

$$B \rightarrow A$$

$$B \rightarrow C$$

$$A \rightarrow B$$

$$A \rightarrow C$$

$$B \rightarrow C$$

Izgled programa:

HANOJ {

PREBACI (SA, POMOCU, NA)

Algoritama formiranja Hamija:

INPUT m

PRINT $homog(m)$

FUNCTION $homog(m)$

IF $m=1$ THEN

$homog = "A-C"$

ELSE

$h1 = homog(m-1)$

FOR $i=1$ TO $len(h1)$

SELECT CASE $mid(h1, i, 1)$

CASE "B"

$h1 = left(h1, i-1) + "C" + right(h1, len(h1)-i)$

CASE "C"

$h1 = left(h1, i-1) + "B" + right(h1, len(h1)-i)$

END SELECT

NEXT i

$h2 = homog(m-1)$

FOR $i=1$ TO $len(h2)$

SELECT CASE $mid(h2, i, 1)$

CASE "A"

$h2 = left(h2, i-1) + "B" + right(h2, len(h2)-i)$

CASE "B"

$h2 = left(h2, i-1) + "A" + right(h2, len(h2)-i)$

END SELECT

NEXT i

$homog = h1 + "A-C" + h2$

END IF

END FUNCTION

$m=3$ (rekurzivno) broj prebacivanja je $2^m - 1$

1° A-C

2° A-B A-C B-C

3° A-C A-B C-B A-C B-A B-C A-C

$3+2-1$

STRING

A = "ČELIK ZENKA"

LEFT(A, 5) daje "ČELIK"

RIGHT(A, 6) daje "ZENKA"

MID(A, 5, 3) daje "KLZ"

II maćim:

```
premjesti(sa, ma) { cout << sa << "-" << ma << endl; }
```

```
int main() {
```

```
    cin << m
```

```
    A[], B[], C[];
```

```
    prebaci(A, B, C, m);
```

```
}
```

```
FUNCTION prebaci(sa, prebo, ma, koliko)
```

```
if (m=1)
```

```
    premjesti(sa, ma);
```

```
else
```

```
    prebaci(sa, ma, prebo, koliko-1)
```

```
    premjesti(sa, ma);
```

```
    prebaci(prebo, sa, ma, koliko-1)
```

```
END FUNCTION
```

RED (queue)

- elementarna linearna dinamička struktura podataka
- operacije: ENQUEUE (ulazak u red, tj. operacija umetanja/insert)
DEQUEUE (izlazak iz reda, operacija ublajanja/delete)
- osobine:
 - ima čelo/head koje ukazuje na element na početku reda
 - ima zatečje/tail koje ukazuje na slobodno mjesto za ulazak u red
 - izlazak iz reda može se primijeniti samo za element na čelu
 - ulazak u red može se izvršiti samo na zatečju

Pr (implementacija reda preko niza)

a) Q

1	2	3	4	5	6	7	8	9	10	11	12
						15	6	9	8	4	

head(Q)=7 tail(Q)=12

b) red Q

1	2	3	4	5	6	7	8	9	10	11	12
3	5					15	6	9	8	4	17

tail(Q)=3 head(Q)=7

dodaje 17, 3, 5 na a)

c) red Q

1	2	3	4	5	6	7	8	9	10	11	12
3	5						15	6	9	8	17

tail(Q)=3 head(Q)=8

uzima 15 iz b)

a) red Q ima 5 elemenata: 15, 6, 9, 8, 4 na čelu je 7, na lokacijama 7, ..., 11

b) ENQUEUE(Q, 17), ENQUEUE(Q, 3), ENQUEUE(Q, 5)
na red a)

c) stavljaj makom operacije $DEQUEUE(Q)$ na redn pod b)

Kada bi u redn Q izvršile 5 operacija $ENQUEUE$ dobili bi „overflow“ (prekoraćenje) jer bi $\underline{tail(Q) = head(Q)}$

$ENQUEUE(Q, x)$

$Q \{ tail[Q] \} \leftarrow x$

if $tail[Q] = length[Q]$

then $tail[Q] \leftarrow 1$

else $tail[Q] \leftarrow tail[Q] + 1$

$DEQUEUE(Q)$

$x \leftarrow Q[head[Q]]$

if $head[Q] = length[Q]$

then $head[Q] \leftarrow 1$

else $head[Q] \leftarrow head[Q] + 1$

return x

1. koristeći primjer 1. kao model ilustrirajte rezultat svoje operacije sekvence

$ENQUEUE(Q, 4)$, ~~$ENQUEUE(Q, 1)$~~

$ENQUEUE(Q, 1)$

~~$DEQUEUE(Q)$~~

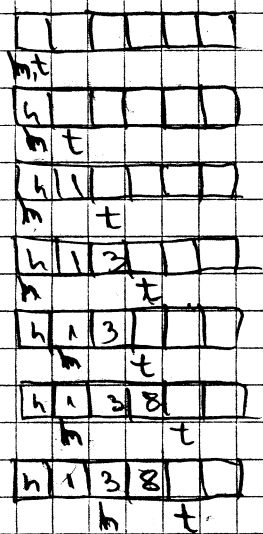
$ENQUEUE(Q, 8)$

$DEQUEUE(Q)$

4) 1
1) 8

na početnom praznom redn Q smještenom u nizu od $Q[1, \dots, 6]$ (od 6 elemenata)

NAPOMENA: nije sagrađeno ispitivanje "overflow-a" ni "underflow-a" (zadato je da se skine element a red je prazan)



2. Dodati procedure ENQUEUE i DEQUEUE tako da prijavljuju "underflow" i "overflow"

if ($\text{tail}[Q] - \text{head}[Q] + 1 == \text{length}[Q]$)

OR

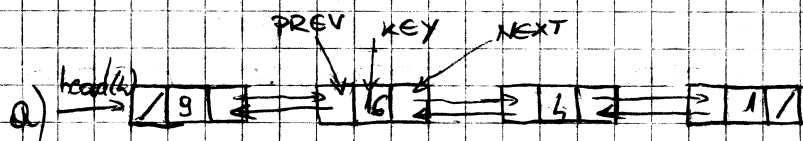
($\text{tail}[Q] + 1 == \text{head}[Q]$)

then print "queue overflow"

3. Pokazati kako se implementira red koristeći 2 steka.

VEZANA LISTA (ulazna lista)

bice na ispitu



L-lista (glava, čelo lista)

PREV (previous, prethodni)

NEXT (sljedeći)

KEY (ključni)

"1" označava tzv. NIL tj. nedefinisana vrijednost i koristi se za označavanje ili detekciju kraja (početka) liste

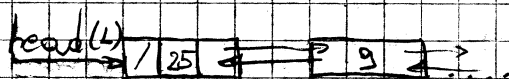
Ovo je bio primjer jedne vezane liste L. (dvostruko vezane liste) - ima i PREV i NEXT

b) U listi pod a) izvršimo LIST_INSERT(L, x), gdje je key [x] = 25

(NAPOMENA: x je pointerskog tipa;

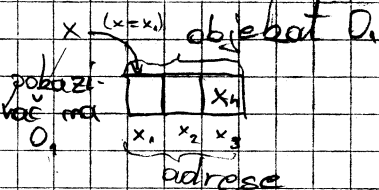
pointer (mem. lokacija)

LIST_INSERT ubacuje element (objekat - jer imamo pored key i adrese prev i next, na početak liste.

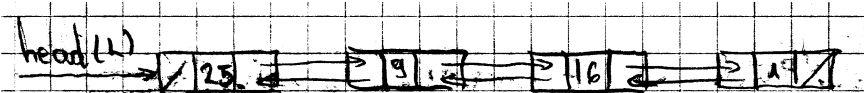


c) U listi pod b) izvršimo LIST_DELETE(L, x), gdje x pokazuje na objekat sa ključem k

kako pristupiti proizvoljnom elementu (mpr. k. u listi) prema poziciji u poretku?



O1 je prethodni za O2, xn upućuje na O2 tj. adresu xn, a x1 upućuje na O1 tj. x1 - početak od O1



Najopštija elementarna lista je dvostruko vezana lista.

PRETRAŽIVANJE (SEARCHING) VEZANE LISTE

LIST-SEARCH(L, b)

Možemo koristiti tail kao bod redova (eventualno stekova) da bi od head prebismo next, next... došli na kraj, a i ne moramo koristiti tail jer imamo "na - kraju".

Npr. LIST-SEARCH(L, 16)

Ide od početka liste i traži ključni element 16.

OSOBINE/METODE/OPERACIJE:

head(L) daje adresu i objekta u listi

prev(x) daje adresu prethodnog objekta u odnosu na objekat sa adrese x

key(x) daje ključnu vrijednost objekta sa adrese x

next(x) daje adresu sljedećeg objekta u odnosu na objekat sa adrese x

tail(L) adresa posljednjeg objekta u listi (za razliku od redova gdje je i a ne posljednji)

Kod listi, za razliku od nizova, imamo podatke raznih tipova, a mi samo radimo sa adresama. Bitno nam je da dajemo na početak - na adresu L.

Pr/ LIST_SEARCH(L, 16)

Treba da vrati adresu objekta čija je ključna vrijednost jednaka 16.

Algoritam:

```
P = head(L)
DO
  IF (key(P) = k)
    RETURN P
ELSE
  P = next(P)      (uzima sljedeće P)
WHILE (P != tail(L))
RETURN "/"
```

Za liste, za razliku od nizova gdje je poredak određen indeksima, u vezanim listama poredak je određen pointerima.

UMETANJE OBJEKATA U LISTU:

LIST_INSERT(L, x)

adresa tog novog podatka

U adresu L ubacimo ono sa adrese x - LIST_INSERT

P = head(L)

head(L) = x

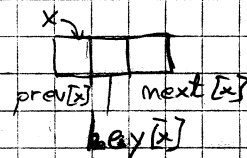
prev(P) = x

next(x) = P

prev(x) = "/"

9, 16, 4, 1

P = 9
head(L) = 25
prev(P) = 25
next(25) = 9
prev(25) = "/"

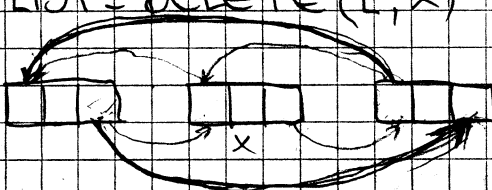


Ako hocemo da ubacimo negdje izmedu
recimo 9, 16, 4, 1 pomjeramo samo 16 i 4
a ne sve kao kod niza;

L ₁	x ₁
L ₂	x ₂
L ₃	x ₃
L ₄	x ₄
L ₅	x ₅
L ₆	x ₆

BRISANJE IZ LISTE:

LIST_DELETE(L, x)



25, 9, 16, 4, 1

x = 4
next(x) = 1
prev(x) = 16
prev(x) = next(x)
next(x) = prev(x)

IF prev(x) = / THEN

next(x) = /

ELSE IF next(x) = / THEN

prev(x) = /

ELSE
next(prev(x)) = next(x)

prev(next(x)) = prev(x)

END IF

IF prev[x] != "/" THEN

next[prev[x]] = next[x]

ELSE head[L] = next[x]

IF next[x] != "/" THEN

prev(next[x]) = prev[x]

NELINEARNE STRUKTURE PODATAKA

U realnosti često nailazimo na objekte i procese sa složenim unutaranjim vezama gdje je jedan element povezan sa više elemenata i gdje različiti elementi mogu biti povezani sa različitim brojem drugih elemenata. Pored toga i svako od veza može biti pridružena neka numerička vrijednost. Ovakve objekte i procese modeliramo nelinearnim strukturama podataka koje su sposobne da predstavljaju višedimenzionalne relacije na fleksibilan način.

Tipične operacije koje se primjenjuju na nelinearne strukture su slične onim kod linearnih, pa imamo:

- obilazak svih elemenata u nekom definisanom nelinearnom poretku
 - pretraživanje zadane vrijednosti
 - pristup proizvoljnom elementu
 - umetanje novog elementa
 - brisanje postojećeg elementa
- ali imamo i neke specifične operacije:
- nalaženje različitih puteva
 - nalaženje najbržih rastojanja, itd.

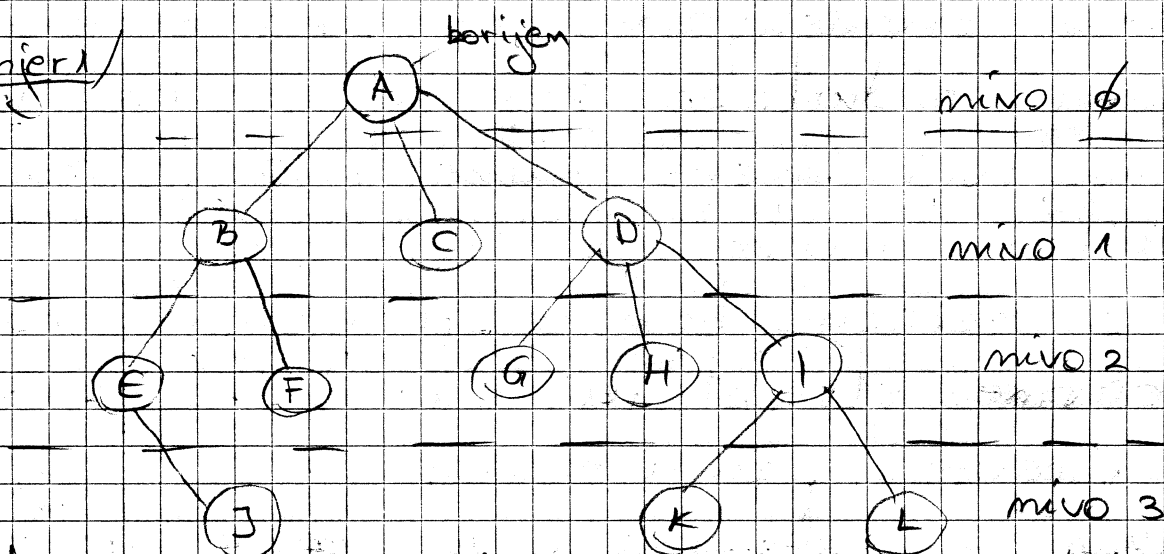
Osnovne vrste struktura su:

- NELINEARNA STABLA i stabla su imaju specijalan slučaj grafova
- GRAFOVI

Stabla (korijena):

DEF: Stablo T je konačan neprazan skup elemenata proizvoljnog tipa (elemente zovemo čvorovi) takav da - postoji jedan poseban čvor koji se naziva korijen - ostali čvorovi se mogu razdvojiti u $m \geq 0$ disjunktivnih skupova T_1, \dots, T_m koji su također stabla. Ova stabla se nazivaju podstabla korijena.

Primjer 1

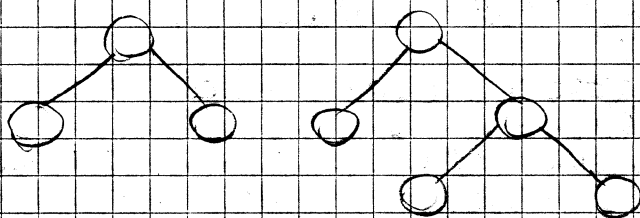


stablo sa 12 čvorova (iz \forall čvora može se doći do bilo kojeg drugog čvora)

korijen neće biti list

Listovi: C, F, G, H, J, K, L

Primjer 2



šema (model sastavljen od više stabala)

Memorijska reprezentacija

Najčešće se koristi ulamčana reprezentacija (vezana lista)
(vezana, borištenjem pointera)

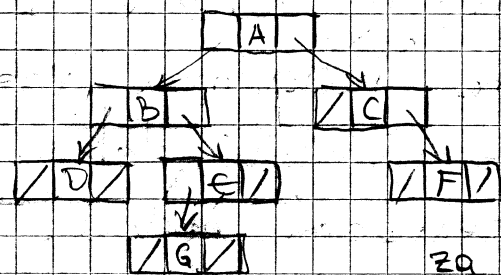
Primjer linearnog stabla:



Ako stablo ima n čvorova, maksimalan broj veza (grana) koje može imati 1 čvor je $n-1$.

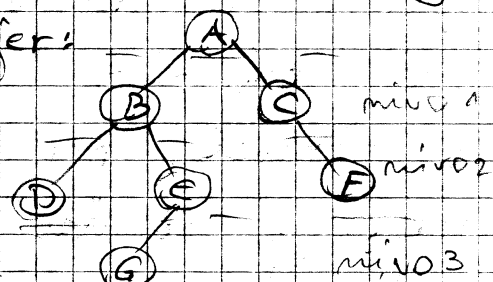
Reprezentacija stabla zavisi od problema koji rješavamo.

Ulamčana reprezentacija:

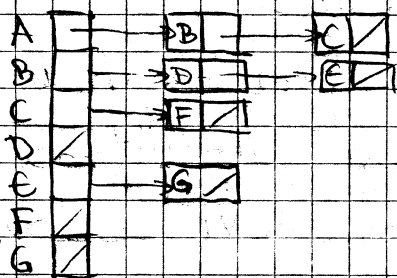


za binarno stabla

Primjer:



Reprezentacija preko niza vezanih lista



(susjedni članovi)

za one koji nemaju susjedne (dijete) stavimo nulu tj. /

Najduži put je put od 3 grane = nivo 3

Binarna stabla - primjena: Huffmanovi kodovi

Jedna od klasičnih primjena je kompresija podataka pri njihovom čuvanju ili prijenosu, da bi se smanjilo memorijsko zauzeće ili vrijeme prijenosa. Neka se npr. poruke generišu od abecede koja ima x različitih simbola. Prolazi se problem kodiranja i dekodiranja poruke. Pri čemu se svaki simbol predstavlja nekim binarnim kodom određene dužine tako da dužina poruke bude minimalna, a proces kod. i dekodiranja ostane dovoljno prost. Također, pretpostavlja se da su poznate vjerovatnoće p_1, \dots, p_x izražene u procentima sa kojima se javljaju pojedini simboli u poruci pri čemu je $\sum_{i=1}^x p_i = 100$. Pokazuje se da skraćivanje dužine kodirane poruke može da se postigne kodovima različite dužine, pri tom zbog jednoznačnosti dekodiranja ovo treba da budu prefiksni kodovi. To znači da binarni kod ni jednog simbola ne smije da bude sadržan na početku neke binarne sekvence bilo kojeg drugog binarnog simbola. Sam postupak izbora optimalnih prefiksnih kodova se obavlja primjenom Huffmanovog algoritma pa se ovi kodovi nazivaju Huffmanovi kodovi.

HUFFMAN (W, e)

INIT-QUEUE (H, e)

FOR i=1 TO e DO

z = GETNODE

w(z) = w[i]

PQ-INSERT (H, z)

END FOR

FOR i=1 TO e-1 DO

z = GETNODE

rade iz reda
2 čvora sa
najmanjom
vjerovatno-
šćom

x = PQ-MIN-DELETE (H)

y = PQ-MIN-DELETE (H)

w(z) = w(x) + w(y)

left(z) = x

right(z) = y

PQ-INSERT (H, z)

END FOR

RETURN z

inicijalizacija reda H

/* predvidi da red H
može primiti najviše
e elemenata */

/* kreiraj objekat tipa
"čvor" dodijeli ga varijabli z */

// osobina čvora z je vjerovatnoća pri

element-objekat
"čvor" z */

GET - uzeti
H-red

e - broj simbola

w - niz vjerovatnoća
(u %)

NODE - čvor

// novi čvor

// traži najmanji element u
redu, vrati ga i izbriši iz reda

// novi čvor će imati osobinu vjerovat-
noća jednaku zbiru dvije upravo
nađene

// z će biti roditelj čvorova sa prethodno
nađenim min. vjerovatnoćama

// ubaci u red novi čvor

tzv. "GREEDY" algoritam
(pothlepni)

1. Naći optimalan prefiks kod za tekst u kome
se pojavljuju simboli predstavljani sljedećom
tablicom.

SIMBOLI	A	B	C	D	E	F
VJEROVATNOĆE (%)	42	6	31	7	4	10
	w[1]	w[2]				w[6]

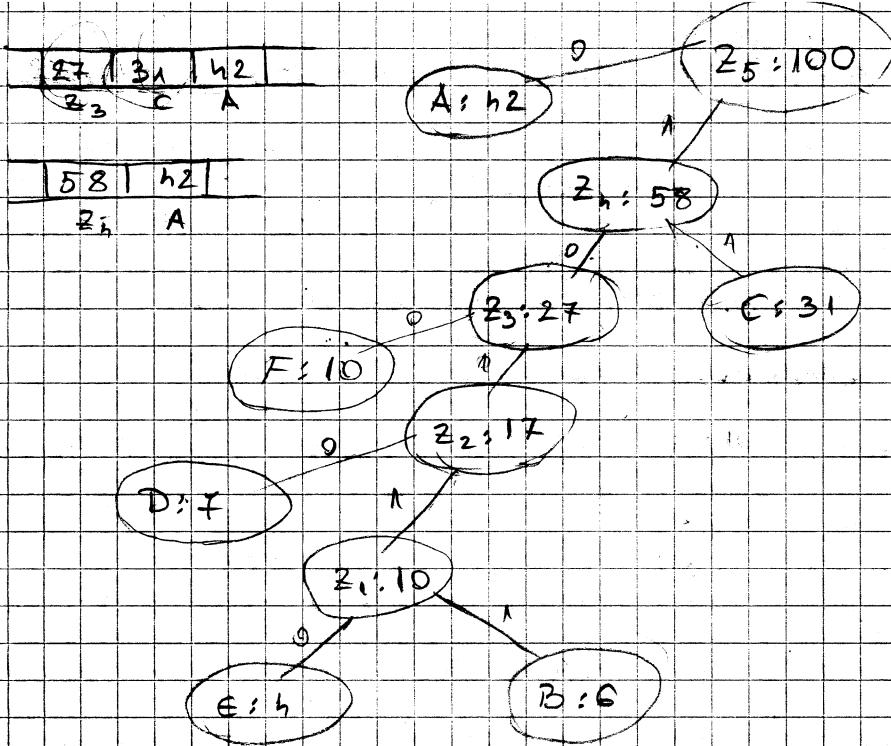
tail	H	head
10 7 31 6 42		
F E D C B A		

e=6

10 10 7 31 42	
31 F D C A	

17 10 31 42	
31 F C A	

pravićemo čvor z čija su djeca
najmanja (4 i 6) - izbacujemo ih
a dodajemo njihovu zbiru na početak



E: 10110

F: 100

B: 10111

C: 11

D: 1010

A: 0

Dakle, ako je veličina polaznog fajla bila 100000 bita nakon ovog sortiranja biće

A će zauzeti 12000 bitova

B: 6000

C: 31000

D: 7000

E: 4000

F: 10000

Tipično kodiranje:

A: 000

B: 001

C: 010

D: 011

E: 100

F: 1011

8-bitno kodiranje:

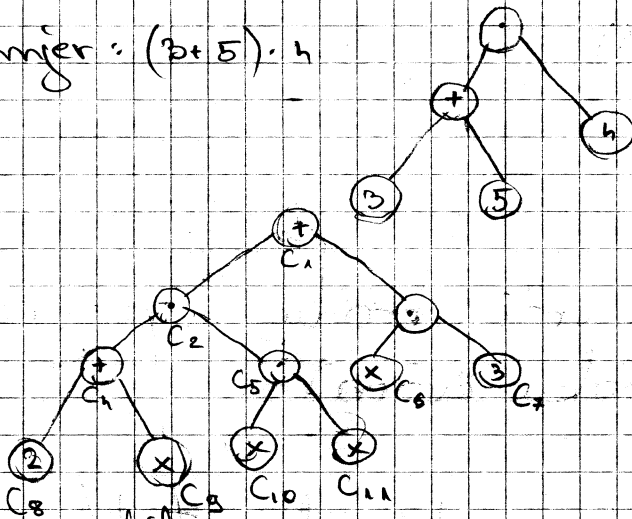
Za E smo imali 4000 (3) a sad 10110 (5) pa imamo

$$\frac{1}{3} \cdot 12000 + \frac{5}{3} \cdot 6000 + \frac{2}{3} \cdot 31000 + \frac{1}{3} \cdot 7000 + \frac{5}{3} \cdot 4000 + 10000$$

Primjena stabla - predstavljanje aritmetičkih izraza

1. $3x + (2 + x)x^2$ predstaviti binarnim stablom

Primjer: $(3 + 5) \cdot 4$



slika 1

Obilasci stabla

Rekurzivne realizacije:

1) PREORDER (root)
if (root \neq nil) then
 r P(root)
 T₁ PREORDER(left(root))
 T₂ PREORDER(right(root))
end if

najprije posjećuje r, zatim T₁,
pa T₂, ... na kraju T_k

2) INORDER (root)
if (root \neq nil) then
 T₁ INORDER(left(root))
 r P(root)
 T₂ INORDER(right(root))
end if

najprije posjećuje T₁, pa r,
pa T₂, ... T_k

3) POSTORDER (root) najprije obilazi T_1, T_2, \dots, T_k
 if (root \neq nil) then pa na kraju r
 T_1 POSTORDER (left (root))
 T_2 POSTORDER (right (root))
 r P (root)
 end if

P(i) je procedura koja vrši obradu ^{nad svim čvorovima} i posjećuje čvor (i mpr ispisuje oznaku čvora)

2. Izvršiti svaku od procedura obilaska stabla na primjeru sa slike 1.

1) PREORDER (C₁) - početni poziv (mpr. u glavnom programu)

C₁ C₂ C₄ C₈ C₉ C₅ C₁₀ C₁₁ C₃ C₆ C₇
 + . + 2 X . X X . X 3 (pre. notacija)

2) INORDER (C₁) C₈ C₄ C₉ C₂ C₁₀ C₅ C₁₁ C₁ C₆ C₃ C₇

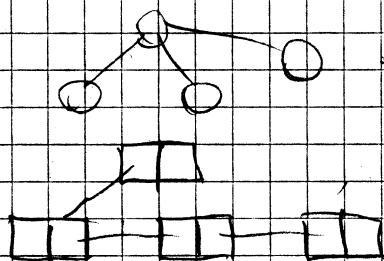
2 + X . + X . X + X 3

3) POSTORDER (C₁) obilazi listove po redu

C₈ C₉

2 X + X X . X 3 . +

Svako stablo se može predstaviti kao binarno stablo (lijevo-sin, desno-brat pa se nadovezuje)



$$a_0 x^0 + a_1 x^1 + a_2 x^2 + \dots + a_m x^m$$

a_i koef. monoma
 x^i stepen monoma

Kako predstaviti polinom (pomoću koje strukture,

Niz nam nije interesantan

$$P[m] = \{a_0, a_1, \dots, a_{m-1}\}$$

Primjer izračunavanja pol. za neko x :

$$3 + 4x - 5x^2$$

$$x = 5$$

$$v = 0$$

$$v = v + 3 \quad (=3)$$

$$v = v + 4 \cdot (x)^1 \quad (=23)$$

$$v = v - 5 \cdot x^2 = (-108)$$

Imamo 6 operacija (bilo bi
 bio pol. većeg stepena
 bilo bi previše posla - zato
 nam nije zanimljiv)



Implementiramo polinom pomoću vezane liste.
 (elementi su vezani znakom +)

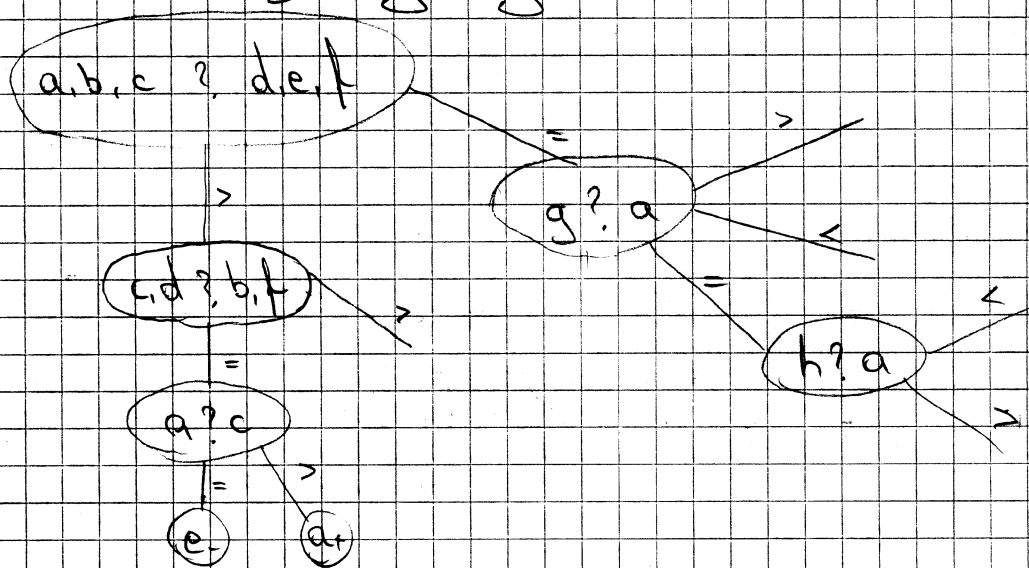
možemo pisati pseudokod za pojedinačnu pr. (++)

a možemo i upotrijebiti

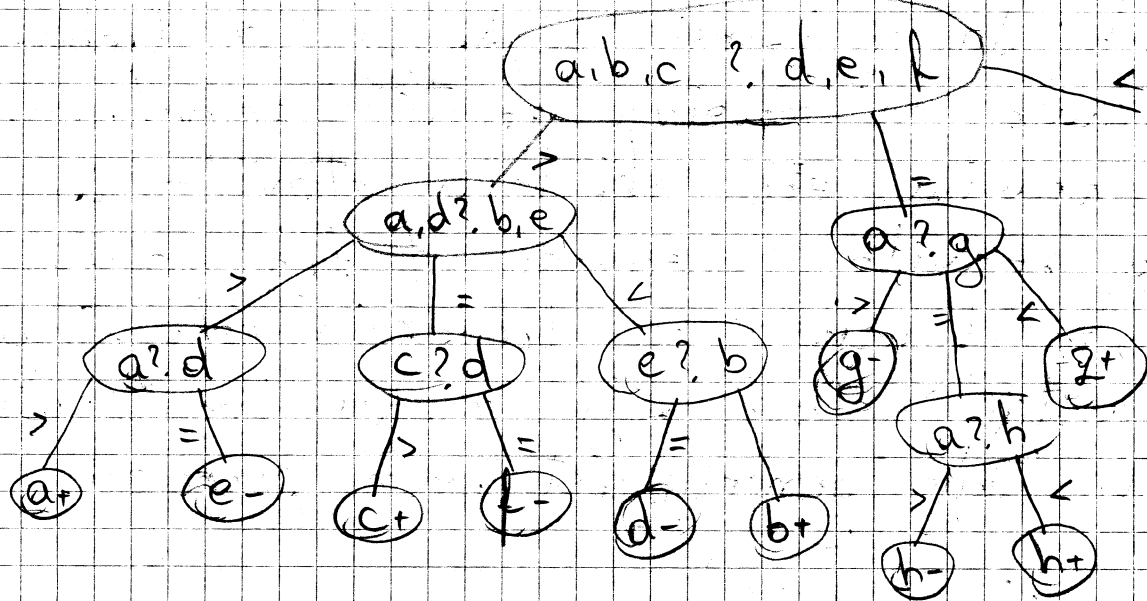
Primjena stabla: Stablo odlučivanja

Jedno moguće modeliranje načina rješavanja problema je modeliranje stablom odlučivanja, koje direktno odslika strukturu programskog rešenja.

1. Neka je dato 8 novčića a, b, c, d, e, f, g, h od kojih je jedan lažan i ima različitu težinu ~~razlika~~ od drugih. Ako je na raspolaganju vaga potrebno je odrediti koji je novčić lažan i da li je lakši ili teži od ostalih uz minimalni broj mjerenja.



$a, b, c > d, e, f$
 $c, d > b, f$ ($a = e$)
 $c ? a$



U ovom primjeru zbog njegove jednostavnosti, stablo odlučivanja se može generisati direktno, a na osnovu njegova dijagram toba program zbog rješavanja. Međutim u drugim slučajevima do stabla odlučivanja se dolazi posredno, preko tabele odlučivanja

Pravilo	1	2	3	4	5
Uслов 1	Y	N	N	N	N
Uслов 2	-	Y	Y	N	N
Uслов 3	-	Y	N	Y	N
Akcija 1	x		x		x
Akcija 2	x	x	x		

Pravilo 1 se može razbiti na 4 pravila:

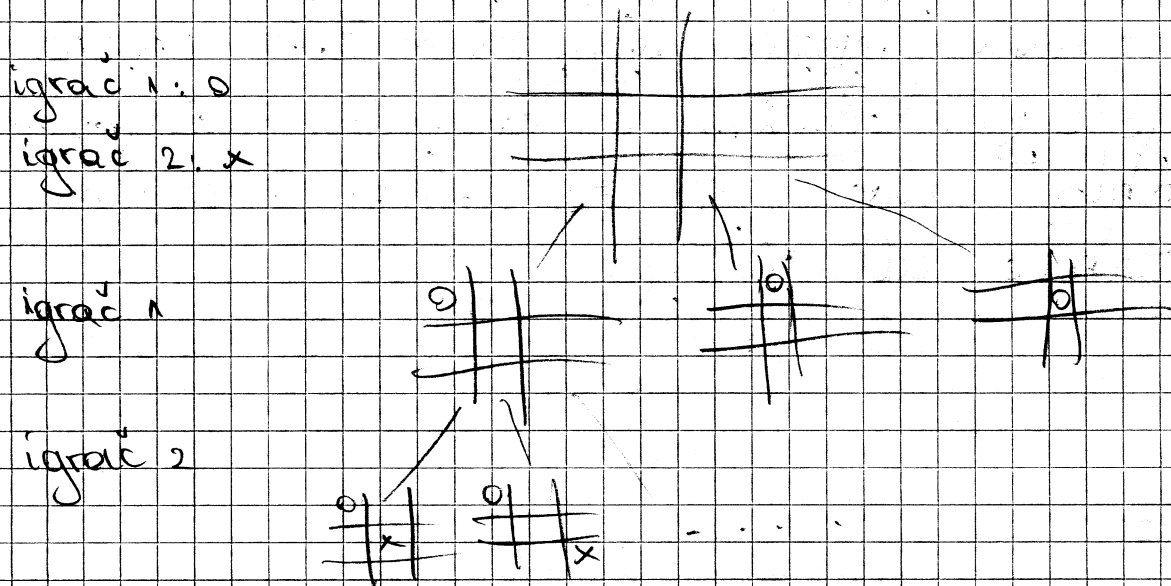
Y	Y	Y	Y
Y	Y	N	N
Y	N	Y	N
X	X	X	X
X	X	X	X

Svaka kolona predstavlja jedno pravilo. Pravilo određuje pod kojim uslovima se akcije izvršavaju tako da očigledno odgovaraju jednoj IF naredbi sa složenim uslovom npr. Pravilo 2 kaže da ako uslov 1 nije ispunjen a uslovi 2 i 3 jesu

Akcija 2 treba da se izvrši a Akcija 1 ne treba.
Dobro pravilo se naziva prostim pravilom.
Ako se u pravilu za neki uslov stavi -, to
znači da pravilo ne zavisi od tog uslova.
U pravilu pored uslova sa logičkim ishodom
postoje i opštiji uslovi koji mogu imati više
ishoda. što daje generalnost ovom prikazu.

Primjena stabla: stabla igara

Stabla se veoma često koriste i u realizaciji
raznih igara na računaru kao što su šah,
kružići i krstići, itd.



DO OVDJE TREBA ZA ISPIT

Implementacije polinoma

$$P(m) = \sum_{i=0}^m a_i x^i$$

a_i su koeficijenti, $a_i \in F$, F -polje (najčešće je $F = \mathbb{Z}$)

- operacije:

- sabiranje: $P_1(m_1) = \sum_{i=0}^{m_1} a_{1i} x^i$, $P_2(m_2) = \sum_{i=0}^{m_2} a_{2i} x^i$

$$P_3(m_3) = \sum_{i=0}^{m_3} a_{3i} x^i, \quad P_3 = P_1 + P_2$$

$$m_3 \leq \max(m_1, m_2)$$

$$a_{3i} = a_{1i} + a_{2i}$$

- odazimanje svodimo na sabiranje

- polinom se često predstavlja u vektorskom zapisu

- množenje skalarom

$b \in F$

$$b \cdot P = \sum_{i=0}^m b a_i x^i$$

- množenje polinoma

$$P_1 = \sum_{i=0}^{m_1} a_i x^i, \quad P_2 = \sum_{i=0}^{m_2} b_i x^i$$

$$P_1 \cdot P_2 = \sum_{i=0}^{m_1+m_2} c_i x^i$$

$$c_k = \sum_{i+j=k} a_i b_j$$

- dijeljenje polinoma

$$P_1 : P_2 = Q + \frac{R}{P_2} \Leftrightarrow P_1 = Q \cdot P_2 + R$$

- izvođenje polinoma

$$(1) \dots a_0 + a_1 x + a_2 x^2 + \dots + a_m x^m =$$

$$= a_0 + x(a_1 + a_2 x + \dots + a_m x^{m-1}) =$$

$$= a_0 + x(a_1 + x(a_2 + x(a_3 + \dots + x a_m))) \dots (2)$$

Hornerova
šema

u izrazu (1) imamo ukupno $\frac{m}{2}$ sabiranja i $\frac{m \cdot (m+1)}{2}$ množenja

u izrazu (2): $\frac{m}{2}$ sabiranja i $\frac{m}{2}$ množenja

m_1 $P_1 \rightarrow \boxed{1|0} \rightarrow \boxed{2|1} \rightarrow \boxed{3|2|1}$

m_2 $P_2 \rightarrow \boxed{2|0} \rightarrow \boxed{3|1} \rightarrow \boxed{4|2} \rightarrow \boxed{5|3|2|1}$

$O(m_1 + m_2)$ $(P_1 + P_2) \rightarrow \boxed{3|0} \rightarrow \boxed{4|1} \rightarrow \boxed{7|2|1}$

m_1 - monomi polinoma P_1

m_2 - monomi polinoma P_2

U predstavljanju polinoma vezanom listom (pointerški) trebalo bi postići: - da svaki element te liste predstavlja monom; - svi elementi su različitog stepena; - elementi liste su sortirani po stepenima

1. Napraviti algoritam izračunavanja polinoma (preko varijante (2)). Polinom je dat, unosimo: tj. unosimo polinom.

$$P_0 = a_0 + P_1 \quad P = P(x, a, m)$$

$$P_1 = a_1 + P_2 \quad \text{Function } P(x, i, m)$$

IF $i = m$ THEN

ELSE $P = a_i$

$P = a_i + x \cdot P(x, i+1, m)$

END IF

END Function

x - nepoznata
 i - najmanji stepen
 m - stepen polinoma

$$\begin{aligned}
 (2, 0, 3) &= a_0 + x \cdot (2, 1, 3) = a_0 + x \cdot (a_1 + x \cdot (2, 2, 3)) = \dots \\
 &= a_0 + x (a_1 + x (a_2 + x (2, 3, 3))) = \dots \\
 &= a_0 + x (a_1 + x (a_2 + x (a_3)))
 \end{aligned}$$

MNOŽENJE POLINOMA

Napisati algoritam složenosti $O(m^2)$ za množenje 2 polinoma stepena ne većeg od m .

```

INPUT m
FOR i=0 TO m
    INPUT a(i)
NEXT i
INPUT m (m <= m)
FOR i=0 TO m
    INPUT b(i)
NEXT i
DIM C(m*m) // C(i)=0
    ∀ i ∈ {0, ..., m*m}
FOR i=0 TO m
    FOR j=0 TO m
        C(i+j) = C(i+j) + a(i) * b(j)
    NEXT j
NEXT i
PRINT C(0);
FOR i=1 TO m*m
    PRINT "+"; C(i); "x^"; i;
NEXT i
    
```

$$\begin{aligned}
 P_m(x) &= a_m x^m + a_{m-1} x^{m-1} + \dots + a_1 x^1 + a_0 x^0 \\
 Q_m(x) &= b_m x^m + b_{m-1} x^{m-1} + \dots + b_1 x^1 + b_0 x^0
 \end{aligned}$$

```

FOR i=0 TO m
    INPUT a(i)
    P(i)(x) = P(i)(x) + a(i) * x^i
NEXT i
FOR j=0 TO m
    INPUT b(j)
    Q(j)(x) = Q(j)(x) + b(j) * x^j
NEXT j
FOR i=0 TO m
    FOR j=0 TO m
        R(i+j)(x) = (a(i) * b(j)) * x^{i+j} + R(i+j)(x)
    NEXT j
NEXT i
    
```

$$\begin{aligned}
 P_0(x) &= a_0 \\
 P_1(x) &= a_0 + a_1 x
 \end{aligned}$$

$$\begin{aligned}
 P_0 \cdot Q_0(x) &= a_0 \cdot b_0 \cdot x^0 \\
 P_m \cdot Q_m(x) &= a_m \cdot b_m \cdot x^{m+m}
 \end{aligned}$$

$$\begin{aligned}
 &(a_m \cdot b_m) x^{m+m} + (a_m \cdot b_{m-1} + a_{m-1} \cdot b_m) x^{m+m-1} + \dots + (a_m \cdot b_0 + a_{m-1} \cdot b_1 + \dots + a_0 \cdot b_m) x^{m+0} + \dots + a_0 \cdot b_0 x^0
 \end{aligned}$$

Problem n kraljica

Na ploči poput šahovske, ali dimenzija $m \times m$ razmjestiti n kraljica tako da se nijedne dvije ne raspadažu (da nisu ni u istom redu ni u istoj koloni ni na istoj dijagonali)

D-dama (kraljica)

$n = 1$

D

$n = 2$

D	

 nema rj.

$n = 3$ I varijanta

D	-	-
-	/	
-		/

 ne može

II varijanta

/	/	/
D	-	-
-	-	D

 nema rj.

$n = 4$

D			
		D	
/		/	
	/		/

 \Rightarrow

D			
	D		
			D
/	/	/	/

zbir indeksa u matrici je ključni

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

$m[m][n] \leftarrow \{\{\emptyset, \emptyset, \dots\}, \{\emptyset, \emptyset, \dots\}, \dots, \{\emptyset, \emptyset, \dots\}\}$

for ($i=1; i \leq m; i++$) { // petlja za redove

for ($j=1; j \leq m; j++$) { // petlja za kolone

if (PoljeSlobodno(m, i, j)) { $m[i][j] = 1$; break; }

```
else if (j == m) { i -- ;
```

```
    j := pozicijaDama (m, i, j);
```

```
    m[i][j] =  $\phi$ ;
```

```
}
```

```
if (i ==  $\phi$ ) { cout << "Nema(višć) rješenja"; return b; }
```

```
}
```

1. Kako provjeriti da li se na nekoj od dijagonala bojim pripada a_{pq} matrice A nalazi 1, gledajući elemente od 1 do $(p-1)$ vrste.

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & & & \\ & & a_{pq} & \\ & & & \end{bmatrix}$$

ULAZ: A, p, q, m

IZLAZ: 0 (metačno) v 1 (tačno)

FUNCTION PostojiNaDijagonali(A, p, q, m)

FOR $i = 1$ TO $p-1$

if ($q-i \geq \phi$ and $A(p-i, q-i) == 1$) RETURN 1

if ($q+i \leq m+1$ and $A(p-i, q+i) == 1$) RETURN 1

END FOR

RETURN 0

ENDFUNCTION

5	3	6
5		1
2	7	4

// OGRANIČAVANJE

FUNCTION NemaRjesenje(....

if ImamoObidjenihPolja() and
NemaMogucihPoteza()

RETURN TRUE

END FUNCTION

numerisanje polja: $1 \dots m^2$

ako postoji sekvenca (rjesenje) a_1, \dots, a_{m^2}

onda je rjesenje $a_{m^2} \dots a_1$

SA ISPITA

2. Neka se x prirodna br. nalaze u binarnom zapisu u 2 niza dužine n tako da svaki element niza predstavlja 1 cifru nula ili 1. Potrebno je zbir ta dva broja smjestiti u također binarnom zapisu u nizu $n+1$. Dati formalno zapis problema (opis, ulaz i izlaz) i algoritam koji rješava postavljem problem.

Binarnih mjesta

1
2

max broj

$$2^0 = 1$$

$$2^1 + 2^0 = 3$$

1 1 1 1 1
1 1 1 1 1
1 1 1 1 0

uvijek može
biti \geq samo
 $\geq a \quad 1$

$$2(2^n - 1) = 2^{n+1} - 2$$

0 0
0 1 1
1 0 2
1 1 3
1 0 0 2

$a = 0 \quad 1$
 $b = 1 \quad 0$
 $c = 1 \quad 1$

INPUT M

FOR

$i = 1$ TO M

INPUT $a[i], b[i]$

IF $i \neq 0, 1$ GOTO [0]

END IF

IF $(i=0, j=0)$ THEN $i+j=0$

IF $(i=0, j=1)$ THEN $i+j=1$

IF $(i=1, j=0)$ THEN $i+j=1$

IF $(i=1, j=1)$ THEN $i+j=0$

Opis: sabiranje 2 binarna broja

Ulaz: dva niza a i b

Izlaz: niz binarnih cifara kojima je predstavljen
zbir 2 bin. br. (niza a i b)

ULAZ: A, B

ULAZ: C

START

ostatak = 0

FOR i = 1 TO m

$C = A(i) + B(i) + \text{ostatak}$

if $(C = 2)$

$C(i) = 0$

ostatak = 1

else if $(C = 3)$

$C(i) = 1$

ostatak = 1

else

$C(i) = C$

ostatak = 0

END IF

END FOR

$C(m+1) = \text{ostatak}$

END

Lista čvorova

Lista čvorova := $\{(a, 29), (b, 23), (c, 20), (d, 10), (e, 9), (f, 7), (g, 2)\}$

// $i := 1$

$x := (a, 29)$

Lista čvorova := $\{(a, 29) \dots (f, 7)\}$

$y := (f, 7)$

Lista čvorova = $\{(a, 29) \dots (e, 9)\}$

$z := (z, 8)$ // z_1 je novi čvor sa frekvencijom = zbiru frek. čvorova

Lista čvorova := $\{(a, 29), (b, 23), (c, 20), (d, 10), (e, 9), (z, 8)\}$

S_1 kreiraj $((z, 8), (g, 2), (f, 7))$ // crtamo stablo

// $i := 2$

$x := (z, 8)$

$LČ := \{(a, 29), (b, 23), (c, 20), (d, 10), (e, 9)\}$

$y := (e, 9)$

$LČ := \{(a, 29), (b, 23), (c, 20), (d, 10)\}$

$z := (z_2, 18)$ // z_2 je novi čvor = zbiru čvorova z_1 i e

$LČ := \{(a, 29), (b, 23), (c, 20), (d, 10), (z_2, 18)\}$

S_2 kreiraj $((z_2, 18), (z_1, 8), (e, 9))$ // crtamo stablo

// $i := 3$

$x := (d, 10)$

$LČ := \{(a, 29), (b, 23), (c, 20), (z_2, 18)\}$

$y := (z_2, 18)$

$LČ := \{(a, 29), (b, 23), (c, 20)\}$

$z := (z_3, 28)$ // z_3 novi čvor ... = $d + z_2$

$LČ := \{(a, 29), (b, 23), (c, 20), (z_3, 28)\}$

S_3 kreiraj $\{(z_3, 28), (d, 10), (z_2, 18)\}$ // crtamo stablo

// $i := 4$

$x := (c, 20)$

$LČ := \{(a, 29), (b, 23), (z_3, 28)\}$

$y := (b, 23)$

$LČ := \{(a, 29), (z_3, 28)\}$

$z := (z_4, 43)$ // z_4 novi čvor ... = $c + b$

5. DIM a(100), b(100)

FOR i=1 TO 90

a(i)=i

NEXT i

FOR i=1 TO 90

br = int (rnd(1) * (91-1)) + 1

b(i) = a(br)

h = 0

FOR j=1 TO 91-i

IF a(j) <> b(i) THEN

h = h + 1

a(h) = a(j)

END IF

NEXT j

NEXT i

FOR i=1 TO 90

PRINT b(i)

NEXT i

→ $\checkmark : \{ (a, 28), (z_3, 28), (z_n, 43) \}$

5n. breiraj $\{ (z_n, 43), (c, 20), (b, 23) \}$ // crtamo stablo

// i := 5

x := (z_3, 28)

$\checkmark : \{ (a, 29), (z_n, 43) \}$

y := (a, 29)

$\checkmark : \{ (z_n, 43) \}$

z := (z_5, 57) // z_5 je novi čvor ... = z_3 + a

$\checkmark : \{ (z_n, 43), (z_5, 57) \}$

55. breiraj $\{ (z_5, 57), (z_3, 28), (a, 20) \}$ // crtamo stablo

// i := 6

x := (z_n, 43)

$\checkmark : \{ (z_5, 57) \}$

$$\gamma := (z_5, 57)$$

$$LC := \{\}$$

$$z := (z_6, 100) \quad // z_6 \text{ novi čvor} \dots = z_n + z_5$$

$$LC := \{ (z_6, 100) \}$$

Se kreiraj $((z_6, 100)(z_n, 13)(z_5, 57))$ // crtamo stablo

// nema više iteracija. Sada crtamo $S_1 U S_2 U S_3 U S_4 U S_5 U S_6$

Na grane koje vode od roditelja do djece s lijeve strane stavljamo nule, a na one koje vode do djeteta s desne str. stavljamo jedinice. Citanje koda vrši se tako što idući putem od korijena stabla do lista zapisujemo brojeve koji se nalaze na granama koje prolazimo.

I varijanta (Sve se radi na jednom nizu)

poč. stanje	23	50	50	16	27	16	69
1. iter	50	23	50	16	27	16	69
2.	50	50	23	16	27	16	69
3.	50	50	23	16	27	16	69
4.	50	50	27	23	16	16	69
5.	50	50	27	23	16	16	69
6.	69	50	50	27	23	16	16

II varijanta (dviije liste L_1 -početna, L_2 -krajnja)

$$L_1 \{23, 50, 50, 16, 27, 16, 69\}, L_2 \{\}$$

$$1 \text{ iteracija } L_1 \{23, 50, 50, 16, 27, 16\} \quad L_2 \{69\}$$

$$2 \quad L_1 \{23, 50, 50, 16, 27\} \quad L_2 \{69, 16\}$$

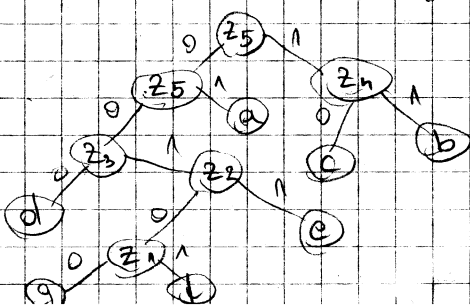
$$3 \quad L_1 \{23, 50, 50, 16\} \quad L_2 \{69, 27, 16\}$$

$$4 \quad L_1 \{23, 50, 50\} \quad L_2 \{69, 27, 16, 16\}$$

$$5 \quad L_1 \{23, 50\} \quad L_2 \{69, 50, 27, 16, 16\}$$

$$6 \quad L_1 \{23\} \quad L_2 \{69, 50, 50, 27, 16, 16\}$$

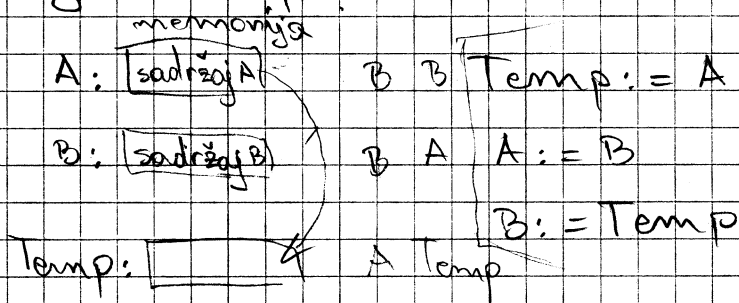
$$7 \quad L_1 \{\} \quad L_2 \{69, 50, 50, 27, 23, 16, 16\}$$



- Matematičko modeliranje - rješavanje problema
(smanjimo bočku od 1 km 1000 puta tj. na 1 m
pa je težina $1000 \cdot 1000 \cdot 1000 = 10^9$)
- Fizičko modeliranje (smanjimo most od 1 km
na 1 m, da li će se sve varijable smanjiti 1000
puta? to ne znamo i zato nam treba mat-
ematičko mod.)
- Algoritam je niz elementarnih koraka koji vode
ka rješenju nekog problema. (ti koraci se ne
mogu dati uprosti)
- Programiranje se uvijek radi na najnižem nivou
apstrakcije.
- Kompleksnost algoritma zavisi od metoda rješa-
vanja određenog problema.
- "O" notacija je na neki način aproksimacija
(bitna nam je samo zavisnost vremena pri
rješenju problema i odabir metode za što brže
rješavanje)
- Apstraktan tip podataka (int, double, definišemo
određene tipove podataka - skalare, vektore, kompl-
eksne br. za koje definišemo matem. operacije
omogućuje kombinaciju različitih tipova pod.
(npr. množenje vektora i kompl. broja)
- Imamo 3 faze rješavanja problema:
 - 1) neformalno opisivanje problema (matematički)
 - 2) apstraktni tipovi pod. (globalni alg. zapisanu
pseudokodu)

3) strukture podataka (stvarni program)

- Algoritamske strukture: 1) sekvencijalna linijska struktura (linija za linijom bez skokova se rješava problem)

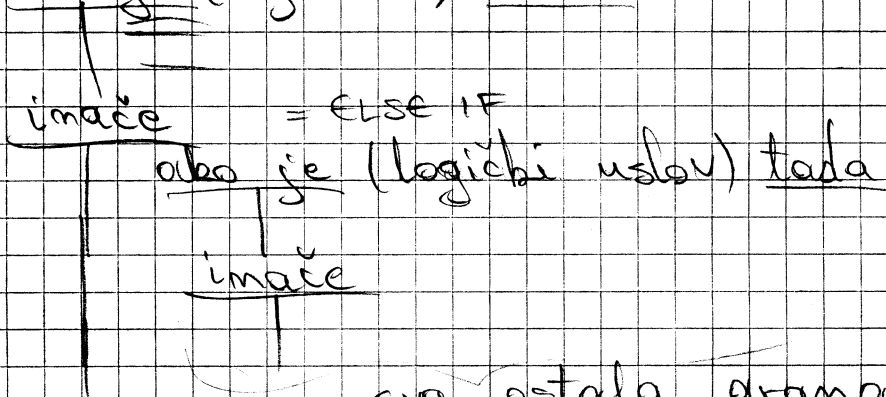


2) struktura grananja (izvedene strukture)

a) jednostrano grananje (jednostrano)
ako je (log. uslov) tada

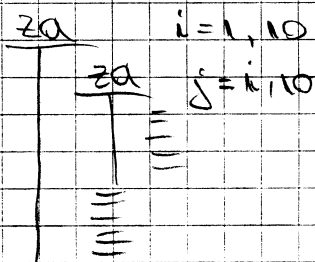
b) dvostrano

ako je (log. uslov) tada



sva ostala grananja su izvedena iz jednostranog i dvostranog

3) ciklična struktura (FOR, DO-WHILE)



dok je (log. uslov)

ponavljaj

tijelo do-while strukture se prođe bar jednom dok je (log. uslov)

Osnove složenosti algoritama:

- Algoritam je postupak za rješavanje određenih problema neke klase. Sastoji se od niza instrukcija. Algoritam mora biti reprodukcibilan (da ima svrhu; da se koristi)

- Zahtjevi za algoritam (da bi algoritam bio prihvatljiv, prikladan za izvođenje):

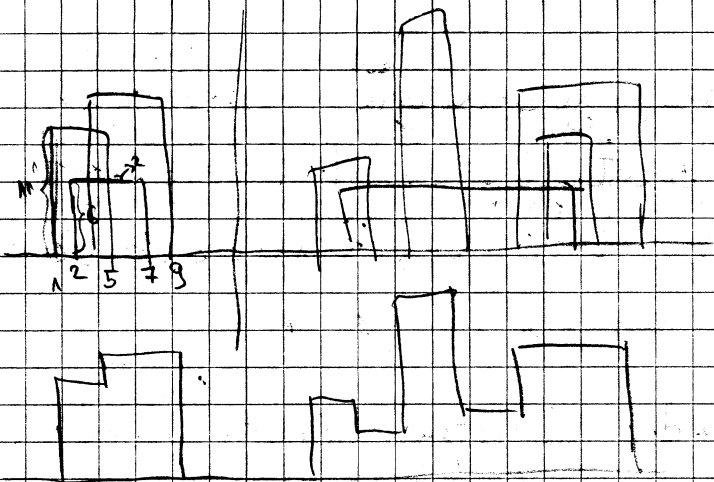
- postoji početna instrukcija
- po završetku bilo koje inst. mora se znati sljedeća instrukcija
- postupak se završava od konačno mnogo koraka
- sastoji se od konačno mnogo instrukcija

- Svojstvo algoritma:

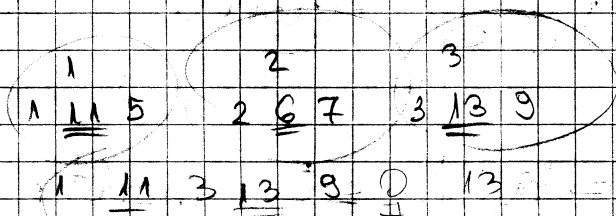
- 1) ulaz (svaki alg. ima 0 ili više ulaza)
- 2) izlaz (svaki alg. ima bar 1 izlaz koji nazivam rezultatom, rješenjem)
- 3) konačnost (alg. završava u konačno mnogo koraka)
- 4) ~~razumna~~ složenost (nebo želi da dobije sadržaj ključa za dekriptiranje za razumno vrijeme)

Analizu algoritama radimo zato što postoji više algoritama za rješavanje nekog problema. Zavisí od: vremena (broj operacija i inst.)
vremenske i prostorne (veličina memorije - koliko treba mem.) složenosti.

Zadaca: Ispitati Skyline algoritam.



treba da se nađe:



Imamo n objekata zadanih preko ovih trojki

Princip optimalnosti (najbolja vrijednost)

Problem diskretnosti fja (optimum može fje:)

$f(x_1, x_2, \dots, x_n) = f(\vec{x})$ diferencijabilna (nije problem)

$\min f(\vec{x})$

podograničenje $g(\vec{x}) = 0$

$\min f(\vec{x})$

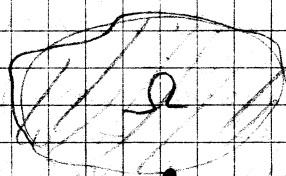
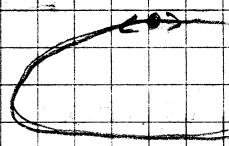
podogr. $g(\vec{x}) \leq 0$

$\min f(\vec{x})$

$x_1 \in \{0, 1, 2, \dots, 12, \dots\}$

$x_2 \in \{ \dots \}$

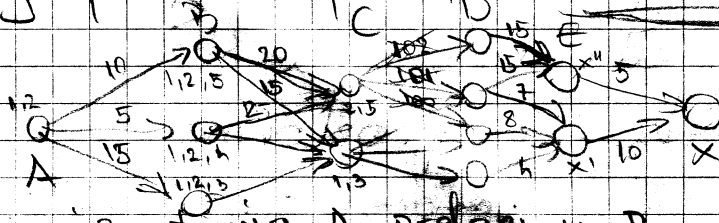
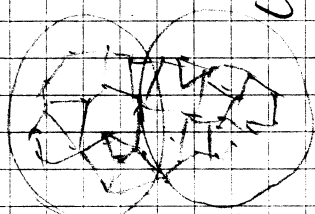
$x_3 \in \{ \dots \}$



zadovoljeni
Slaterovi
uslovi

problem nije diskretan
(ne možemo diferencijirati)
nego kombinatoran

Problem trgovačkog putnika - problem hermenekizma

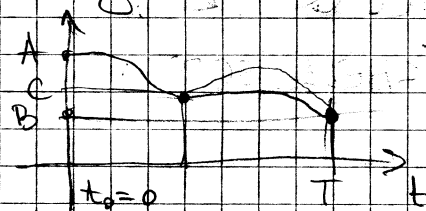


mreža
ima počet
i kraj

izbacujemo najdužu granu a vidimo računamo da li ostaje moguće rješenje (da ne ostavljamo slijepu ulicu)

- Optimalan put je optimalan po svakom svom dijelu.

stanje A treba prevesti u stanje B, za neko vrijeme T



$$P = \frac{W}{t} \quad P = \frac{dW}{dt}$$

brzina promjene energije zavisi od snage

Dinamičko programiranje uvijek možemo primijeniti ako je problem razdvojen na etape (faze).

Algoritmi vanjskog i unutrašnjeg sortiranja:
alg. unutrašnjeg - broj elemenata nije veliki (sortiramo ključeve - identificiraju svaki slog pomalob i mogu se sastojati od više adresa)

Podjela algoritama sortiranja:

a) Sortiranje zasnovano na poredanju i premještanju algoritmi unutrašnjeg sortiranja (kada ima malo pod. koji mogu stati na unutarnju mem.)
b) Algoritmi sa adresnim kalkulacijama (alg. vanjskog sortiranja) - na vanjskoj (eksternoj memoriji). Ovdje spadaju ^{zastjeva} prema eksternoj mem.

1) Proxmap sort

2) Radix sort

} imaju linearnu složenost (kada imamo vrlo veliki br. podataka za sortiranje)

- Unutrašnja sortiranja mogu biti:

a) Sortiranje transpozicija (BUBBLE SORT)

b) Algoritmi umetanja (~~INSERTION~~^{INSERTION} SORT i TREE SORT)

c) Sortiranje redovima čebanja (SELECTION SORT i
HEAP SORT)

d) Podijeli i vladaj (MERGE SORT i QUICK SORT)

e) Alg. smanjivanja prirastaja (SHELL SORT)

7, 17, 13, 14.5

Sortiranje selekcijom

Opis: Prođe se poljem i pronađe se najmanji element. Zatim se najmanji element zamijeni s početnim. Dalje se promatra ostatak polja (bez početnog elementa) i ponavlja isti postupak.

Za dati niz brojeva a_1, a_2, \dots, a_m odrediti niz tih istih brojeva tako da je $a'_1 \leq a'_2 \leq \dots \leq a'_m$.

Algoritam

```
INPUT m
FOR i=1 TO m
    INPUT A(i)
    B(i) = A(i)
NEXT i

FOR i=1 TO m-1
    FOR j=i+1 TO m
        IF B(i) > B(j) THEN
            m = B(i)
            B(i) = B(j)
            B(j) = m
        END IF
    NEXT j
NEXT i

FOR i=1 TO m
    PRINT B(i)
NEXT i

END
```

7, 17, 13, 14, 5
i=1
17, 13, 14, 5
j=2
j=3
j=4
j=5
i=2

SORTIRANJE UMETANJEM (INSERTION SORT)

Opis: Za vrijeme rada algoritma, početni komad polja je već sortirano, a ostatak polja nije sortirano. U jednom prolasku algoritam uzima prvi element iz nesortiranog dijela, te ga umetne na „pravo mjesto“ (u smislu sortiranog redoslijeda) u sortirani dio, pri čemu dolazi do pomicanja nekih elemenata za jedno mjesto dalje. Dakle, jednim prolaskom dužina početnog sortiranog dijela se povećava za 1, a dužina nesortiranog dijela se smanji za 1.

Algoritam: od manjeg ka većem

```
INPUT m
FOR i = 1 TO m
    INPUT a(i)
NEXT i

FOR i = 2 TO m
    k = a(i)
    j = i - 1
    WHILE (j > 0) AND (a(j) > k) DO
        a(j+1) = a(j)
        j = j - 1
    END WHILE
    a(j+1) = k
END FOR
```

od većeg ka manjem

```
a(j) < k
a(j) = a(j+1)
j = j + 1
```

SORTIRANJE ZAMJENOM SUSJEDNIH ELEMENATA (BUBBLE SORT / METODA DIREKTNE ZAMJENE)

Opis: Prolazimo poljem od početka prema kraju i uspoređujemo susjedne elemente. Ako je neki element veći od sljedećeg zamijenimo im vrijednosti. Kad na taj način dođemo do kraja polja najveća vrijednost doći će na posljednje mjesto. Nakon toga ponovljamo postupak na skraćenom polju (bez zadnjeg elementa). Algoritam se smije zaustaviti čim on u nekom prolazu ustanovi da nema parova elemenata koje bi trebalo zamijeniti.

Algoritam:

INPUT N

POS := N

DO {

 BOUND := POS

 POS := 0

 FOR i = 1 TO BOUND - 1 DO

 IF $a[i] > a[i+1]$ THEN

$a[i] \leftrightarrow a[i+1]$

 POS := i

 END IF

 PRINT a[i]

 END FOR

} WHILE (POS \neq 0)

PARTICIJSKO SLAGANJE - QUICK SORT

Opis: Riječ je o rekursivnom algoritmu tipa "podijeli pa vladaj". Odabere se 1. element u polju tzv. stožer. Svi ostali elementi se razvrstavaju ispred (lijevo) odnosno iza (desno) stožera ovisno o tome da li su \leq ili \geq od stožera. Da bi se polje sortiralo do kraja potrebno je zasebno sortirati lijevo i desno podpolje a to postizemo rekursivnim pozivom istog algoritma.

Algoritam:

```
QUICKSORT (a, low, high) {  
    IF (low < high) THEN  
        j = PARTITION (a, low, high)  
        QUICKSORT (a, low, j-1)  
        QUICKSORT (a, j+1, high)  
    END IF  
PARTITION (a, down, up) {  
    i = down  
    j = up  
    WHILE (i < j) DO  
        WHILE (a[i] ≤ pivot) AND (i < j) DO  
            i = i + 1  
        END WHILE  
        WHILE (a[j] > pivot) DO  
            j = j - 1  
        END WHILE  
        IF (i < j) THEN a[i] ↔ a[j]  
        END IF  
    END WHILE  
    a[down] = a[j]  
    a[j] = pivot  
    RETURN j  
}
```


MERGE SORT (SORTIRANJE SPAJANJEM)

Opis: Reč je o rekursivnom algoritmu „podijeli pa vladaj“. Zadano polje podijeli se na 2 manja polja podjednake dužine. Ta 2 polja zasebno se sortiraju rekursivnim pozivima istog algoritma. Zatim se mala sortirana polja sažimaju u jedno.

Algoritam

```
SortSpajanjem(A, p, q)
    ako je  $(p < q)$  onda je {
         $r \leftarrow \lfloor \text{velicina}((p+q)/2) \rfloor$ 
        SortSpajanjem(A, p, r)
        SortSpajanjem(A, r+1, q)
        SpojiSortirane(A, p, r, q)
    }
```

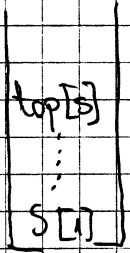
Manja: Korištenje dodatne memorije zbog prepisivanja polja

Prednost: Mogućnost primjene algoritma kad podaci me stanu u glavnu memoriju već su pohranjeni u (sekvencijalnoj) arhitekturi.

STOG (STACK)

Stack je specijalna vrsta liste u kojoj se sva ubacivanja i izbacivanja obavljaju na jednom kraju koji se zove vrh ($top[S]$). Također se naziva LIFO (last in - first out) te pushdown lista.

Prvi element steka $S[i]$ se nalazi na dnu, a posljednji umetnuti element je $top[S]$.



Operacije: PUSH za umetanje elementa.
POP za brisanje

STACK-EMPTY(S)

IF $top[S] = 0$ THEN

RETURN true

ELSE

RETURN false

PUSH(S, x)

$top[S] \leftarrow top[S] + 1$

$S[top[S]] \leftarrow x$

POP(S)

IF STACK-EMPTY(S) THEN

ERROR "under flow"

ELSE

$top[S] \leftarrow top[S] - 1$

RETURN $S[top[S] - 1]$