ALGORITMI I STRUKTURE PODATAKA 2

Pripremila: Lena Joković

ACN K11-23

① FIND-PEAK(a, n)

```
a[0]=-∞ , a[n+1]=-∞
low=1 ; high=n
while (low=high) do
    mid=(low+high)/2
    if (a[mid]>a[mid-1]) and a[mid]>a[mid+1]) then
        return mid
    end_if
    if (a[mid]>a[mid-1] and a[mid]<a[mid+1]) then
        low=mid+1
    else high=mid-1
    end_if
end_while
```

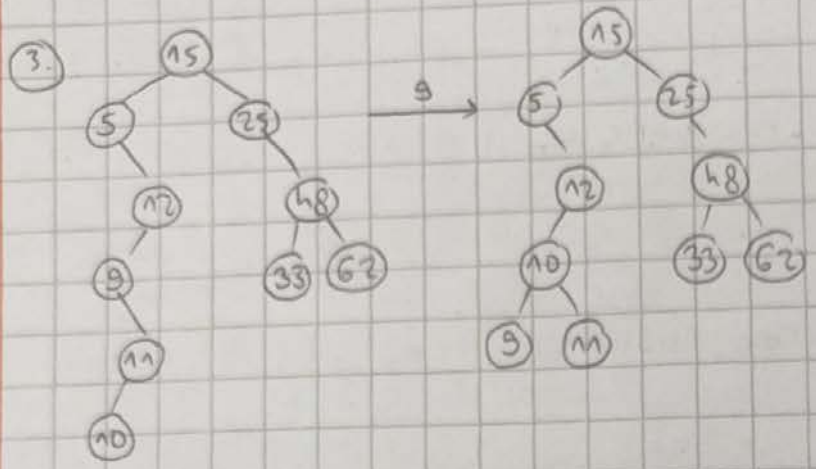
ACA

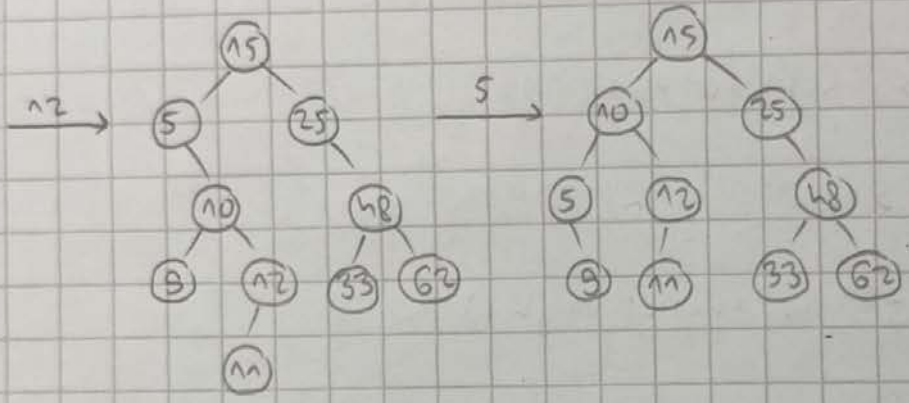
② BST_SUM(root, sum)

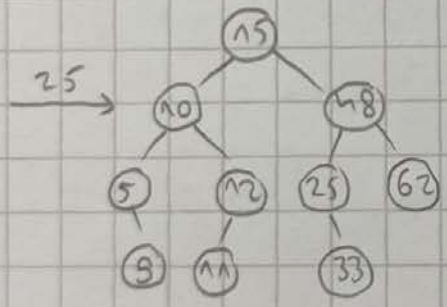
```
STACK_INIT(Slow), STACK_INIT(Shigh)
tmp=root
while (tmp!=nil) do PUSH(Slow, tmp), tmp=left(tmp)
tmp=root
while (tmp!=nil) do PUSH(Shigh, tmp), tmp=right(tmp)
low=POP(Slow), high=POP(Shigh)
while (key(low)<key(high)) do
    if (key(low)+key(high)=sum) then return low, high
    if (key(low)+key(high)<sum) then
        if (right(low)!=nil) then ... low=...
        else low=POP(Slow)
    else
        if (left(high)!=nil) then ... high=...
        else high=POP(Shigh)
    end_if
end_while
return nil, nil
```

АСП Кл-23

3.


15
5
25
12
48
9
10
33
62
11
9


12
5
15
25
10
48
8
12
33
62
11
5
15
10
25
5
12
48
9
11
33
62


25
15
10
48
5
12
25
62
9
11
33

КЛ-23

АСП

(4) 1 5 7 16 19 21 34 39 49 54 55 61 63 76 89 90 93 96 99

- трета на 9 49 63 91 96

9 -> 5 поредена => 0

49 -> 7 поредена => 9

63 -> 6 поредена => 13

91 -> 6 поредена => 0

96 -> 3 поредена => 18

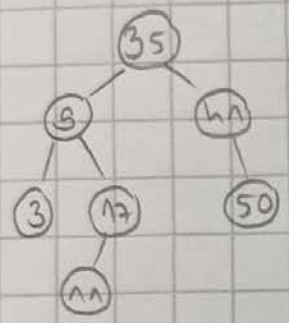
- резултат: [0, 9, 13, 0, 18]

- търсен број поредена: 27

АСП Вл-23

(5) - ако чвор има лево подстабло -> претходник је најдешњи чвор у левом подстаблу

- ако чвор нема лево подстабло -> претходник је најближи предак коме је посматрани чвор у десном подстаблу

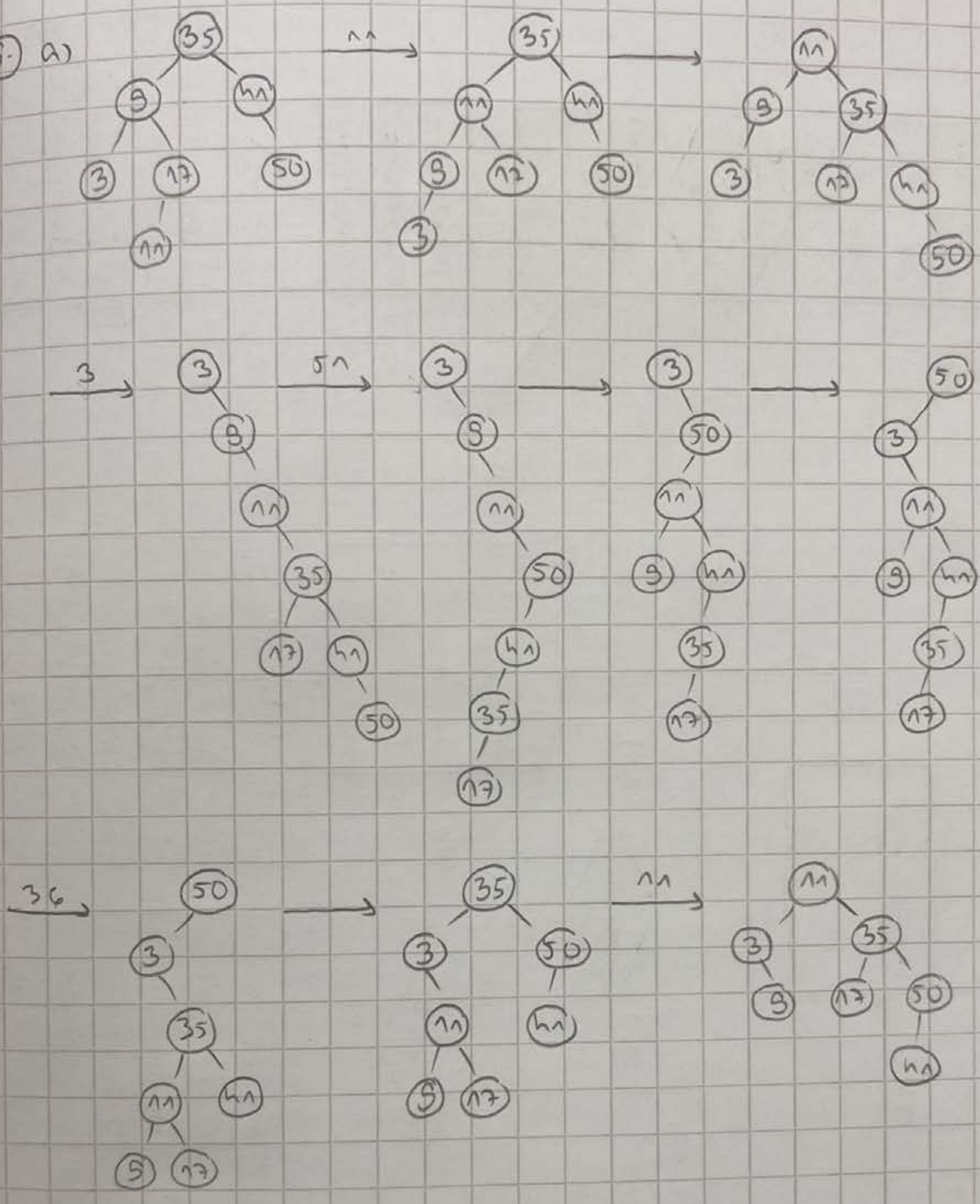


35 претходник: има лево подстабло => 17

17 претходник: нема лево подстабло => 9

К1-23 АСП

6. а)



б) само подешавајуће стабло:

4 + 3 + 6 + 5 + 3 = 21 корак

обично BST стабло:

4 + 3 + 3 + 2 + 4 = 16 корака

АСП К1-22

(1.) -претрата на 4, 8, 16, 16, 8, 4

-транспозиција: => $\frac{24}{6} = 5,67$ поредења

14 12 8 4 11 2 9 10 7 16 22 — 4

14 12 4 8 11 2 9 10 7 16 22 — 4

14 12 8 4 11 2 9 10 7 16 22 — 10

14 12 8 4 11 2 9 10 16 7 22 — 9

14 12 8 4 11 2 9 16 10 7 22 — 3

14 8 12 4 11 2 9 16 10 7 22 — 4

14 8 4 12 11 2 9 16 10 7 22

-пребацување на почеток: => $\frac{24}{6} = 4$ поредења

14 12 8 4 11 2 9 10 7 16 22 — 4

4 14 12 8 11 2 9 10 7 16 22 — 4

8 4 14 12 11 2 9 10 7 16 22 — 10

16 8 4 14 12 11 2 9 10 7 22 — 1

16 8 4 14 12 11 2 9 10 7 22 — 2

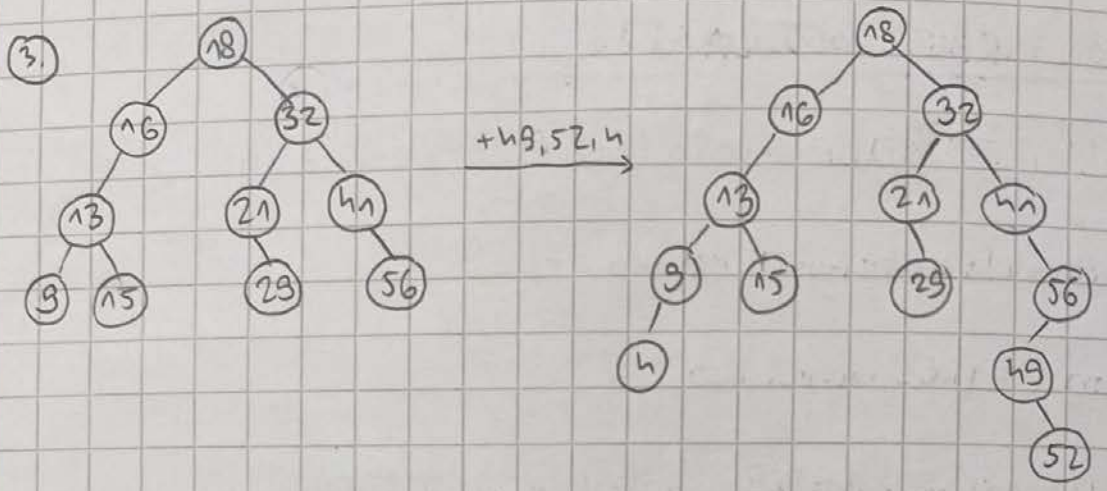
8 16 4 14 12 11 2 9 10 7 22 — 3

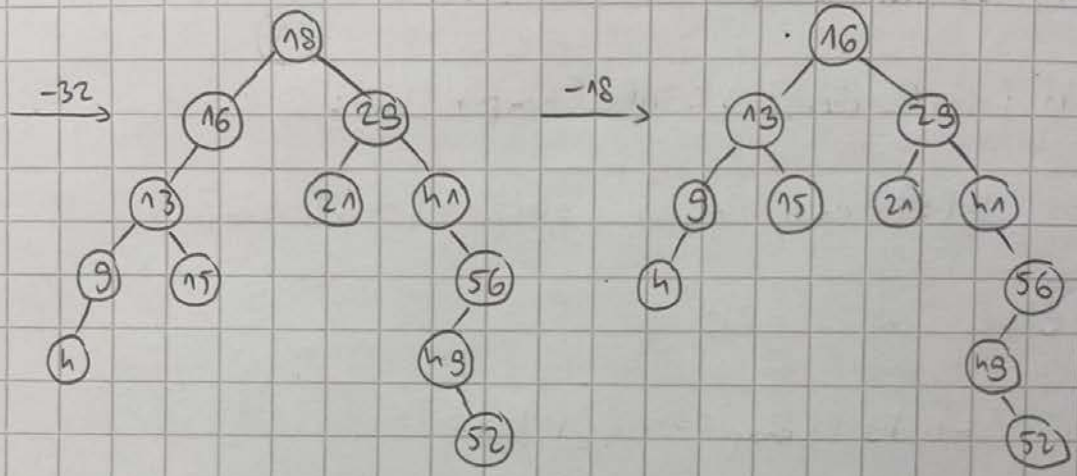
4 8 16 14 12 11 2 9 10 7 22

② RECONSTRUCT_BST(preorder, n)

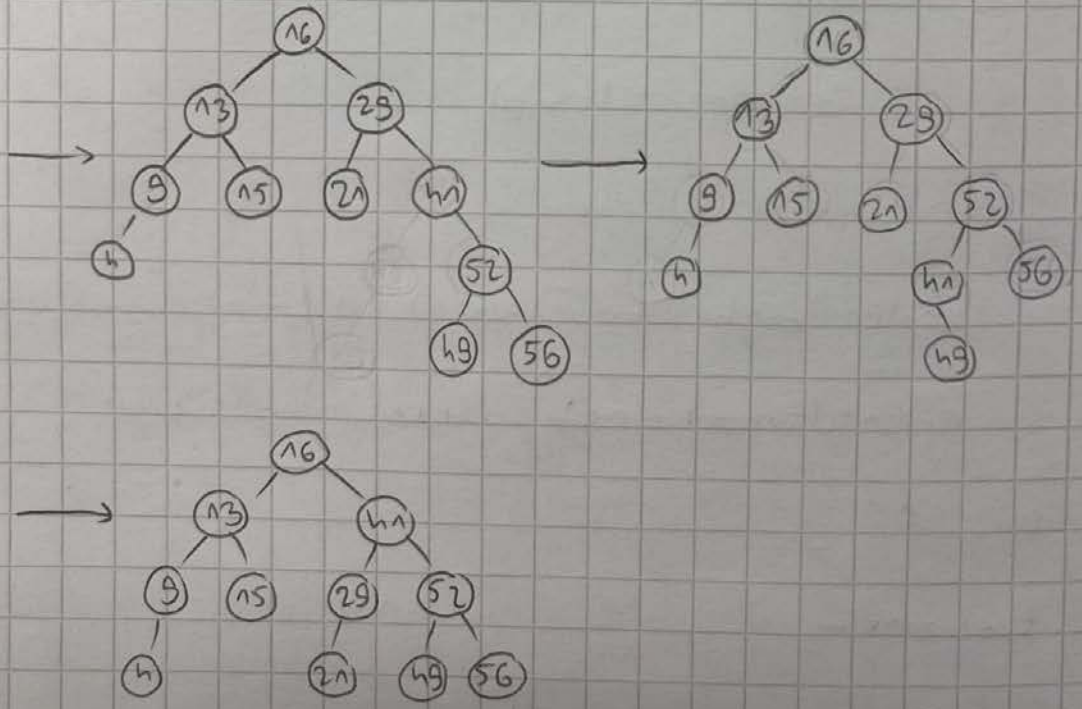
```
root = GETNODE , info(root) = preorder[0]
parent(root) = nil , tmp = root
for (i = 1 to n-1) do
    new = GETNODE , info(new) = preorder[i]
    if (info(new) < info(tmp) then
        left(tmp) = new , parent(new) = tmp
    else
        while (tmp != root) do
            if (tmp = left(parent(tmp)) and
            info(tmp) < info(new) < info(parent(tmp)) then break
            tmp = parent(tmp)
        end_while
        while (right(tmp) != nil) do tmp = right(tmp)
        right(tmp) = new , parent(new) = tmp
    end_if
    tmp = new
end_for
return root
```

АСП ИП-22





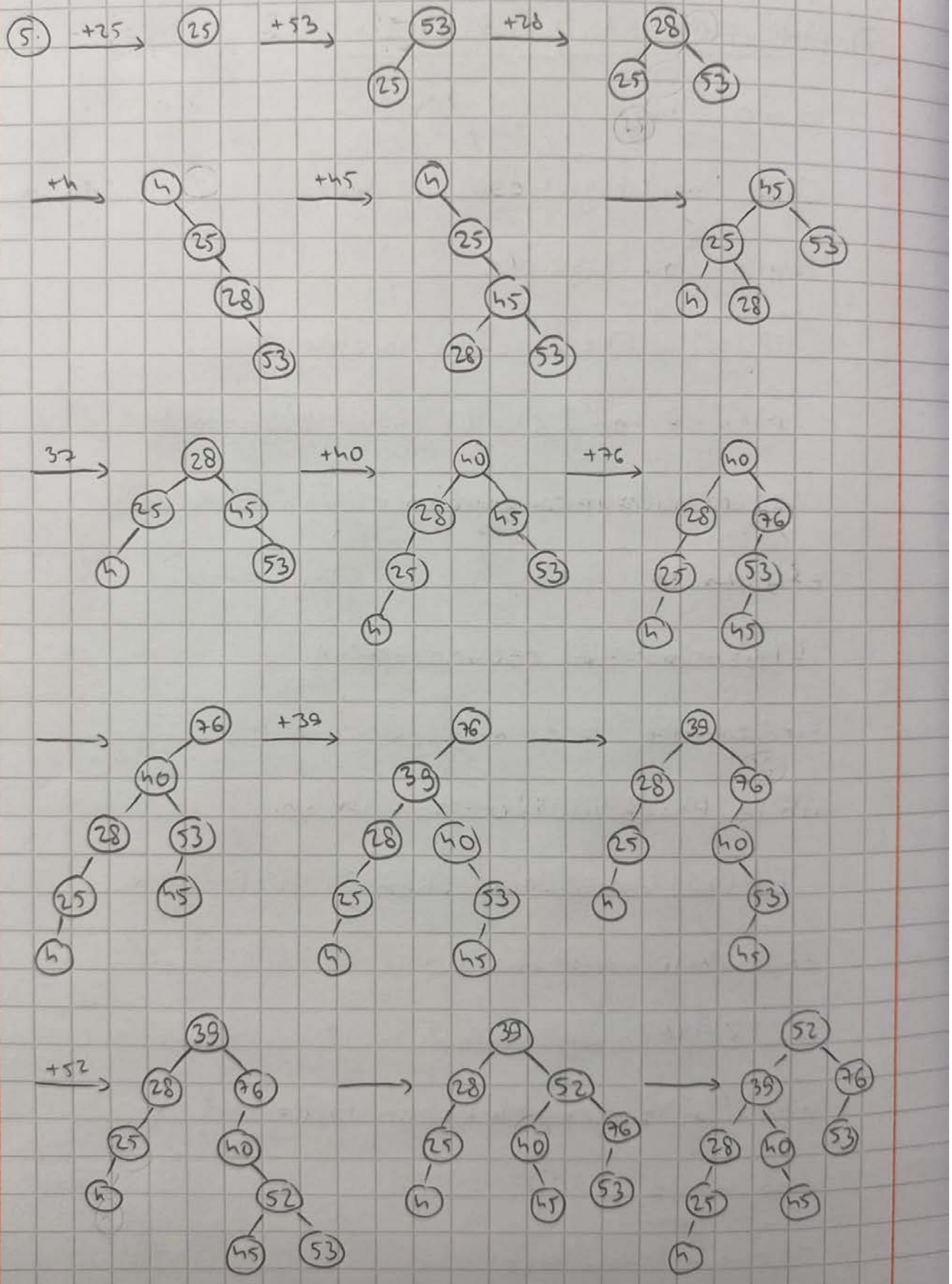
-вирівнювання за повним балансом: 56, 41, 29, 16

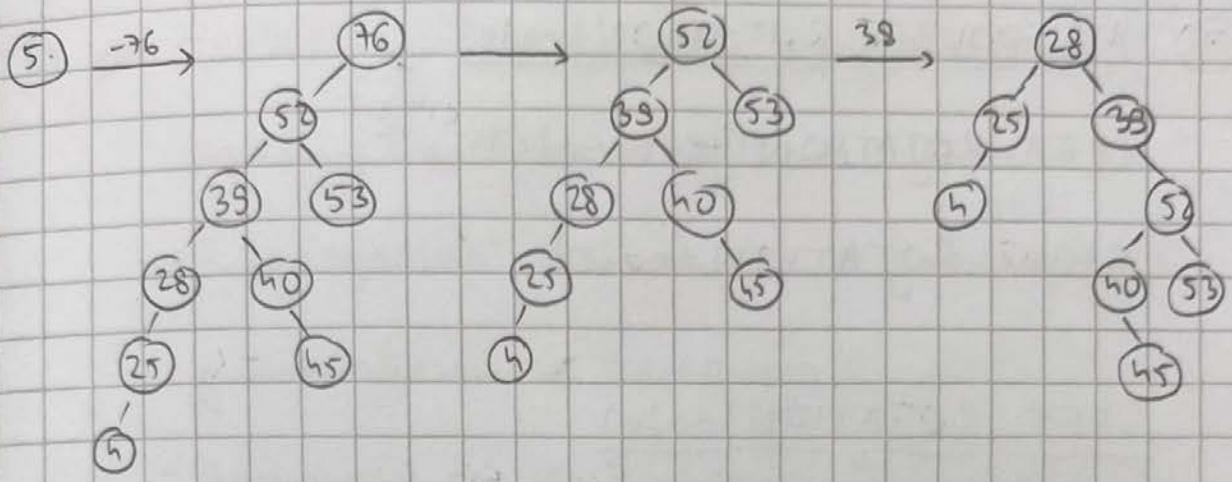


(4) FIND_FIRST_AND_LAST(a, n, k)

```
low=0 , high=n-1
while (low<high) do
    mid=(low+high)/2
    if (a[mid]=k) then break
    else if (a[mid]>k) then high=mid-1
        else low=mid+1
end_while
if(a[mid]!=k) return -1, -1
first=mid, last=mid
while (first>0) do
    if (a[first-1]=k) then first=first-1
end_while
while (last<n-1) do
    if (a[last+1]=k) then last=last+1
end_while
return first, last
```

ACN 번호-22


50
+25
25
+53
53
25
+28
28
25
53
+4
4
25
28
53
+45
4
25
45
28
53
45
25
53
4
28
37
28
25
45
4
53
+40
40
28
45
25
53
4
+76
40
28
76
25
53
4
45
76
40
28
53
25
45
+39
76
40
28
53
25
45
4
39
28
76
25
40
4
53
45
39
28
76
25
40
4
52
45
53
+52
39
28
52
25
40
76
4
45
53
52
39
76
28
40
53
25
45
4

5. 

6. $P(x) = -4x^2 + 12x + 7$

- може се одредити интервал неизвесности, тј. интервал за који важи да су вредности полинома у горњој и доњој граници супротног знака, па онда применити бинарно претраживање

$P(0) = 7$, $P(1) = 15$, $P(2) = 15$, $P(4) = -9$

=> интервал неизвесности $(2, 4)$

$P(3) = 7$ => интервал $(3, 4)$

$P(3.5) = 0$ ✓

- дакле, нула полинома је $x = 3.5$

ACN 시나-22

(7.) AVL_DOUBLE_ROTATION (node)

LEFT_ROTATION (left(node))

RIGHT_ROTATION (node)

LEFT_ROTATION (node)

tmp = right(node)

right(node) = left(tmp)

left(tmp) = node

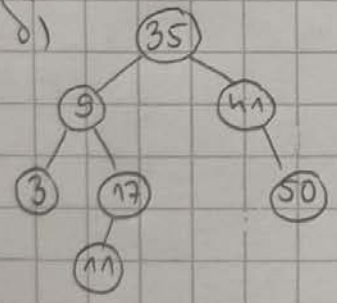
parent(tmp) = parent(node)

parent(node) = tmp

(8.) а) - претпоставка да сви кључеви имају једнаку вероватноћу присуства је да је „случајно" BST настало уметањима у произвољном поретку

- замишљамо да је просечна перформанса претраживања у „случајном" BST-у истог реда $O(\log n)$ као у оптималном BST-у и појача за константан фактор већи.

б)



$$S_n = \frac{P_1 + n}{n} = \frac{11+7}{7} = \frac{18}{7} \approx 2,57$$

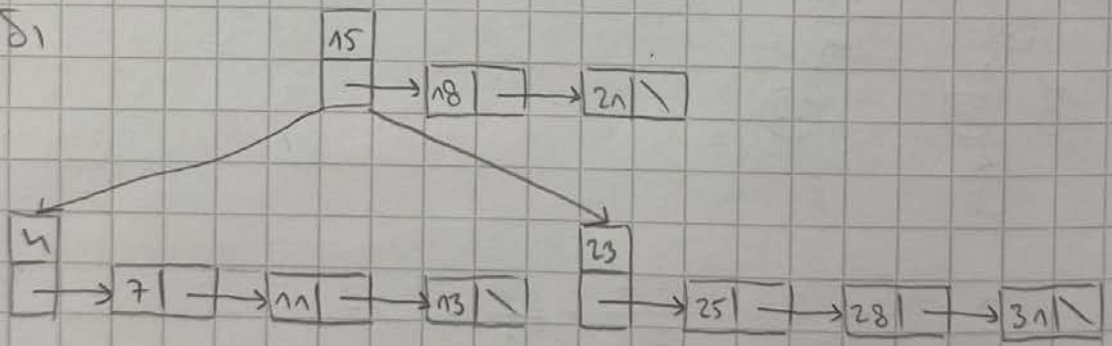
$$U_n = \frac{PE}{n+1} = \frac{PI + 2n}{n+1} = \frac{11+2\cdot 7}{8} = \frac{25}{8} = 3,125$$

Асп К1-21

(1.) а) - користим се BST, чији чвор има поље кључа са адресом прве инструкције неке функције, и друго поље са показивачем на листу других инструкција које припадају тој функцији

- при претрази се налази одговарајућа функција, тј. адреса њене прве инструкције, а затим пролази кроз листу других инструкција

б)

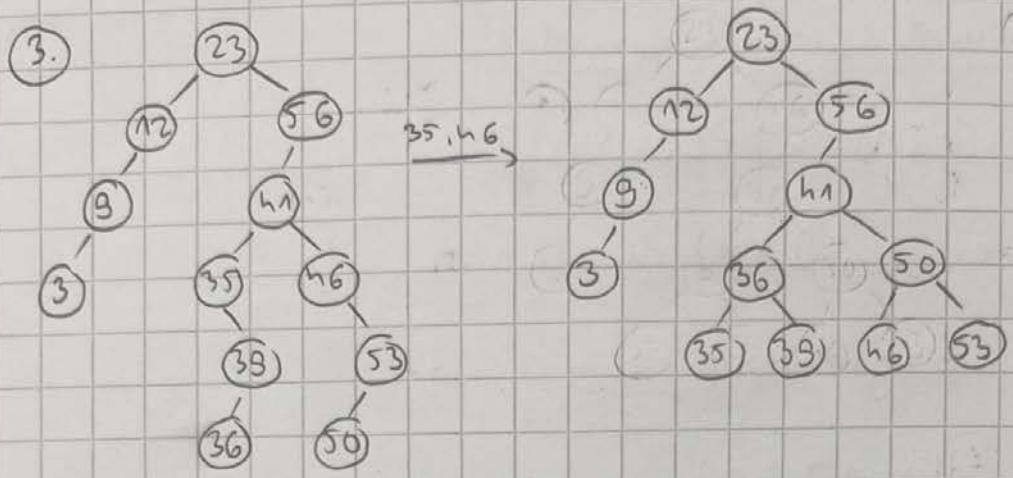


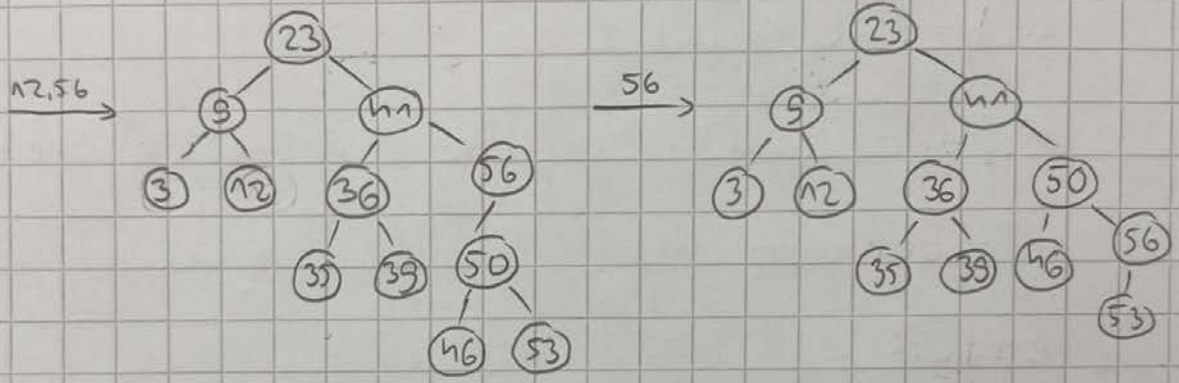
- претражује се кључ 23 у BST
- пролази се кроз листу чвора са кључем 23 до адресе 25

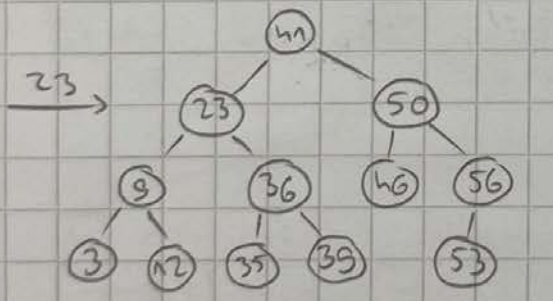
② CALC_SHOP_NUM(price, n, x)

```
if (price[0] > x) return 0
if (price[n-1] < x) return n
low=0, high=n-1
while (low ≤ high) do
    mid = (low+high)/2
    if (price[mid] > x and price[mid-1] ≤ x) then
        return n-mid
    end_if
    if (price[mid] > x) then high=mid-1
    else low = mid+1
    end_if
end_while
```

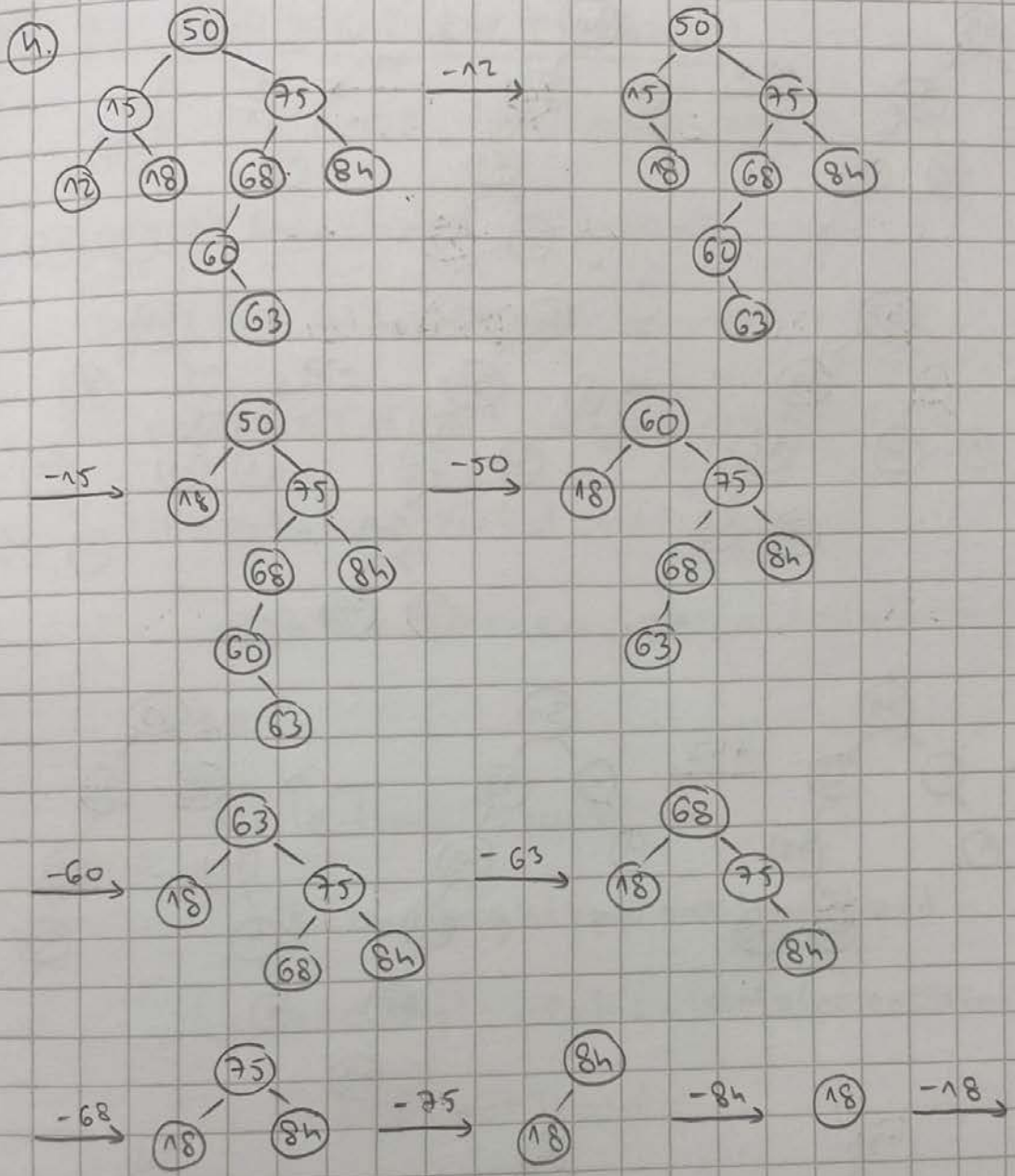
ACN VL1-2N


3.
23
12
56
9
41
3
35
46
38
53
36
50
35,46
23
12
56
9
41
3
36
50
35
39
46
53

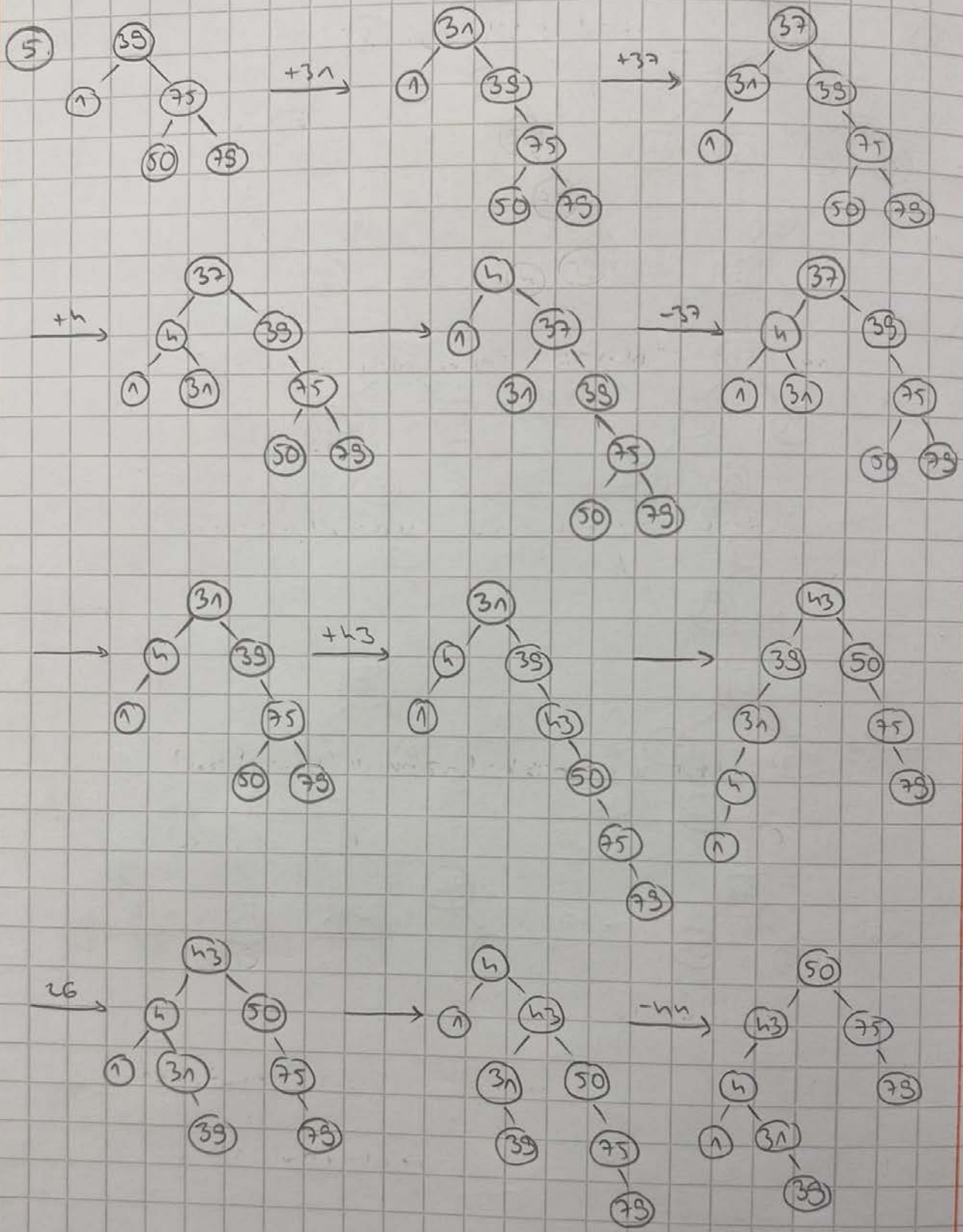

12,56
23
9
41
3
12
36
56
35
39
50
46
53
56
23
9
41
3
12
36
50
35
39
46
56
53


23
41
23
50
9
36
46
56
3
12
35
39
53

4.



АСП №1-21


5
39
1
75
50
79
+31
31
1
39
75
50
79
+37
37
31
39
1
75
50
79
+4
37
4
39
1
31
75
50
79
4
1
37
31
39
75
50
79
-37
37
4
39
1
31
75
50
79
31
4
39
1
75
50
79
+43
31
4
39
1
43
50
75
79
43
39
50
31
75
4
79
1
26
43
4
50
1
31
75
39
79
4
1
43
31
50
39
75
79
-44
50
43
75
4
79
1
31
39

⑥ BST_RECONSTRUCT(post, n)

```
root = GET NODE , info(root) = post[n-1]
parent(root) = nil , tmp = root
for (i = n-2 down to 0) do
    new = GET NODE , info(new) = post[i]
    if (info(new) > info(tmp)) then
        right(tmp) = new , parent(new) = tmp)
    else
        while (tmp != root) do
            if (tmp = right(parent(tmp)) and
            info(tmp) < info(new) < info(parent(tmp)) then break
            tmp = parent(tmp)
        end_while
        while (left(tmp) != nil) do tmp = left(tmp))
        left(tmp) = new , parent(new) = tmp
    end_if
    tmp = new
end_for
return root
```

ACN ΚΛ-2Λ

7. AVL-FIND-CRITICAL-NODE(node)

```
change = true
critical = nil
tmp = node
while (change and parent(tmp) != nil) do
    p = parent(tmp)
    if (tmp = left(p)) then balance(p) = balance(p) + 1
    else balance(p) = balance(p) - 1
    end-if
    if (critical = nil and abs(balance(p)) > 1) then
        critical = p
    end-if
    if (balance(p) = 0) then change = false
    else tmp = parent(tmp)
    end-if
end-while
return critical
```

(8.) а) - умножаване числата по отвадящите вероятностите p_i

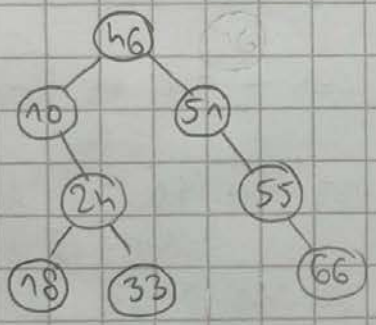
- цена сливане: $C = \sum_{i=1}^{n} p_i(k_i+1)$

б)

k_i:	10	18	24	33	46	51	55	66
p_i:	0.15	0.02	0.12	0.03	0.3	0.25	0.08	0.05

- редом се умножат числата:

46, 51, 10, 24, 55, 66, 33, 18



$\Rightarrow C = 0.3 \cdot 1 + (0.25 + 0.15) \cdot 2 + (0.12 + 0.08) \cdot 3 +$

$+ (0.05 + 0.03 + 0.02) \cdot 4$

$C = 0.3 + 0.8 + 0.6 + 0.4 = 2.1$

АСП К1-20

(1.) - Претрата на 25, 40, 25, 40

- Транспозиция: => $\frac{28}{4} = 7$ поредена

12 7 3 27 25 78 98 21 22 40 51 — 5

12 7 3 25 27 78 98 21 22 40 51 — 10

12 7 3 25 27 78 98 21 40 22 51 — 4

12 7 25 3 27 78 98 21 40 22 51 — 9

12 7 25 3 27 78 98 40 21 22 51

- Пребарување на почеток: => $\frac{19}{4} = 4,75$ поредена

12 7 3 27 25 78 98 21 22 40 51 — 5

25 12 7 3 27 78 98 21 22 40 51 — 10

40 25 12 7 3 27 78 98 21 22 51 — 2

25 40 12 7 3 27 78 98 21 22 51 — 2

40 25 12 7 3 27 78 98 21 22 51

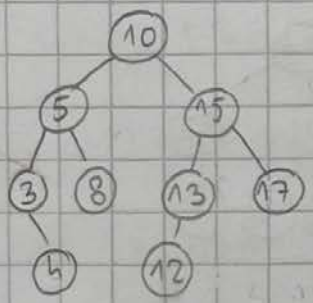
predhodnik

K1-20

(2.) - prethodnik po inorder poretku za zadati čvor

1.) ako ima levo podstablo -> najdesniji čvor u levom podstablu

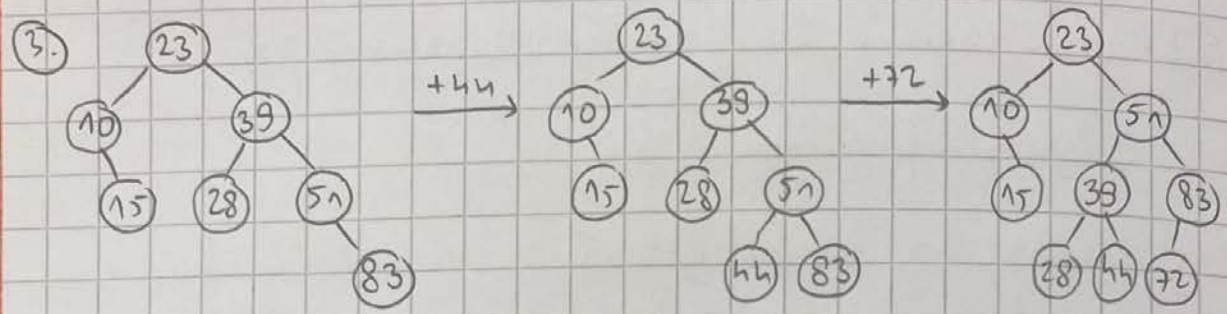
2.) ako nema levo podstablo -> najbliži predak kome je posmatrani čvor u desnom podstablu

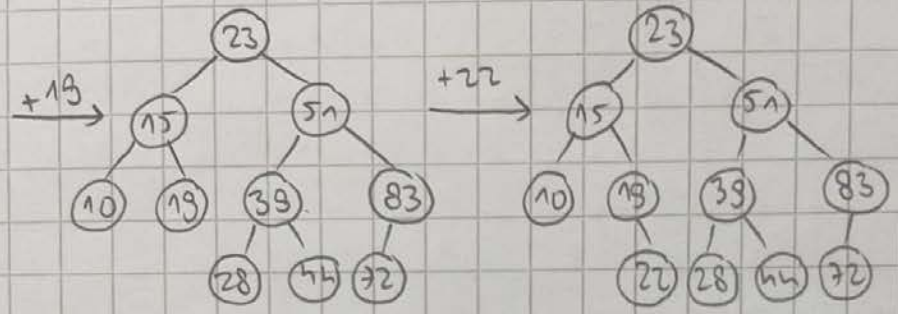


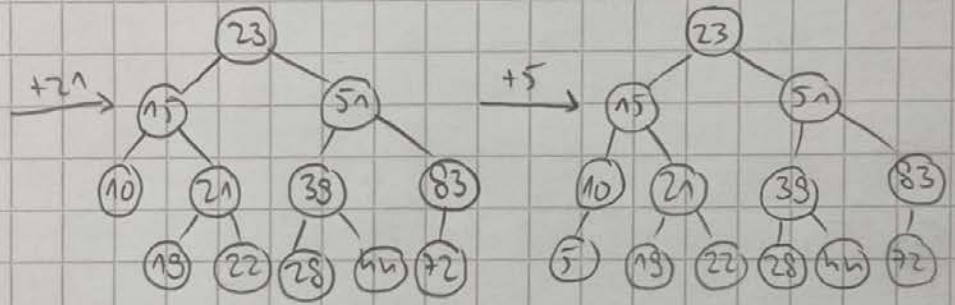
1.) 5 -> 4, 10 -> 8, 13 -> 12, 15 -> 13

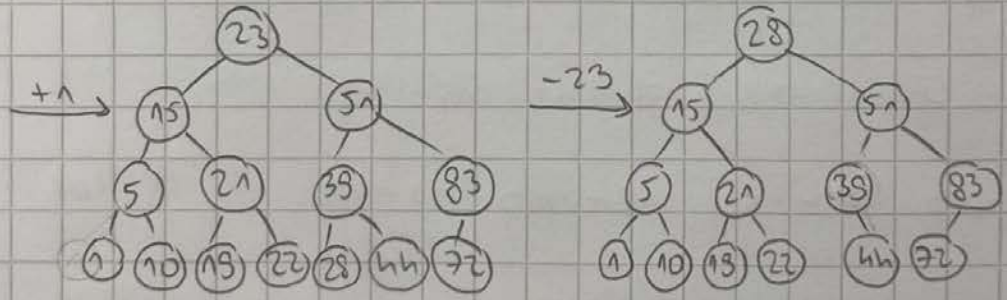
2.) 4 -> 3, 8 -> 5, 12 -> 10, 17 -> 15

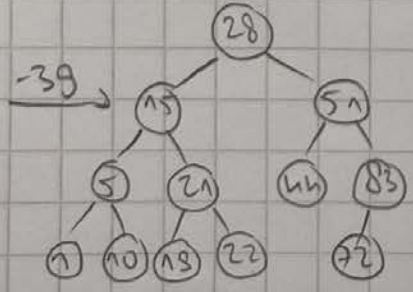
ACП №20


3.
23
10
39
15
28
51
83
+44
23
10
39
15
28
51
44
83
+72
23
10
51
15
39
83
28
44
72


+19
23
15
51
10
19
39
83
28
44
72
+22
23
15
51
10
19
39
83
22
28
44
72

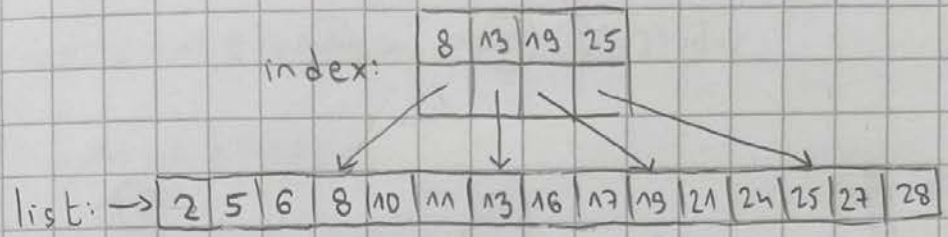

+21
23
15
51
10
21
38
83
19
22
28
44
72
+5
23
15
51
10
21
39
83
5
19
22
28
44
72


+1
23
15
51
5
21
39
83
1
10
19
22
28
44
72
-23
28
15
51
5
21
39
83
1
10
13
22
44
72


-39
28
15
51
5
21
44
83
1
10
13
22
72

АСП

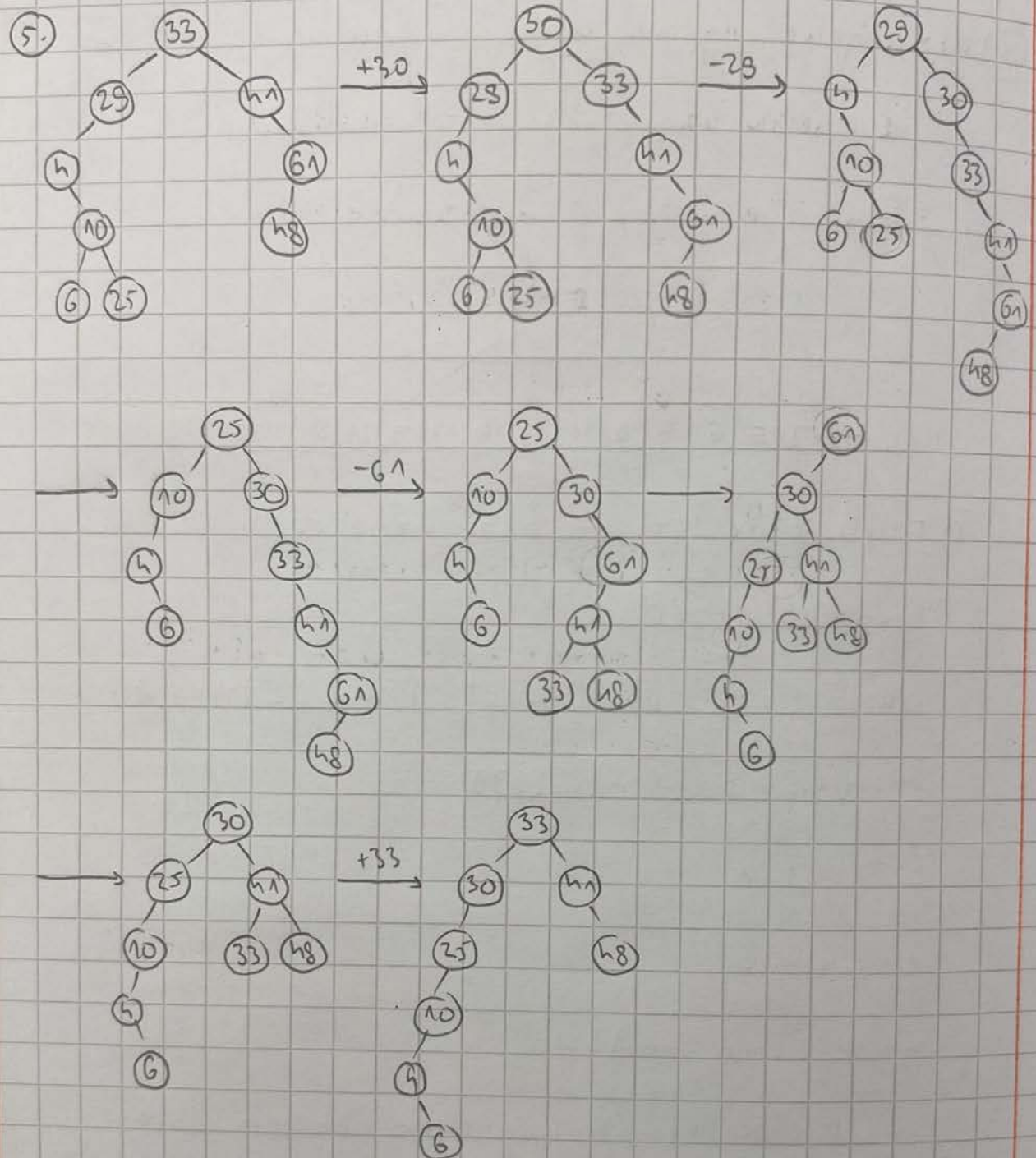
(4) а) - индекс садржи мањи број елемената низова из листе и показивање на те низове у датој листи



б) FIND_KEY_LIST(list, index, size, key)

```
tmp = list, i = 0
while (key > info(index[i]) and i < size) do
    tmp = adr(index[0])
    i = i + 1
end_while
while (tmp != nil) do
    if (info(tmp) = key) then return tmp
    else if (info(tmp) > key) then return nil
    end_if
    tmp = next(tmp)
end_while
return nil
```

ACП №-20


5.
33
29
41
4
61
10
48
6
25
+30
50
28
33
4
41
10
61
6
25
48
-28
29
4
30
10
33
6
25
41
61
48
25
10
30
4
33
6
41
61
48
-61
25
10
30
4
61
6
41
33
48
61
30
27
41
10
33
48
4
6
30
25
41
10
33
48
4
6
+33
33
30
41
25
48
10
4
6

(c) PRINT_ALL_BST(root)

```
n=0, preorder=NEWARRAY(NODE)
STACK_INIT(S), PUSH(S, root)
while (not STACK_EMPTY(S)) do
    next=POP(S)
    while (next != nil) do
        preorder[n]=next, n=n+1
        if (right(next)!=nil) then PUSH(S, right(next))
        next=left(next)
    end_while
end_while
for(i=n-1 down to 0) do
    p=preorder[i]
    if((left(p)=nil or (bst(left(p)) and info(left(p))<info(p))) and
    (right(p)=nil or (bst(right(p)) and info(right(p))>info(p)))) then
        bst(p)=true, print(p)
    else
        bst(p)=false
    end_if
```

АСП К1-20

(2.) FIND-FIRST_POSITIVE(f)

```
low = 0 , high = 1
while (f(high) < 0) do
    low = high , high = high * 2
end-while
while (low < high) do
    mid = low + (high - low)(-f(low)) / (f(high) - f(low))
    if (f(mid) = 0) return mid
    if (f(mid) > 0) then high = mid
    else low = mid
    end-if
end-while
```

- прво одредујемо интервал неизвесности, а онда га методом интерполационог претраживања сужујемо, само да је вредност функције у крајевима супротног знака

(8) a) $S_n = 1 + (U_0 + U_1 + \dots + U_{n-1})/n$

$$S_n = \frac{P1 + n}{n}$$

$$U_n = \frac{PE}{n+1}$$

b) $PE = P1 + 2n$

$$S_n = \frac{PE - 2n + n}{n} = \frac{PE - n}{n} = \frac{n+1}{n} U_n - 1$$

АСП №1-13

(1.) - умейлайте 5, 28, 29, указывайте 14, 15

4	7	10	14	15	15	21	24	25	40	45
1	0	0	1	0	0	0	1	0	1	0

↓ +5

4	5	5	14	15	15	21	24	25	40	45
1	1	0	1	0	0	0	1	0	1	0

↓ +28

4	5	5	14	15	15	21	24	25	28	40
1	1	0	1	0	0	0	1	0	1	1

↓ +29

4	5	5	14	15	15	21	24	28	29	40
1	1	0	1	0	0	0	1	1	1	1

↓ -14

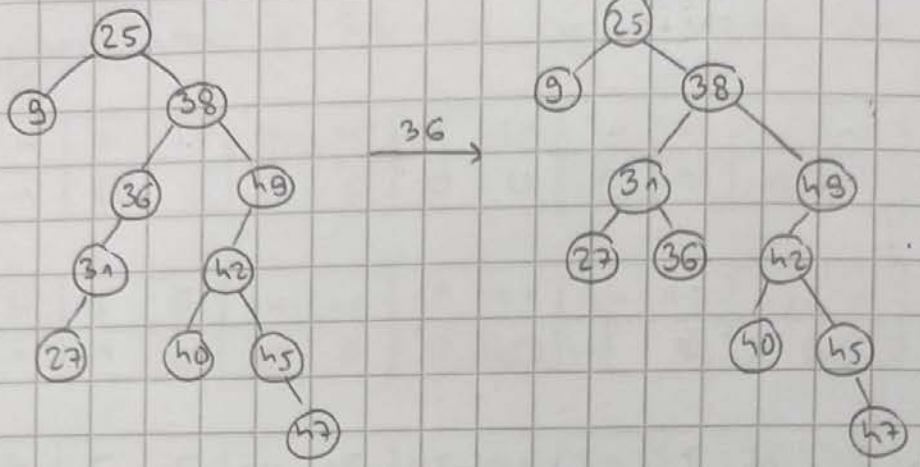
4	5	5	14	15	15	21	24	28	29	40
1	1	0	0	0	0	0	1	1	1	1

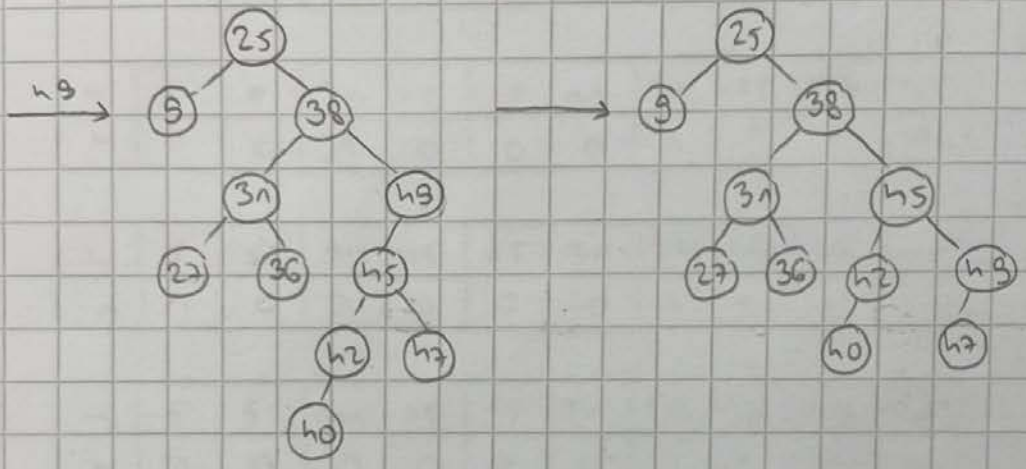
↓ -15

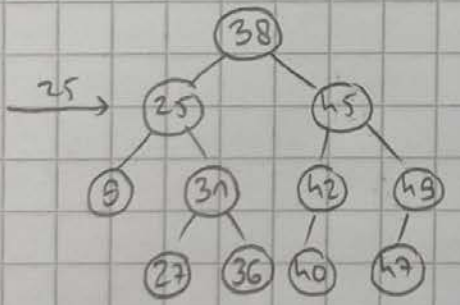
4	5	5	14	15	15	21	24	28	29	40
1	1	0	0	0	0	0	1	1	1	1

ACD Ш1-19

3.

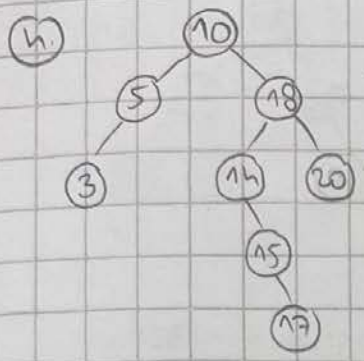

25
9
38
36
49
31
42
27
40
45
47
36
25
9
38
31
49
27
36
42
40
45
47


49
25
9
38
31
49
27
36
45
42
47
40
25
9
38
31
45
27
36
42
49
40
47


25
38
25
45
9
31
42
49
27
36
40
47

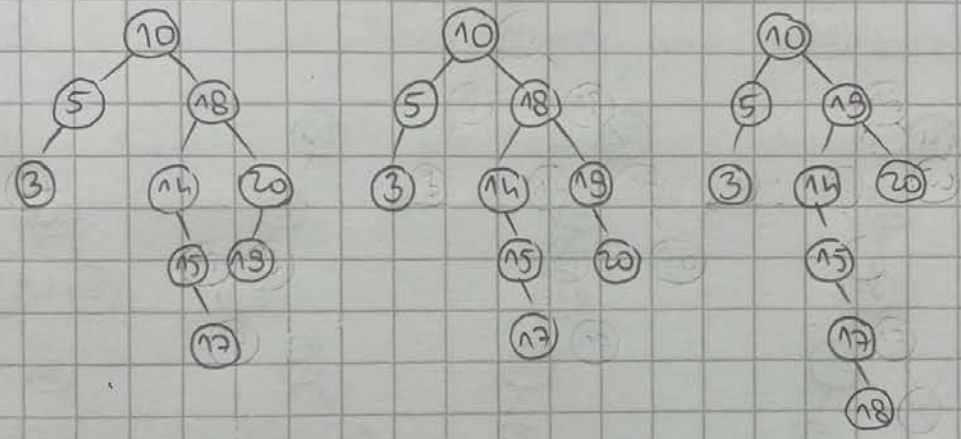
АСП

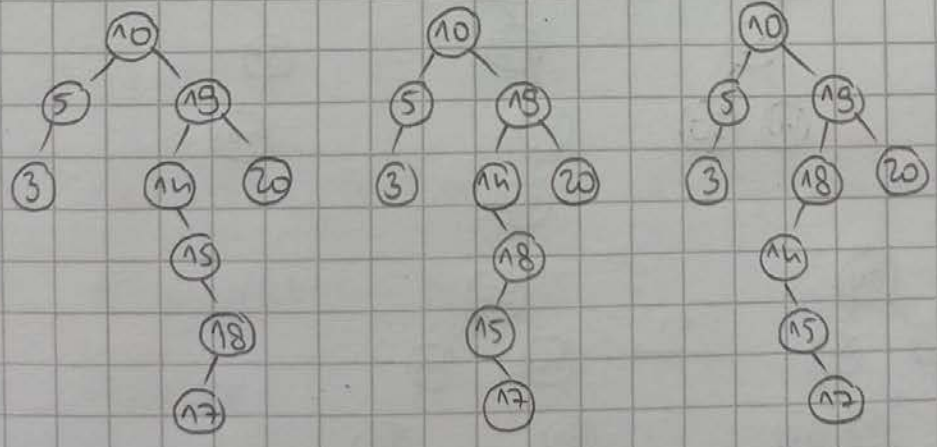
Ид-19



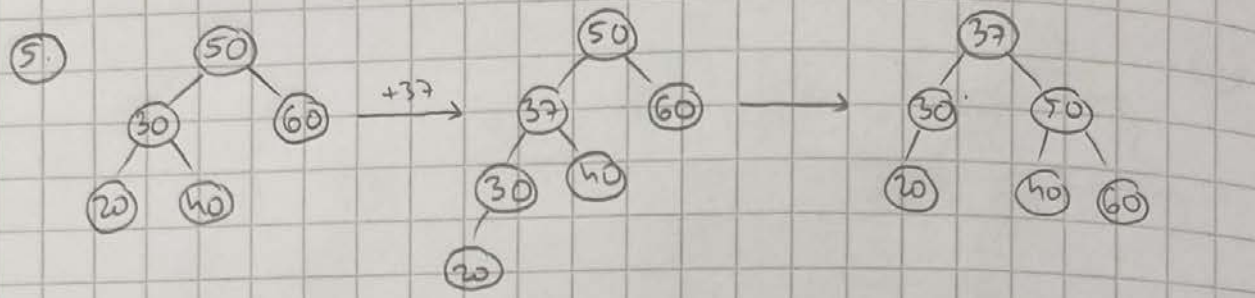
- сви могући изгледи стабла пре брисања

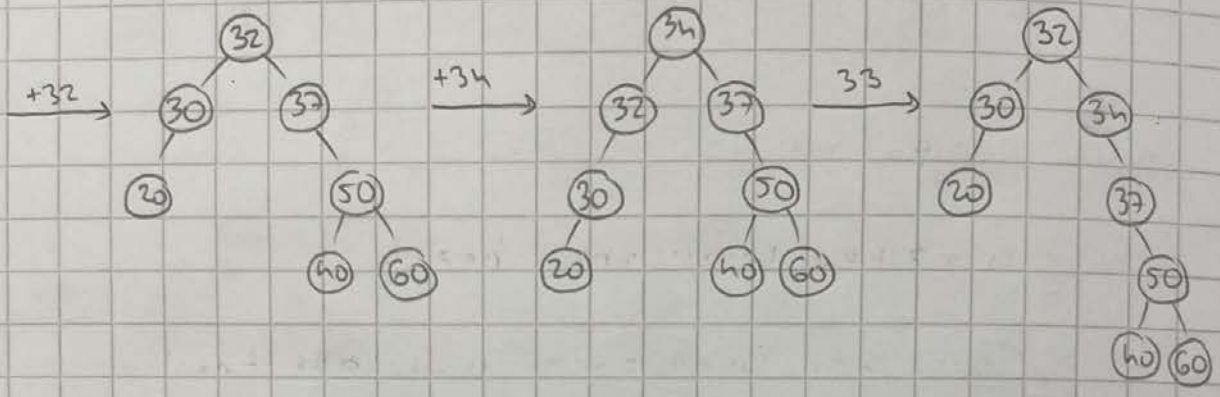
чвора 19:

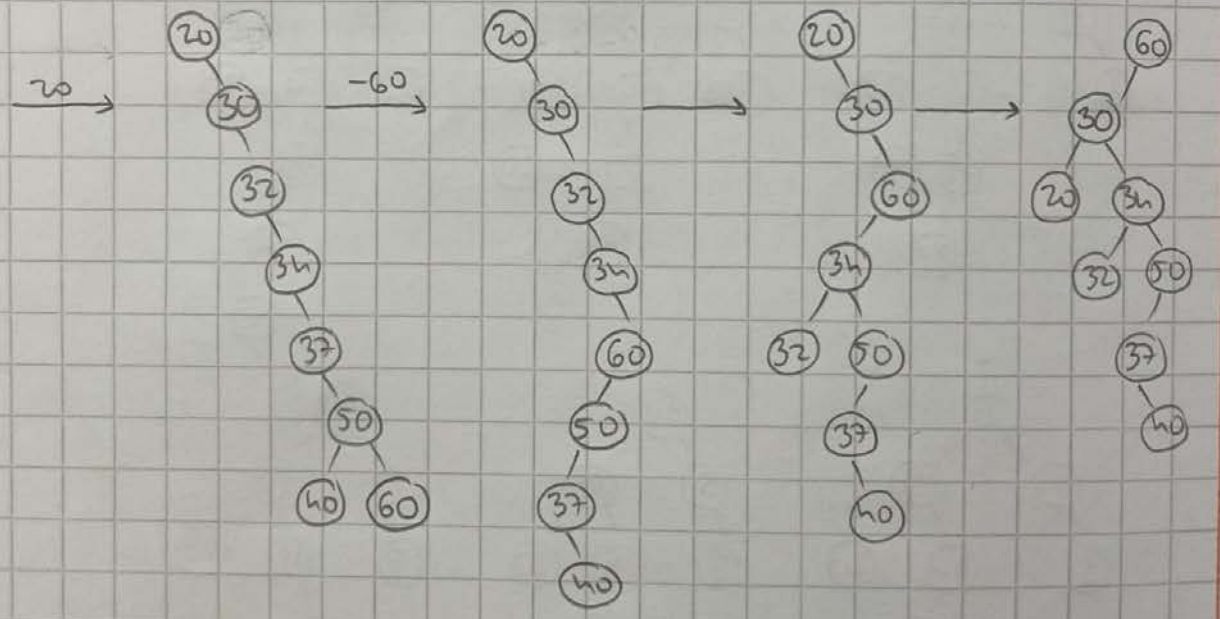


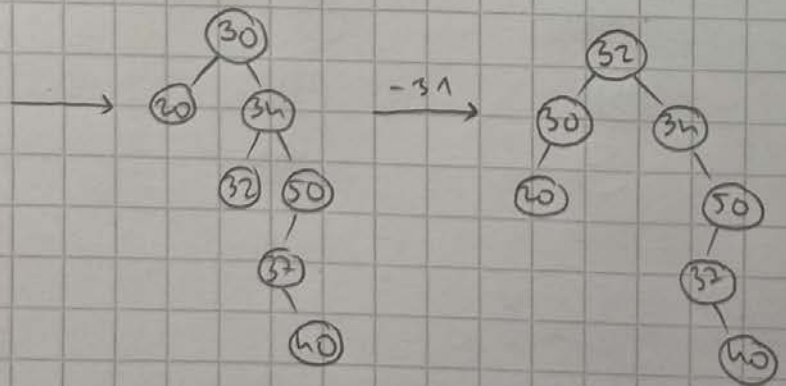


ACП №1-19


5.
50
30
60
20
40
+37
50
37
60
30
40
20
37
30
50
20
40
60


+32
32
30
37
20
50
40
60
+34
34
32
37
30
50
20
40
60
33
32
30
34
20
37
50
40
60


20
20
30
32
34
37
50
40
60
-60
20
30
32
34
60
50
37
40
20
30
60
34
32
50
37
40
60
30
20
34
32
50
37
40


30
20
34
32
50
37
40
-31
32
30
34
20
50
37
40

⑥ SPLAY_SPLIT(root, k)

```
tmp=root
p=nil
while (tmp!=nil) do
    p=tmp
    if(k=key(tmp)) then break
    if(k<key(tmp)) then tmp=left(tmp)
    else tmp=right(tmp)
    end_if
end_while
if (tmp=nil)
    SPLAY(p)
    if(k>key(p)) return p, right(p)
    else return left(p), p
    end_if
else
    SPLAY(tmp)
    return left(tmp), right(tmp)
end_if
```

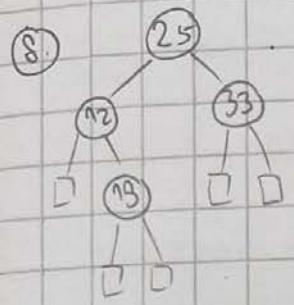
ACN ಸಂಖ್ಯೆ-19

7. FIND_BST_FLOOR(root, key)

```
tmp = root

p = nil

while (tmp != nil) do

    if (k = key(tmp)) then return tmp

    if (k > key(tmp)) then

        p = tmp

        tmp = right(tmp)

    else tmp = left(tmp)

    end_if

end_while

return p
```

ИИ-19

8.



k_i:	12	19	25	33
p_i:	0.2	0.2	0.15	0.05

k_i:	$k_i<12$	$12<k_i<19$	$19<k_i<25$	$25<k_i<33$	$33<k_i$
q_i:	0.05	0.1	0.15	0.05	0.05

а) цена слабна: $C = \sum_{i=1}^{n} p_i(k_i+1) + \sum_{i=0}^{n} q_i k_i$

$C = 0.2\cdot 2 + 0.2\cdot 3 + 0.15 + 0.05\cdot 2 + 0.05\cdot 2 + 0.1\cdot 3 + 0.15\cdot 3 +$

$+ 0.05\cdot 2 + 0.05\cdot 2 = 0.4 + 0.6 + 0.15 + 0.1 + 0.1 + 0.3$

$+ 0.45 + 0.1 + 0.1 = 2.3$

б) корен 19 → $w_l = q_0 + p_1 + q_1 = 0.05 + 0.2 + 0.1 = 0.35$

$w_r = q_2 + p_3 + q_3 + p_4 + q_4 = 0.15 + 0.15 + 0.05 + 0.05 + 0.05 = 0.45$

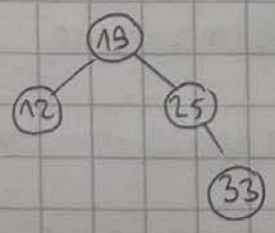
$|w_l - w_r| = 0.1$

корен 25 → $w_l = q_0 + p_1 + q_1 + p_2 + q_2 = 0.05 + 0.2 + 0.1 + 0.2 + 0.15 = 0.7$

$w_r = q_3 + p_4 + q_4 = 0.05 + 0.05 + 0.05 = 0.15$

$|w_l - w_r| = 0.55$

=> за корен се бира 19, а затим 25



АСП ИА-18

(1.) а) $C = \sum_{i=1}^{n} p_i(h_{i+1}) + \sum_{i=0}^{n} q_i h_i$

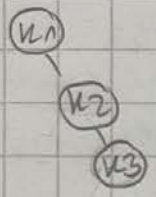
h_i - висина интерног / екстерног чвора

p_i - вероватноћа успешне претраге на интервалу k_i

q_i - вероватноћа неуспешне претраге на интервалу (k_i, k_{i+1})

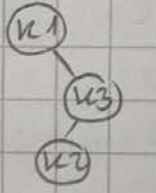
б) $k_1 < k_2 < k_3$, $p_1 = 0.2$, $p_2 = 0.05$, $p_3 = 0.1$

$q_0 = 0.15$, $q_1 = 0.2$, $q_2 = 0.1$, $q_3 = 0.2$



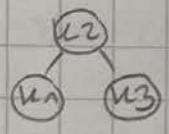
$C = 0.2 + 0.05 \cdot 2 + 0.1 \cdot 3 + 0.15 + 0.2 \cdot 2 + (0.1 + 0.2) \cdot 3 =$

$= 0.2 + 0.1 + 0.3 + 0.15 + 0.4 + 0.9 = 2.05$



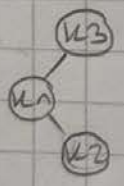
$C = 0.2 + 0.1 \cdot 2 + 0.05 \cdot 3 + 0.15 + (0.2 + 0.1) \cdot 3 + 0.2 \cdot 2 =$

$= 0.2 + 0.2 + 0.15 + 0.15 + 0.9 + 0.4 = 2$



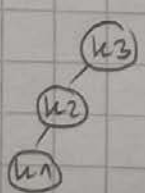
$C = 0.05 + (0.2 + 0.1) \cdot 2 + (0.15 + 0.2 + 0.1 + 0.2) \cdot 2 =$

$= 0.05 + 0.6 + 1.3 = \underline{1.95}$



$C = 0.1 + 0.2 \cdot 2 + 0.05 \cdot 3 + 0.15 \cdot 2 + (0.2 + 0.1) \cdot 3 + 0.2 =$

$= 0.1 + 0.4 + 0.15 + 0.3 + 0.9 + 0.2 = 2.05$



$C = 0.1 + 0.05 \cdot 2 + 0.2 \cdot 3 + (0.15 + 0.2) \cdot 3 + 0.1 \cdot 2 + 0.2 =$

$= 0.1 + 0.1 + 0.6 + 1.05 + 0.2 + 0.2 = 2.25$

② FIND_BST_SUCC(root, key)

```
tmp=root
p=nil
while (tmp != nil) do
  if (key=key(tmp)) then break
  if (key<key(tmp)) then
    p=tmp
    tmp=left(tmp)
  else tmp=right(tmp)
  end_if
end_while
if (right(tmp) != nil) then
    tmp=right(tmp)
    while (left(tmp) != nil) do tmp=left(tmp)
    return tmp
else return p
end_if
```

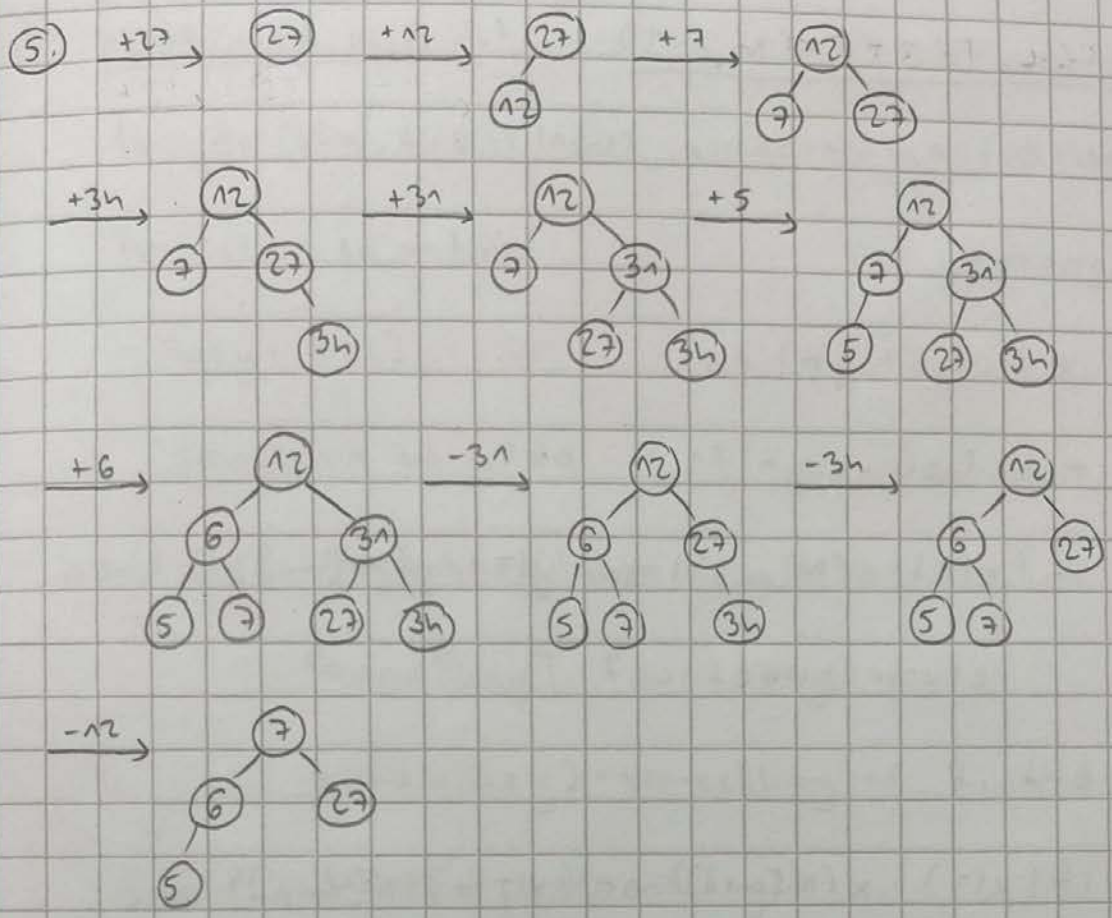
ΑΕΜ 41-18

④ CORRECT_BST(root)

```
n=0 , next=root , STACK_INIT(S) , wrong=nil
while (true) do
    while (next!=nil) do
        PUSH(S, next) , next=left(next)
    end_while
    if (not STACK_EMPTY(S)) then
        next=POP(S)
        a[n]=next , n=n+1
        next=right(next)
    else break
    end_if
end_while
for (i=0 to n-1) do
    if ((a[i]>a[i-1] and a[i]>a[i+1]) or
        (a[i]<a[i-1] and a[i]<a[i+1]) then
        if (wrong=nil) wrong=a[i]
        else SWITCH(wrong, a[i])
    end_if
end_for
```

№1-18

Асп


5.
+27
27
+12
27
12
+7
12
7
27
+34
12
7
27
34
+31
12
7
31
27
34
+5
12
7
31
5
27
34
+6
12
6
31
5
7
27
34
-31
12
6
27
5
7
34
-34
12
6
27
5
7
-12
7
6
27
5

ACN ΚΛ-18

(c) CHECK_VERTEX(M, n, T)

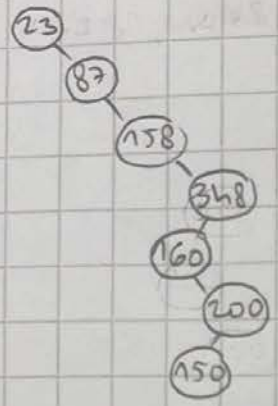
```
low = 0
high = n-1
while (low ≤ high) do
    mid = (low + high) / 2
    if (x(T) = x(M[mid]) and y(T) = y(M[mid])) then
        return true
    end_if
    if (x(T) > x(M[mid]) or (x(T) = x(M[mid]) and
        y(T) > y(M[mid]))) then low = mid + 1
    else high = mid - 1
    end_if
end_while
return false
```

(7) SEARCH(A, K, n, m)

```
found = NEW_ARRAY(100), times = NEW_ARRAY(100)
for (i = 1 to m) do
    key = K[i]
    for (j = 1 to n) do
        if (A[j] = key) then
            found[key] = found[key] + j
            times[key] = times[key] + 1
            MOVE(j, times[key])
            break
        end_if
    end_for
end_for
avg = found / times
return avg
```

ACM ИЛИ-18

(8.) а) 23 → 87 → 158 → 348 → 160 → 200 → 150 → 158

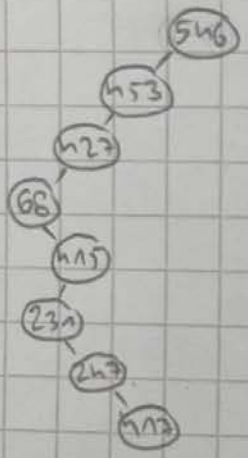


није валидна секвенца

јер 150 не може бити у

десном подстаблу 160

б) 546 → 453 → 427 → 68 → 415 → 231 → 247 → 417 → 300



није валидна секвенца

јер 415 не може бити у

левом подстаблу 415

КЛ-17

АСП

1.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	7	10	11	18	22	25	27	39	41	45	48	56	64	65	78	86

- бинарна претрага на 25 и ни:

4 поредења за 25 + 4 поредења за ни = 8

- => 8 поредења

- вишеструка секвенцијална претрага на 25 и ни:

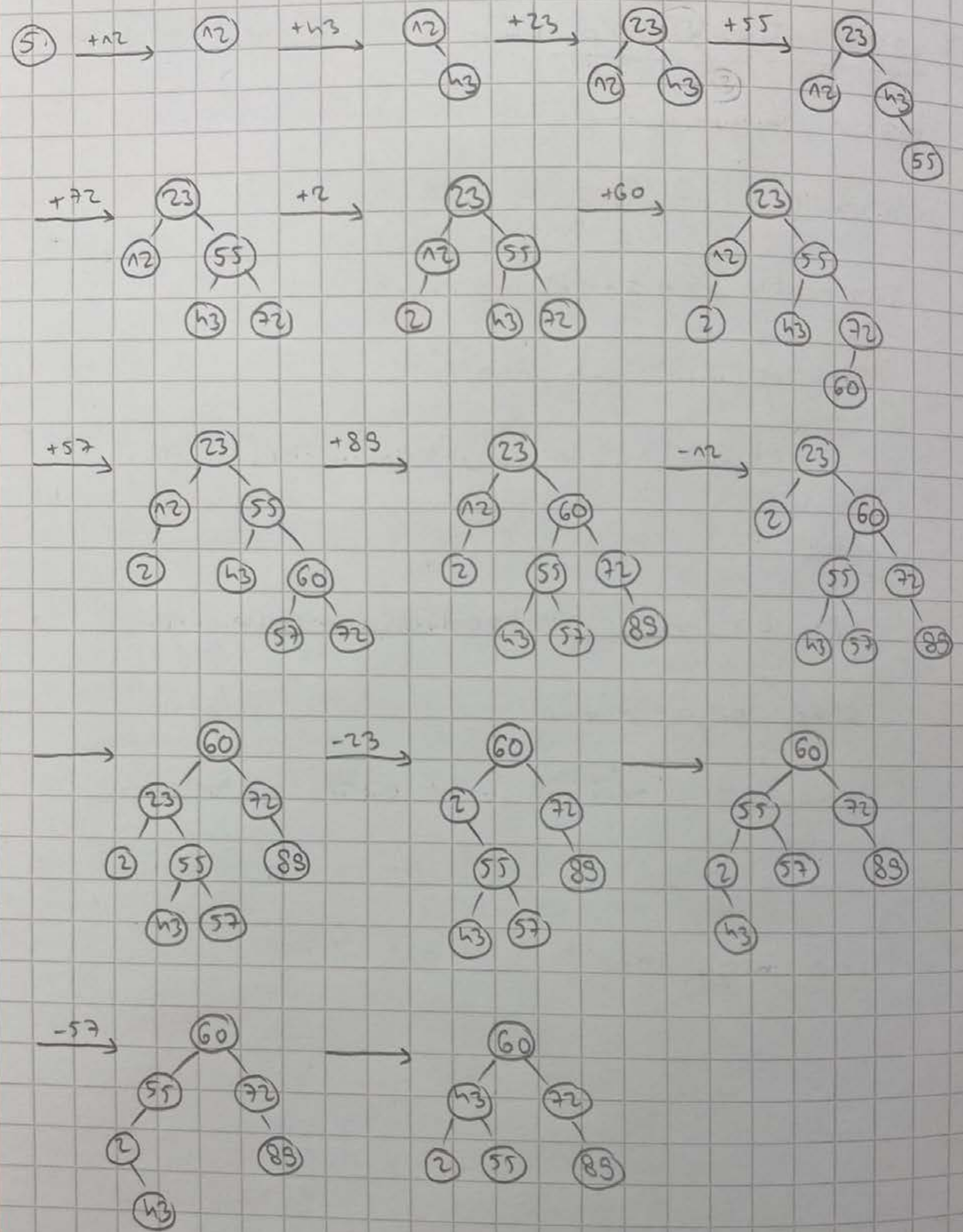
8 поредења за 25 + 6 поредења за ни

=> 14 поредења

(h) ARR-PRED-SUCC (arr, k)

```
low = 1, high = n
while (low ≤ high) do
    mid = (low + high) / 2
    if (arr[mid] = k) then
        return mid-1, mid+1
    end_if
    if (k > arr[mid]) then low = mid+1
    else high = mid-1
    end_if
end_while
return high, low
```

ACП 41-17



ΑΣΠ

⑥ CHECK-AVL-BALANCED(bst)

```
p = root(bst) , QUEUE_INIT (Q)
INSERT (Q, p)
while (not QUEUE-EMPTY(Q)) do
    p = DELETE(Q)
    if (abs (hl(p) - hr(p)) > 1) then return false
    end_if
    if (left(p) != nil) then INSERT (Q, left(p))
    end_if
    if (right(p) != nil) then INSERT (Q, right(p))
    end_if
end_while
return true
```

АСП К1-17

7) а) - принцип одређивања субоптималног BST стабла, у сваком кораку корен подстабла одређујемо као чвор такав да је апсолутна разлика тежине његовог левог и десног подстабла минимална

б) <u>FIND-ROOT(P, Q, n)</u>

```
left = Q[0], right = 1 - Q[0] - P[1]
min = abs(left - right)
for (i=2 to n) do
    left = left + P[i-1] + Q[i-1]
    right = right - Q[i-1] - P[i]
    if (abs(left - right) < min) then
        min = abs(left - right)
    else return i-1
    end_if
end_for
return n
```

8. +50 → 50 +70 → 70 (50) +30 → 30 (50 (70)) +60 → 30 (60 (50) 70)

→ 60 (30 (50) 70) +40 → 60 (40 (30) 50) 70) → 40 (30) 60 (50) 70)

20 → 70 (60 (40 (30) 50)) +45 → 70 (60 (45 (40 (30)) 50)) → 45 (40 (30) 60 (50) 70)

20 → 30 (40 (45 (60 (50) 70)))