ALGORITMI I STRUKTURE PODATAKA 2

Pripremila: Lena Joković

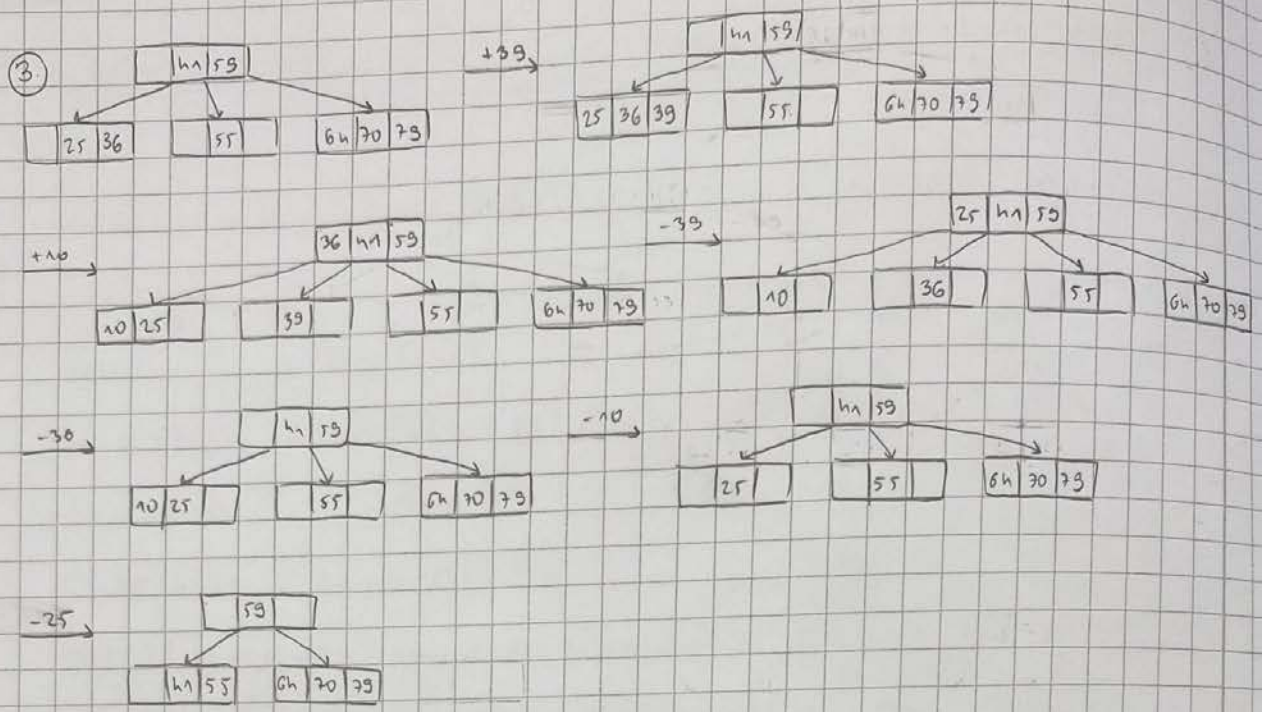
ΑΣΚ2 [K2-23]

① B_DELETE_KEY(root, m, node, key)

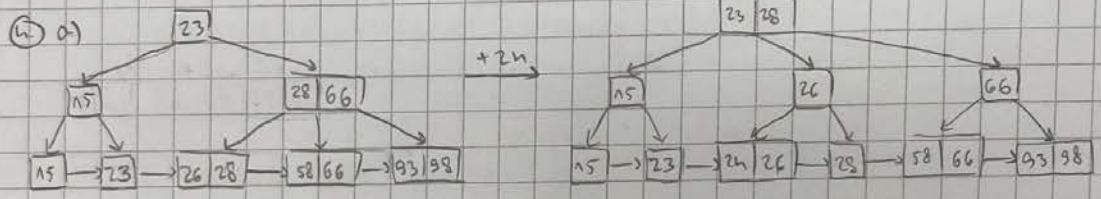
```
i=1
while (node.keys[i]<key) do i=i+1
end_while
temp=node.children[i+1]
while (temp.children[0] != null) do temp=temp.children[0]
end_while
SWAP(node, i, temp, 0)
SHIFT(temp, 0), temp.num=temp.num-1
parent=temp.parent, i=1
while (parent.children[i] != temp) do i=i+1
end_while
if (i=parent.num) then
    left=parent.children[i-1], right=temp, i=i+1
else
    left=temp, right=parent.children[i+1]
end_if
size=left.num+1, left.keys[size]=parent.keys[i]
SHIFT(parent, i), parent.num=parent.num-1
for j in range 1 to right.num do
    size=size+1, left.keys[size]=right.keys[j]
end_for
... (κώδικας για ροζ περιπτώσεις)
```

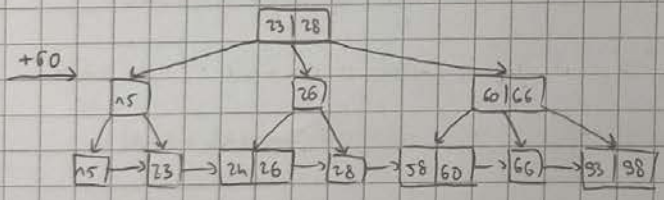
ACM2 | K2-23 |

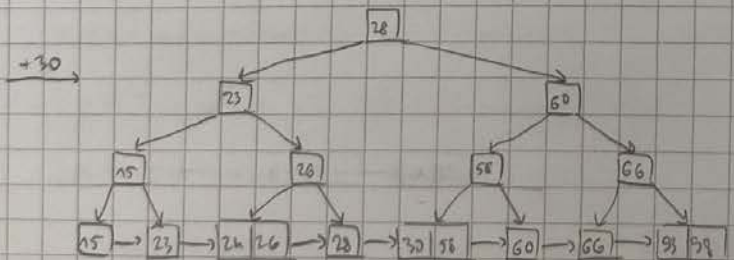
3.

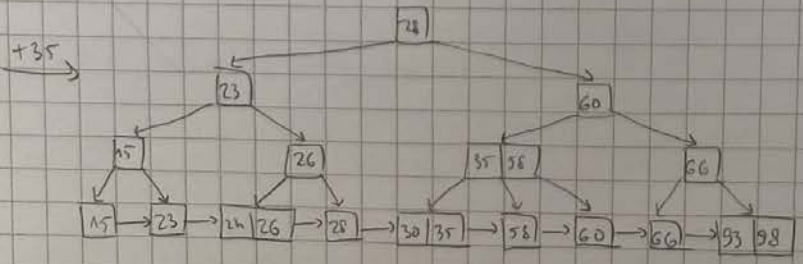


4) a)

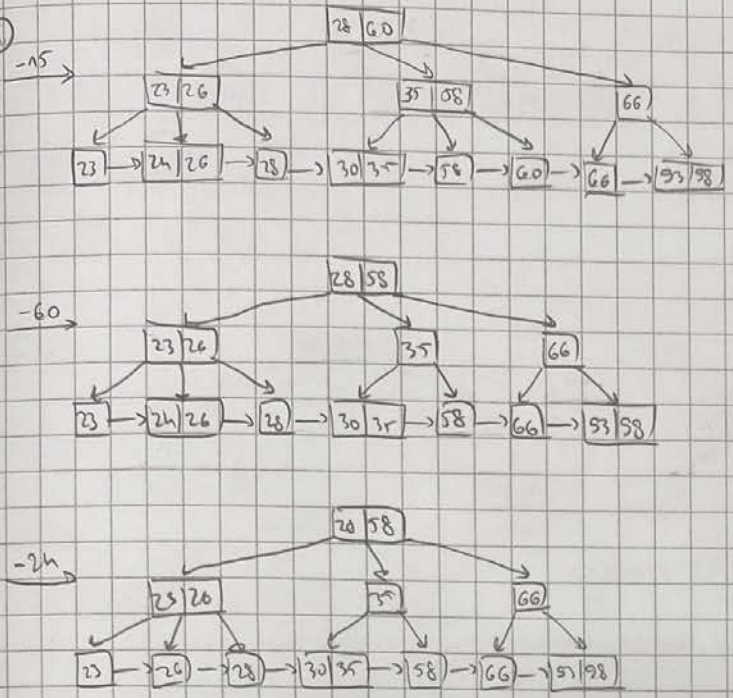








4)



б) - просечан број приступа приликом успешне претраге: 3

- попуњеност табле: $\frac{15}{22} \approx 0.68$

5) - хеш функција се добија сумирањем кодова појединачних карактера стринга, по модулу величине табеле(n)

- униформна хеш функција при довољно дугачким стринговима

A..Z -> 65..90, дужина 10 карактера

а) n = 100

sum ≈ 650..900, h(k) = (sum % 100)

- применљива хеш функција која даје релативно униформно распоређене кодове

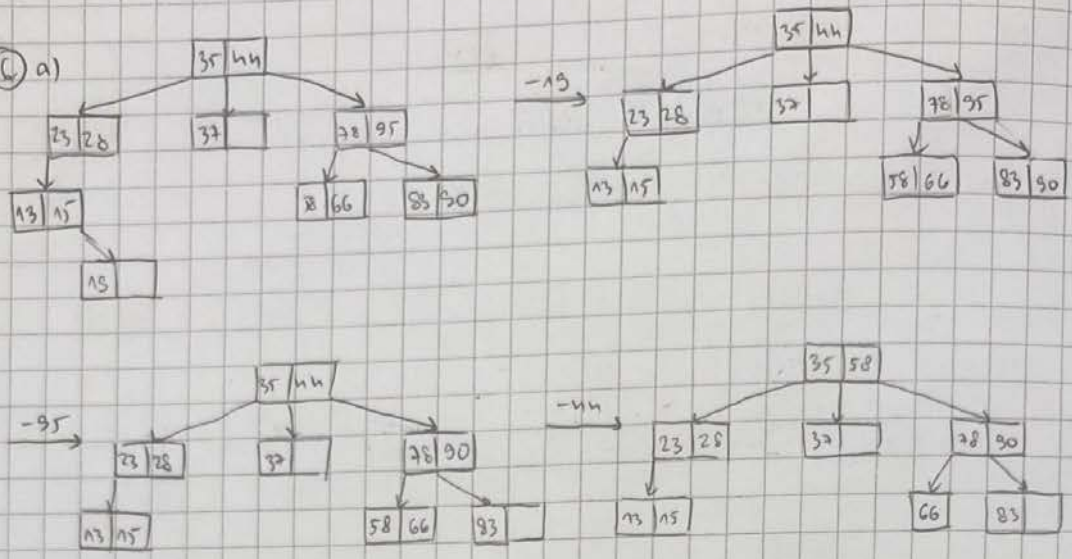
б) n = 1000

sum ≈ 650..900, h(k) = (sum % 1000)

- неприменљива хеш функција, јер не давала сами кодове у распону од 650 до 900, па не велика табела биће неискоришћена

АСП2 К2-23

Q a)

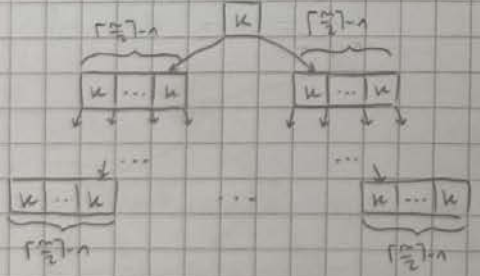


- брисање 19: брисање из листа
- брисање 95: мењање са претходником и брисање из листа
- брисање 44: мењање са следбеником и брисање из листа

б) максимална вероватноћа преклопа у B стаблу реда m

- стабло од n нивоа - (n-2) преклопа
- има најмање $1 + (n-1)(\lceil \frac{m}{2} \rceil - 1)$ кључева
- максимална вероватноћа преклопа је једнака:

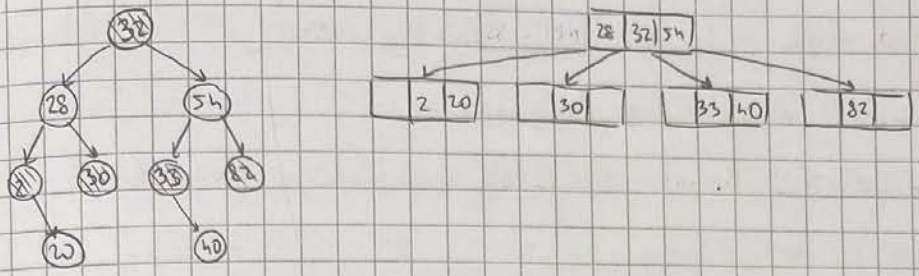
$$\frac{n-2}{1+(n-1)(\lceil \frac{m}{2} \rceil - 1)} \quad , \quad n \gg \Rightarrow \frac{1}{\lceil \frac{m}{2} \rceil - 1}$$

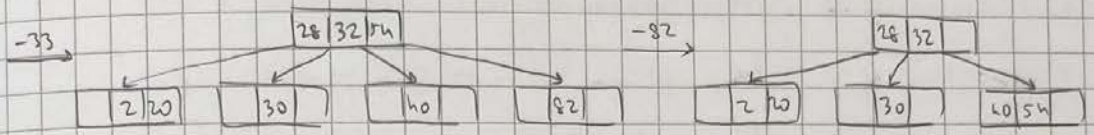


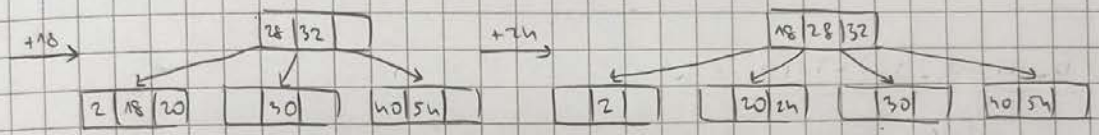
К2-22

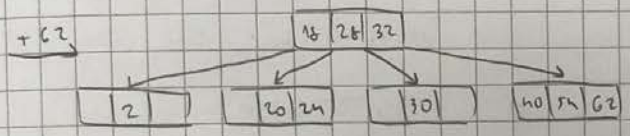
АСПЗ

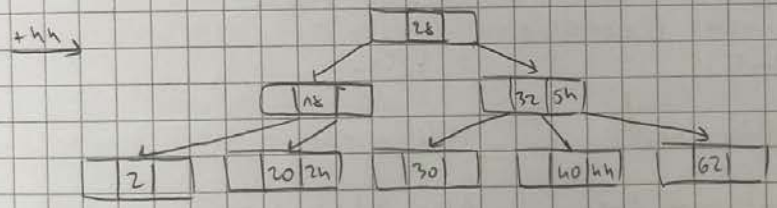
1.

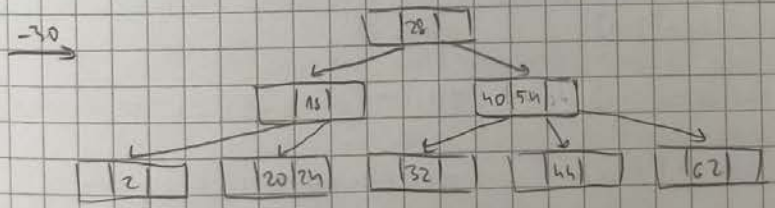

32
28
54
18
30
33
82
20
40
28 32 54
2 20
30
33 40
82


-33
28 32 54
2 20
30
40
82
-82
28 32
2 20
30
40 54


+18
28 32
2 18 20
30
40 54
+24
2
18 28 32
20 24
30
40 54


+62
18 28 32
2
20 24
30
40 54 62


+44
28
18
32 54
2
20 24
30
40 44
62


-30
28
18
40 54
2
20 24
32
44
62

АСПЗ K2-22

② а) B^* стабло реда m ($m>3$), више од два нивоа

два листова минимално попуњена

два унутрашња чвора више од минимално попуњена

из листа се бришу кључеви

1.) спајање у z: чвор, десни брат и леви брат

2.) спајање у z: чвор, десни брат и други десни брат

3.) спајање у z: чвор, леви брат и други леви брат

б) B_STAR_DELETE(node, w)

```
i = INDEX(node.keys, w)
SHIFT(node, i)
node.num = node.num - 1
arr = COPY(node.keys, node.num)
parent = node.parent
i = INDEX(parent.childern, node)
if (i > 1 and i <= parent.num) then
    node1 = parent.childern[i-1]
    node2 = parent.childern[i+1]
else if (i <= parent.num - 1) then
    node1 = parent.childern[i+1]
    node2 = parent.childern[i+2]
else if (i > 2) then
    node1 = parent.childern[i-1]
    node2 = parent.childern[i-2]
end_if
size = node.num
for j in range 1 to node1.num do
    INSERT(arr, node1.keys[j])
    INSERT(arr, node2.keys[j])
    size = size + 2
end_for
if (i = parent.num+1) then i = i-2
else if (i > 1) then i = i-1
end_if
SHIFT(parent, i), mid = size/2
parent.num = parent.num - 1
parent.keys[i] = arr[mid]
for j in range 1 to mid-1
    node1.keys[j] = arr[j]
    node2.keys[j] = arr[mid+j]
end_for
node1.num = node2.num = mid-1
parent.childern[i] = node1
parent.childern[i+1] = node2
```

③ a) HAS_KEY(root,key)

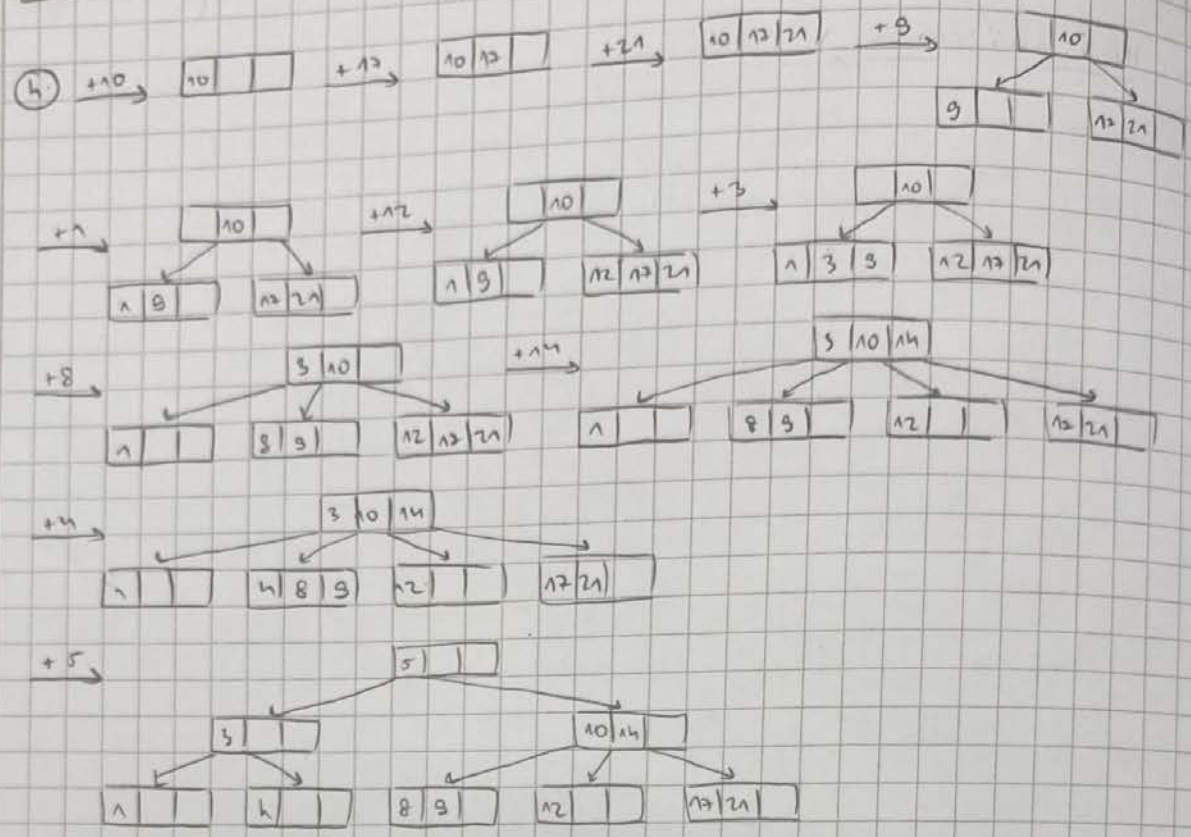
```
temp=root
for i in range 1 to len(key) do
    code=ch(key[i])
    if(temp[code]!=null) then temp=temp[code]
    else return false
    end_if
end_for
if(temp[info]!=null) then return true
else return false
end_if
```

δ) CAN_BE_DIVIDED(root, sequence)

```
if (HAS_KEY(root, sequence)) then
    return true
else
    for i in range 2 to len(sequence) do
        if (CAN_BE_DIVIDED(root, sequence[:i]) and
            CAN_BE_DIVIDED(root, sequence[i:]) then
            return true
        end_if
    end_for
    return false
end_if
```

АСП2 | К2-22

④

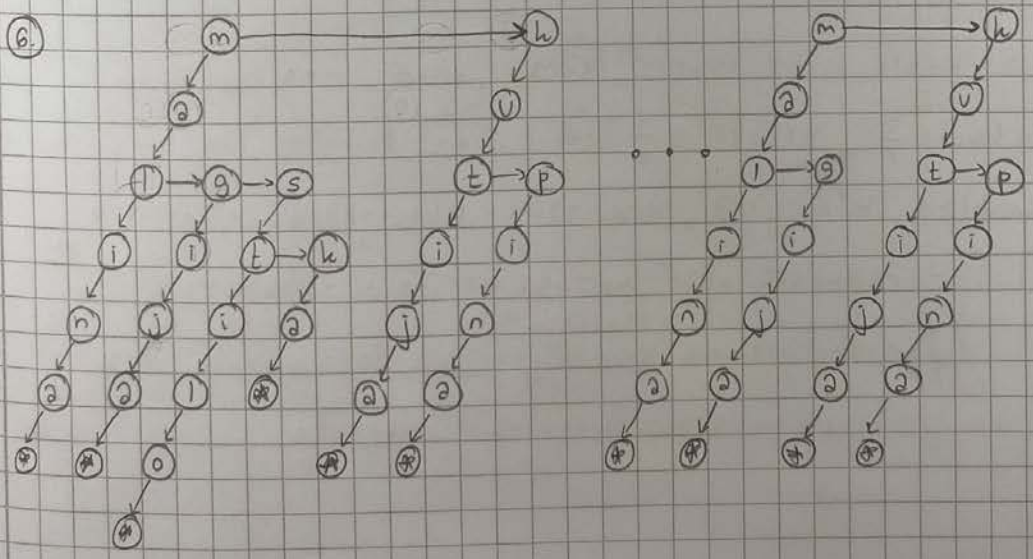


- попуштеност слабља: $\frac{\wedge\wedge}{2n} \approx 0.46$
- просечан број приступа приликом успешне претраге:
  $(1 \cdot 1 + 3 \cdot 2 + 7 \cdot 3)/11 = 2.55$
- просечан број приступа приликом неуспешне претраге: 3

⑤ а)

⑤ δ) INSERT_DATA(root, m, key, info)

⑥



АСП2 | К2-22 |

(7.) број кључева n, висине h

$n \le (m-1)\cdot(1+m+m^2+\dots+m^h) = (m-1)\frac{m^{h+1}-1}{m-1} = m^{h+1}-1$

$m^{h+1} \ge n+1 \Rightarrow \underline{m \ge (n+1)^{\frac{1}{h+1}}}$

- вредност m треба да буде довољно мала да на блоки података на диску стане m-1 кључева, m-1 показивача на запис и m показивача на потомке чвора

(8.) а) ненумерички кључеви → нумерички кључеви

- сваки карактер који се јавља у кључу се може подвргнути, на пример двојним ASCII кодом, и онда постапратити збир кодова свих карактера кључа по модулу величине табеле

$h("azx") = (1+26+24) \% n = 51 \% n$

a → 1, z → 26, x → 24 (редни број слова у абецеди)

б) 8 улаза, кључеви 3, 4, 5, 20, 21

дискретна униформна функција расподеле

$F_z(k_1) = \frac{1}{m}$, $F_z(k_2) = \frac{2}{m}$ ..., $F_z(k_m) = 1$

$H(k) = \lceil n \cdot F_z(k_i) \rceil - 1$

$k_1 = 3$, $k_2 = 4$, $k_3 = 5$, $k_4 = 20$, $k_5 = 21$, $m = 5$, $n = 8$

$F_z(k_i) = \frac{i}{5}$, $H(k_i) = \lceil 8 \cdot \frac{i}{5} \rceil - 1$, $i = 1, \dots, 5$

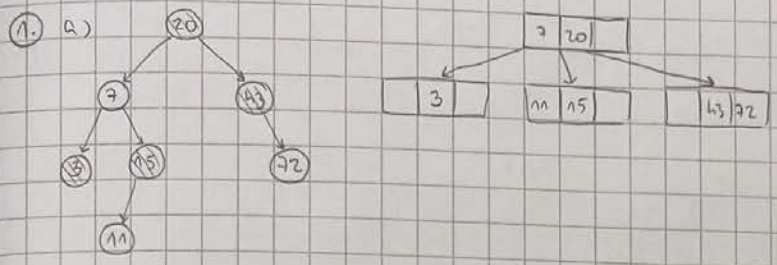
$H(k_1=3) = 1$, $H(k_2=4) = 3$, $H(k_3=5) = 4$, $H(k_4=20) = 6$, $H(k_5=21) = 7$

0	
1	3
2	
3	4
4	5
5	
6	20
7	21

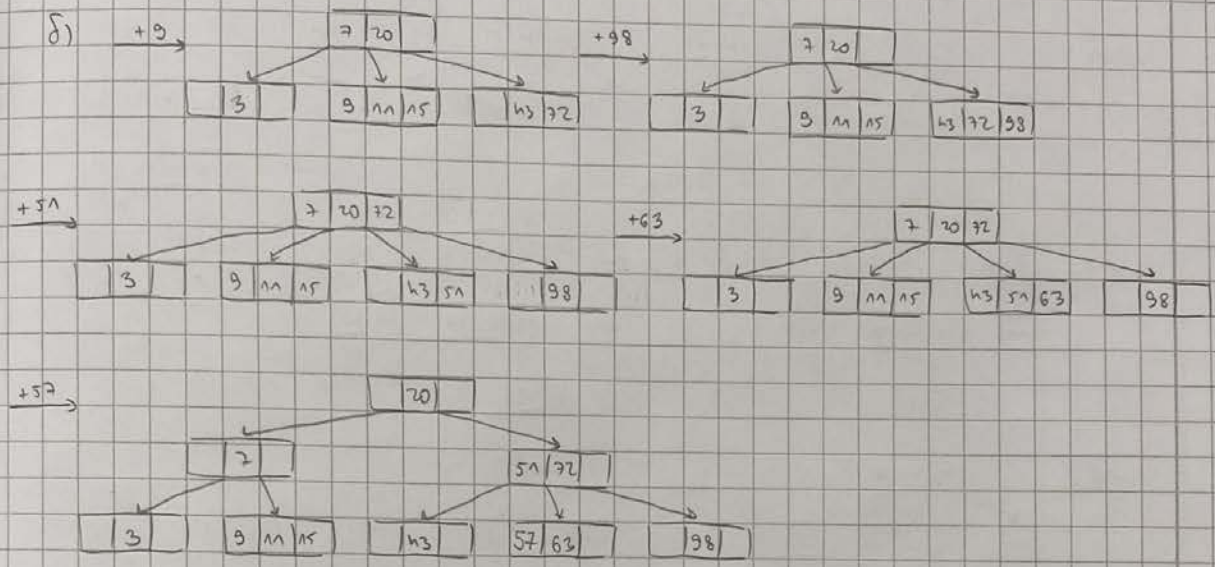
- предност овог метода је једноставно израчунавање жељ функције безконфликта, без обзира на саме вредности кључева

K2-21

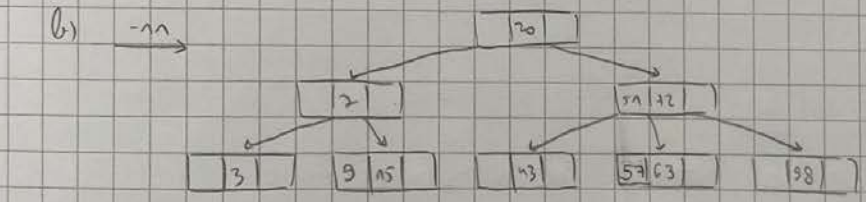
1. a)

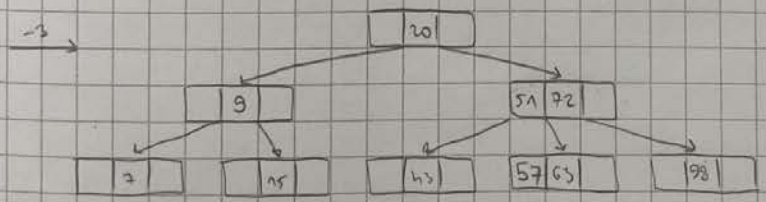

20
7
43
3
15
72
11
7 20
3
11 15
43 72

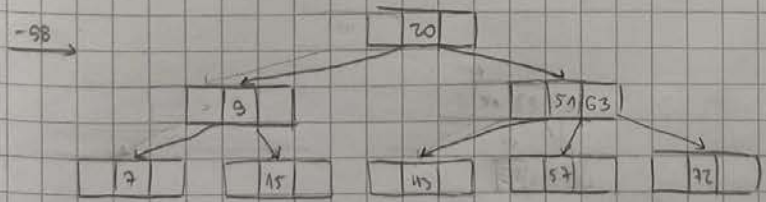
b)


+9
7 20
3
9 11 15
43 72
+98
7 20
3
9 11 15
43 72 98
+51
7 20 72
3
9 11 15
43 51
98
+63
7 20 72
3
9 11 15
43 51 63
98

+57
20
7
51 72
3
9 11 15
43
57 63
98

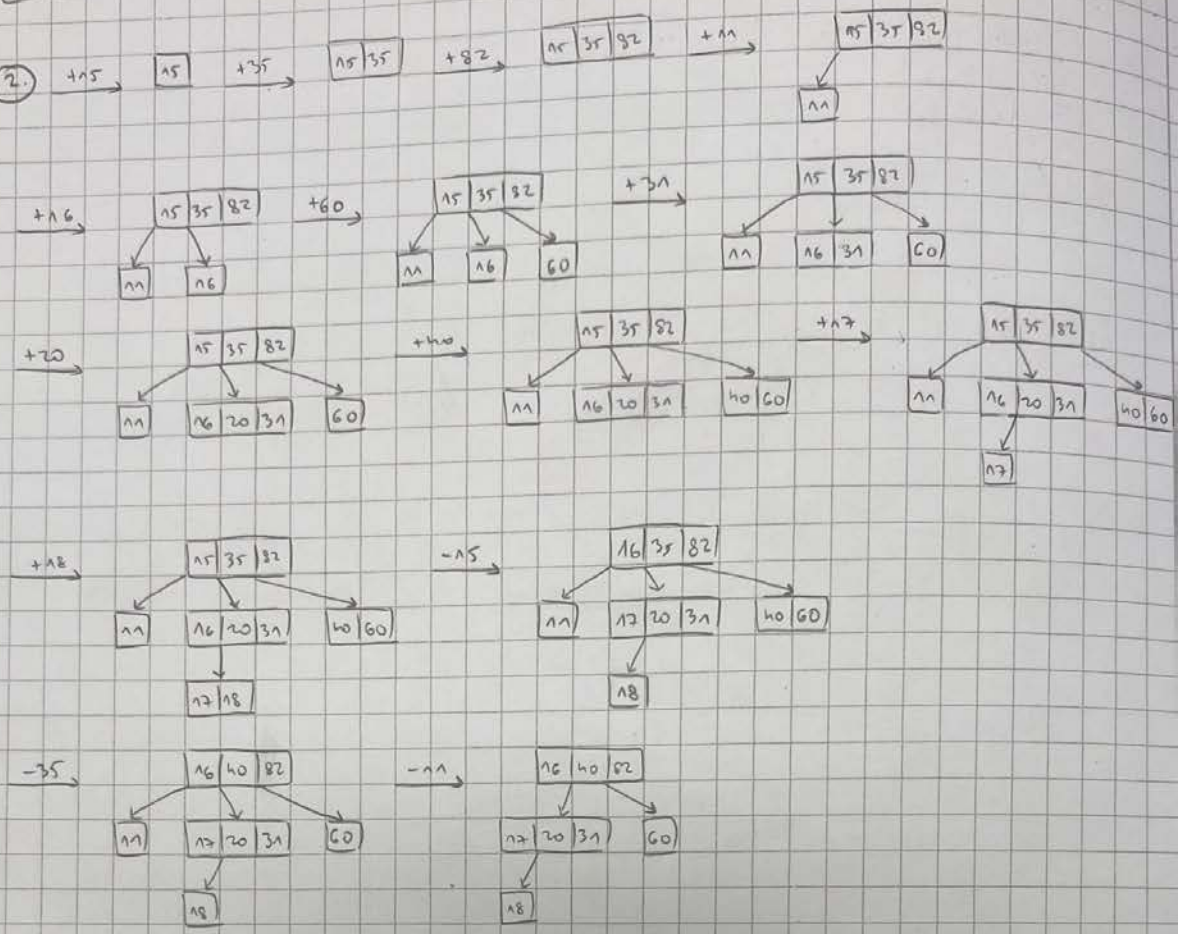

-11
20
7
51 72
3
9 15
43
57 63
98


-3
20
9
51 72
7
15
43
57 63
98


-98
20
9
51 63
7
15
43
57
72

ACn2 k2-21

(2.)



(3.) FIND_PREFIX(root, k)

max_len=0 , len=1 , STACK_INIT(S) , next=root

loop

while (next ≠ null) do

PUSH(S, (next, len))

next = next.left , len = len+1

end_while

if (not STACK_EMPTY(S)) then

next, len = POP(S)

if (next.info = EOF) then next.num = 1

else next.num = ADD_NUM(next)

end_if

if (next.num ≥ k and len > max_len) then max_len = len

end_if next = next.right

K2-21

AC12

(3) FIND_PREFIX(root, k)

```
    else return max_len end-if
end-loop
```

ADD_NUM(node)

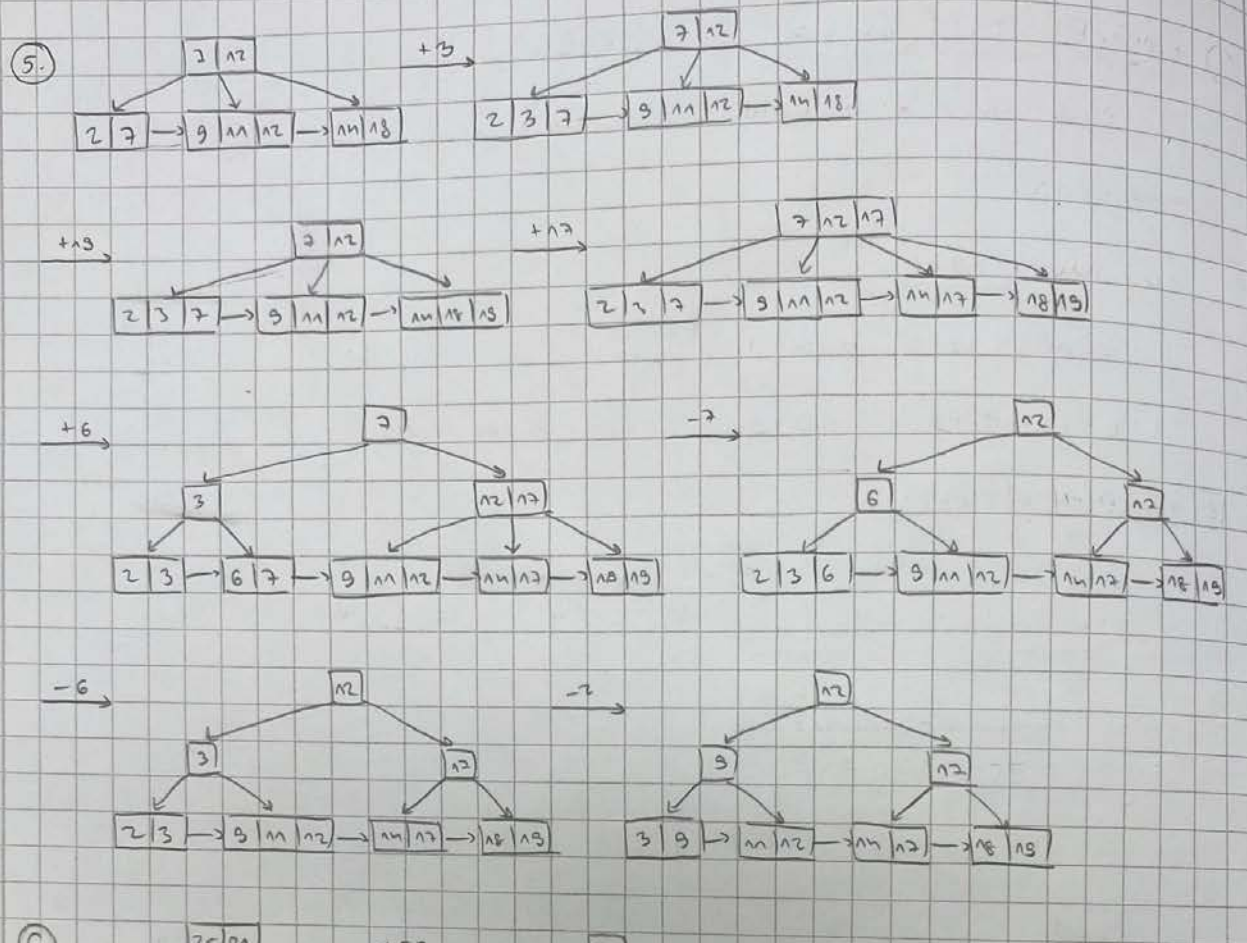
```
temp = node.left , num=0
while (temp ≠ null) do num = num + temp.num
end-while
return num
```

(4) B_RANGE(root, m, a, b)

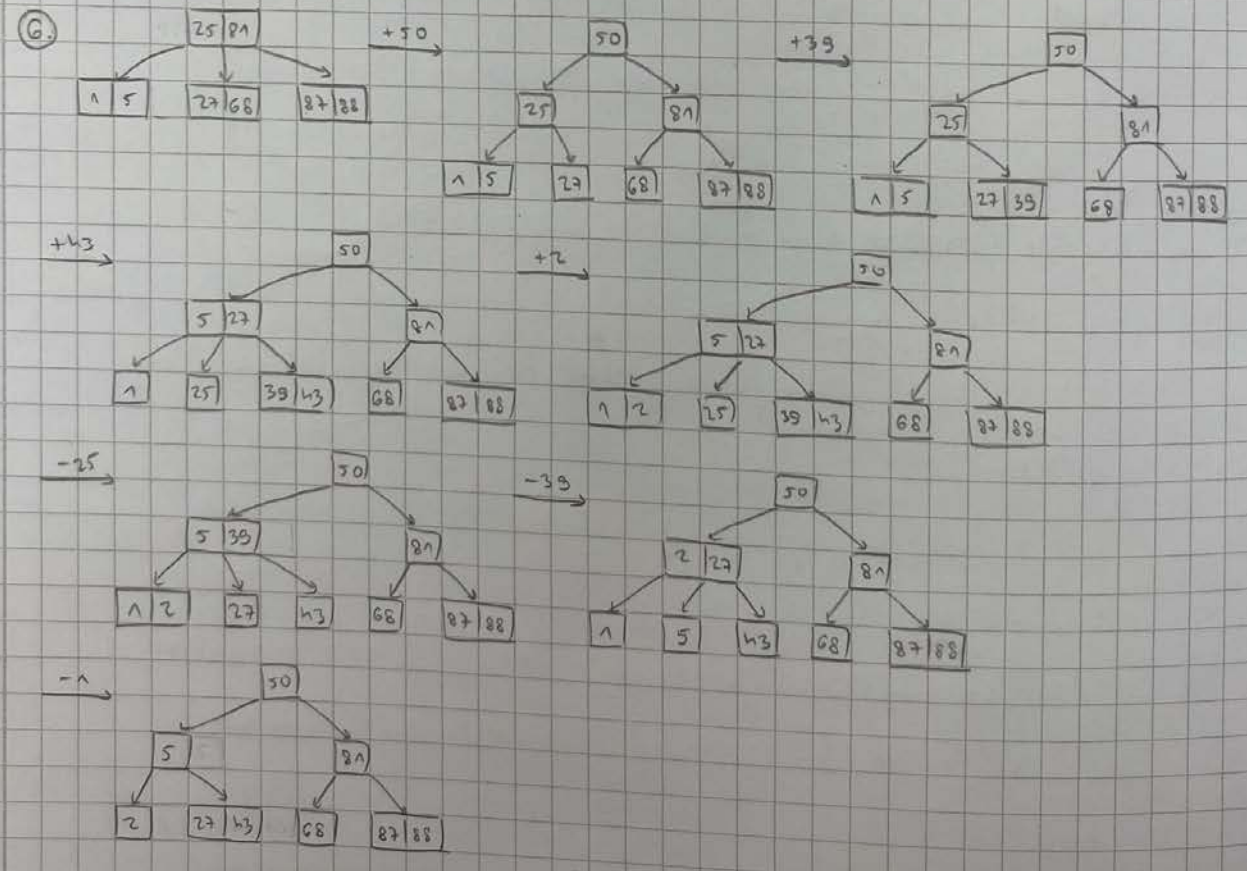
```
keys = NEW_ARRAY
if (root != null) then
    i = 1
    while (i ≤ root.num and root.keys[i] < a) do i = i+1
    end-while
    while (i ≤ root.num and root.keys[i] ≤ b) do
        if (root.keys[i] > a) then
            keys = keys + B_RANGE(root.childern[i], m, a, b)
        end-if
        keys = keys + root.keys[i]
    end-while
    if (root.keys[i-1] < b) then
        keys = keys + B_RANGE(root.childern[i], m, a, b)
    end-if
end-if
return keys
```

ACM2 (K2-21)

5.



6.



(7) клучеви од 0 до 99, хеш табела са 9 улаза

21, 12, 32, 66, 58 - значајно вероватни

h(k) = прва цифра (k+1)

- сваки од кључева 21, 12, 32, 66, 58 се кодира на различито место, па не долази до међусобних колизија

1	2	3	4	5	6	7	8	9
12	21	32		58	66			

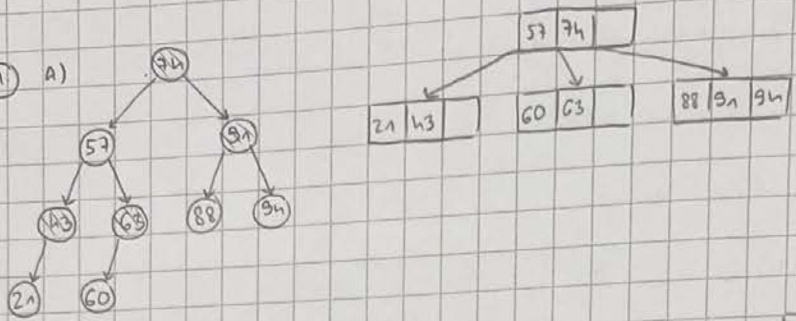
- могу се сместити и кључеви 45, 73, 80, 97

(8) BPLUS_DELETE_KEY(root, m, key)

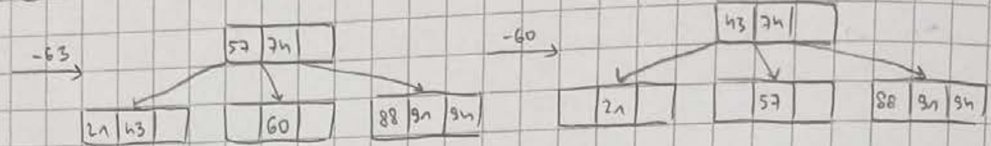
```
temp=root
while (not IS_LEAF(temp)) do
    i=1
    while (i≤temp.num and temp.keys[i]<key) do i=i+1
    end-while
    temp=temp.childern[i]
end-while
j=1
while (j≤temp.num and temp.keys[j]<key) do j=j+1
end-while
if (j=temp.num) then
    temp.parent.keys[i]=temp.keys[j-1]
else
    for k in range j to temp.num-1
        temp.keys[k]=temp.keys[k+1]
    end-for
end-if   temp.num=temp.num-1
```

АСП2 КЗ-19

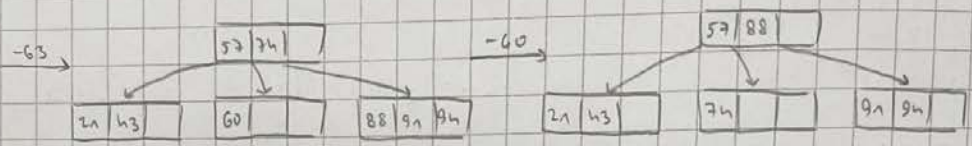
① А)



б)

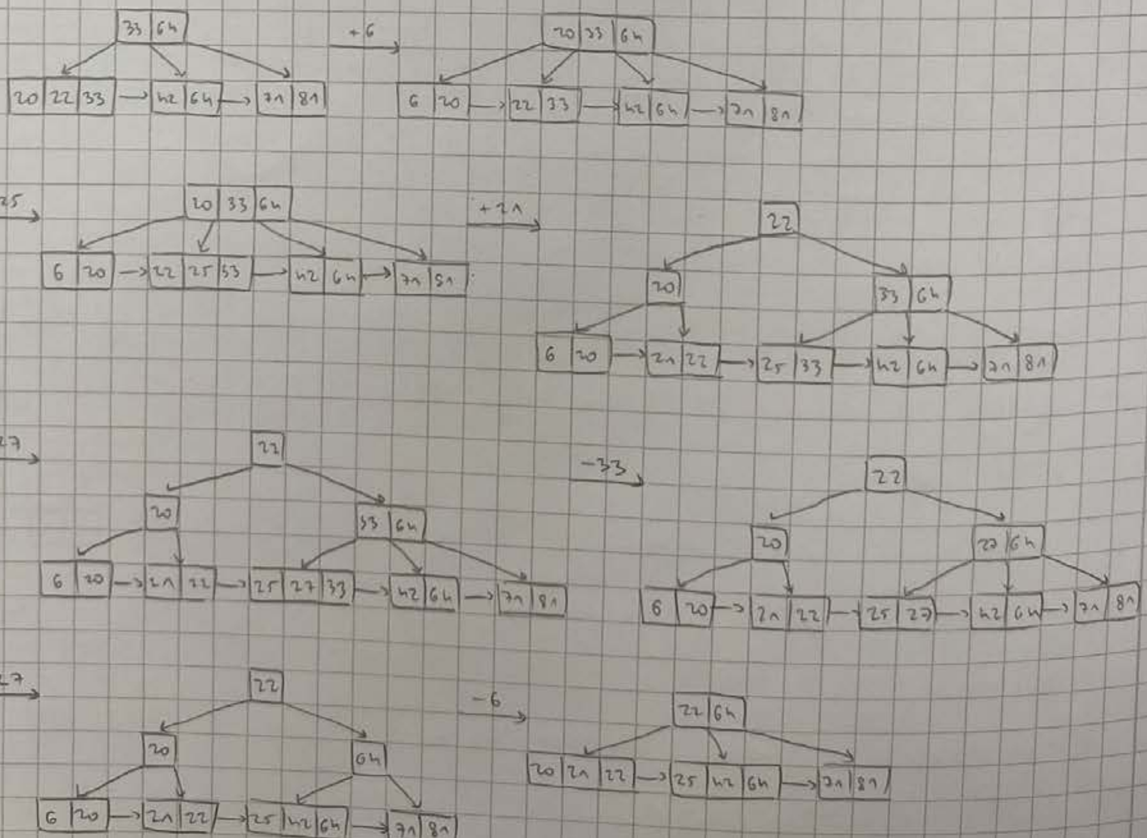


в)

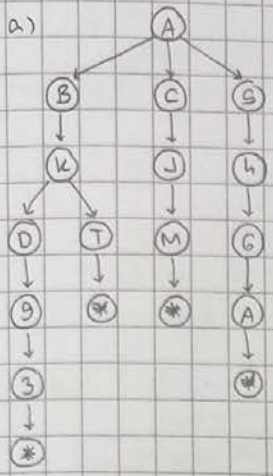


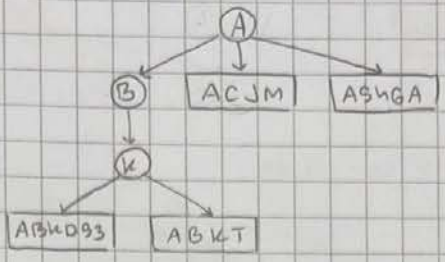
- било би разлике у коначном изгледу стабла, јер би се сви чворови попуњавали са десна на лево и брисања би се позајмица од десног брата (док што у 2-3-4 стаблу није „стварни брат")

②



3. а)





- две јединствене путање од неког чвора до листа се могу смештити у један посебан чвор, који садржи целокупан низ и показивач на запис, па се његова структура разликује од структуре чвора који се гранa.

б) TRIE-COMPACTION(root)

ACN2 | K2-19

(h) DELETE(root, m, key)

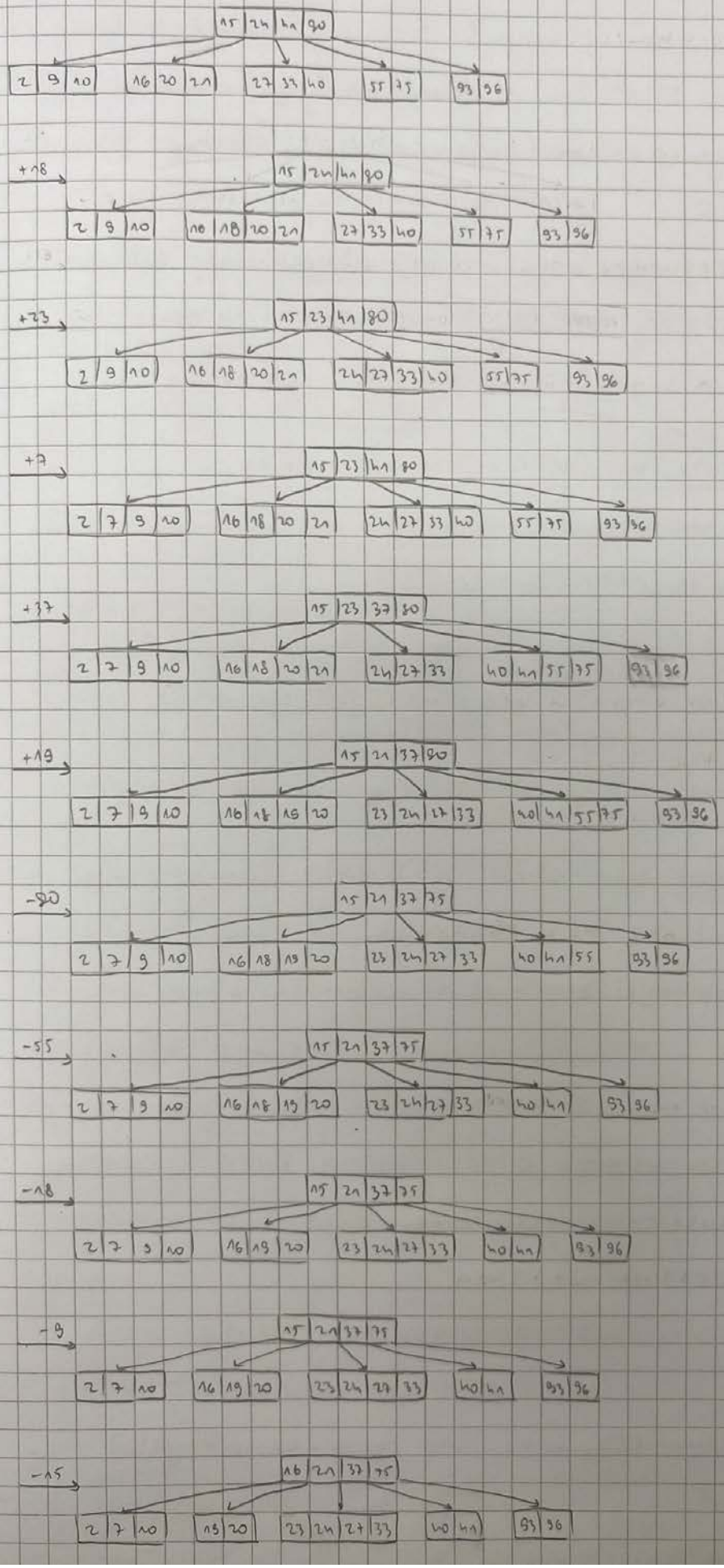
```
temp=root
while (not IS_LEAF(temp)) do
    i=1
    while (i≤temp.num and temp.keys[i]<key) do i=i+1
    end_while
    if (i≤temp.num and temp.keys[i]=key) then break
    else temp=temp.children[i]
    end_if
end_while
DELETE_FROM_NODE(temp, i)
```

DELETE_FROM_NODE(node, i)

```
if (IS_LEAF(node)) then SHIFT(node,i), node.num=node.num-1
    SHIFT(node,i)
    node.num=node.num-1
    if (node.num=0) then DELETE(node) end_if
else
    temp=SUCC(node,i), j=0
    if (temp=node) then temp=PRED(node,i), j=temp.num
    end_if
    node.keys[i]=temp.keys[j]
    DELETE_FROM_NODE(temp,j)
end_if
```

K2-19

5.


15 24 41 80
2 9 10
16 20 21
27 33 40
55 75
93 96
+18
15 24 41 80
2 9 10
16 18 20 21
27 33 40
55 75
93 96
+23
15 23 41 80
2 9 10
16 18 20 21
24 27 33 40
55 75
93 96
+7
15 23 41 80
2 7 9 10
16 18 20 21
24 27 33 40
55 75
93 96
+37
15 23 37 80
2 7 9 10
16 18 20 21
24 27 33
40 41 55 75
93 96
+19
15 21 37 80
2 7 9 10
16 18 19 20
23 24 27 33
40 41 55 75
93 96
-80
15 21 37 75
2 7 9 10
16 18 19 20
23 24 27 33
40 41 55
93 96
-55
15 21 37 75
2 7 9 10
16 18 19 20
23 24 27 33
40 41
93 96
-18
15 21 37 75
2 7 9 10
16 19 20
23 24 27 33
40 41
93 96
-9
15 21 37 75
2 7 10
16 19 20
23 24 27 33
40 41
93 96
-15
16 21 37 75
2 7 10
19 20
23 24 27 33
40 41
93 96

АСП2 К2-19

(6) а) - B^+ стабло чији су кључеви године уписа студената, док је у листу B^+ стабла уз сваку годину уписа налази уланчана листа свих студената који су те године уписали факултет

- оваква структура ефикасно подржава претрагу прве године и секвенцијалну претрагу осталих године до краја траженог периода, а издвајање тражених студената

б) FIND-STUDENTS-IN-RANGE(structure, year1, year2)

```
temp=structure.root
while (not IS-LEAF(temp)) do
    i=0
    while(i≤temp.num and temp.keys[i]<year1) do  i=i+1
    end-while
    temp=temp.childern[i]
end-while
i=1   students=NEW-ARRAY
while (temp.keys[i]<year1) do  i=i+1
end-while
year=temp.keys[i]
while (year≤year2) do
    students=students+temp.info[i]
    if (i<temp.num) then
        i=i+1, year=temp.keys[i]
    else
        temp=temp.next, i=0, year=temp.keys[i]
    end-if
end-while  return students
```

(7) В стабло реда m

256 бита запис, 32 бита кључ

64 бита адресе, 2KB величина блока на диску

- у чвору само кључ:

величина чвора: $\underbrace{(m-1)\cdot 32}_{\text{кључеви}} + \underbrace{(m-1)\cdot 64}_{\text{&записа показивачи}} + \underbrace{m\cdot 64}_{\text{показивачи}} = 160m - 96$ бита

$2000 \cdot 8 \geq 160m - 96$

$m \leq \frac{16000+96}{160} \Rightarrow \underline{m \leq 100}$

- у чвору читав податак:

величина чвора: $\underbrace{(m-1)(32+256)}_{\text{подаци}} + \underbrace{m\cdot 64}_{\text{показивачи}} = 352m - 288$ бита

$2000 \cdot 8 \geq 352m - 288$

$m \leq \frac{16000+288}{352} \Rightarrow \underline{m \leq 46}$

- ако се у чвору смешта читав податак, ред стабла је мањи и нешто ће доказити до скупих операција преламања и спајања.

(8) а) - метод дељења: $h(k) = k \% n$, n - величина табеле

- једноставност хеш-функције

- за n треба избегавати парне бројеве и степене бројева 2 и 10

б) $n = 3$, кључеви 19, 10, 4, 37, 49, 52

$h(19) = 1$, $h(10) = 1$, $h(4) = 4$, $h(37) = 1$, $h(49) = 4$, $h(52) = 8$

- велика вероватноћа колизија: {19, 10, 37} и {4, 49}

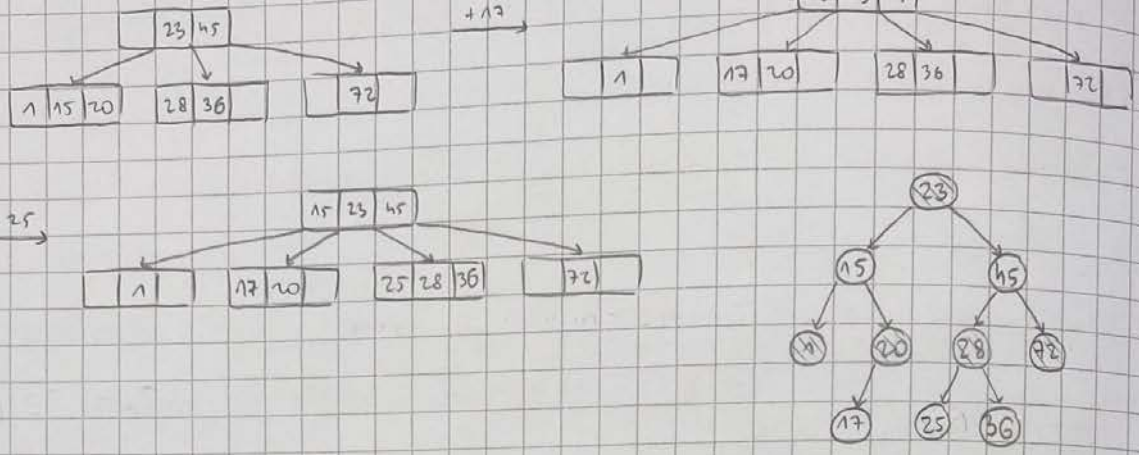
$n = 8$

$h(19) = 3$, $h(10) = 2$, $h(4) = 4$, $h(37) = 5$, $h(49) = 1$, $h(52) = 4$

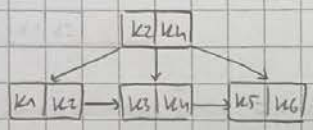
- мања вероватноћа колизија: {4, 52}

АСП2 | К2-18 |

1.

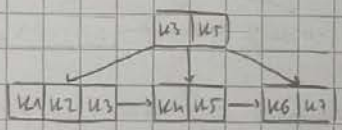


2. а) B^+ стабло, $m=3$, $h=1$



- Просечан број приступа приликом успешне претраге: 2
- Просечан број приступа приликом неуспешне претраге: 2

б) B^+ стабло, $m=4$, $n=7$



- Просечан број приступа приликом успешне претраге: 2
- Просечан број приступа приликом неуспешне претраге: 2

8. а) - Минимална савршена хеш функција је хеш функција при којој са n кључева у табели са n чворова не долази до колизија

б) кључеви 54, 91, 57, 23, 28 $\Rightarrow n=5$

$H(K) = (K \% 10) / 2$

0	1	2	3	4
91	23	54	57	28

③ CALC (root)

```
max = 0 , pow = 0 , temp = root
loop
    i = 9
    while (i ≥ 0 and temp[i] = null) do i = i - 1
    end_while
    if (i = -1) then break
    end_if
    max = i · 10^pow + max , pow = pow + 1
    temp = temp[i]
end_loop
min = 0 , pow = 0 , temp = root
loop
    if (temp[10] != null) then break
    end_if
    i = 0
    while (i ≤ 9 and temp[i] = null) do i = i + 1
    end_while
    if (i = 10) then break
    end_if
    min = i · 10^pow + min , pow = pow + 1
    temp = temp[i]
end_loop
return max - min
```

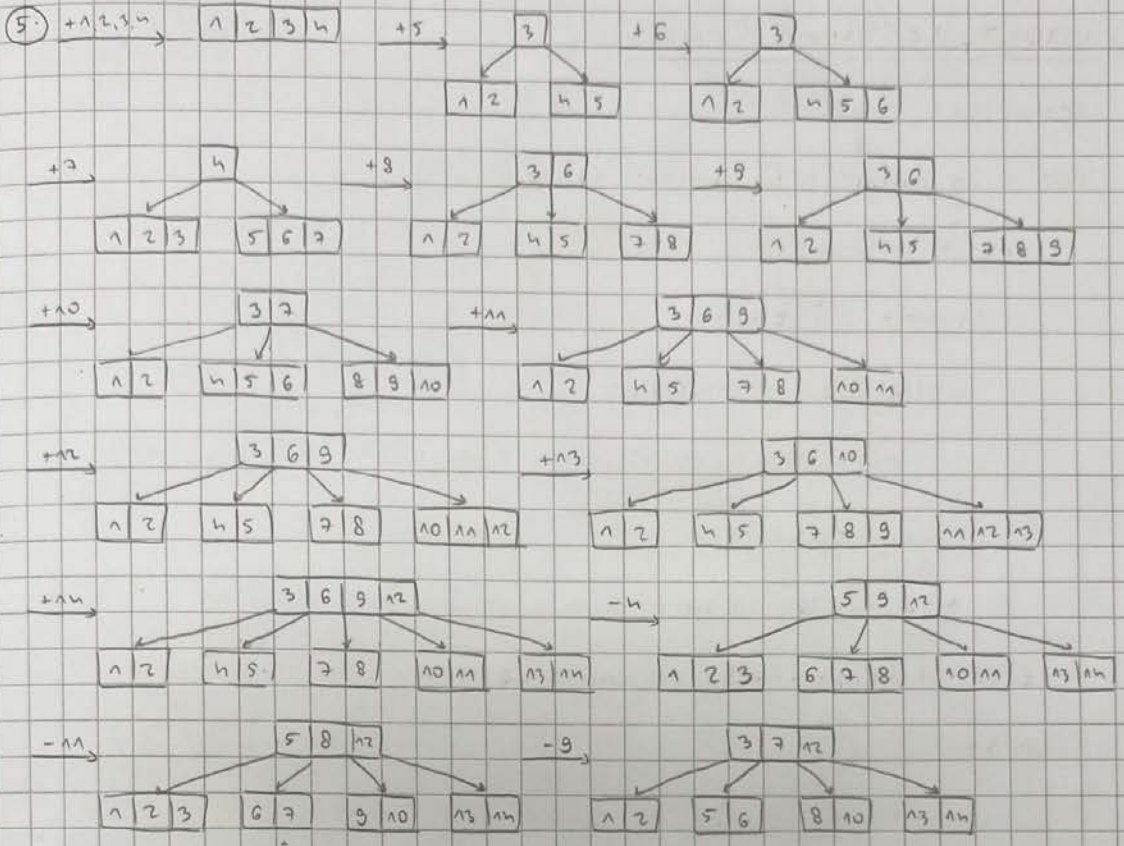
ACM2 HW2-18

(4) B_DELETE_NO_BORROW(root, m, node, key)

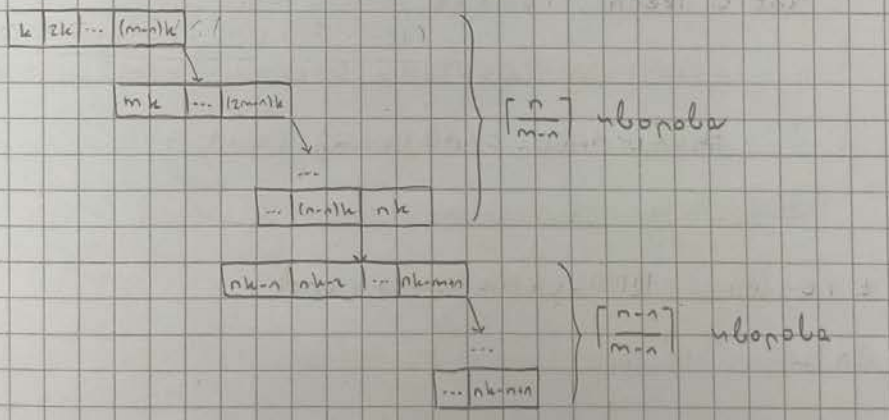
```
i=1
while (node.keys[i] < key) do i=i+1
end_while
for j in range i to node.num-1 do
    node.keys[j] = node.keys[j+1]
end_for
node.num = node.num-1
parent = node.parent , i=1
while (parent.childern[i] != node) do i=i+1
end_while
if (i = parent.num+1) then
    left = parent.childern[i-1], right = node , i=i-1
else
    left = node , right = parent.childern[i+1]
end_if
size = left.num+1 , left.keys[size] = parent.keys[i]
for j in range 1 to right.num do
    size = size+1 , left.keys[size] = right[j]
end_for
left.num = size , DELETE(right)
for j in range i to parent.num-1 do
    parent.keys[j] = parent.keys[j+1]
    parent.childern[j+1] = parent.childern[j+2]
end_for
parent.num = parent.num-1
```

К2-18

5.



7.



$$\Rightarrow h = \left\lceil \frac{n}{m-1} \right\rceil + \left\lceil \frac{n-1}{m-1} \right\rceil - 1 \quad , \; n, k \gg m$$

ACN2 | K2-18 |

⑥ COUNT_KEYS(root, format)

```
temp=root , num=0
for i in range 1 to len(format) do
    char = format[i]
    if (char = ".") then
        while (temp != null) do
            num=num + COUNT_KEYS (temp.left , format[(i+1):])
            temp= temp.right
        end_while return num
    else if (char = "*") then continue
    else
        if (format[i+1] = "*") then
            while (temp != null) do
                num=num + COUNT_KEYS (temp, format[(i+2):])
                temp= FIND_CHAR(temp, char)
            end_while return num
        else temp= FIND_CHAR (temp, char)
        end_if
    end_if
    if (temp = null) then return num end_if
end_for
if (temp = EOF) then num=num+1 end_if return num
```

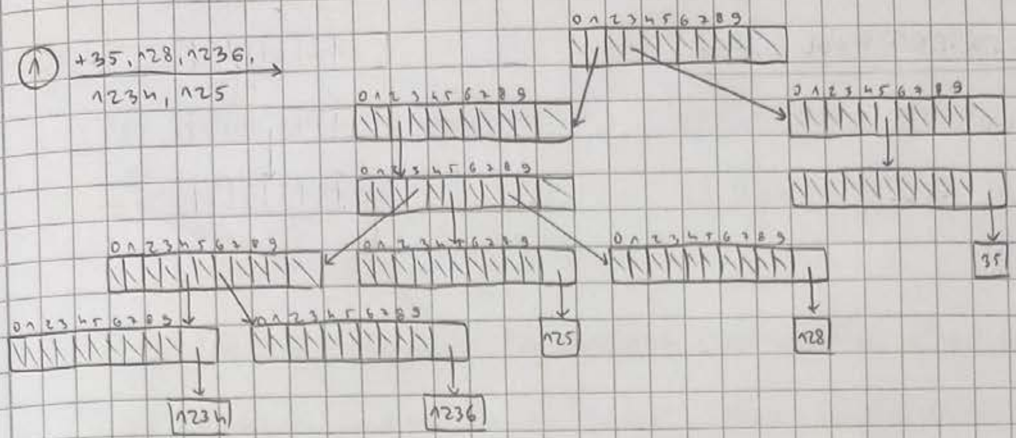
FIND_CHAR(temp, char)

```
while ( temp and temp.info < char) do temp=temp.right end_while
if ( temp and temp.info = char ) then return temp.left
else return null
end_if
```

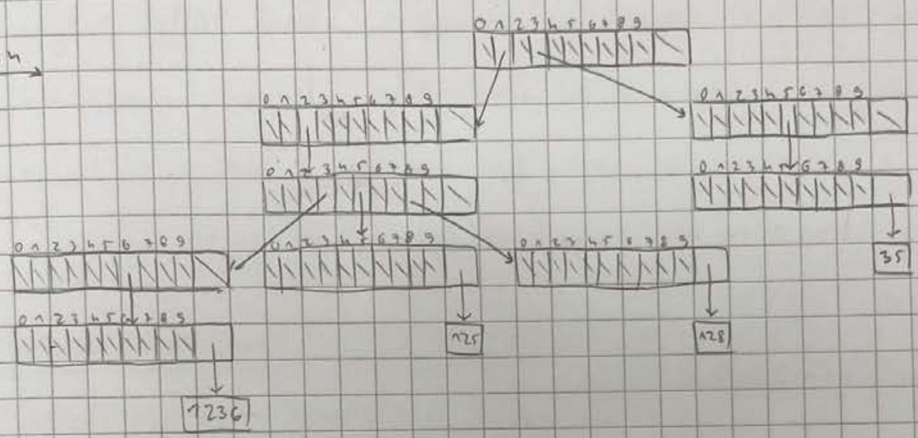
K2-17

АСП2

① +35, 128, 1236, 1234, 125



-1234



② В стабло реда m

број приступа подацима

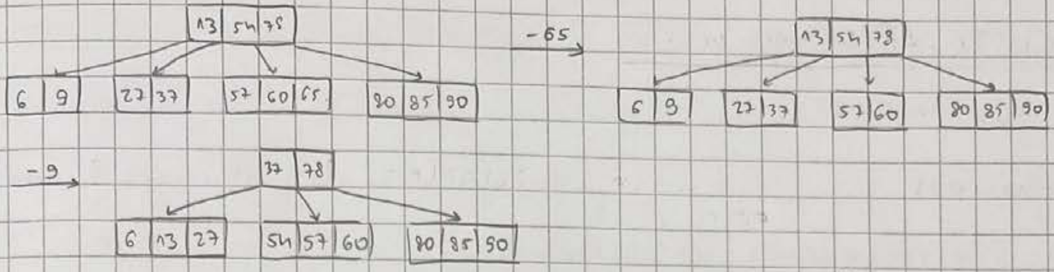
ACM2 K2-17

③ B_PLUS_INSERT (root, m, key)

```
temp = root
while (not IS_LEAF(temp)) do
    i = 1
    while (i ≤ temp.num and temp.keys[i] < key) do i = i+1
    end_while
    temp = temp.children[i]
end_while
j = 1
while ( j ≤ temp.num and temp.keys[j] < key ) do j = j+1
end_while
if ( j ≤ temp.num) then
    for k in range temp.num+1 down to j+1 do
        temp.keys[k] = temp.keys[k-1]
    end_for
else UPDATE_PARENT(temp, key) // temp.parent.keys[i] = key
end_if
temp.keys[j] = key , temp.num = temp.num+1
if (temp.num > m-1) then
    node = NEW_NODE(m) , mid = temp.num/2
    for k in range 1 to temp.num - mid do
        node.keys[k] = temp.keys[mid+k] , node.num = node.num+1
    end_for
    temp.num = mid , node.parent = temp.parent
    SHIFT (temp.parent , i)
    temp.parent.keys[i] = temp.keys[mid]
    temp.parent.children[i+1] = node
```

К2-17

4.



- при брисању кључева 57|60|65, а затим 6|9, чвор не може да позајми кључ ни од десног, левог, другог десног ни другог левог брата, па долази до спајања чворова 3-у-2
- преостали кључеви у 3 чвора и 2 раздвојна кључа се поређају у низ, средњи иде у родитељски чвор, а преостали кључеви се распоређују у 2 чвора

5. TRANSFORM(root)

```
QUEUE_INIT(Q), INSERT(Q, (root, null))
while (not QUEUE_EMPTY(Q)) do
    node, parent = DELETE(Q)
    if (node.color = black) then
        new = NEW_NODE, new.keys[2] = node.key
        if (parent != null) then CONNECT(parent, new)
        else new-root = new
        end_if
    else ADD(parent, node.key), new = parent
    end-if
    if (node.left) then INSERT(Q, (node.left, new)) end_if
    if (node.right) then INSERT(Q(node.right, new)) end-if
end-while
return new-root
```

ACN2 K2-17

(c) DELETE_KEYS(root, prefix)

```
STACK_INIT(S)
temp=root
for i in range 1 to len(prefix) do
    char=prefix[i]
    while (temp≠null and temp.info<char) do temp=temp.right
    end_while
    if (temp=null or temp.info>char) then return
    end_if
    PUSH(S, temp), temp=temp.left
end_for
DELETE_FROM_ROOT(temp)
temp=POP(s), parent=POP(s)
while (not STACK_EMPTY(S) and parent.left=temp and temp.right=null) do
    DELETE(temp)
    temp=parent, parent=POP(S)
end_while
start=parent.left
if (start=temp) then
    parent.left=temp.right, DELETE(temp)
else
    while(start.right!=temp) do start=start.right
    end_while
    start.right=temp.right, DELETE(temp)
end_if
```

[К2-17]

⑦ m-арно стабло, број чворова n

а) - максимална висина: минимално попуњено

$$h_{max} = \left\lceil \frac{n}{m-1} \right\rceil - 1$$

- минимална висина: максимално попуњено

$$n \leq (m-1)(1 + m + m^2 + \ldots + m^h) = (m-1)\frac{m^{h+1}-1}{m-1} = m^{h+1} - 1$$

$$h_{min} = \lceil \log_m (n+1) \rceil - 1$$

б) - максималан број листова: максимално попуњено

$$b_{max} = m\left[\frac{n}{2(m-1)}\right] + \left\lceil \left(n - m\left[\frac{n}{2(m-1)}\right]\right) / (m-1) \right\rceil + \ldots$$

- минималан број листова: минимално попуњено

$$b_{min} = \left\lceil \frac{n}{m-1} \right\rceil$$

⑧ 100 уноса, метод конверзије основе

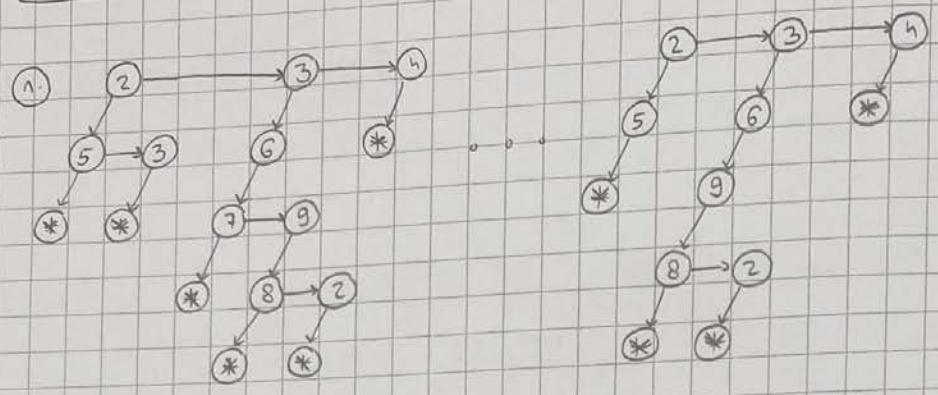
а) H(key)

```
t = 0, pow = 0
while (key > 0) do
    t = t + (key % 10) · q^pow
    key = key / 10, pow = pow + 1
end-while
return t % 100
```

б) EQ_CLASS_NUM(K, n)

```
num = 0, arr = NEW-ARRAY(100)
for i in range 1 to n do
    code = H(K[i])
    if (arr[code] = 0) then
        arr[code] = 1, num = num + 1
    end-if
end-for
return num
```

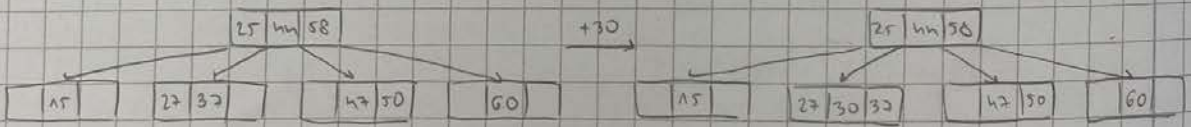
АСП2 **К2-16**

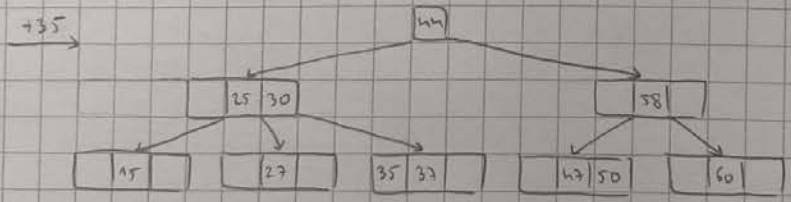


② а) – уметање у дун чвор 2-3-н стабла се врши тако што се нови кључ додаје у родитеља, чвор се дели, и остала два нулвена чвора постају нули у новим чворовима, а нови кључ долази на одговарајућу позицију као нулвени у једном од новог чвора

– постоје разлике у односу на класично B стабло због позиције чворова и тога да се у родитеља увек додаје претходни нули чвор, а не средњи у новодобијеном низу.

б)





3. B_DEL_BORROW(node, m)

```
parent=node.parent
i=1
while (parent.childern[i] ≠ node) do i=i+1
end_while
if (i ≤ parent.num) then right=parent.childern[i+1] end_if
if (i>1) then left=parent.childern[i-1] end_if
if (right and right.num > ⌈m/2⌉-1) then
    node.num=node.num+1
    node.keys[node.num]=parent.keys[i]
    parent.keys[i]=right.keys[1]
    SHIFT_L(right), right.num=right.num-1
else if (left and left.num > ⌈m/2⌉-1) then
    node.num=node.num+1, SHIFT_R(node)
    node.keys[1]=parent.keys[i-1]
    parent.keys[i-1]=left.keys[left.num]
    left.num=left.num-1
else print("Nemoguca pozajmica")
end_if
```