ალგორითმები და მონაცემთა სტრუქტურები

ბინარული ძებნა.

ზ. კუჭავა, LaTeX

1 ამოცანა:

[1]58გვ:72ეგვ, [2]409-417გვ:419-427ეგვ, [3]46-51გვ:64-69ეგვ,[4]333-337გვ, [5]56-59გვ:75-78ეგვ, [6]131-132ეგვ, [7]120-125გვ:137-142ეგვ, [10]274-278გვ, [8]67გვ, [9]395-398გვ, [11]58გვ, [13]700გვ

2 ფსევდოკოდი

[1]58გვ:72ეგვ 1.

სურ. 1: K&R

Listing 1: C code

```
low = 0;
high = n - 1;
while(low <= high)
{
    mid = (low+high) / 2;
    if(x < v[mid])
        high = mid - 1;
    else if(x > v[mid])
        low = mid + 1;
    else         /* found match */
        return mid;
}
return -1; /* no match */
```

Listing 2: Assembler code

```
binsearch:
movl    %edi, -20(%rbp)
movq    %rsi, -32(%rbp)
movl    %edx, -24(%rbp)
movl    $0, -4(%rbp)
movl    -24(%rbp), %eax
subl    $1, %eax
movl    %eax, -8(%rbp)
jmp     .L6
.L10:
movl    -4(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
movl    %eax, %edx
shrl    $31, %edx
addl    %edx, %eax
sarl    %eax
movl    %eax, -12(%rbp)
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rdx
movq    -32(%rbp), %rax
addq    %rdx, %rax
movl    (%rax), %eax
cmpl        %eax, -20(%rbp)
jge     .L7
movl    -12(%rbp), %eax
subl    $1, %eax
movl    %eax, -8(%rbp)
jmp     .L6
.L7:
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rdx
movq    -32(%rbp), %rax
addq    %rdx, %rax
movl    (%rax), %eax
cmpl        %eax, -20(%rbp)
jle     .L8
movl    -12(%rbp), %eax
addl    $1, %eax
movl    %eax, -4(%rbp)
jmp     .L6
.L8:
movl    -12(%rbp), %eax
jmp     .L9
.L6:
movl    -4(%rbp), %eax
cmpl        -8(%rbp), %eax
jle     .L10
movl    $-1, %eax
.L9:
popq    %rbp
```

while ციკლში შედარებების ოპტიმიზაცია
2.

სურ. 2: ”ოპტიმიზაცია”

Listing 3: C code

```
low = 0;
high = n - 1;
mid = (low + high)/2;
while(low <= high && x != v[mid])
{
    if( x > v[mid])
        low = mid+1;
    else
        high = mid -1;
    mid = ( low + high)/2;
}
if(x==v[mid])
    return mid; /* found match */
else
    return -1;  /* no match */
```

Listing 4: Assembler code

```
binsearch:
movl    %edi, -20(%rbp)
movq    %rsi, -32(%rbp)
movl    %edx, -24(%rbp)
movl    $0, -4(%rbp)
movl    -24(%rbp), %eax
subl    $1, %eax
movl    %eax, -8(%rbp)
movl    -4(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
movl    %eax, %edx
shrl    $31, %edx
addl    %edx, %eax
sarl    %eax
movl    %eax, -12(%rbp)
jmp     .L6
.L10:
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rdx
movq    -32(%rbp), %rax
addq    %rdx, %rax
movl    (%rax), %eax
cmpl        %eax, -20(%rbp)
jle     .L7
movl    -12(%rbp), %eax
addl    $1, %eax
movl    %eax, -4(%rbp)
jmp     .L8
.L7:
movl    -12(%rbp), %eax
subl    $1, %eax
movl    %eax, -8(%rbp)
.L8:
movl    -4(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
movl    %eax, %edx
shrl    $31, %edx
addl    %edx, %eax
sarl    %eax
movl    %eax, -12(%rbp)
.L6:
movl    -4(%rbp), %eax
cmpl        -8(%rbp), %eax
jge     .L9
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rdx
movq    -32(%rbp), %rax
addq    %rdx, %rax
movl    (%rax), %eax
cmpl        %eax, -20(%rbp)
jne     .L10
.L9:
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rdx
movq    -32(%rbp), %rax
addq    %rdx, %rax
movl    (%rax), %eax
cmpl        %eax, -20(%rbp)
jne     .L11
movl    -12(%rbp), %eax
jmp     .L12
.L11:
movl    $-1, %eax
.L12:
popq    %rbp
```

switch კონსტრუქციის გამოყენებით
3.

სურ. 3: Switch.

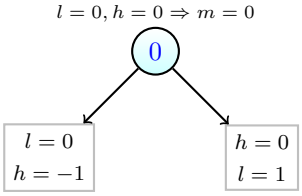
Listing 5: C code

```
low = 0;
high = n - 1;
while(low <= high)
{
    mid = (low+high) / 2;
    switch(((x - v[mid])>0) - ((x
        - v[mid])<0))
    {
        case (1) :
        low = mid + 1;
        break;
        case (-1) :
        high = mid - 1;
        break;
        default : /* found match
            */
        return mid;
    }
}
return -1; /* no match */
```

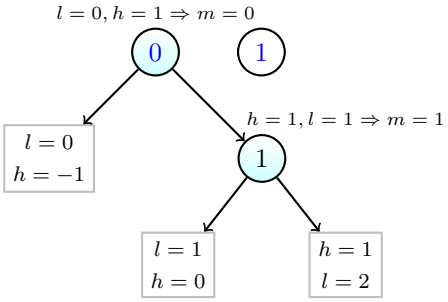
Listing 6: Assembler code

```
binsearch:
        movl    %edi, -20(%rbp)
movq    %rsi, -32(%rbp)
movl    %edx, -24(%rbp)
movl    $0, -4(%rbp)
movl    -24(%rbp), %eax
subl    $1, %eax
movl    %eax, -8(%rbp)
jmp     .L6
.L10:
movl    -4(%rbp), %edx
movl    -8(%rbp), %eax
addl    %edx, %eax
movl    %eax, %edx
shrl    $31, %edx
addl    %edx, %eax
sarl    %eax
movl    %eax, -12(%rbp)
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rdx
movq    -32(%rbp), %rax
addq    %rdx, %rax
movl    (%rax), %eax
movl    -20(%rbp), %edx
subl    %eax, %edx
movl    %edx, %eax
testl   %eax, %eax
setg    %al
movzbl  %al, %edx
movl    -12(%rbp), %eax
cltq
leaq    0(,%rax,4), %rcx
movq    -32(%rbp), %rax
addq    %rcx, %rax
movl    (%rax), %eax
movl    -20(%rbp), %ecx
subl    %eax, %ecx
movl    %ecx, %eax
shrl    $31, %eax
movzbl  %al, %eax
subl    %eax, %edx
movl    %edx, %eax
cmpl    $-1, %eax
je      .L7
cmpl    $1, %eax
jne     .L8
movl    -12(%rbp), %eax
addl    $1, %eax
movl    %eax, -4(%rbp)
jmp     .L6
.L7:
movl    -12(%rbp), %eax
subl    $1, %eax
movl    %eax, -8(%rbp)
jmp     .L6
.L8:
movl    -12(%rbp), %eax
jmp     .L9
.L6:
movl    -4(%rbp), %eax
cmpl    -8(%rbp), %eax
jle     .L10
movl    $-1, %eax
.L9:
popq    %rbp
```

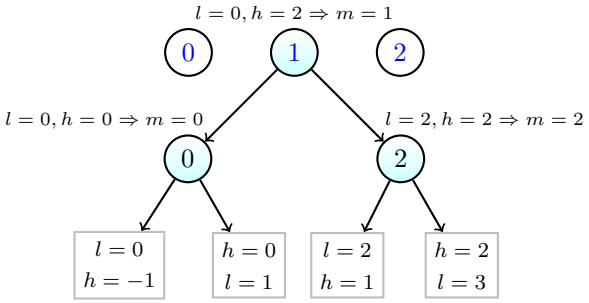
2.1 ბინარული ძებნის ხის შემთხვევები.



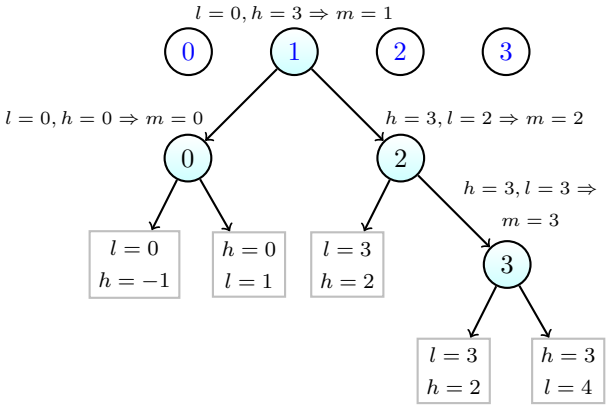
სურ. 4: N=1



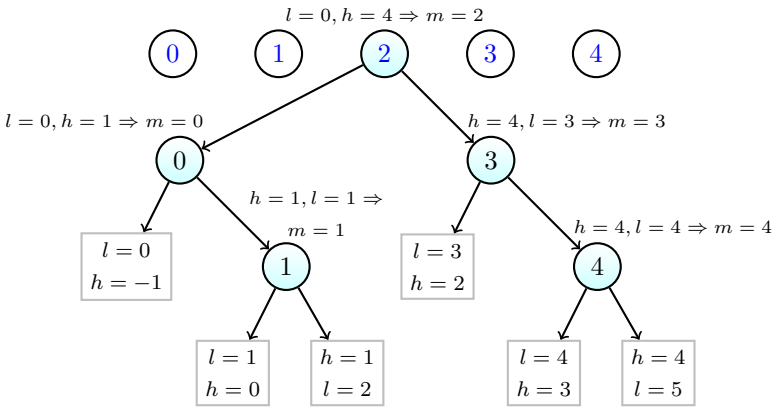
სურ. 5: N=2



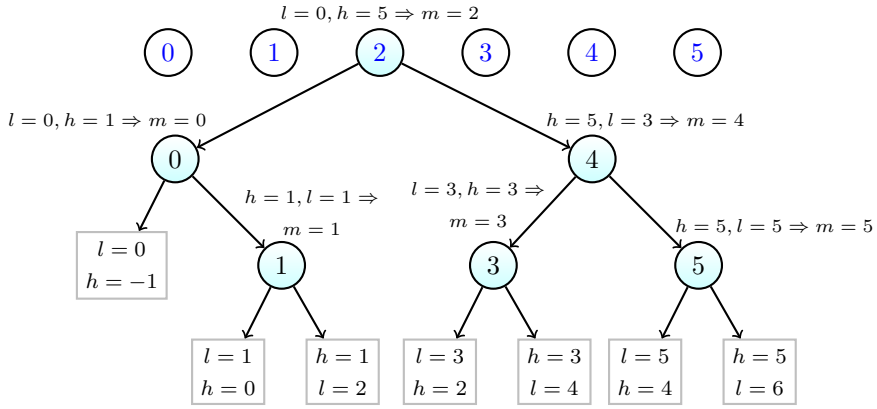
სურ. 6: N=3



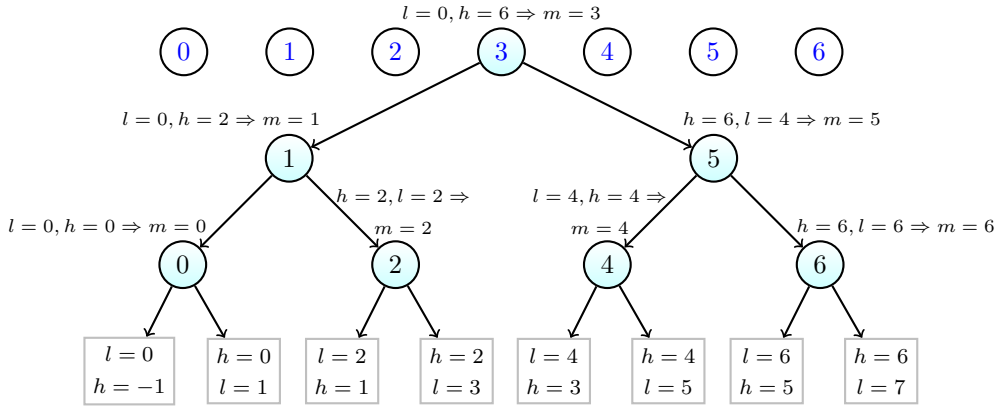
სურ. 7: N=4



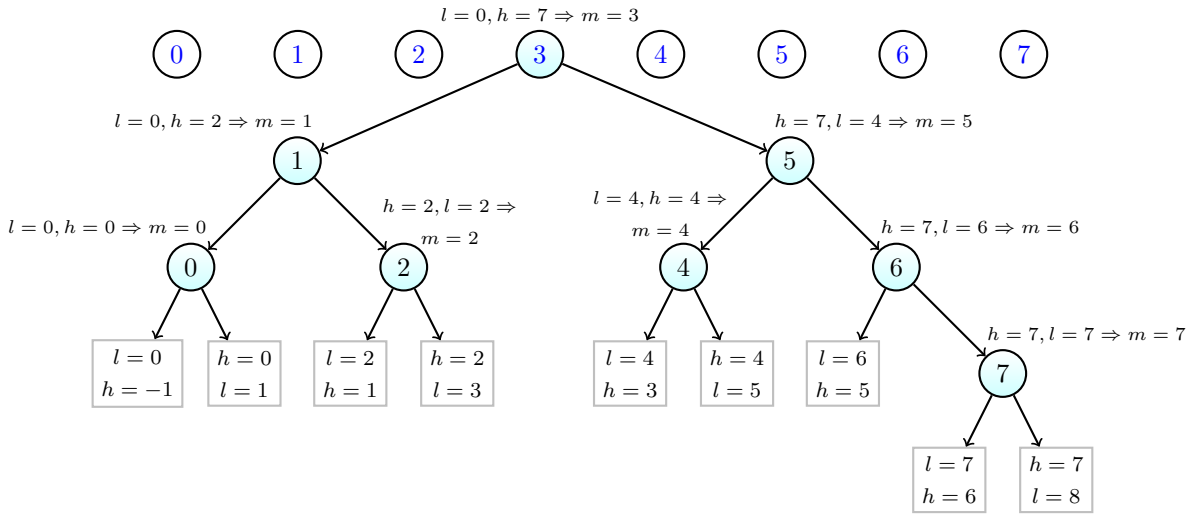
სურ. 8: N=5



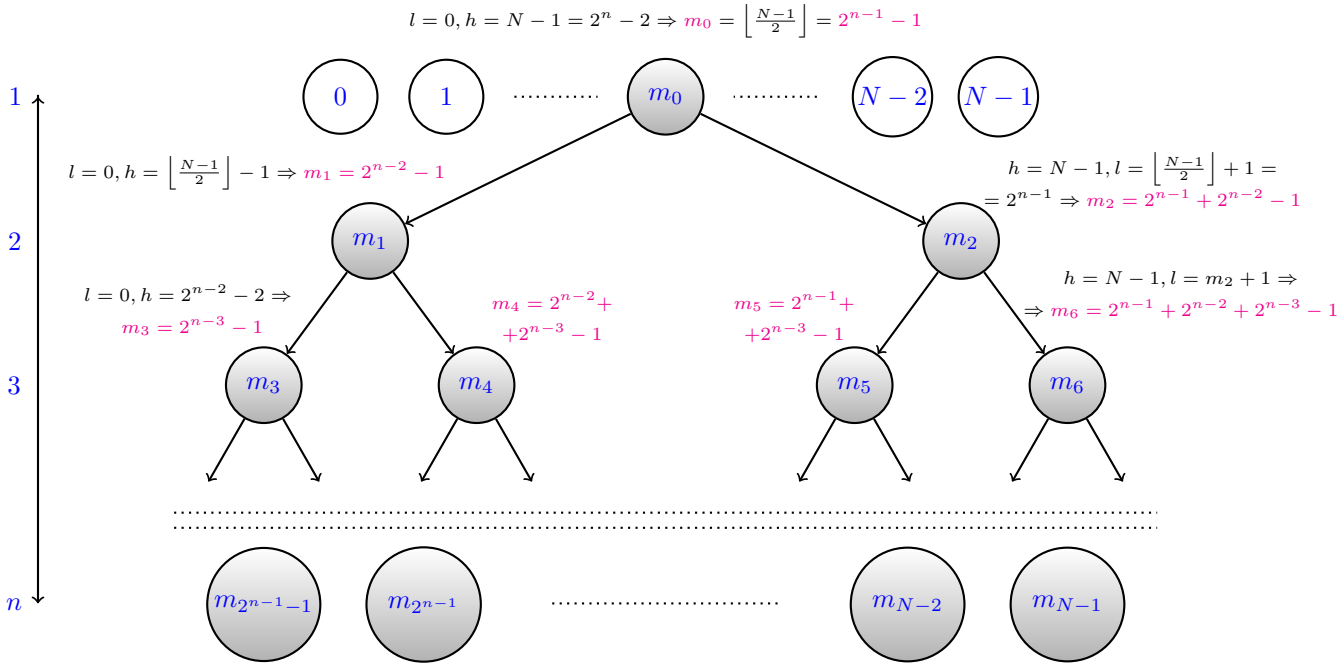
სურ. 9: N=6



სურ. 10: N=7



სურ. 11: N=8



სურ. 12: სრულყოფილი ხე

3 ანალიზი.

3.1 სრულყოფილი ხე.

ანალიზი ჩავატაროთ 1-ლი კოდისთვის შედარებათა რაოდენობის მიმართ. განვიხილოთ შემთხვევა როდესაც მასივის ელემენტთა რაოდენობა არის $N = 2^n - 1$, ანუ განსაზღვრავს სრულყოფილ (perfect [15]102pg:113epg, [14]) ხეს (12) , რომელსაც მაქსიმალურ სიღრმეზე n აქვს ყველა შესაძლო ფოთოლი, რაოდენობით 2^{n-1}.

3.1.1 საუკეთესო და უარესი შემთხვევები.

საუკეთესო შემთხვევა, ბუნებრივია, გვაქვს როდესაც პირველივე არჩეული ინდექსისთვის ვიპოვეთ საძიებელი ელემენტი. შედარებათა რაოდენობა ორია.
უარესია შემთხვევა როდესაც საძიებელი ელემენტი არა მოიძებნა, რადგან n სიღრმეზე მოხვედრის შემდეგ ციკლის პირობა $\text{low} \leqslant \text{high}$ აღმოჩნდება მცდარი.

3.1.2 საშუალო შემთხვევა.

3.1.2.1 მარტივი დათვლა. ხშირად მიღებულია ბინარული ძებნის საშუალო შემთხვევის გამარტივებული ვარიანტის განხილვა, როდესაც ვთვლით, რომ მასივის ელემენტში საძიებელი მნიშვნელობის პოვნას ჭირდება მხოლოდ ერთი შედარება.

3.1.2.1.1 საძიებელი მნიშვნელობა მასივშია. ჩავთვალოთ ჯერ, რომ საძიებელი მნიშვნელობა მასივშია, ანუ საძიებელი მნიშვნელობა აუცილებლად მოიძებნება ბინარული ძებნის ხის რომელიმე წვეროში ან ფოთოლში. თუ საძიებელი მნიშვნელობა ნაპოვნია პირველი სიღრმის დონეზე, მაშინ, ბუნებრივია ჩატარებული იქნება ერთი შედარება ამ დონეზე არსებული ერთადერთი კვანძისთვის. თუ საძიებელი მნიშვნელობა ნაპოვნია მეორე სიღრმის დონეზე, მაშინ, ჩატარებული იქნება ორი შედარება მეორე დონეზე არსებული ოთხივე კვანძისთვის. საძიებელი მნიშვნელობის k-ურ დონეზე პოვნის შემთხვევაში, ამ დონეზე განლაგებული ყოველი 2^{k-1} კვანძისთვის შესრულებული იქნება k რაოდენობის შედარება. თუ ჩავთვლით, რომ ყოველ წვეროში ან ფოთოლში საძიებელი მნიშვნელობის პოვნა თანაბარმოსალოდნელია და ტოლია $\frac{1}{N}$, მაშინ, T შედარებათა რაოდენობის საშუალო შემთხვევისთვის ვღებულობთ

$$ET = \sum_{k=1}^{n} \frac{k \cdot 2^{k-1}}{N} = \frac{1}{2N} \cdot \sum_{k=1}^{n} k \cdot 2^k = \frac{1}{2N} S_n \tag{1}$$

S_n–ის ფორმულის გამოსაყვანად განვიხილოთ შემდეგი გამოსახულება

$$\begin{aligned}
&\frac{1}{2}S_n = S_n - \frac{1}{2}S_n = \\
&= \left(1 \cdot 2^1 + 2 \cdot 2^2 + 3 \cdot 2^3 + \cdots + (n-1) \cdot 2^{n-1}\right) + n \cdot 2^n - \\
&- \left[1 \cdot 2^0 + \left(2 \cdot 2^1 + 3 \cdot 2^2 + \cdots + (n-1) \cdot 2^{n-2} + n \cdot 2^{n-1}\right)\right] = \\
&= n \cdot 2^n - 1 - \left[2^1 + 2^2 + \cdots + 2^{n-1}\right] = \\
&= n \cdot 2^n - 1 - 2\left[2^{n-1} - 1\right] = 2^n(n-1) + 1
\end{aligned}$$

რაც გვაძლევს

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

ამდენად, საშუალოსთვის ვღებულობთ

$$ET = \frac{2^n(n-1)+1}{N} = n - 1 + \frac{n}{2^n - 1} = n + O(1) = \log N + O(1) \tag{3}$$

3.1.2.1.2 საძიებელი მნიშვნელობის მასივში ყოფნა უცნობია. დავუშვათ, რომ საძიებელი მნიშვნელობის ყოფნა $N = 2^n - 1$ სიგრძის მქონე მასივში არ არის ცნობილი. ამით წინა განხილულ შემთხვევას, ანუ ბინარული ძებნის ხის კვანძებში ან ფოთლებში საძიებელი მნიშვნელობის პოვნას, ემატება კიდევ ვერ მოძებნის შემთხვევები, რომლებიც 1-ლი კოდისთვის წარმოდგენენ $n+1$ დონეზე გადასვლას. ვინაიდან ეს გადასვლა n დონეზე მყოფი ყოველი ფოთლისთვის შეიძლება ორი განსხვავებული გზით მოხდეს, ამიტომ ვერ მოძებნისთვის გვაქვს დამატებითი $2 \cdot 2^{n-1} = 2^n$ ვარიანტი. თუ ჩავთვლით, რომ ვერ მოძებნას, ისევე როგორც მოძებნას, ერთი შედარება ჭირდება, მაშინ, ფაქტიურად, ვღებულობთ წინა პუნქტში უკვე განხილულ შემთხვევას $n+1$ დონის მქონე სრულყოფილი ხისთვის და საშუალო ისევ 3 ფორმულით გამოითვლება:

$$ET = \frac{2^{n+1}n + 1}{N + 2^n} = n + \frac{n+1}{2^{n+1} - 1} = n + O(1) = \log N + O(1) \tag{4}$$

3.1.2.2 ზოგადი დათვლა. როგორც 1 კოდის შესაბამისი ასემბლერის კოდიდან ჩანს, while ციკლის ყოველი იტერაციისთვის სრულდება ერთი შედარება (ასემბლერის cmpl ინსტრუქცია, მონიშნულია წითელი ფერით), თუ მომდევნო ნაბიჯი მარცხნივ არის გადასადგმელი და ორი შედარება თუ მომდევნო ნაბიჯი მარჯვნივ არის გადასადგმელი ან თუ საძიებელი ელემენტი ნაპოვნია. თავად while ციკლს ემსახურება კიდევ ერთი შედარება. ამდენად ბინარული საძიებელი ხის 12 ყოველი წვეროსთვის სრულდება 3 შედარება, თუ საძიებელი ელემენტი ამ წვეროშია ნაპოვნი და სრულდება 2 ან 3 შედარება, თუ საძიებელი ელემენტი ამ წვეროში არაა ნაპოვნი (დათვლილია ციკლის იტერაციისთვის საჭირო შედარებაც).

ლიტერატურა

[1] Brian W. Kernighan, Dennis M. Ritchie, *The C Programming Language*, Second Edition

[2] Donald Ervin Knuth, *The Art of Computer Programming*, Volume 3, Second Edition

[3] Jeffrey J. McConnell, *Analysis of Algorithms: an Active Learning Approach* , 2001

[4] Loudon K. *Mastering Algorithms with C*, 1999

[5] Robert Sedgewick, *Algorithms in C (3rd ed.) Parts 1-4 Fundamentals, Data Structures, Sorting, Searching*, 1998

[6] Hemant Jain, *Problem Solving in Data Structures and Algorithms Using C*,

[7] Manber Udi, *Introduction Introduction to Algorithms A Creative Approach*, 1989

[8] Weiss, Mark Allen, *DataStructure and algorithm analysis in C++*, 2014

[9] Michael T. Goodrich, Roberto Tamassia, David M. Mount, *Data Structures and Algorithms in C++*,Wiley, 2011

[10] Paul Deitel, Harvey Deitel, *C How to Program with an introduction to C++*, 8-th Edition

[11] Stefan Hougardy, Jens Vygen, *Algorithmic Mathematics*, Springer International Publishing, 2016

[12] n1570-C11.pdf (ISO/IEC 9899:2011)

[13] Thomas H. Cormen Charles E. Leiserson Ronald L. Rivest Clifford Stein , *Introduction to Algorithms*, Third Edition

[14] wiki - Types of binary trees.

[15] Gayle Laakmann Mcdowell, *CRACKING the CODING INTERVIEW* , 6th Edition, 2015