

Puuuurimine

Targo Tennisberg

Aprill 2014

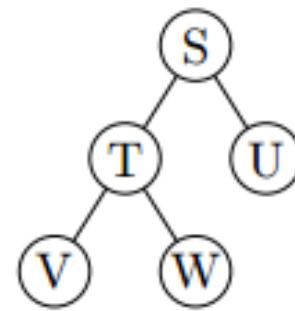
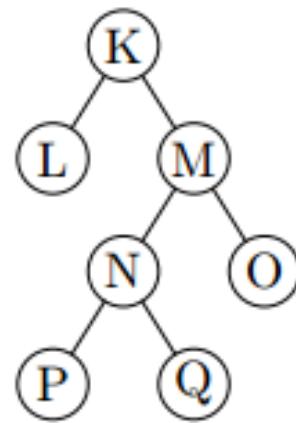
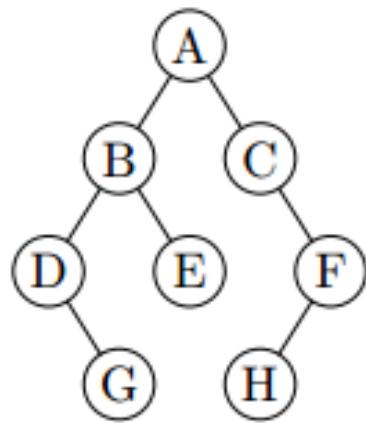
<http://www.targotennisberg.com/eio>

<http://www.targotennisberg.com/tarkvara>

Kava

- Kahendpuu
- Kahendotsimise puu
- Fenwicki puu
- Lõikude puu

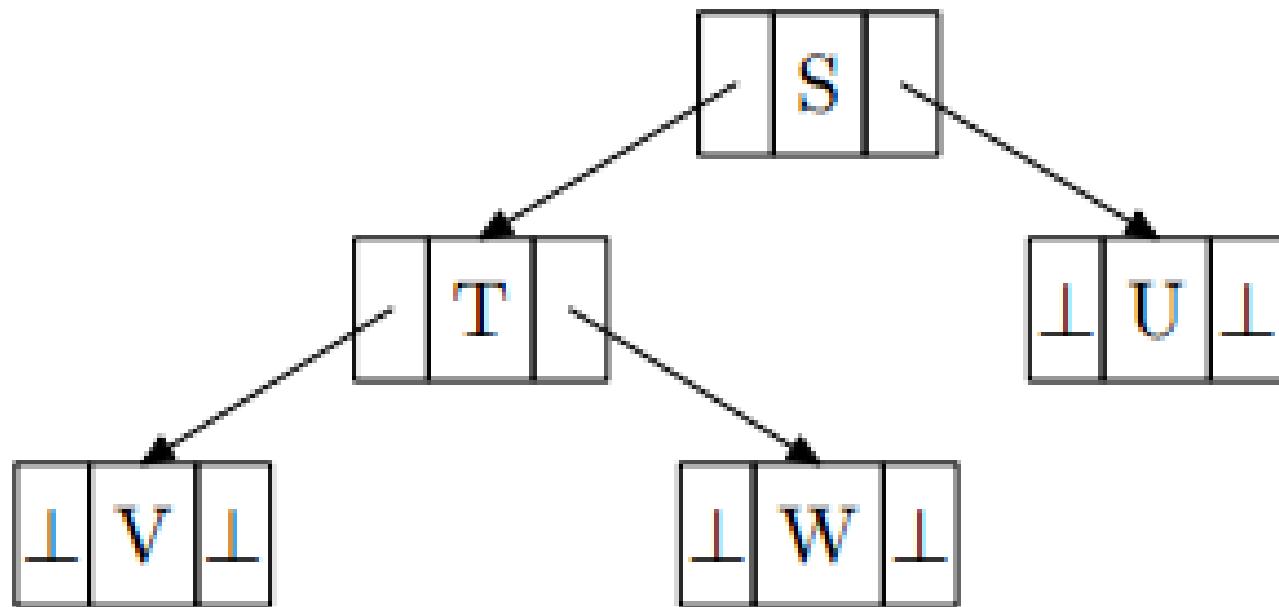
Kahendpuu



Kahendpuude erijuhud

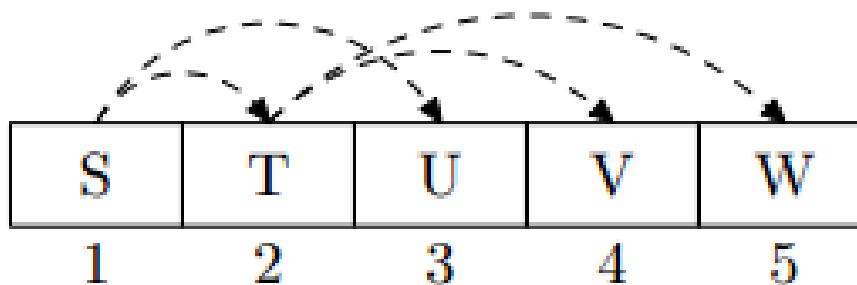
- Homogeenne kahendpuu
- Kompaktne kahendpuu
- Täielik kahendpuu

Kahendpuu „loomulik“ esitus



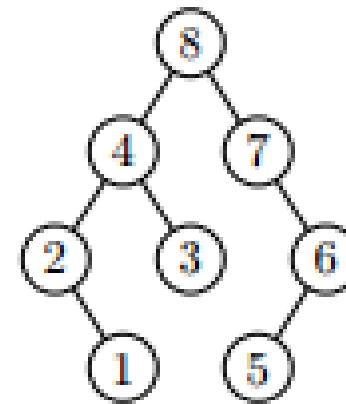
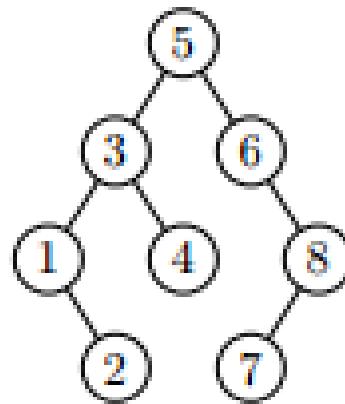
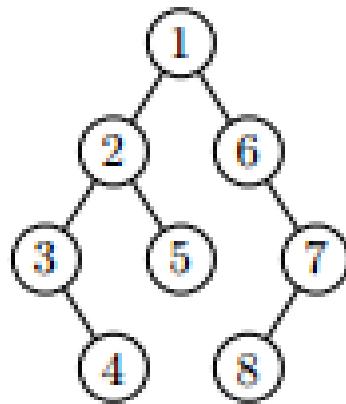
Kahendpuu esitus massiivina

- Sobib hästi täielike ja kompaktsete puude jaoks
- a_i vasak alluv on a_{2i}
- a_i parem alluv on a_{2i+1}
- a_i ülemus on $a_{i/2}$



Kahendpuu läbimine

- Eesjärjestus
- Keskjärjestus
- Lõppjärjestus



Kahendotsingu puu

- Vasaku alampuu kõik väärтused on juurtipu väärтusest väiksemad
- Parema alampuu kõik väärтused on juurtipu väärтusest suuremad
- Alampuud on ise ka kahendotsingu puud

Otsimine otsingupuus

```
function Find-recursive(key, node): // call initially with
node = root

if node = Null or node.key = key then
    return node

else if key < node.key then
    return Find-recursive(key, node.left)

else
    return Find-recursive(key, node.right)
```

Lisamine otsingupuuusse

```
bool BinarySearchTree::add(int value) {          bool BSTNode::add(int value) {  
    if (root == NULL) {                          if (value == this->value)  
        root = new BSTNode(value);            return false;  
        return true;                         else if (value < this->value) {  
    } else                                     if (left == NULL) {  
        return root->add(value);           left = new BSTNode(value);  
    }                                         return true;  
}                                            } else  
                                              return left->add(value);  
} else if (value > this->value) {  
    if (right == NULL) {  
        right = new BSTNode(value);  
        return true;  
    } else  
        return right->add(value);  
}  
return false;  
}
```

Eemaldamine otsingupuust

```
bool BinarySearchTree::remove(int value) {
    if (root == NULL)
        return false;
    else {
        if (root->getValue() == value) {
            BSTNode auxRoot(0);
            auxRoot.setLeftChild(root);
            BSTNode* removedNode = root->remove(value, &auxRoot);
            root = auxRoot.getLeft();
            if (removedNode != NULL) {
                delete removedNode;
                return true;
            } else
                return false;
        } else {
            BSTNode* removedNode = root->remove(value, NULL);
            if (removedNode != NULL) {
                delete removedNode;
                return true;
            } else
                return false;
        }
    }
}
```

```
BSTNode* BSTNode::remove(int value, BSTNode *parent) {
    if (value < this->value) {
        if (left != NULL)
            return left->remove(value, this);
        else
            return NULL;
    } else if (value > this->value) {
        if (right != NULL)
            return right->remove(value, this);
        else
            return NULL;
    } else {
        if (left != NULL && right != NULL) {
            this->value = right->minValue();
            return right->remove(this->value, this);
        } else if (parent->left == this) {
            parent->left = (left != NULL) ? left : right;
            return this;
        } else if (parent->right == this) {
            parent->right = (left != NULL) ? left : right;
            return this;
        }
    }
}

int BSTNode::minValue() {
    if (left == NULL)
        return value;
    else
        return left->minValue();
}
```

Fenwicki puu

- Tuntud ka kui „binary indexed tree“ (BIT)
- Ülesanne: Meil on n karpi ja võimalikud operatsioonid:
 - Lisada kuulike mingisse karpi
 - Leida, mitu kuulikest on karpides i kuni j
- Naiivne lahendus on $O(1)$ lisamiseks ning $O(n)$ otsimiseks

Fenwicki puu

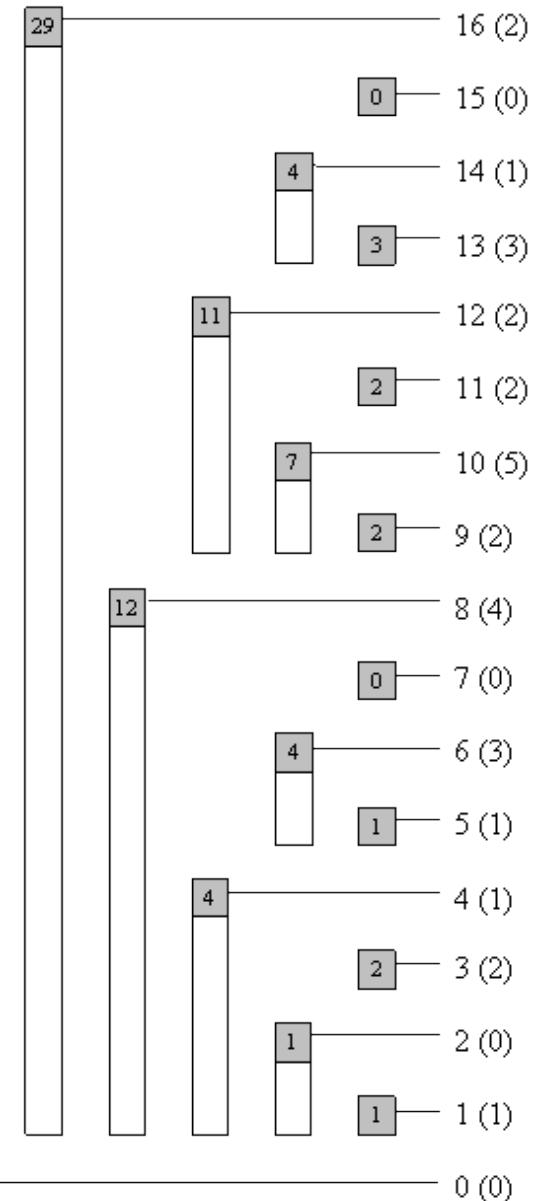
- $F[i]$ – kuulikeste arv karbis i
- $C[i]$ – kuulikeste arvude summa $F[0] + \dots + F[i]$
- $\text{Tree}[i]$ – kuulikeste arvude summa **vastutusalaga** i

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
tree	1	1..2	3	1..4	5	5..6	7	1..8	9	9..10	11	9..12	13	13..14	15	1..16

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
f	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
c	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29
tree	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29

Fenwicki puu

- Oletame, et soovime esimese 13 karbi summat
- 13 kahendsüsteemis on 1101
- $c[1101] = \text{tree}[1101] + \text{tree}[1100] + \text{tree}[1000]$



Viimase mittenullise biti isoleerimine

- $X \& -X$
 - Bitwise operaator
- Proovige, mida see teeb!
- Miks?

Esimese n karbi lugemine

```
int read(int idx)
{
    int sum = 0;
    while (idx > 0){
        sum += tree[idx];
        idx -= (idx & -idx);
    }
    return sum;
}
```

Kuulikeste lisamine

```
void update(int idx ,int val)
```

```
{
```

```
    while (idx <= MaxVal)
```

```
{
```

```
        tree[idx] += val;
```

```
        idx += (idx & -idx);
```

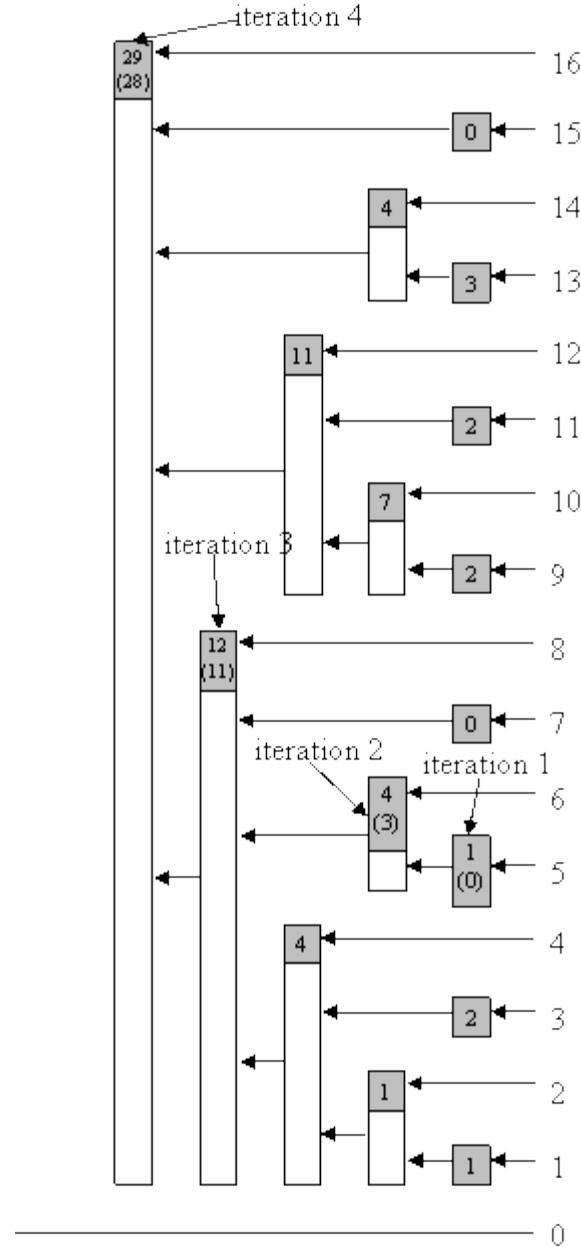
```
}
```

```
}
```

Lisamine: näide

iteration	idx	position of the last digit	idx & -idx
1	$5 = 101$	0	$1 (2^0)$
2	$6 = 110$	1	$2 (2^1)$
3	$8 = 1000$	3	$8 (2^3)$
4	$16 = 10000$	4	$16 (2^4)$
5	$32 = 100000$	---	---

Lisamine: näide



Fenwicki puu - ülesanne

- http://community.topcoder.com/stat?c=problem_statement&pm=6551&rd=9990
- Given a sequence of **N** integers in the range **[0,M)**, find the sum of medians of **N-K+1** contiguous subsequences of length **K**.
(1<=N<=250,000; 1<=K<=1000, M=65536).

Lähim esivanem (Lowest Common Ancestor, LCA)

- Probleem: leida evolutsioonipuus kahe liigi lähim ühine esivanem

Range Minimum Query (RMQ)

- Kuidas leida massiivi alamlõigus väikseima elemendi indeks?

$$\text{RMQ}_A(2,7) = 3$$

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
2	4	3	1	6	7	8	9	1	7

RMQ - eeltöötlus

- Et see päring oleks võimalikult kiire, on vajalik eeltöötlus
- Koostame näiteks 2D tabeli, kus on kõikvõimalikud miinimumid
- Jõumeetod – $O(n^3)$

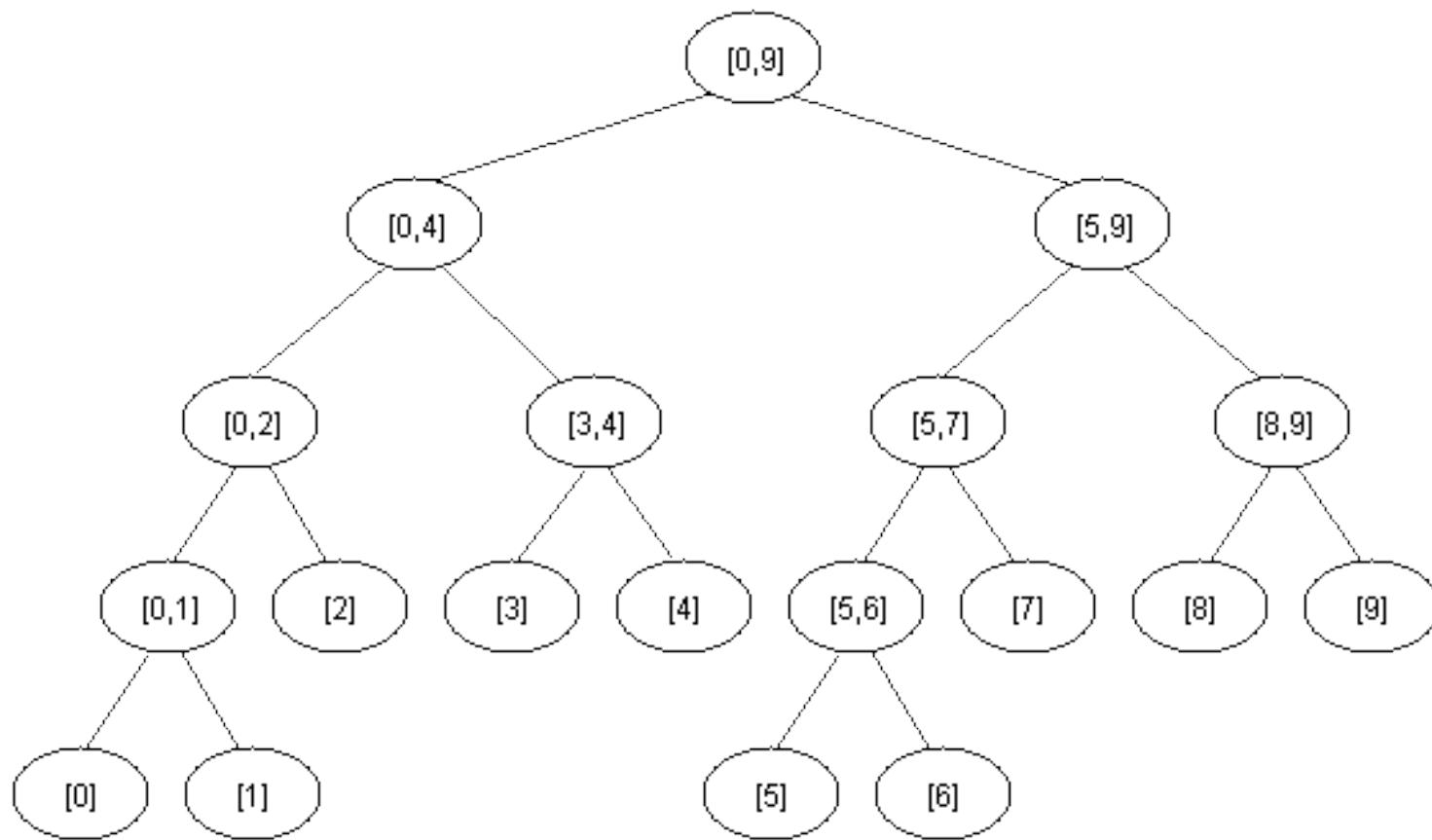
RMQ eeltöötlus – DP lähenemine

```
void process1(int M[MAXN][MAXN], int A[MAXN], int N)
{
    int i, j;
    for (i = 0; i < N; i++)
        M[i][i] = i;
    for (i = 0; i < N; i++)
        for (j = i + 1; j < N; j++)
            if (A[M[i][j - 1]] < A[j])
                M[i][j] = M[i][j - 1];
            else
                M[i][j] = j;
}
```

Keerukus $O(n^2)$

Lõikude puu

- Puu iga sõlm hoiab infot vastava lõigu kohta



Lõikude puu koostamine

```
void initialize(int node, int b, int e, int M[MAXIND], int A[MAXN], int N)
{
    if (b == e)
        M[node] = b;
    else
    {
        //compute the values in the left and right subtrees
        initialize(2 * node, b, (b + e) / 2, M, A, N);
        initialize(2 * node + 1, (b + e) / 2 + 1, e, M, A, N);
        //search for the minimum value in the first and
        //second half of the interval
        if (A[M[2 * node]] <= A[M[2 * node + 1]])
            M[node] = M[2 * node];
        else
            M[node] = M[2 * node + 1];
    }
}
```

Lõikude puust pärimine

```
int query(int node, int b, int e, int M[MAXIND], int A[MAXN], int i, int j)
{
    int p1, p2;

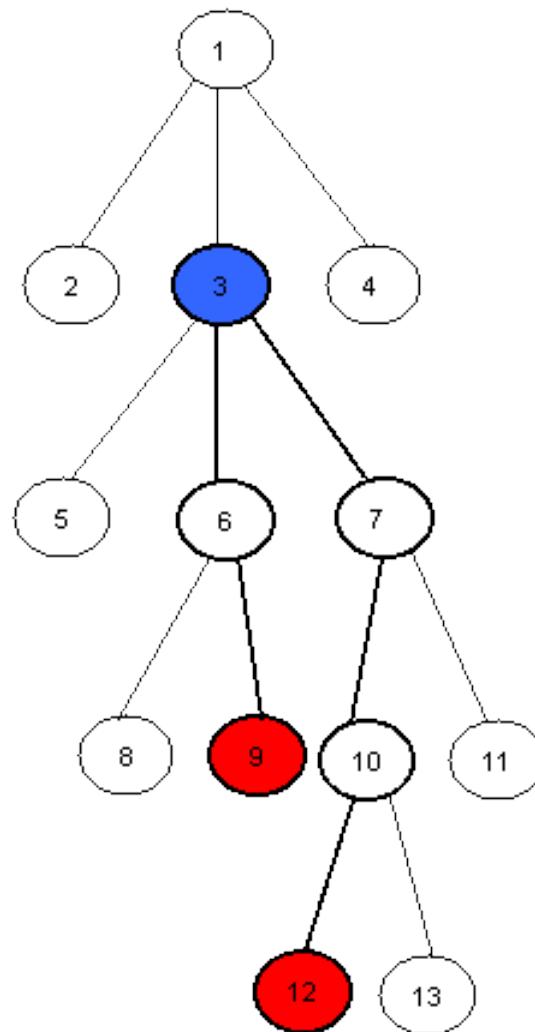
    //if the current interval doesn't intersect
    //the query interval return -1
    if (i > e || j < b)
        return -1;

    //if the current interval is included in
    //the query interval return M[node]
    if (b >= i && e <= j)
        return M[node];

    //compute the minimum position in the
    //left and right part of the interval
    p1 = query(2 * node, b, (b + e) / 2, M, A, i, j);
    p2 = query(2 * node + 1, (b + e) / 2 + 1, e, M, A, i, j);

    //return the position where the overall
    //minimum is
    if (p1 == -1)
        return M[node] = p2;
    if (p2 == -1)
        return M[node] = p1;
    if (A[p1] <= A[p2])
        return M[node] = p1;
    return M[node] = p2;
}
```

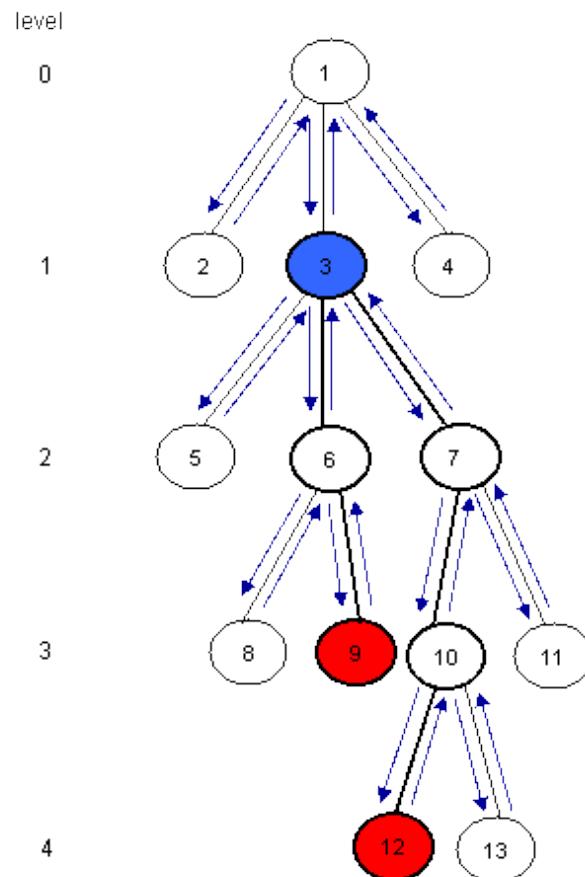
LCA jälle



$$\text{LCA}_T(9, 12) = 3$$

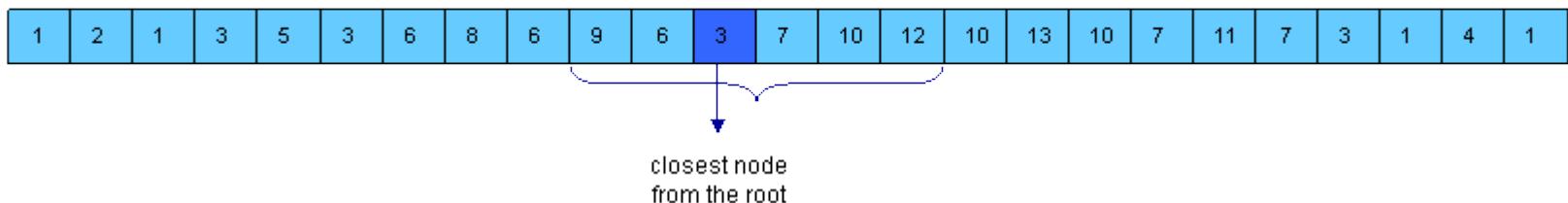
LCA - läbikäimine

- Käime puu sügavuti läbi
- Leiame kõrgeima sõlme u ja v vahel



Euler Tour:

$$\text{LCA}_T(9, 12) = 3$$



LCA ja RMQ

- Koostame 3 massiivi:
- $E[1, 2*N-1]$ – läbikäimisel külastatud sõlmed
- $L[1, 2*N-1]$ – läbikäimisel külastatud sõlmede sügavused
- $H[1, N]$ – sõlmede esmakordsed esinemised massiivis E

LCA ia RMQ

$$\text{LCA}_T(10, 15) = E[12] = 3$$

E:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	2	1	3	5	3	6	8	6	9	6	3	7	10	12	10	13	10	7	11	7	3	1	4	1

E[10...15]

$$\text{RMQ}_L(10, 15) = 12$$

L:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
0	1	0	1	2	1	2	3	2	3	2	1	2	3	4	3	4	3	2	3	2	1	0	1	0

H:

1	2	3	4	5	6	7	8	9	10	11	12	13
1	2	4	24	5	7	13	8	10	14	20	15	17

$$R[9] = 10 \quad R[12] = 15$$

$$\text{LCA}_T(u, v) = E[\text{RMQ}_L(H[u], H[v])]$$

Lõikude puu - ülesanne

- <http://poj.org/problem?id=2374>