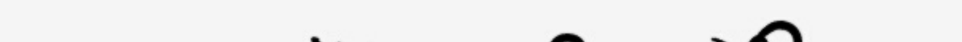
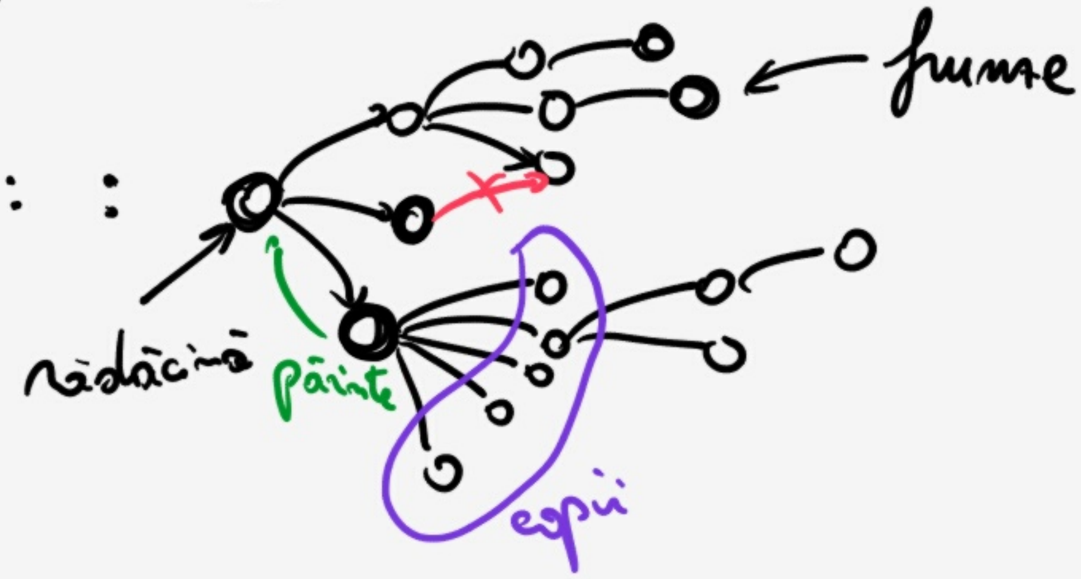
Arbori

Listă:



Arbore: :



Arbore binar: arbore în care fiecare nod are maxim 2 copii

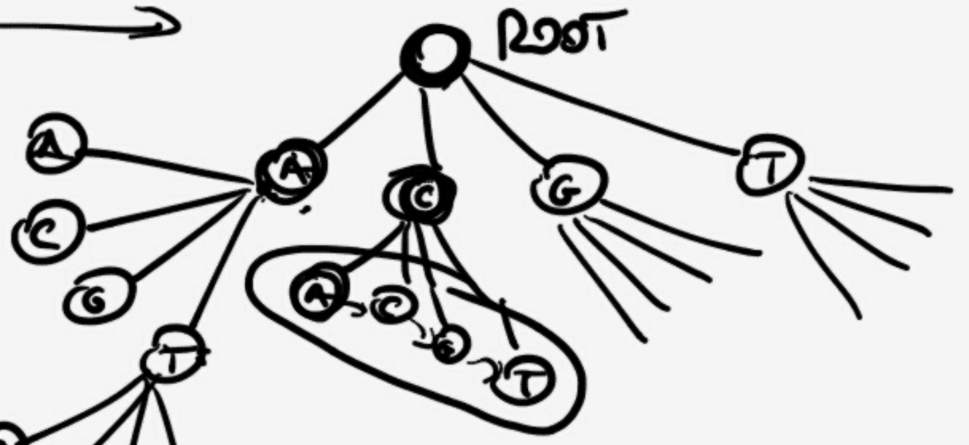
Ref: ... AGCCTAGCATA ... AAAAAAAAAAATACCG

Seq: CCTAGC

27, 28, 29, 30, 31, 32

33

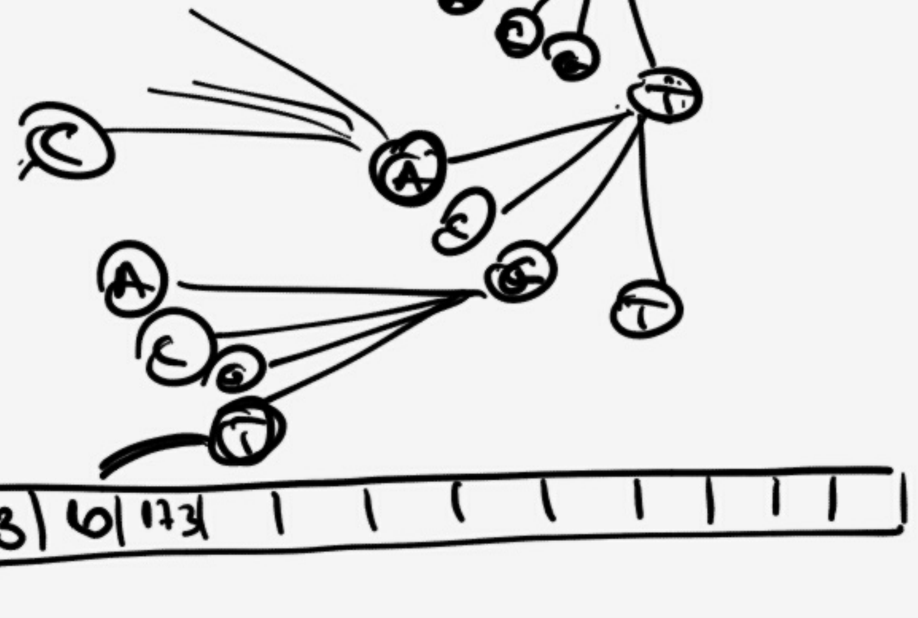
Prefix trees



Ref: 36 chars

Seq: 5 caractere

(A, C, G, T)

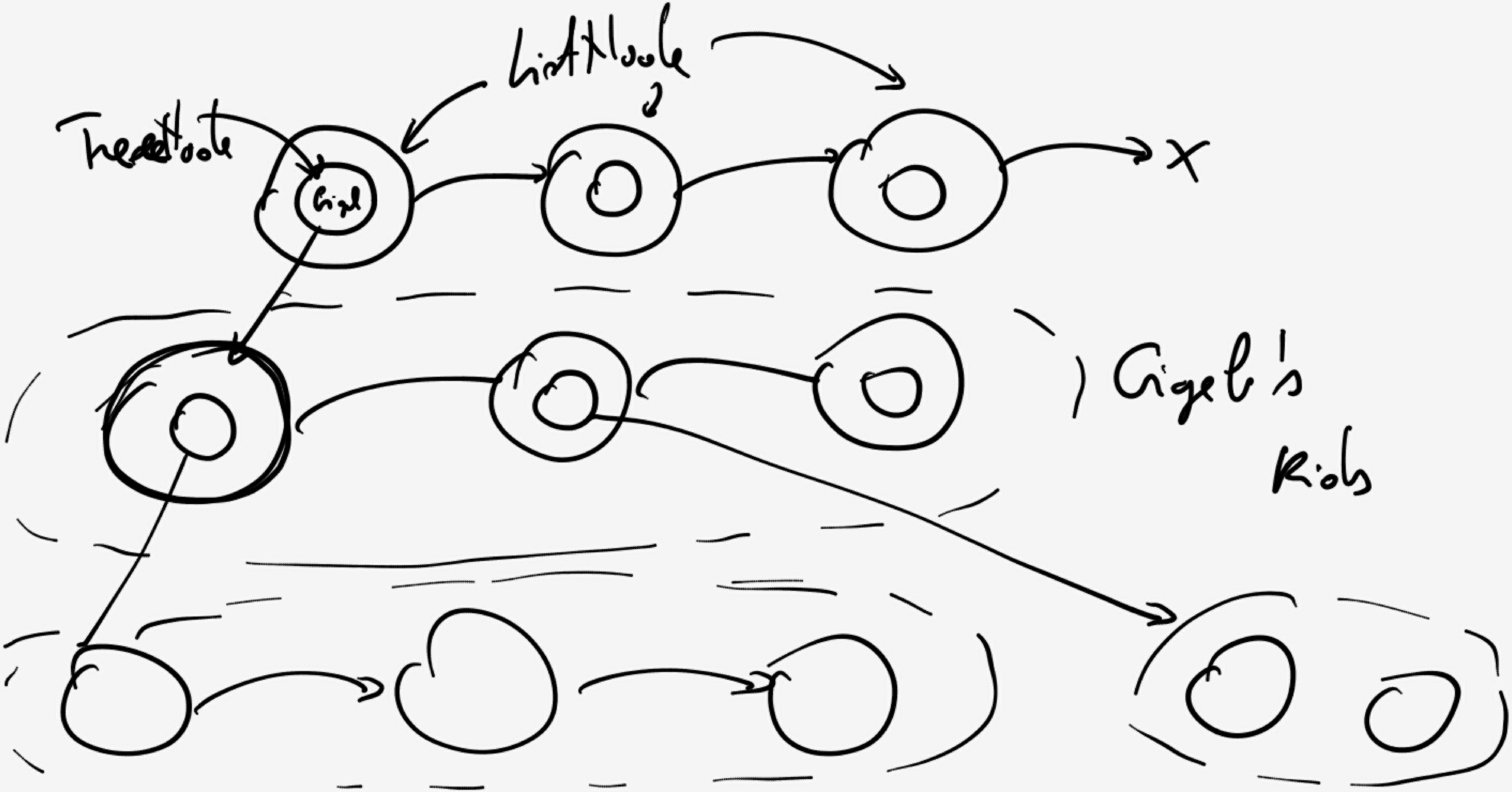


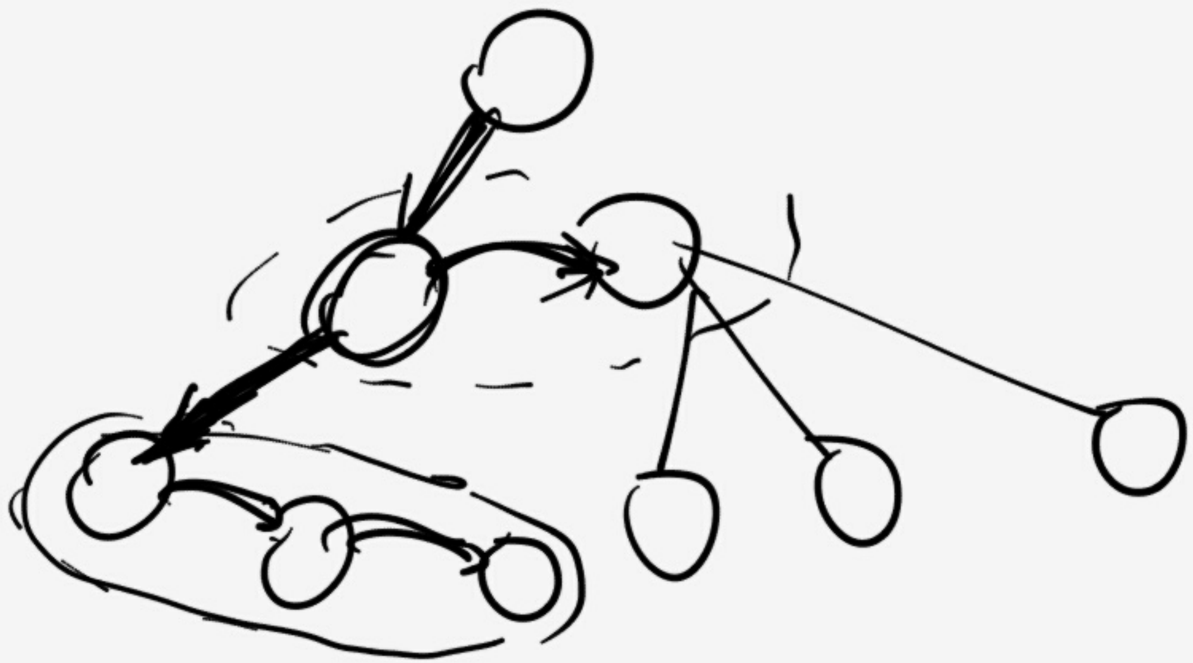
Implementare

```
struct TreeNode {
    struct ListNode *children;
    char * name;
};
```

```
struct ListNode {
    struct TreeNode * payload;
    struct ListNode *next;
}
```

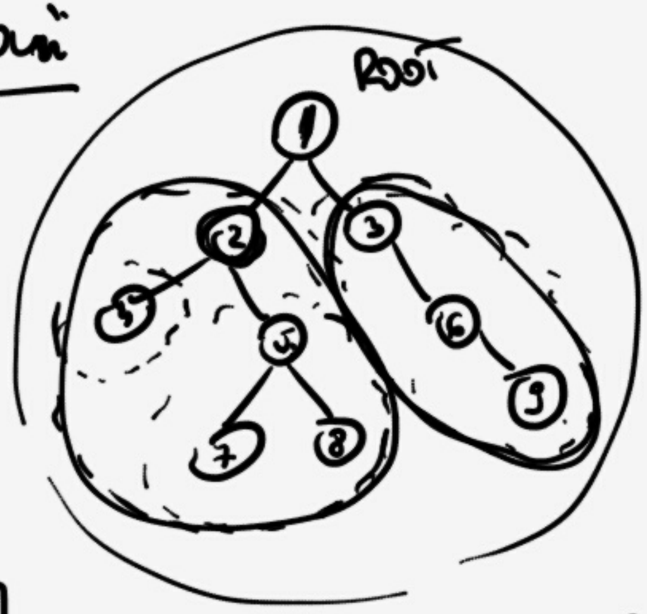
```
struct TreeNode *createNode (char * name) {
    struct TreeNode * newNode = (struct TreeNode *) malloc(
                                    sizeof(struct TreeNode));
    newNode -> children =
```


ListNode
TreeNode
Gigel
Gigel's
Kids



```
struct TreeNode {
    char *name;
    struct TreeNode *children;
    struct TreeNode *next;
};
```

Arbori binari




```
struct BinaryTreeNode {
    char *name;
    struct BinaryTreeNode * left;
    struct BinaryTreeNode * right;
};
```

S, R, D → 4, 2, 7, 5, 8, 1, 3, 6, 9

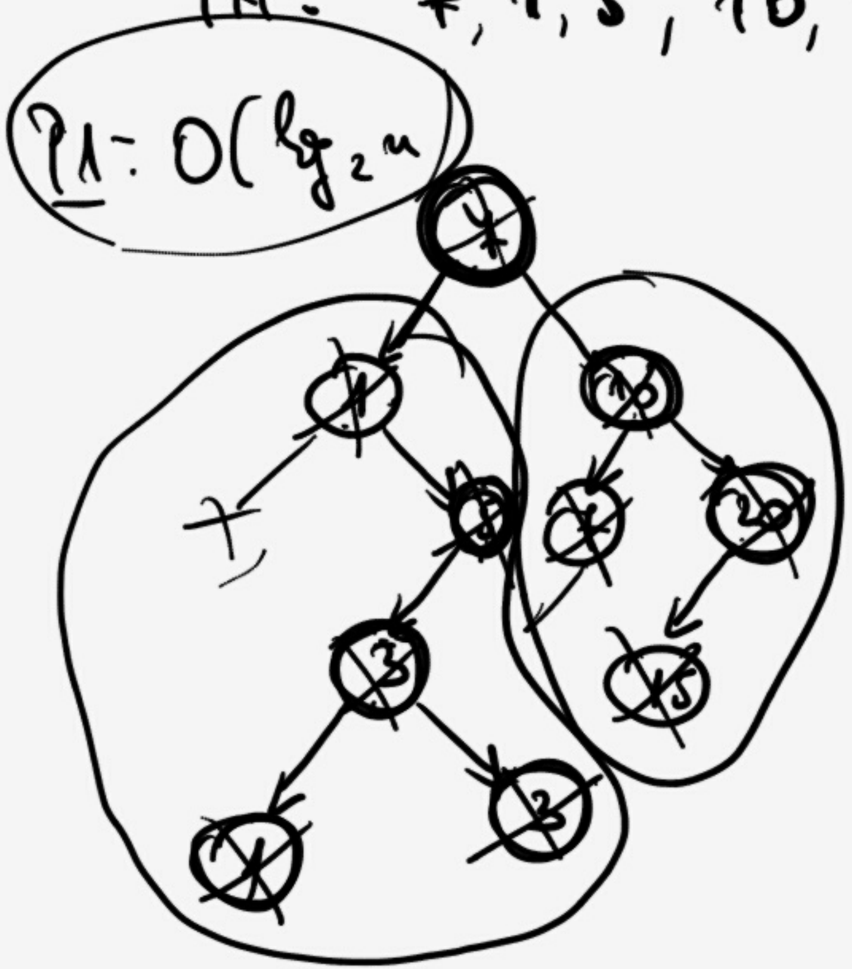
R, S, D → 1, 2, 4, 5, 7, 8, 3, 6, 9

D, S, R → 9, 6, 3, 8, 7, 5, 4, 2, 1

Tree Sort

IN: 7, 1, 5, 10, 3, 7, 20, 15, 3, 1

P1: $O(\log_2 n$



P2: S,R,D → $O(n)$

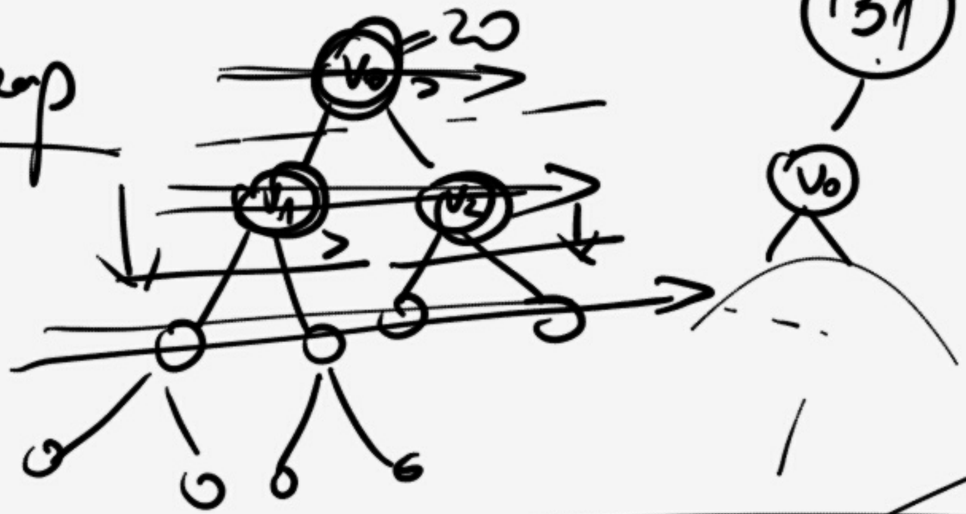
```
parse (Node n)
    parse (n -> left)
    print (n -> data)
    parse (n -> right)
```

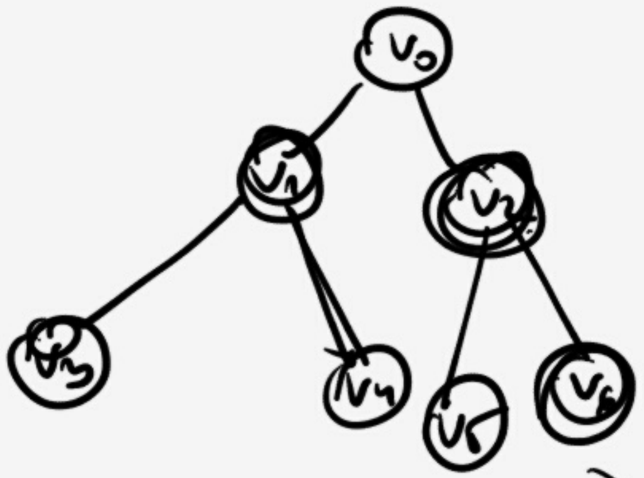
1, 1, 3, 3, 5, 7, 7, 10, 15, 20

T: $O(\log_2 n + n) \rightarrow O(n)$

Heap

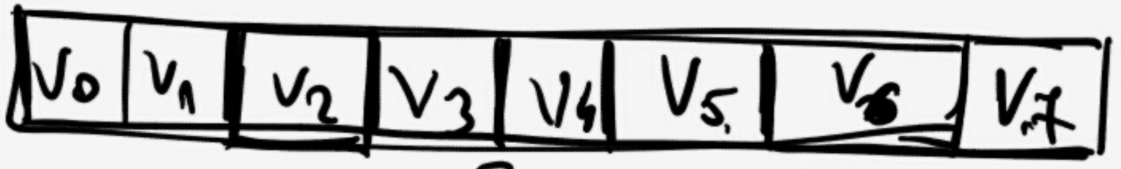
P1: Build Heap





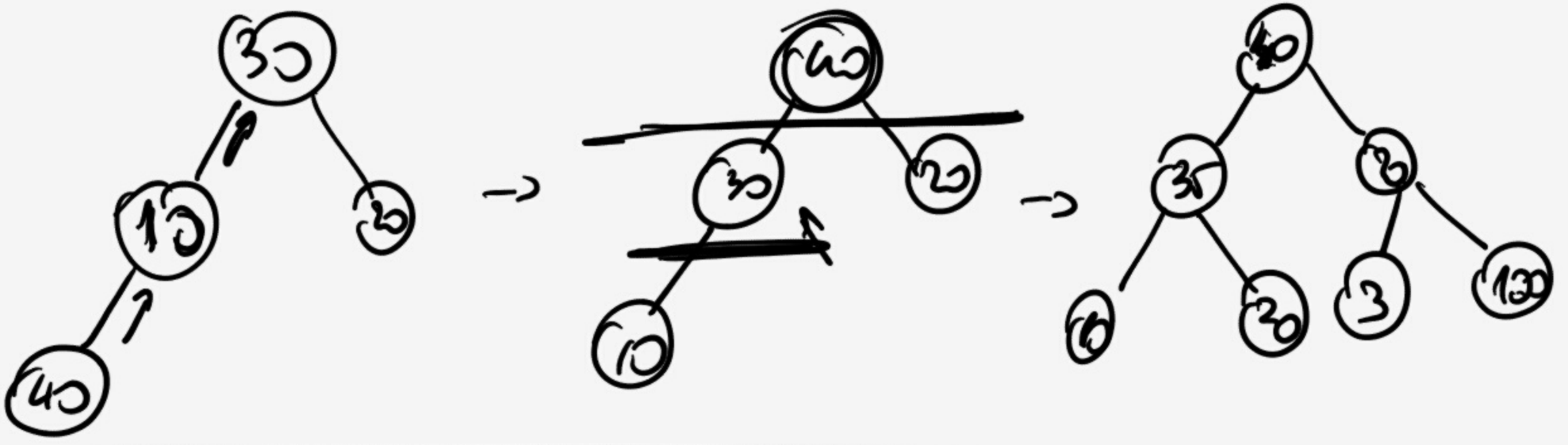
P2:

V



$$C(i) = (2 \cdot i + 1, 2 \cdot i + 2)$$

$$P(i) = \lceil (i-1)/2 \rceil$$



V:

3	7	1	5	2	7	3	10	4	1	7	5
0	1	2	3	4	5	6	7	8	9	10	11

$n-1 = 10/2 \rightarrow 5$