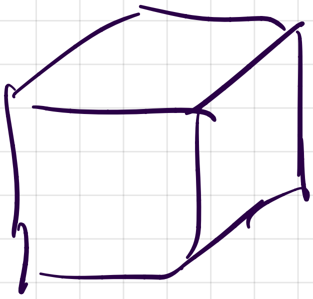
Структуры данных



← set(i, x) $O(1)$

→ ask(l, r) $O(\log n)$

Массив.

list()

a = [1, 2, 3]

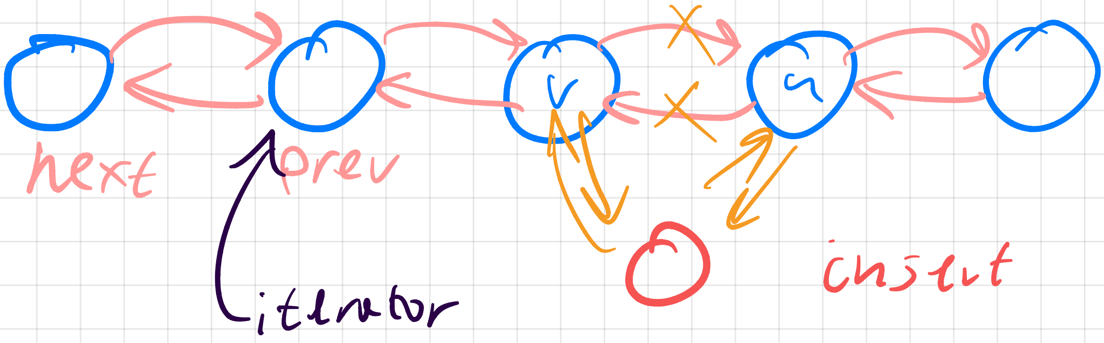
по индексу

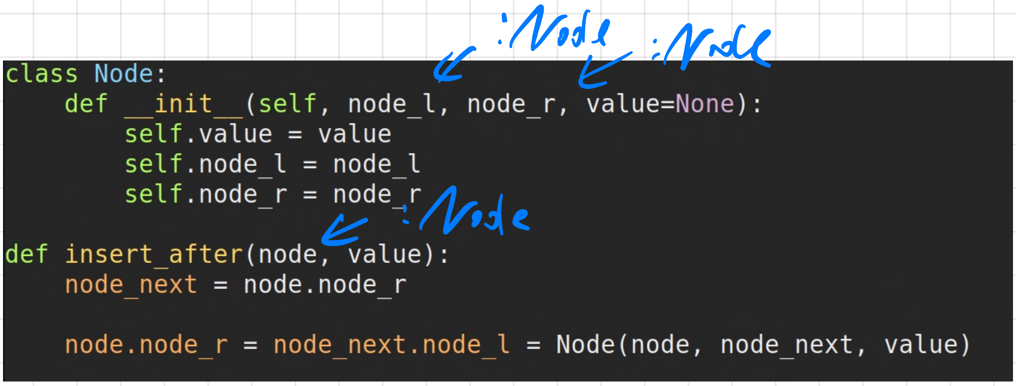
a[0] = 2 ← записать эл-т

print(a[1]) ← прочитать эл-т

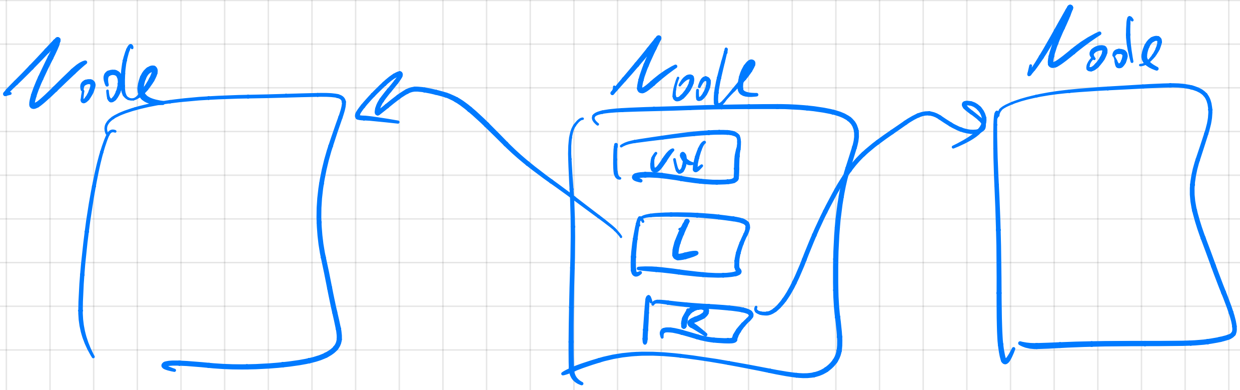
$O(1)$

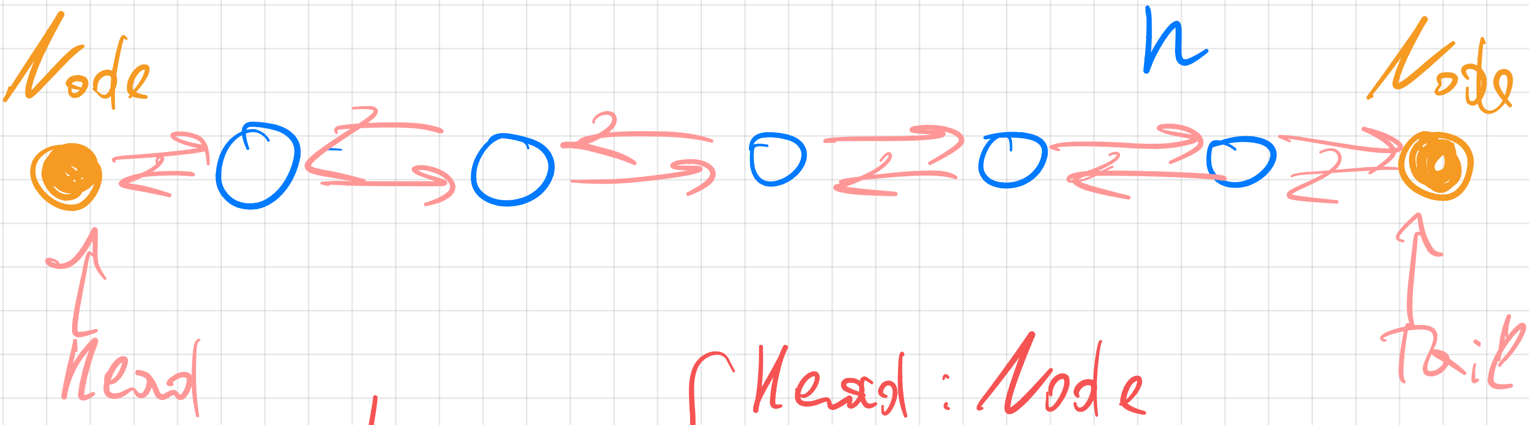
Связный список.





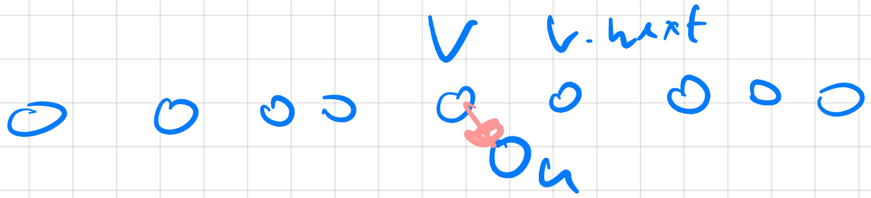
```
class Node:
    def __init__(self, node_l, node_r, value=None):
        self.value = value
        self.node_l = node_l
        self.node_r = node_r

def insert_after(node, value):
    node_next = node.node_r

    node.node_r = node_next.node_l = Node(node, node_next, value)
```





$$\text{List} = \begin{cases} \text{Head} : \text{Node} \\ \text{Tail} : \text{Node} \end{cases}$$

```
def   print(l: List)                          O(n)
      head : Node = l.head
      tail : Node - l.tail

      v : Node =head.next
      while v ≠ tail :          iterator
            print(v.value)
            v = v.next
```



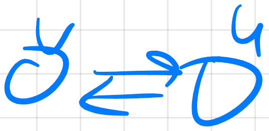
insert

u = Node(value = 10, L = none,
R = none)

$O(1)$

w = v.next

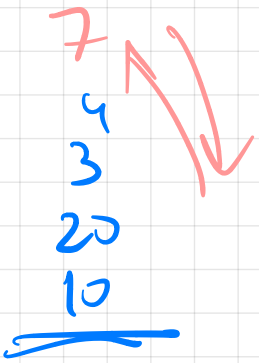
v.next = u
u.prev = v



w.prev = u
u.next = w

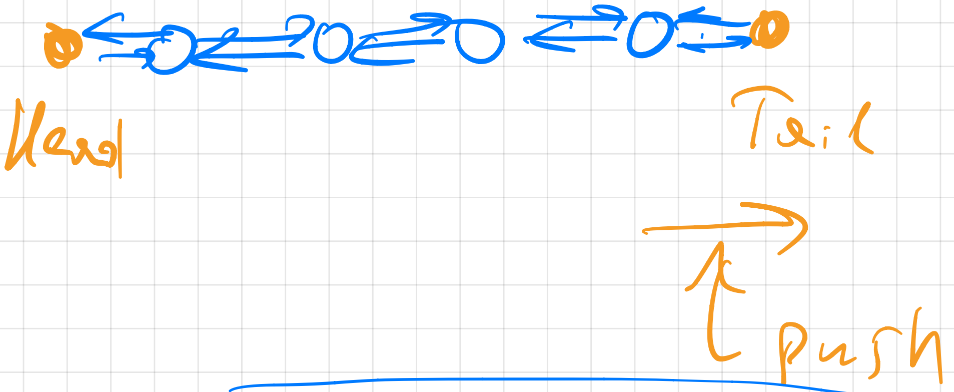
Stack, Queue, Deque
(Стэк, очередь, дек)

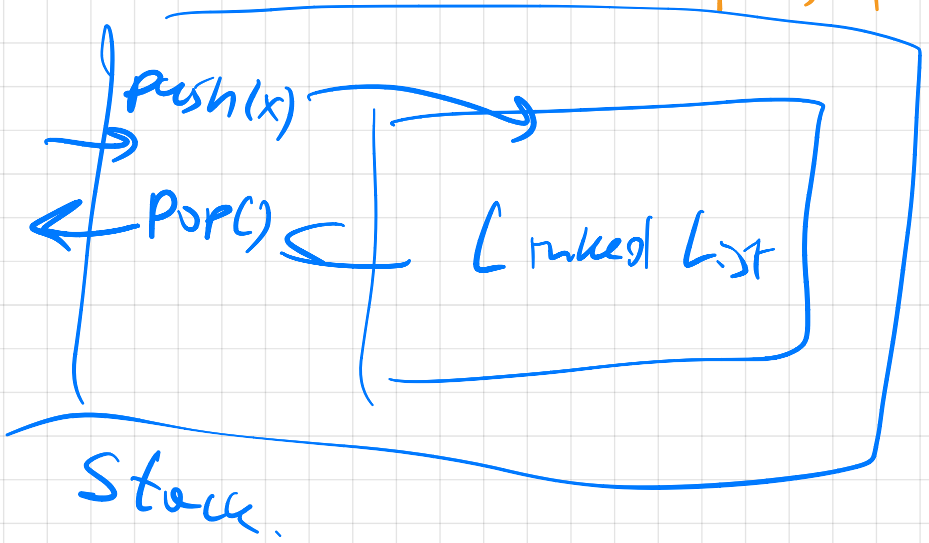
Стэк



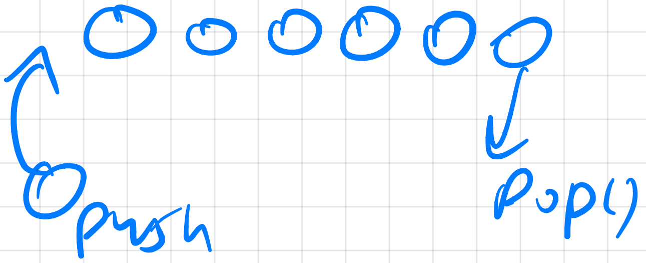
push(x)
pop() → int

$O(1)$



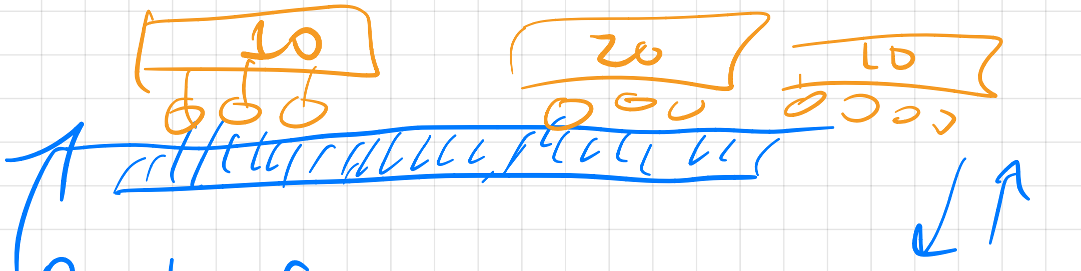


Очередь.



$O(1)$

Дек.



push_front
pop_front

$O(1)$

push_back()
pop_back()

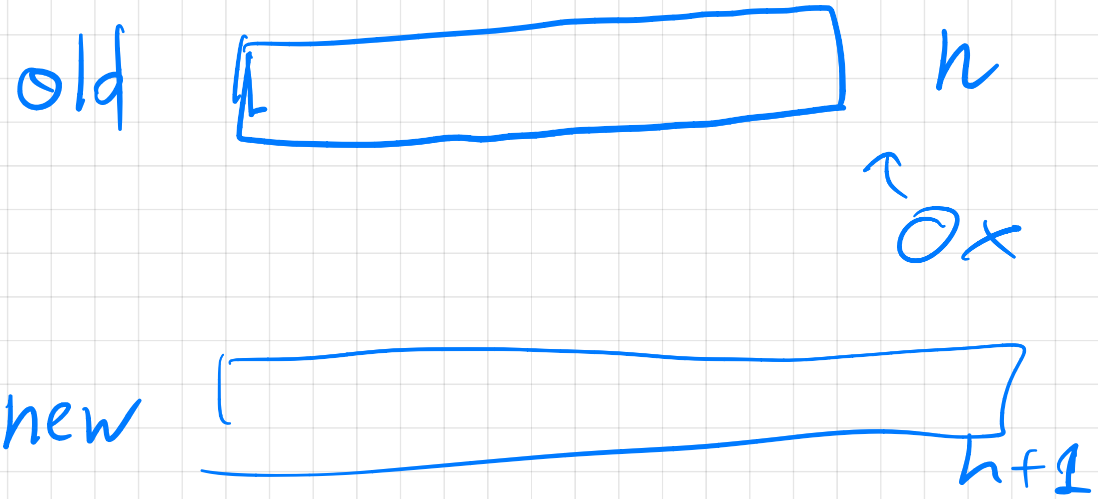
Стэк => Дэк => Linked List

Очередь =>

Vector (list)

Глупый Алгоритм

Массив + push_back()
pop_back()

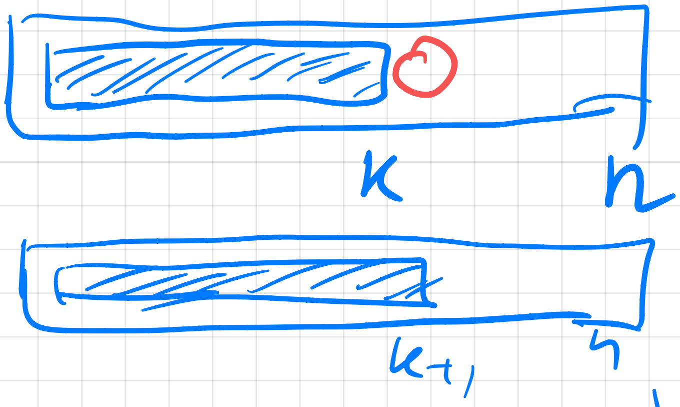


```
Def push_back(old, n, x)
    new = [0 for i in range(n+1)]
    for (i=0..n-1):
        new[i] = old[i]
    new[n] = x
    return (new, n+1)
```

$$0 \quad \underbrace{+1 \quad +1 \quad +1}_{n \times \text{push-Back}}$$

$$\sum_{i=0}^{n} i = \frac{n(n+1)}{2} = \Theta(n^2)$$

vector



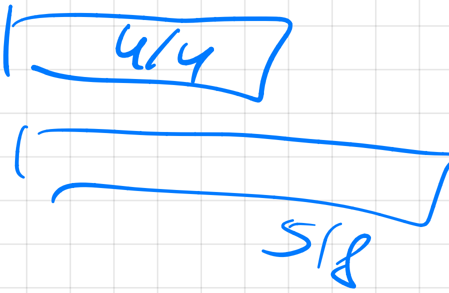
```
vector {
    arr
    n   //len(arr)
    k
}
```

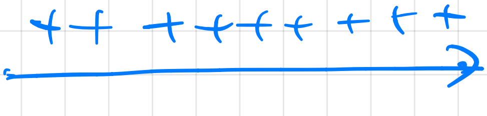
```
push_back(v: vector, x: int)
    if    v.k < v.n:
          v.arr[v.k] = x
          v.k += 1
          return v
```

$O(1)$

```
    new = alloc(2n)

    for    i = 0 .. v.n-1
           new[i] = v.arr[i]

    v.arr = new
    v.n = 2·n

    v.arr[v.k] = x
    v.k += 1
    return v
```

$O(n)$



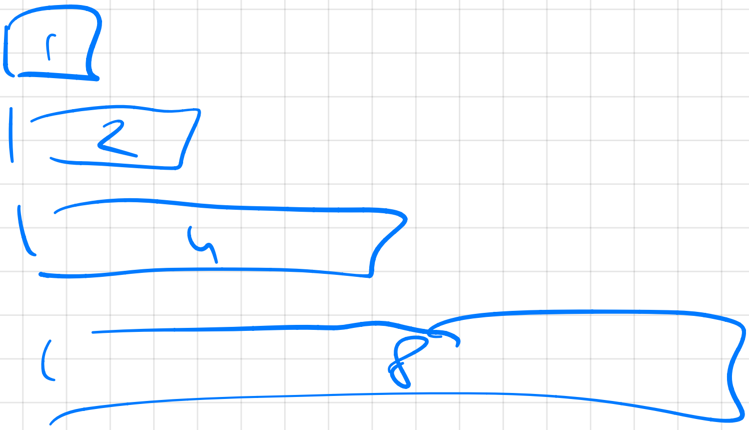


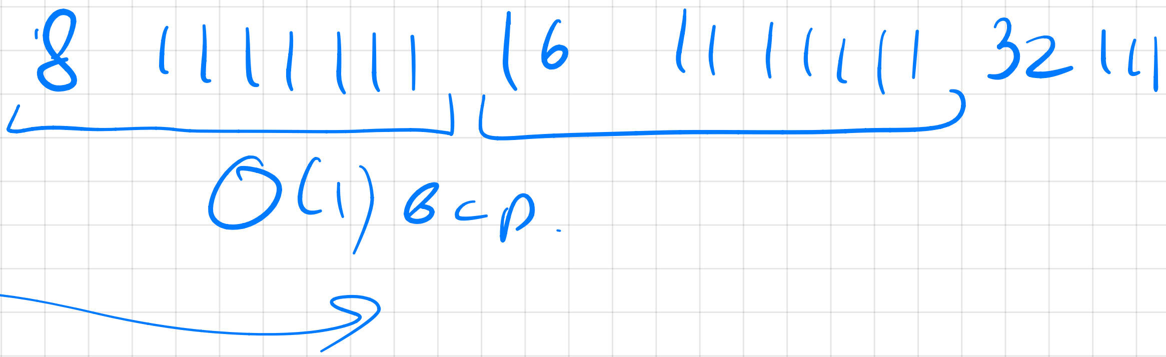
$n=2^k$

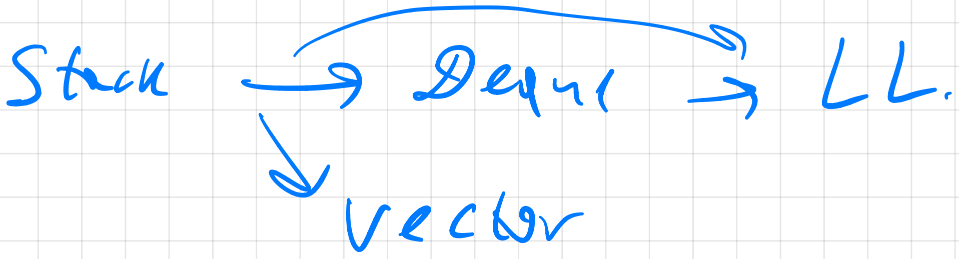
$$\left.\begin{array}{ll} n = 2^k & \rightarrow O(n) \\ n+1 & O(1) \\ n+2 & O(1) \\ \vdots & O(1) \\ & O(1) \\ 2n = 2^{k+1} & O(1) \end{array}\right]$$

$$\frac{1 \times n + (n-1) \times 1}{n} = O(1)$$

Время опер.
в среднем.



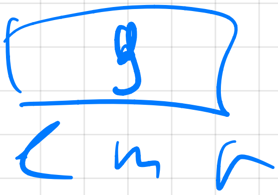




Бинарный Поиск

$\boxed{3\ 2\ 1\ 4\ 8\ 16\ 7}$ O(n) линейный поиск

$\boxed{1\ 1\ 2\ 3\ 4\ 4\ 6\ 7\ 8}$ 6



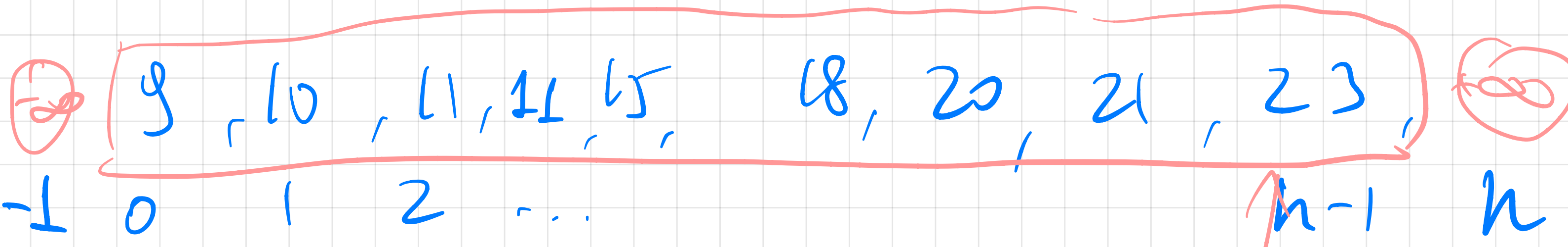
? 6

```
Search (l, r, x)
    if l > r:
        return -1.
    m = (l + r) // 2
    if a[m] == x
        return m
    if a[m] > x
        return Search (l, m-1)
    else
        return Search (m+1, r)
```

$$\text{Search}(0, n-1, x)$$

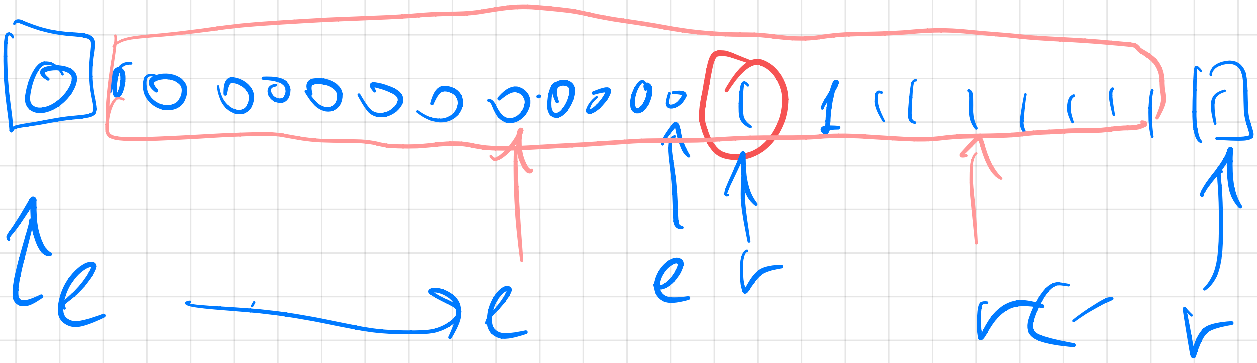
$$O(\log n)$$

Бинарный Поиск на Инвариантах



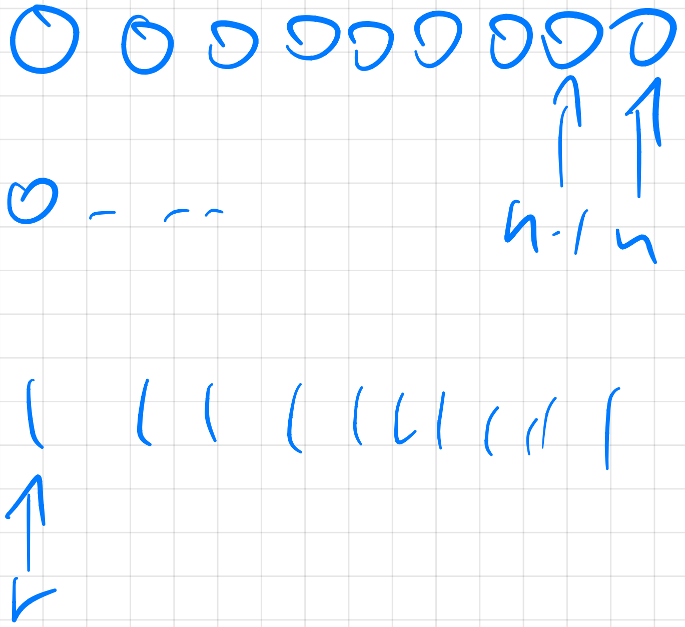
первый элемент $\geq x$

$a_i \geq x$

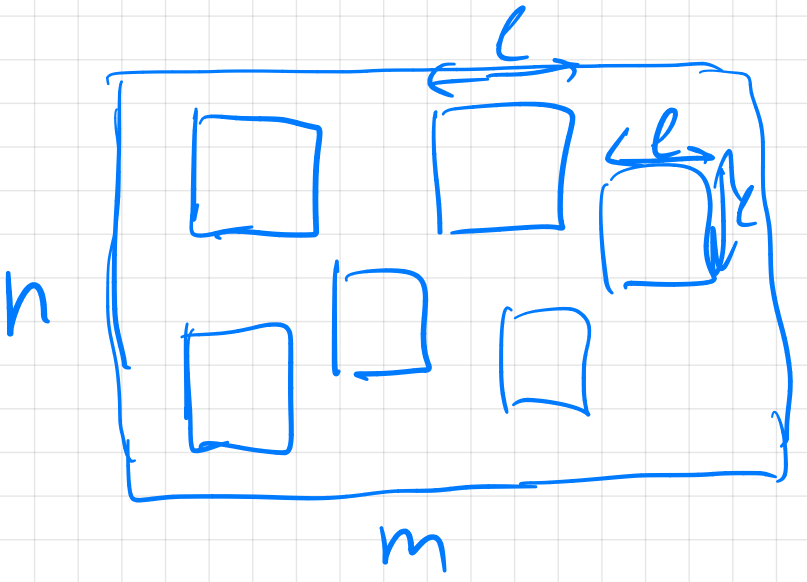


← инвариант, что $f(l) = 0$
$f(r) = 1$

```
l = -1
r = n
while   r - l > 1
        m = (l+r)//2
        if   a[m] ≥ x :        (f(m) == 1?)
                r = m          (r ← m)
        else
                l = m
return  r      (Если r == n можно
                вернуть -1)
```



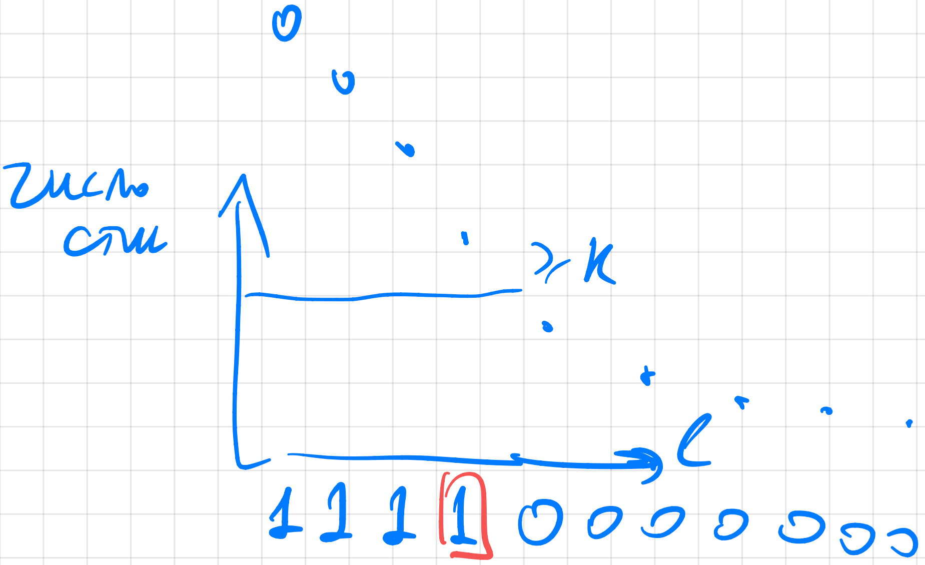
Бинарный поиск по ответу



$= k$ стикеров

l: $\lfloor \frac{n}{l} \rfloor \cdot \lfloor \frac{m}{l} \rfloor$

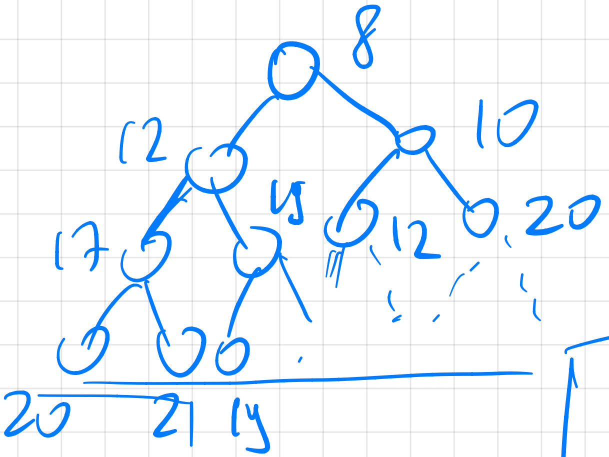
MAX l: $\lfloor \frac{n}{l} \rfloor \lfloor \frac{m}{l} \rfloor \geq k$ ← $f(l)$



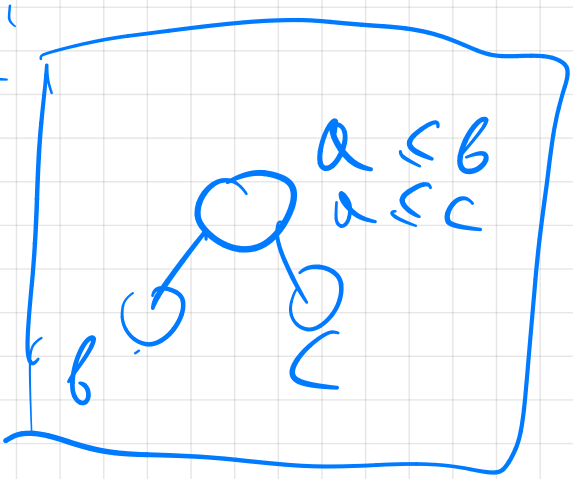
```
L = 0                    // f(L)=1
R = min(n,m)+1           // f(R)=0

while R-L>1:
    m=(L+R)//2
    if f(m)==1:
        L=m
    else
        R=m
```

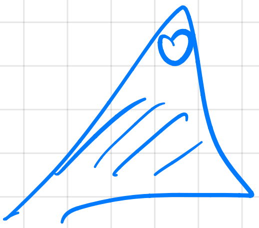
$O(\log n)$

Бинарная куча.

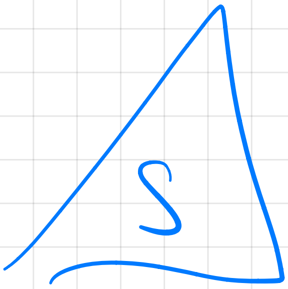


или

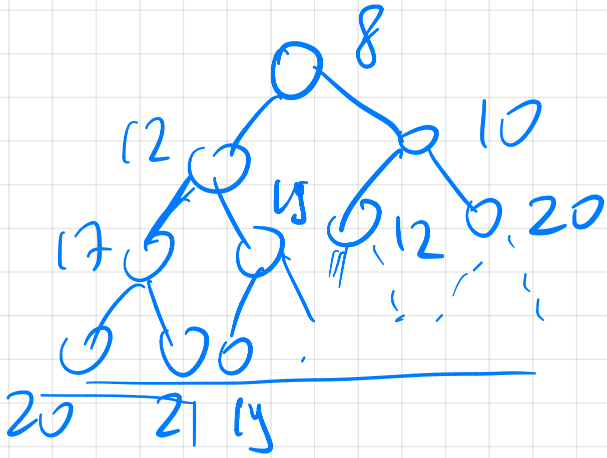


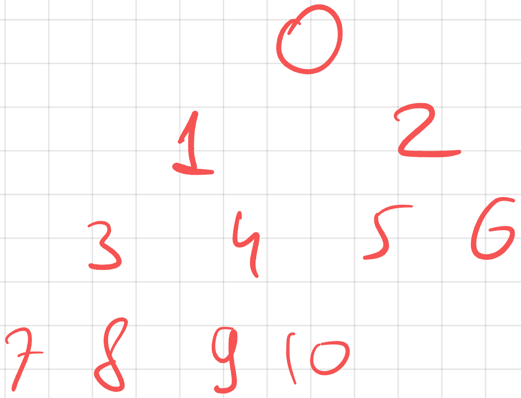


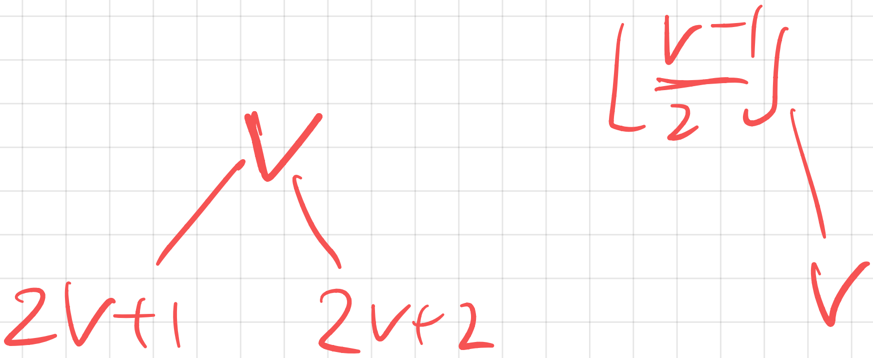
в вершине записан min эл-т



$\min(S) = O(1)$







$[8, 12, 10, 17, 14, 12, 20 \ldots\ldots]$
a

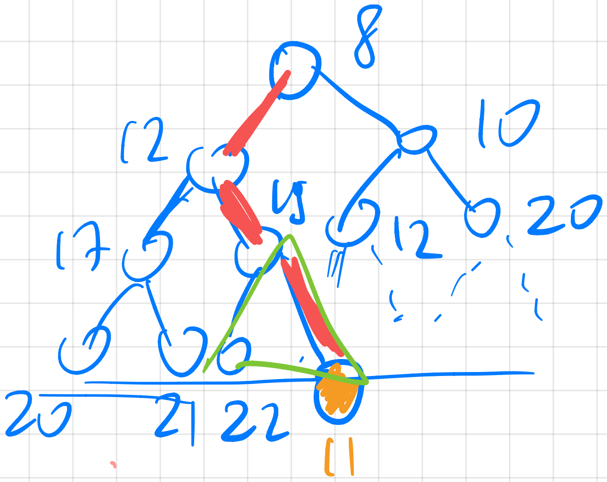
```
min():
    return a[0]
```

add(x)

a.push_back(x)

siftUp(len(a) - 1)

$O(\log n)$

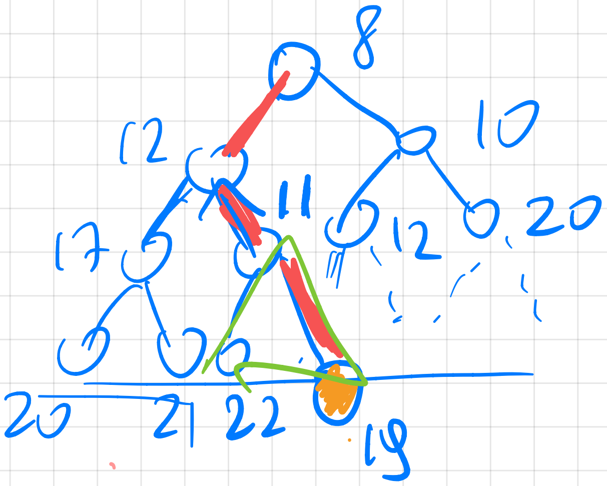


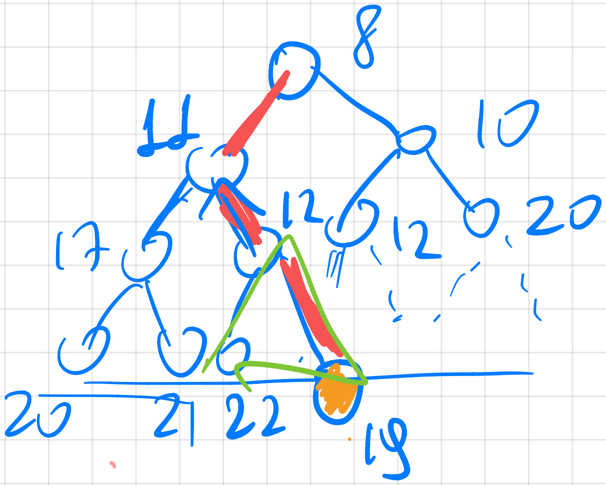
k:

$1+2+2^1+..+2^k \le n$

$2^k \le n$

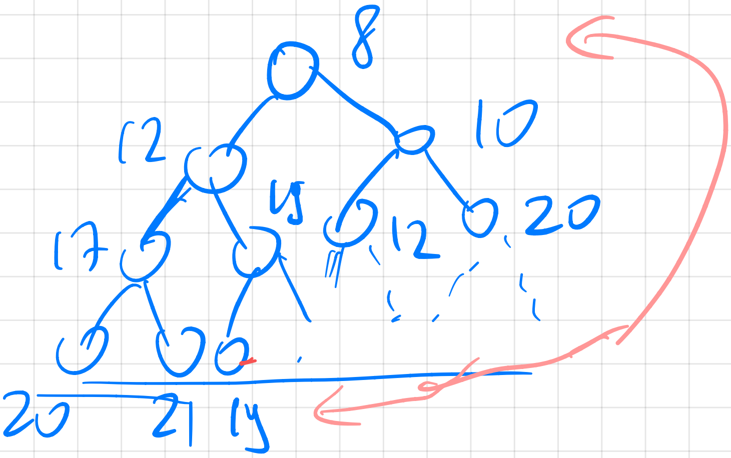
$k \le \log_2 n$

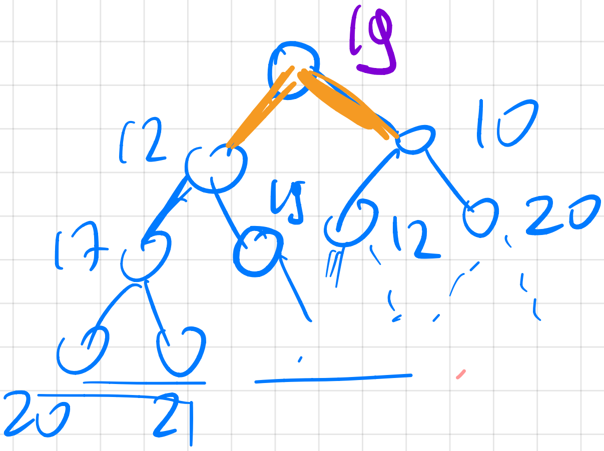


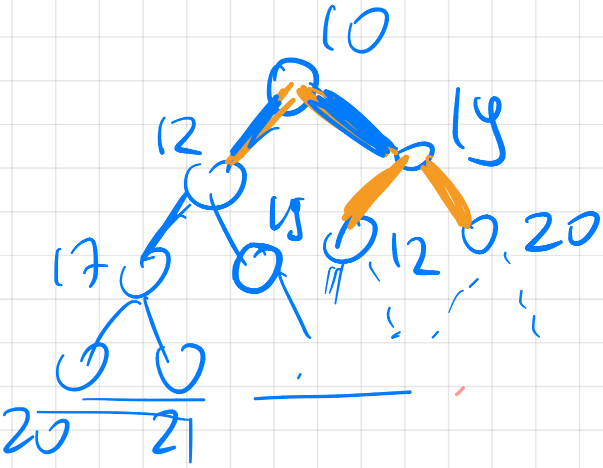


```
siftUP(i)
    while $i \neq 0$ and $a[i] < a[\frac{i-1}{2}]$
        swap($a[i]$, $a[\frac{i-1}{2}]$)
        $i = \frac{i-1}{2}$
```

```
ExtMin():        $O(\log n)$
    swap($a[0]$, $a[len(a)-1]$)
    a.pop()
    siftDown(0)
```







```
SiftDown(v):
    while True:
        mn = v
        for u: [2v+1, 2v+2]:
            if ∃ a[u] u a[u] < a[mn]:
                mn = u
        if mn == v:
            STOP
        else
            swap(a[v], a[mn])
            v = mn
```

$O(\log n)$