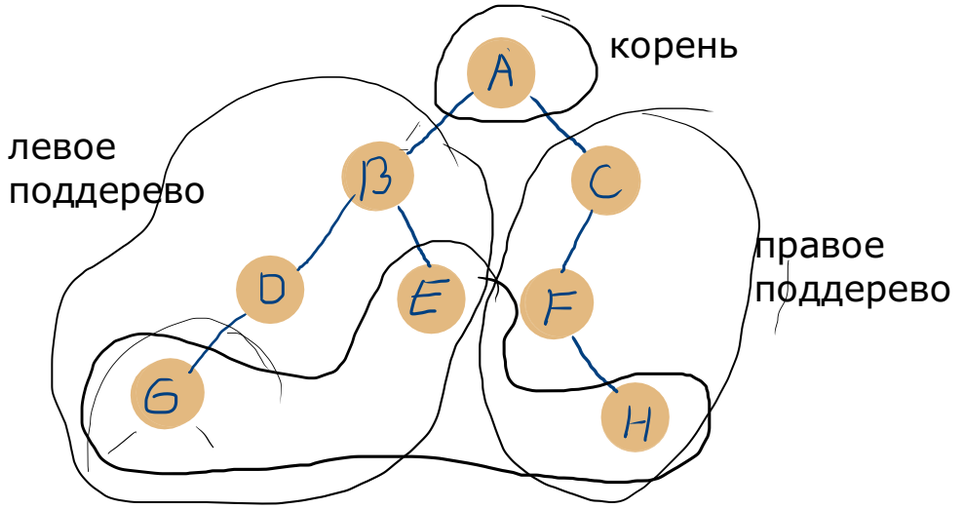
Двоичное дерево

Бинарное дерево - конечное множество узлов, которое:
- пусто (пустое дерево)
- состоит из трех непересекающихся множеств узлов:
  - корневой
  - левое поддерево
  - правое поддерево



Терминология:

B родитель D, E

D левый ребенок B

E правый ребенок B

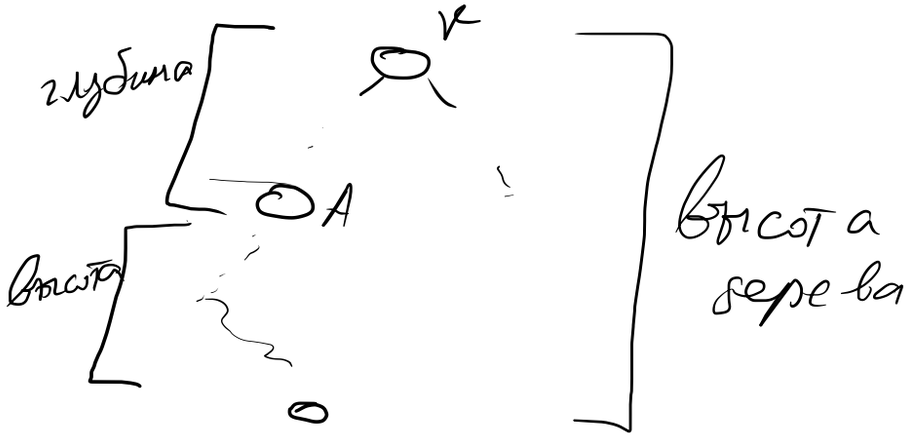
G, D, E потомки узла B

B, A предки узла D

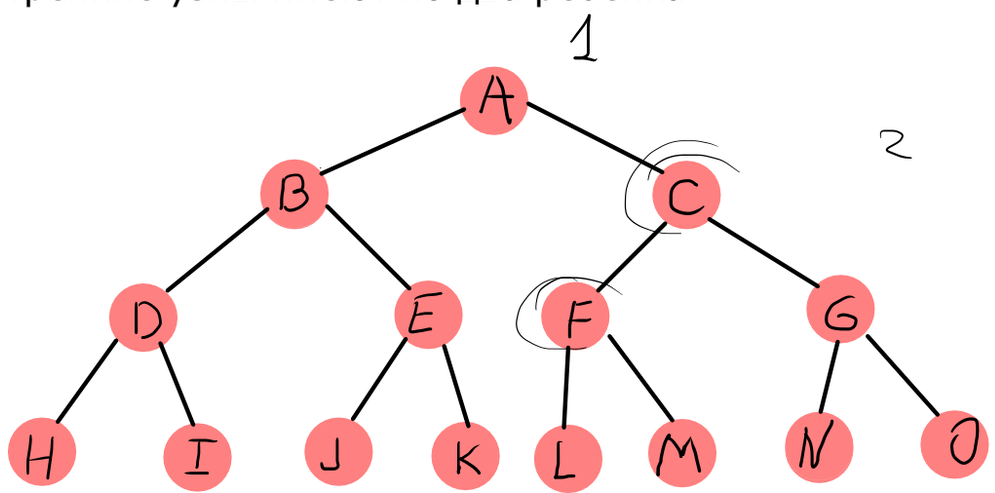
Глубина узла - число ребер на пути от корня к узлу

Высота узла - число ребер в самом длинном пути от узла до листа

Высота дерева - высота корня

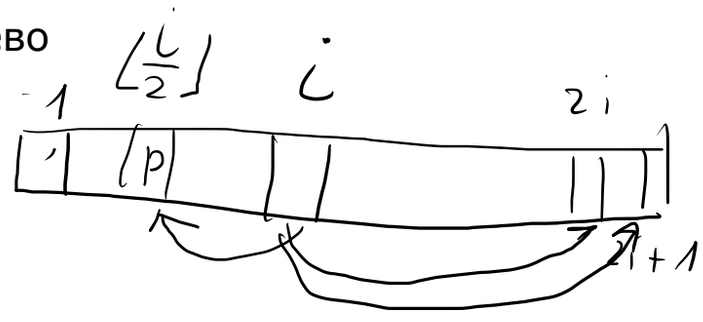


Полное двоиное дерево:
- все листья имеют одну и ту же глубину
- все внутренние узлы имеют по два ребенка



Двоичная куча

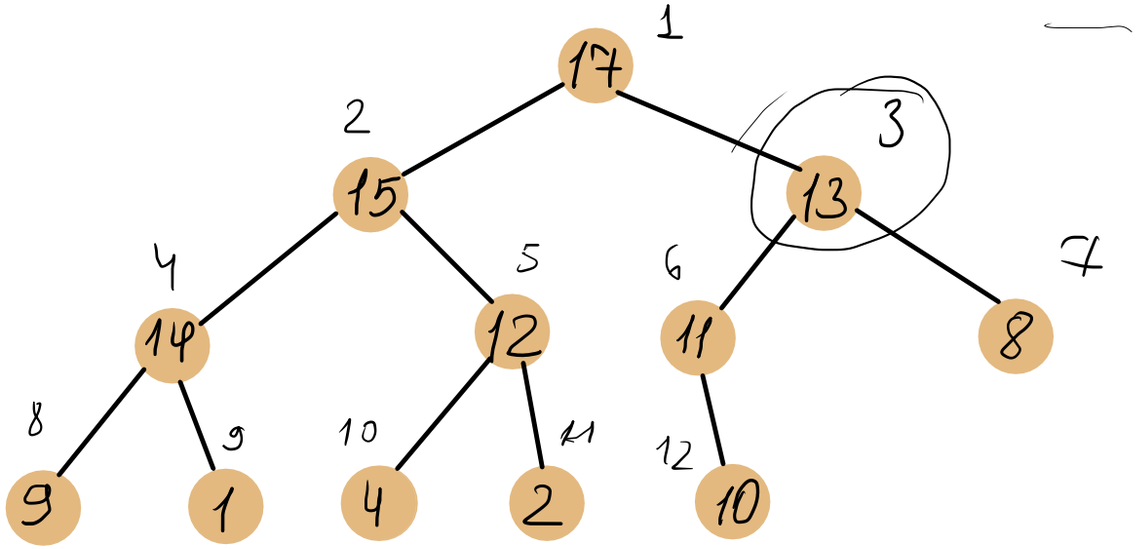
Пирамида, binary heap - массив, почти полное двоичное дерево



Хранение: A[1] - корень дерева
parent(i) - индекс родителя $\lfloor \frac{i}{2} \rfloor$
left(i) - индекс левого ребенка $2i$
right(i) - индекс правого ребенка $2i+1$

2 типа кучи:

- невозрастающся $A[parent(i)] \geq A[i]$
- неубывающая $A[parent(i)] \leq A[i]$



17	15	13	14	12	11	8	9	1	4	2	10

тип - невозрастающая

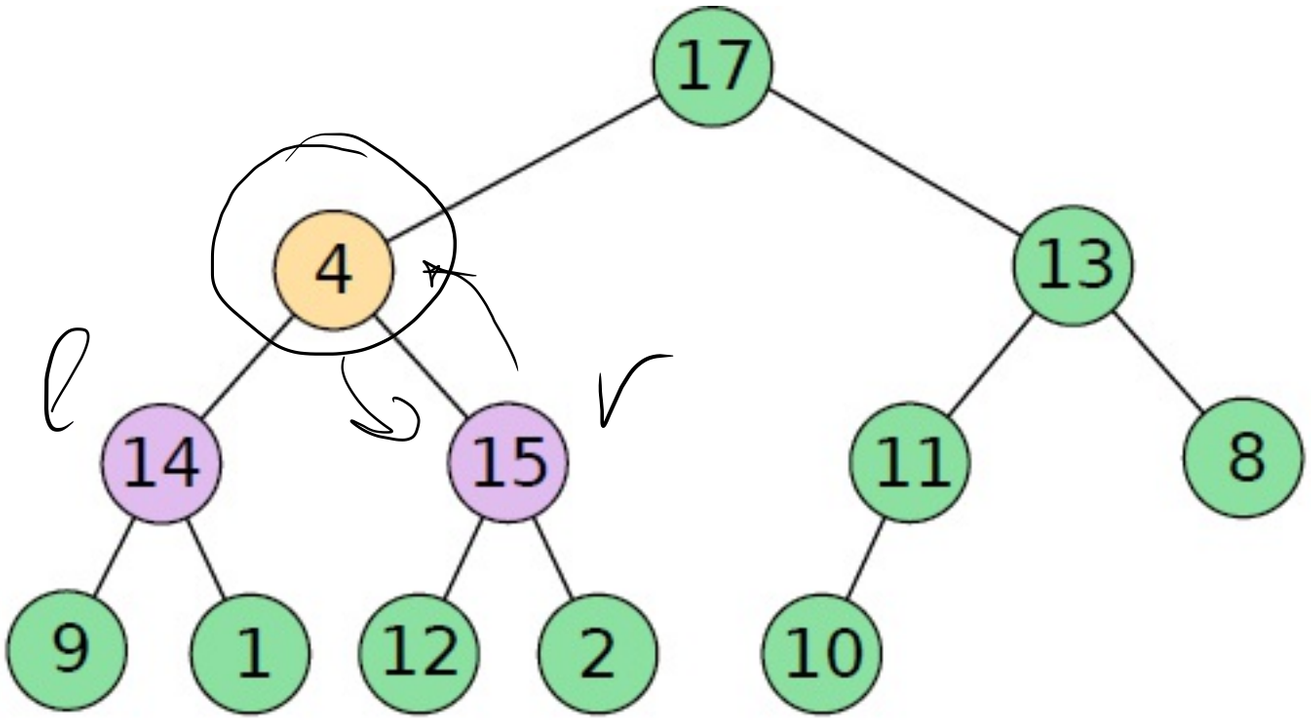
$i = 3$ $A[3] = 13$

$A[parent(i)] = A[\lfloor \frac{3}{2} \rfloor] = A[1] = 17$

$A[2i] = 11$

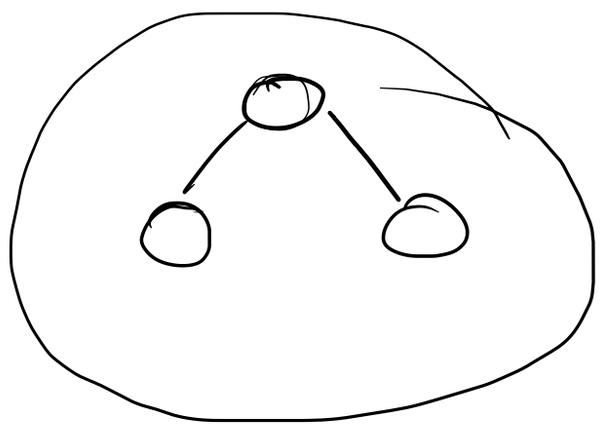
$A[2i+1] = 8$

Восстановление свойства невозрастания кучи

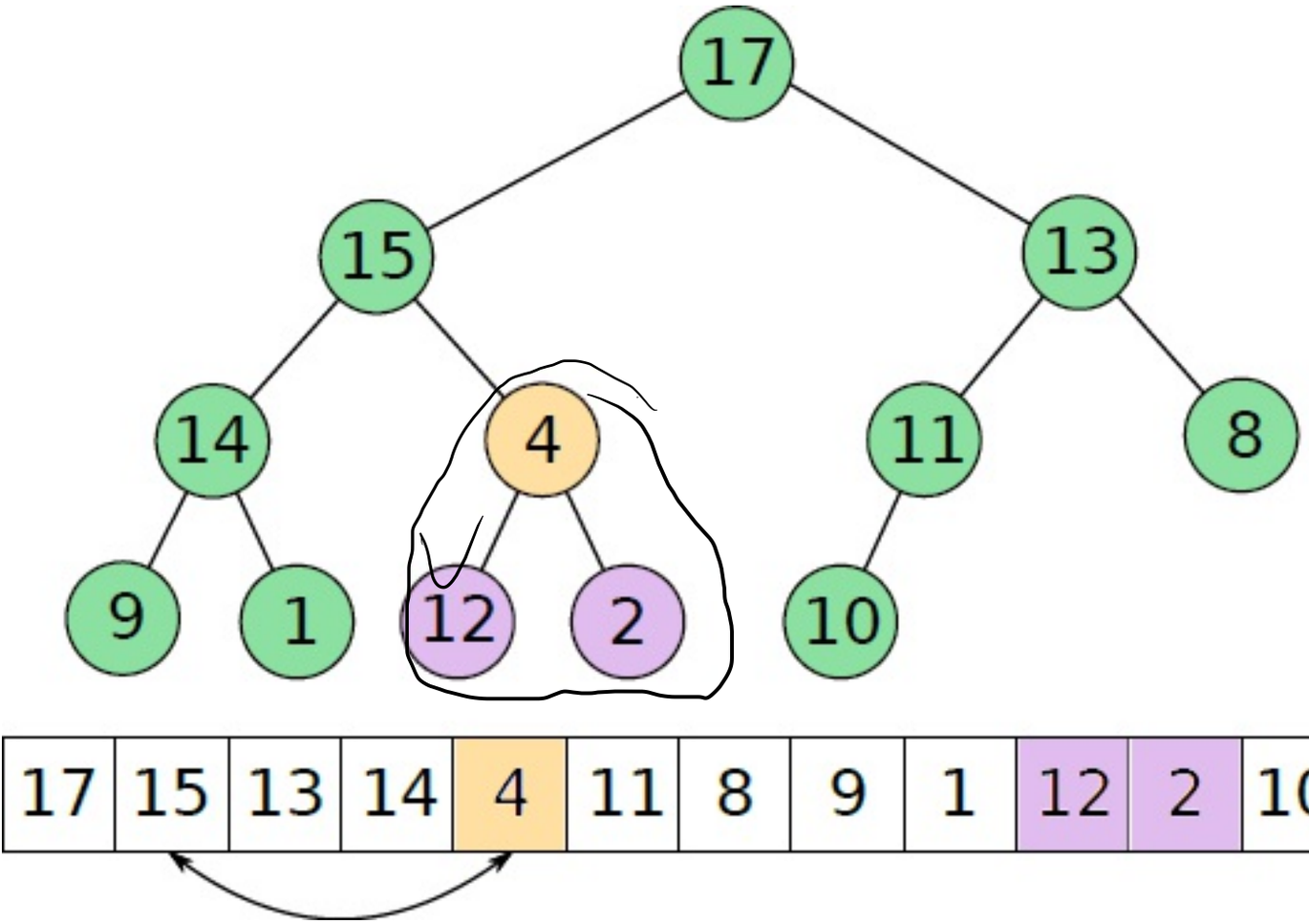


17	4	13	14	15	11	8	9	1	12	2	10
1	2	3	4	5							

```
MAX-HEAPIFY(A, i)
1  l = LEFT(i)
2  r = RIGHT(i)
3  if l ≤ A.heap-size и A[l] > A[i]
4        largest = l
5  else largest = i
6  if r ≤ A.heap-size и A[r] > A[largest]
7        largest = r
8  if largest ≠ i
9        Обменять A[i] и A[largest]
10       MAX-HEAPIFY(A, largest)
```



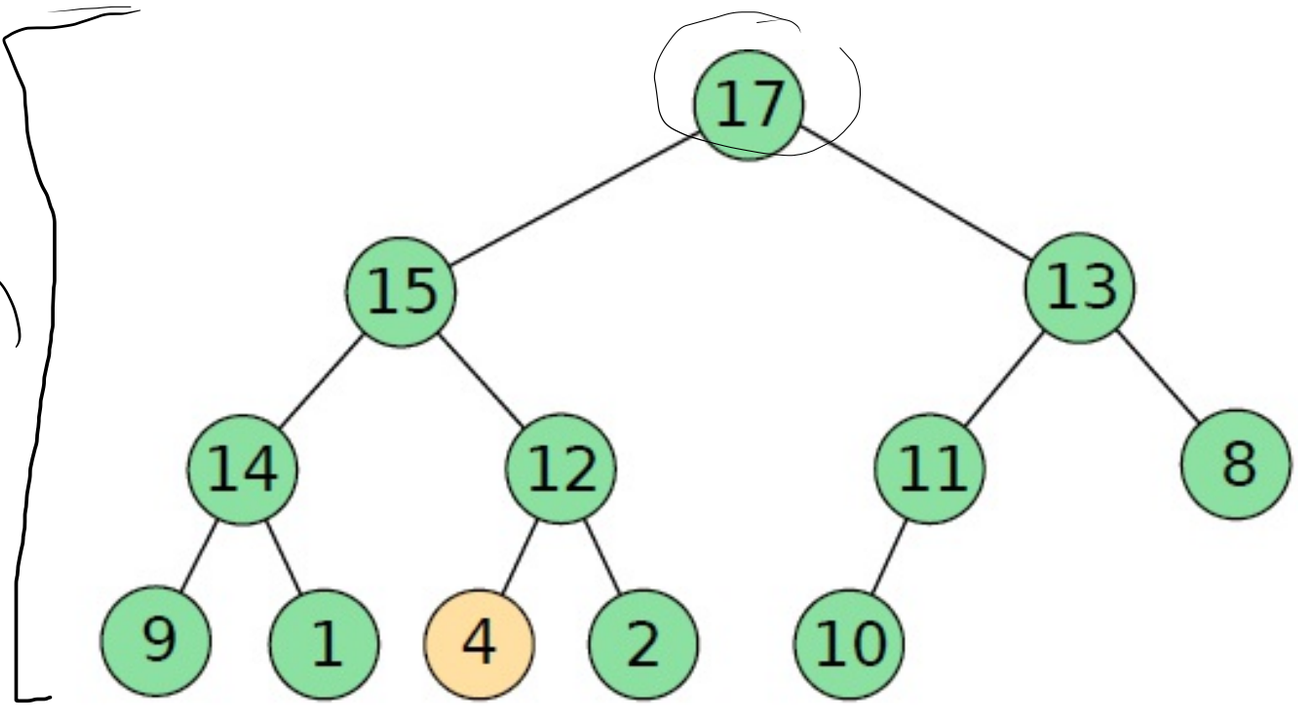
Восстановление свойства невозрастания кучи



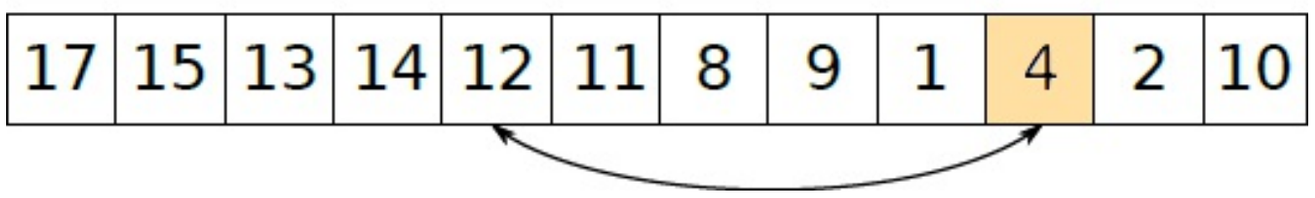

```
MAX-HEAPIFY(A, i)
 1  l = LEFT(i)
 2  r = RIGHT(i)
 3  if l ≤ A.heap-size и A[l] > A[i]
 4        largest = l
 5  else largest = i
 6  if r ≤ A.heap-size и A[r] > A[largest]
 7        largest = r
 8  if largest ≠ i
 9        Обменять A[i] и A[largest]
10        MAX-HEAPIFY(A, largest)
```

Восстановление свойства невозрастания кучи

$O(\lg(n))$



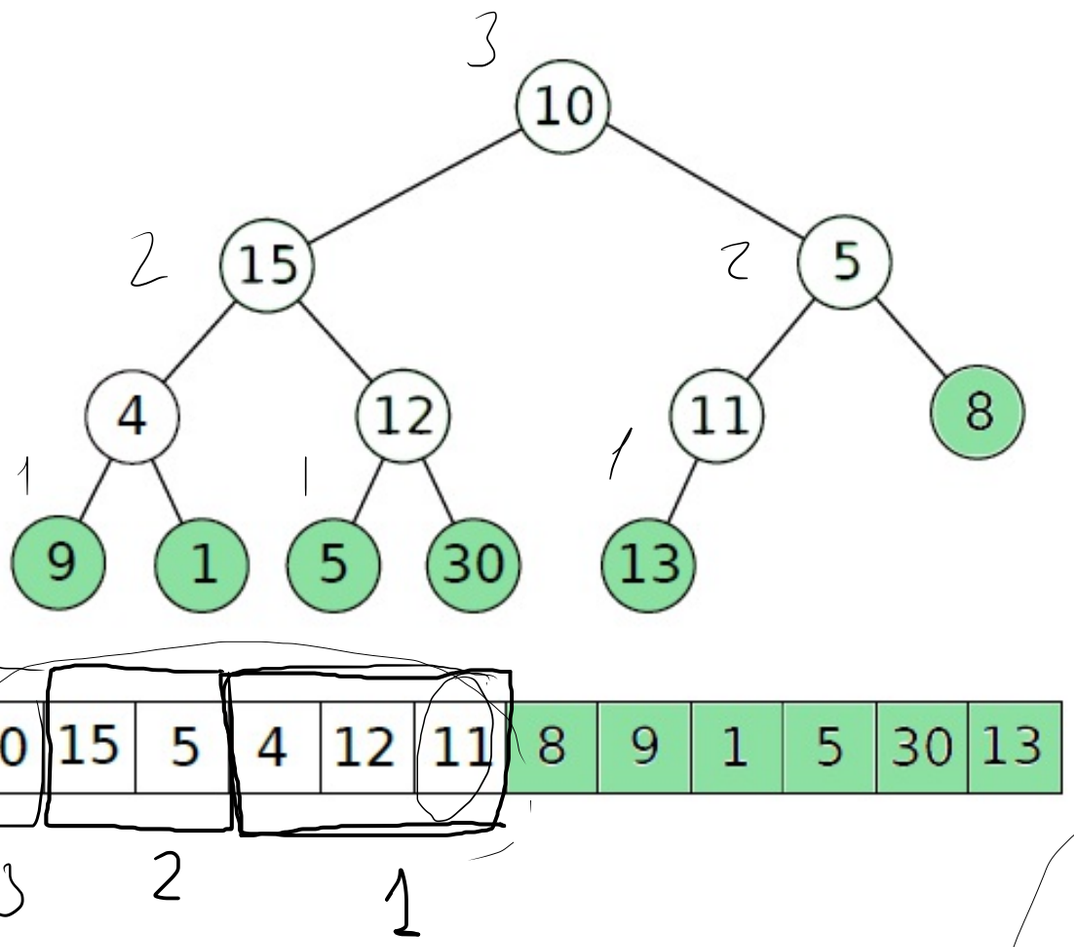
$\lg(n)$




```
MAX-HEAPIFY(A, i)
 1  l = LEFT(i)
 2  r = RIGHT(i)
 3  if l ≤ A.heap-size и A[l] > A[i]
 4      largest = l
 5  else largest = i
 6  if r ≤ A.heap-size и A[r] > A[largest]
 7      largest = r
 8  if largest ≠ i
 9      Обменять A[i] и A[largest]
10      MAX-HEAPIFY(A, largest)
```

O(высота поддерева)

Создание кучи



BUILD-MAX-HEAP(A)

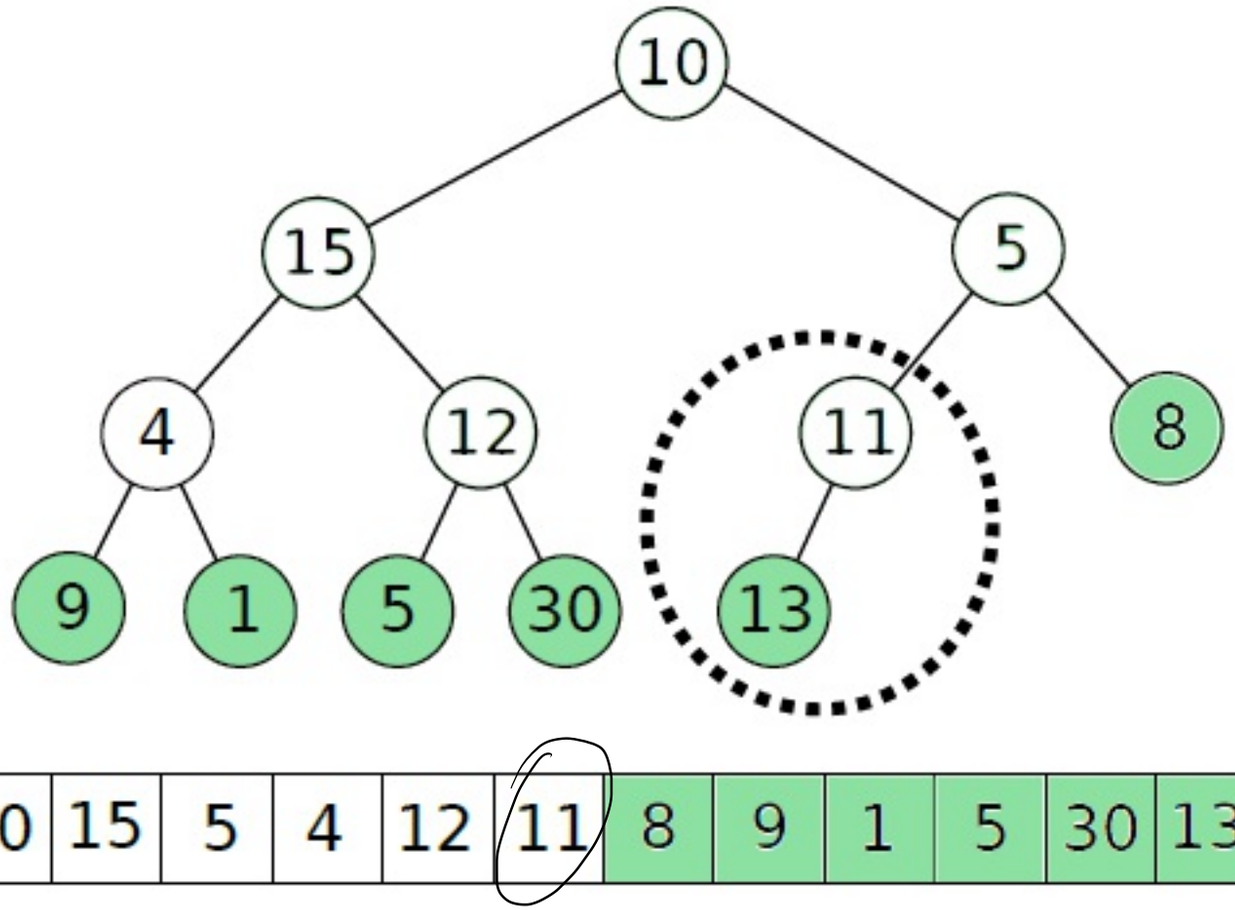
1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

$\frac{n}{2} \cdot 0 + \frac{n}{4} \cdot 1 + \frac{n}{8} \cdot 2 + \frac{n}{16} \cdot 3 + \ldots$

$$\sum_{K=2}^{\log n} \frac{n}{2^k}(k-1) = n$$

Создание кучи

! невозрастает.

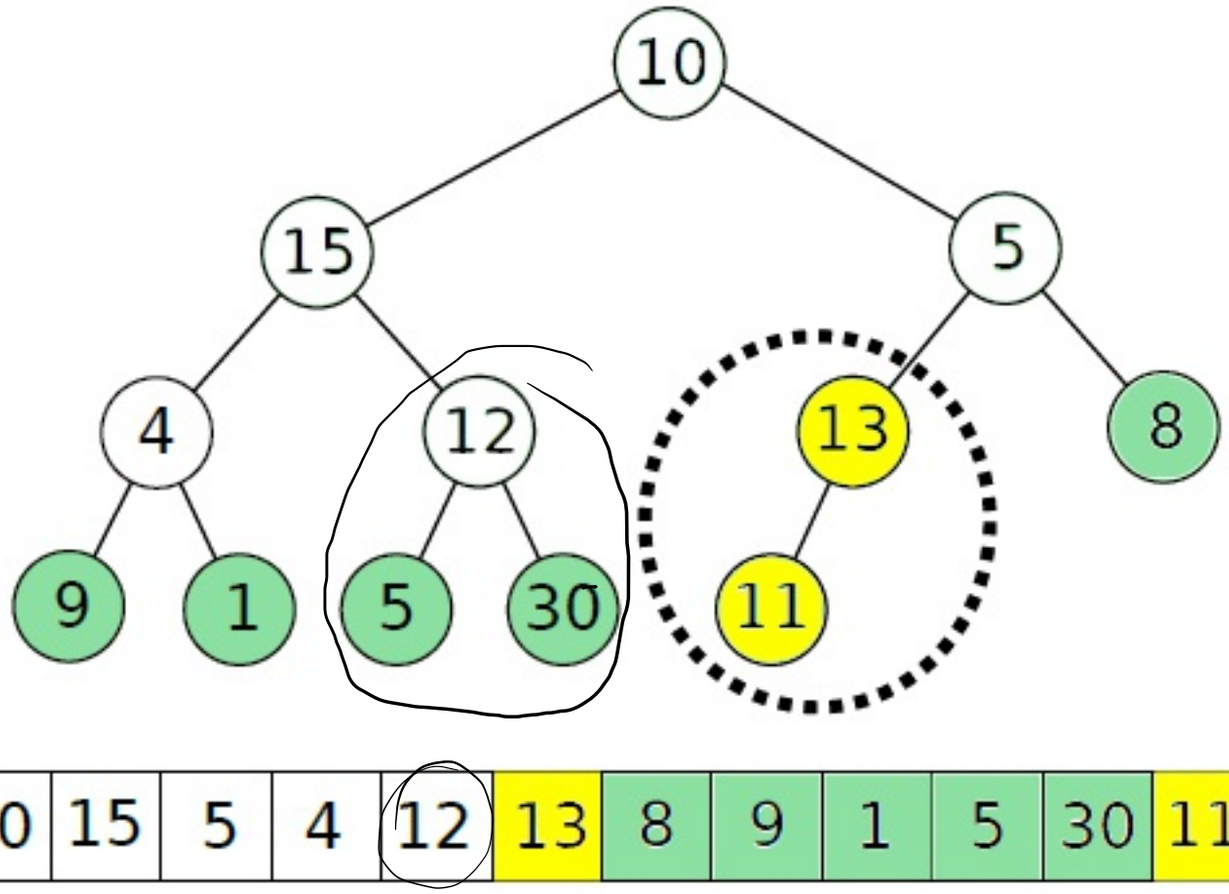


10	15	5	4	12	11	8	9	1	5	30	13

BUILD-MAX-HEAP(A)

1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

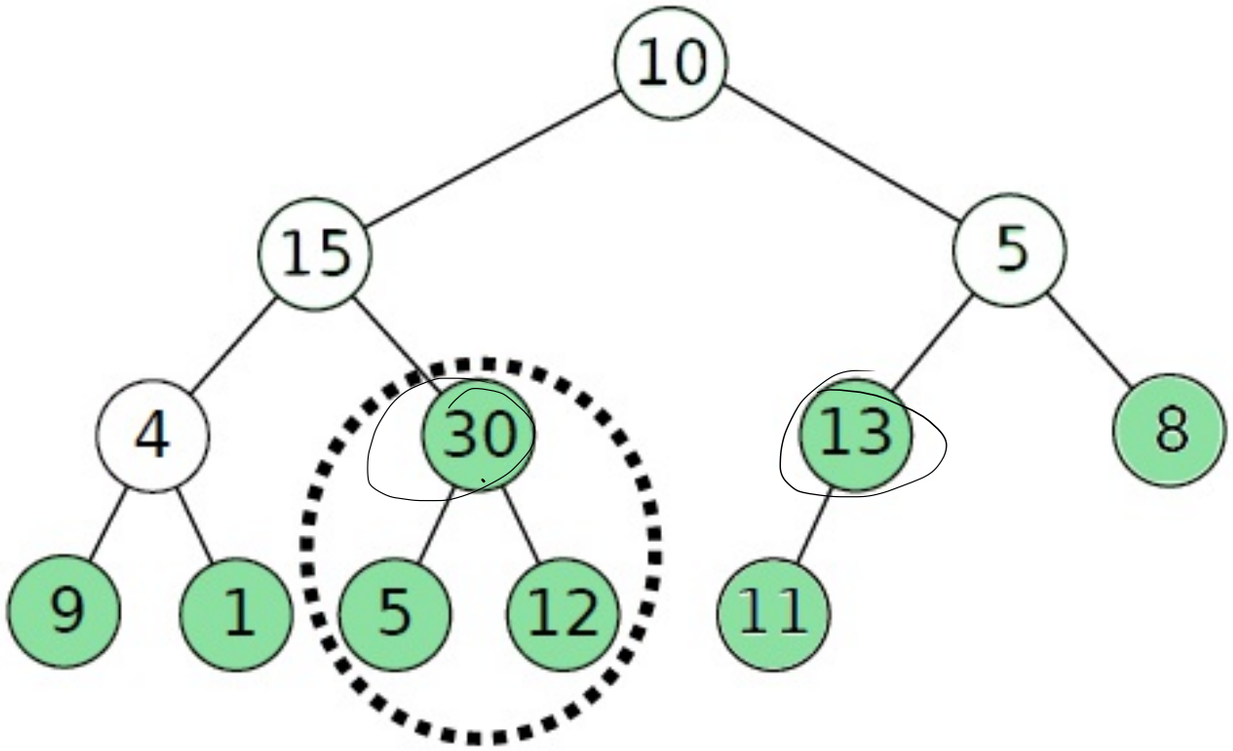
Создание кучи



BUILD-MAX-HEAP(A)

1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

Создание кучи

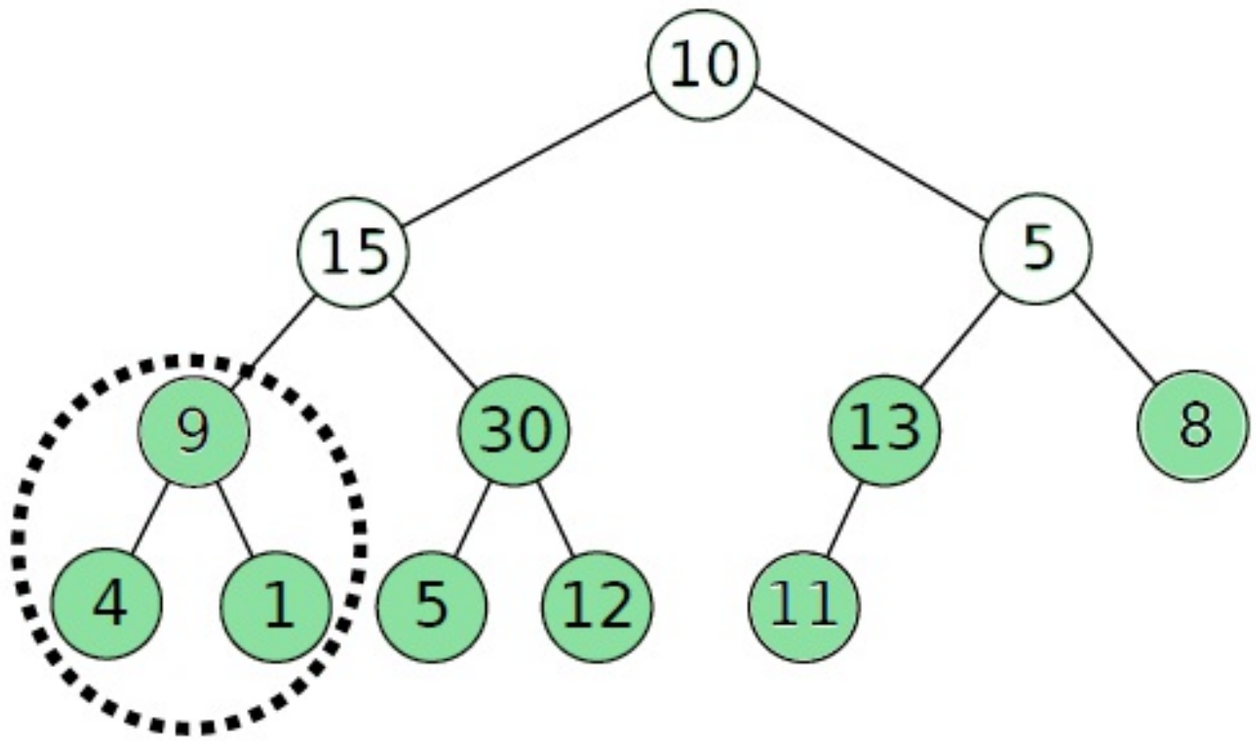


10	15	5	4	30	13	8	9	1	5	12	11

BUILD-MAX-HEAP(A)

1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

Создание кучи

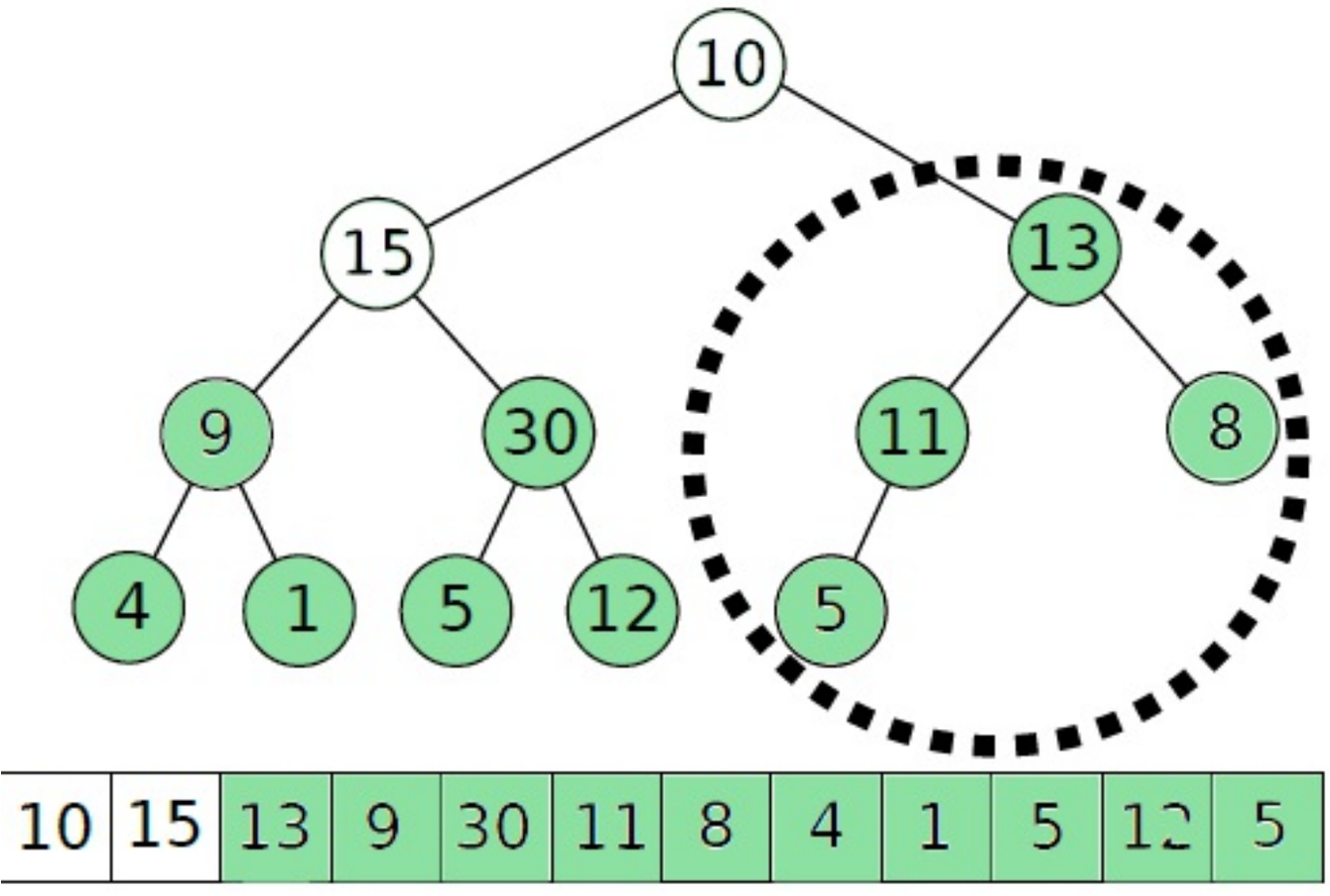


10	15	5	9	30	13	8	4	1	5	12	11

BUILD-MAX-HEAP(A)

1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

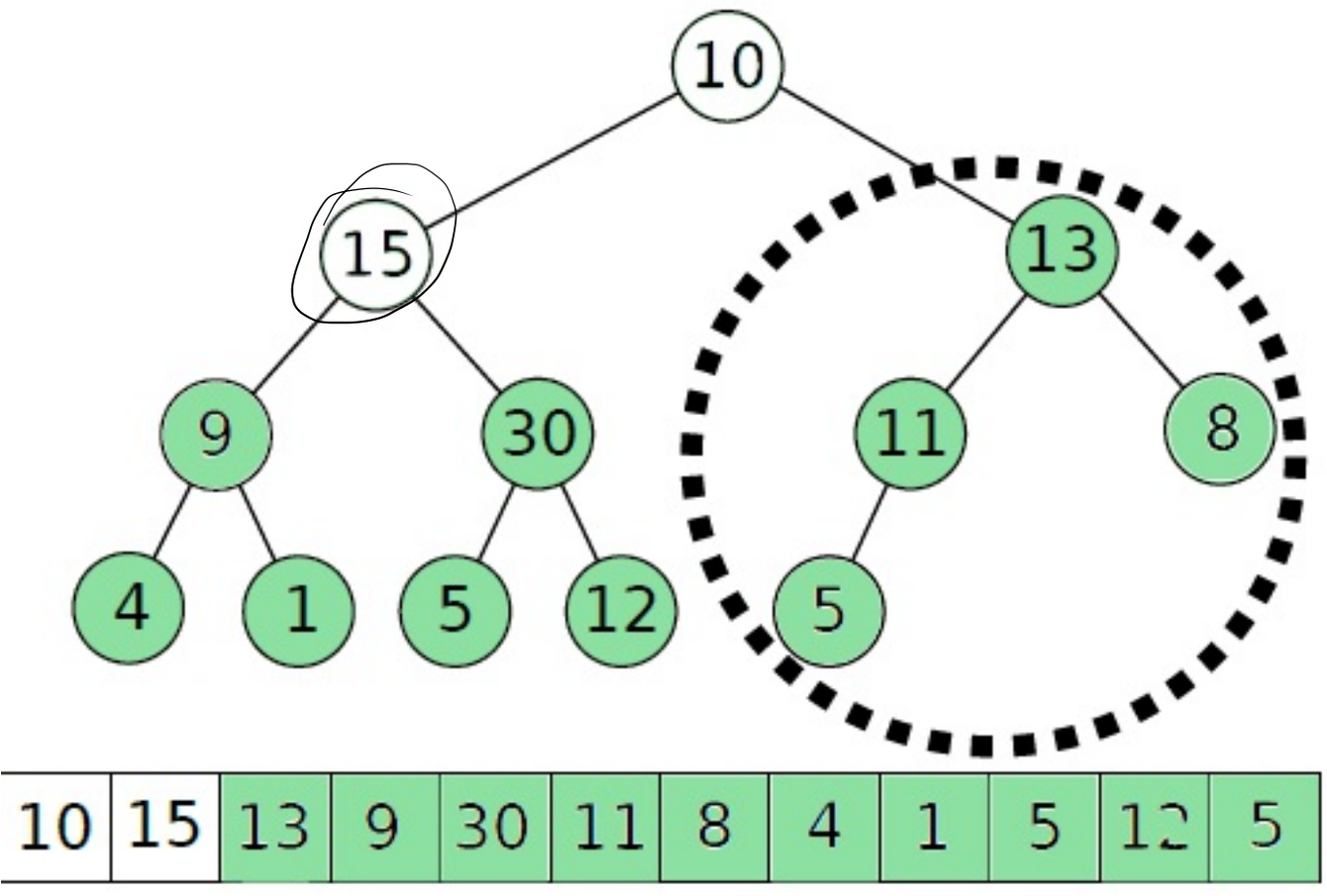
Создание кучи



BUILD-MAX-HEAP(A)

1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

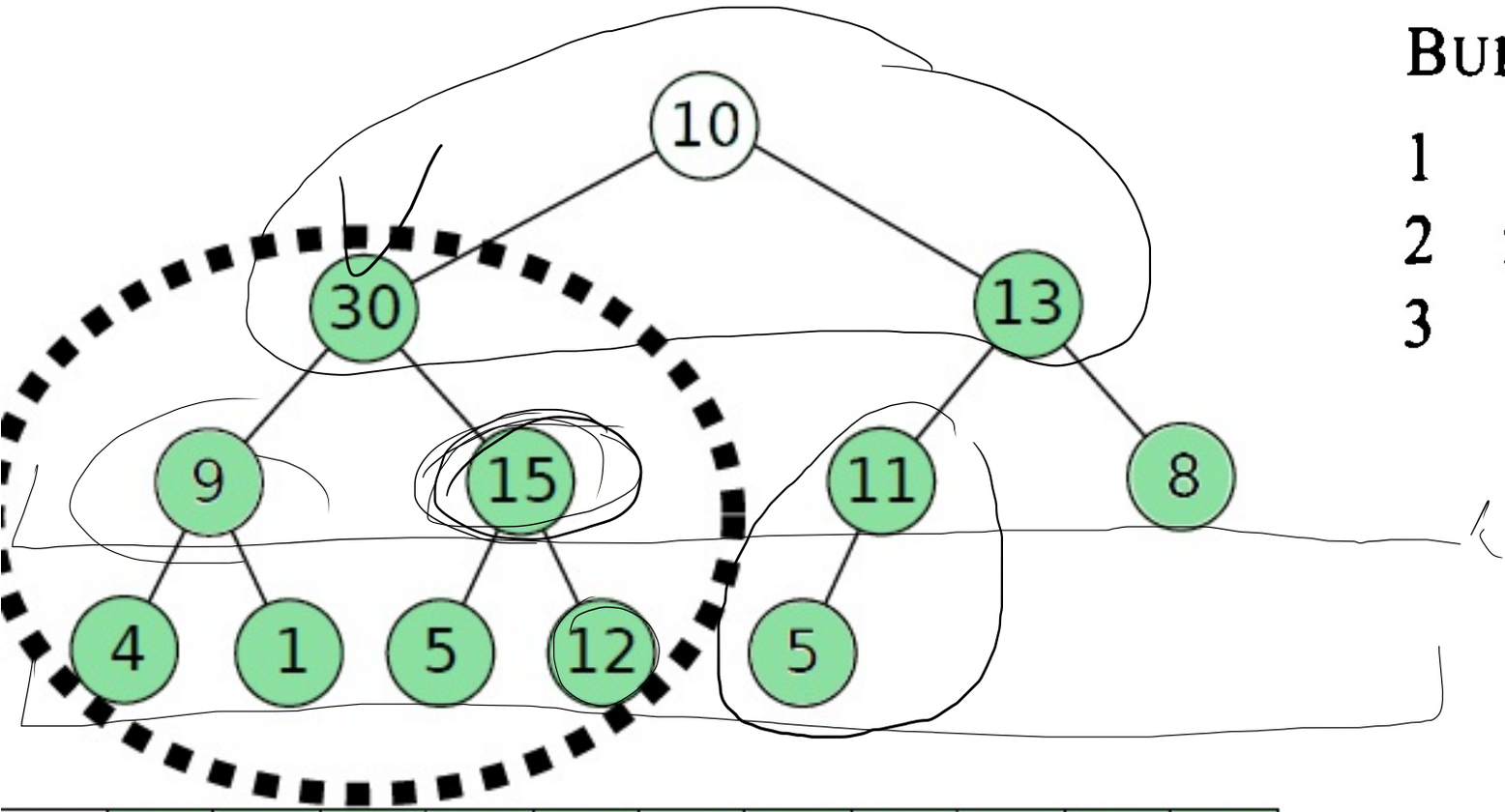
Создание кучи



BUILD-MAX-HEAP(A)

1 $A.heap\text{-}size = A.length$
2 **for** $i = \lfloor A.length/2 \rfloor$ **downto** 1
3 MAX-HEAPIFY(A, i)

Создание кучи



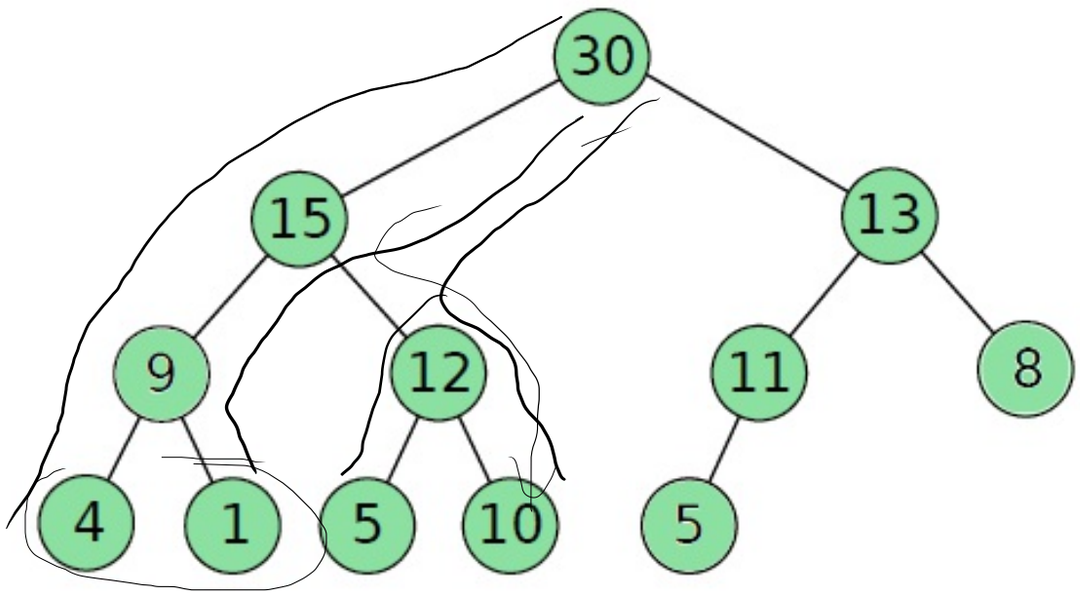
10	30	13	9	15	11	8	4	1	5	12	5

```
BUILD-MAX-HEAP(A)
1  A.heap-size = A.length
2  for i = ⌊A.length/2⌋ downto 1
3      MAX-HEAPIFY(A, i)
```

30

10 13

Создание кучи




```
BUILD-MAX-HEAP(A)
1  A.heap-size = A.length
2  for i = ⌊A.length/2⌋ downto 1
3      MAX-HEAPIFY(A, i)
```

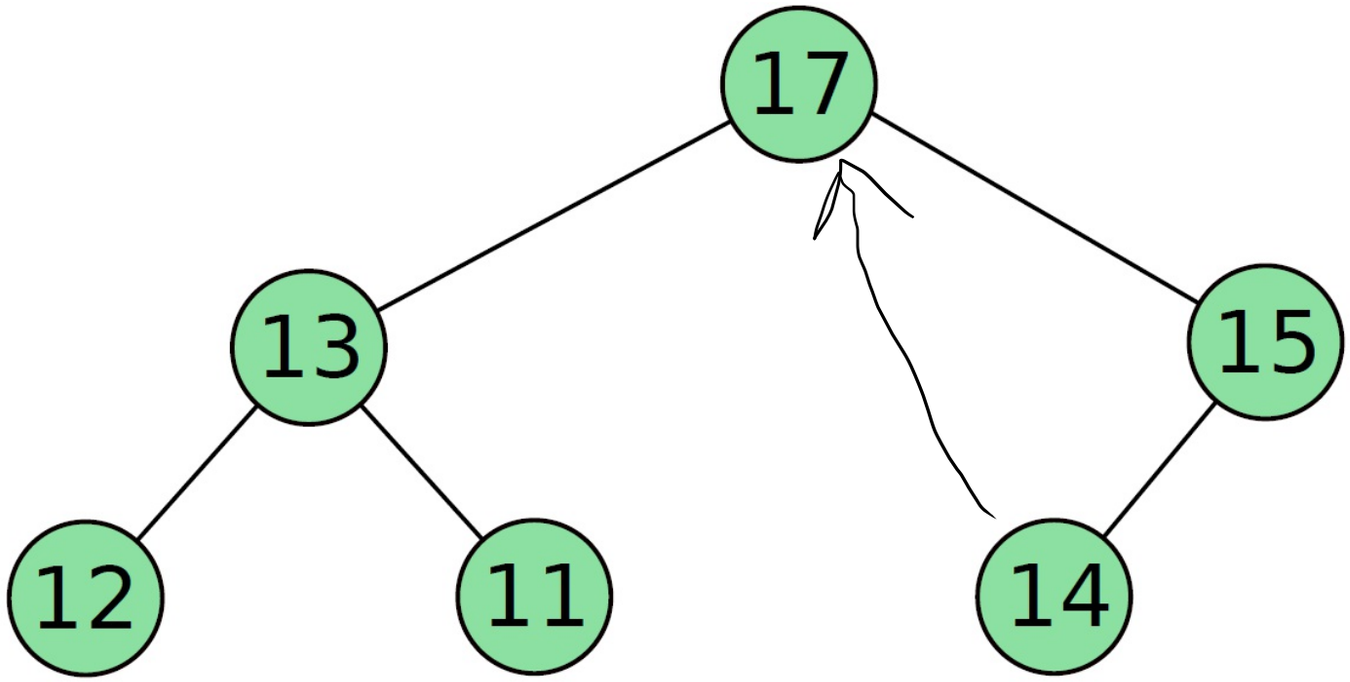
$\frac{n}{2}$

$O(\log n)$

$O(n \log n)$

30	15	13	9	12	11	8	4	1	5	10	5

Сортировка кучей

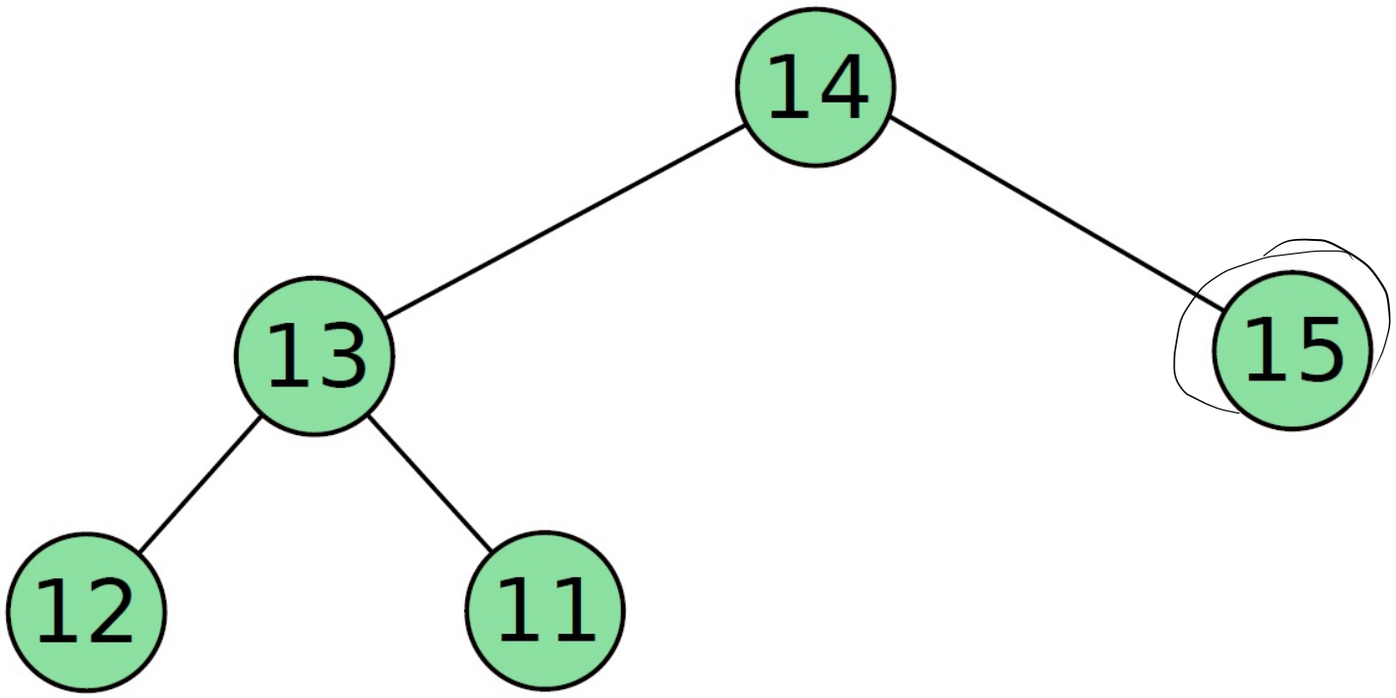


17	13	15	12	11	14

HEAPSORT(A)

1 BUILD-MAX-HEAP(A) $O(n)$
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY($A, 1$)

Сортировка кучей

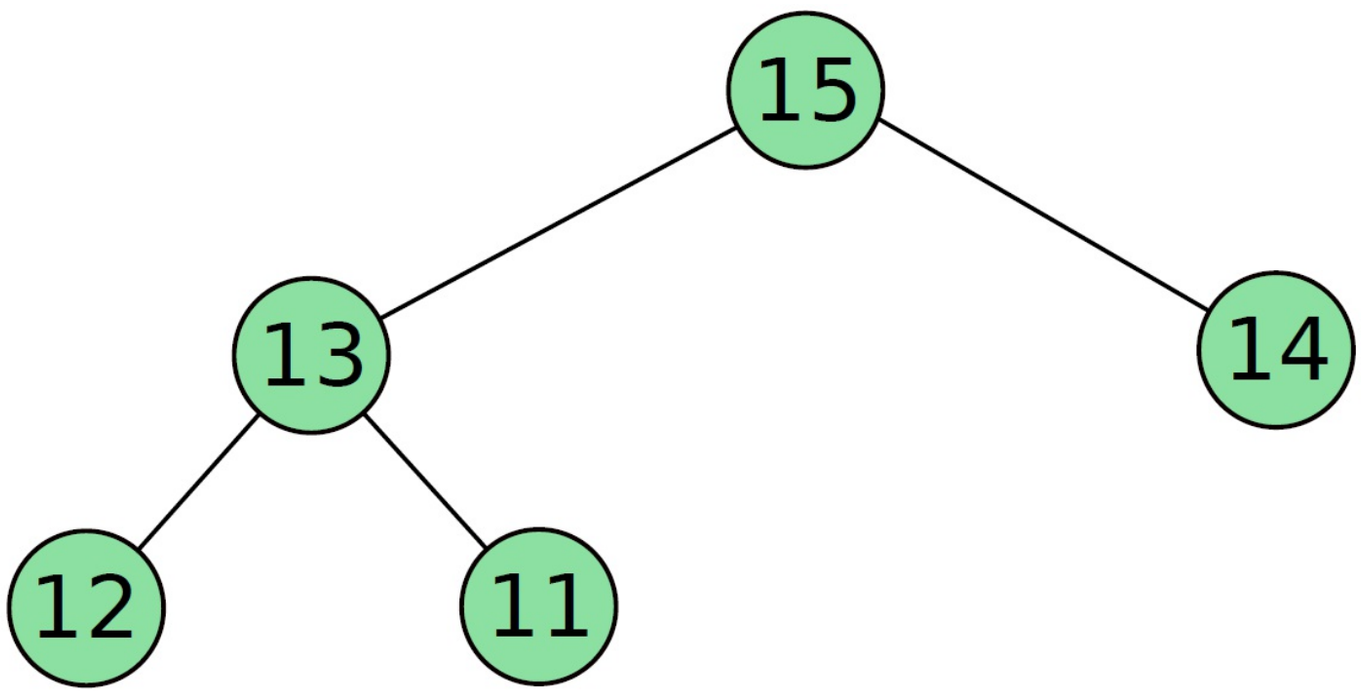


14	13	15	12	11	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY($A, 1$)

Сортировка кучей

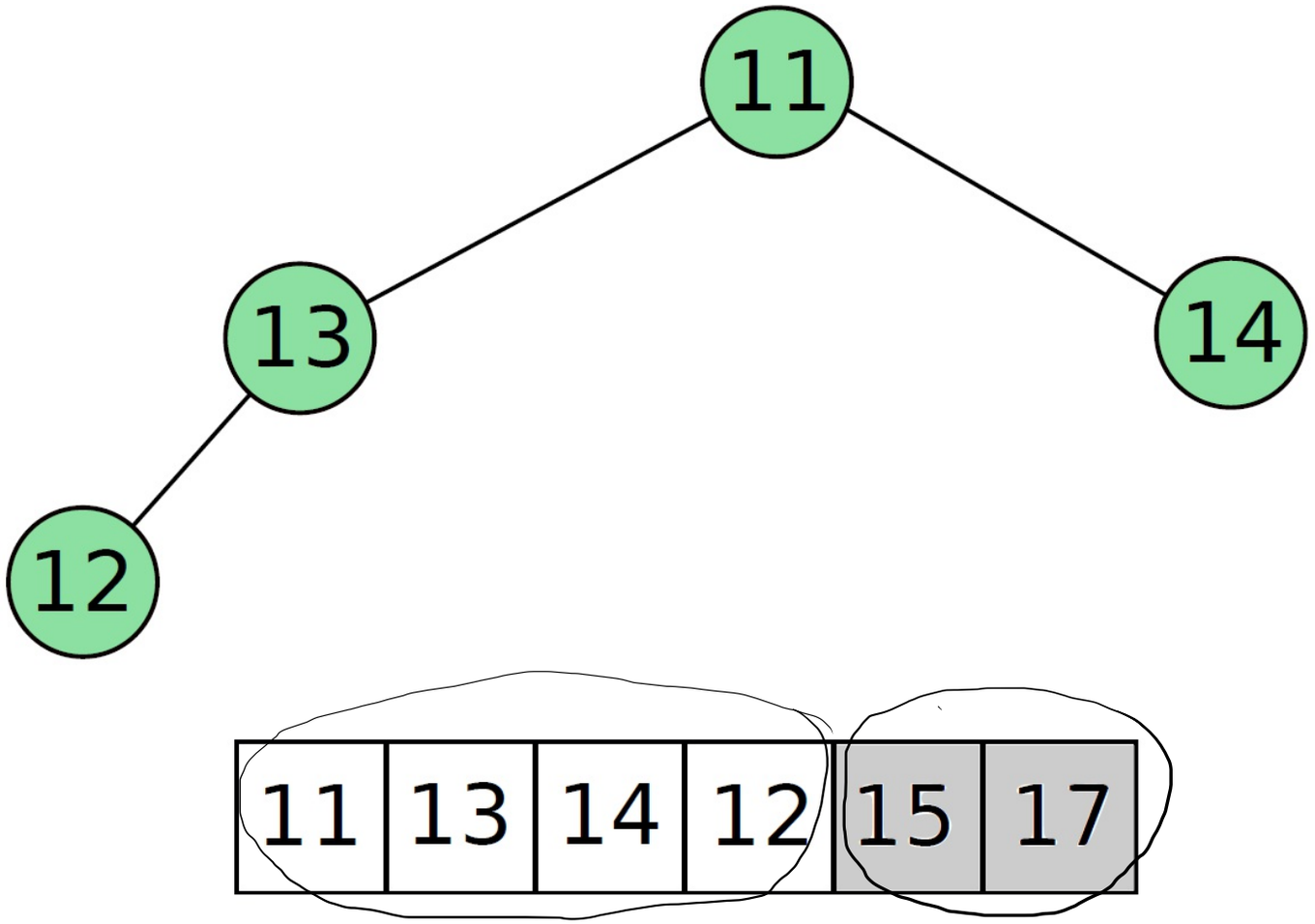


15	13	14	12	11	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY$(A, 1)$

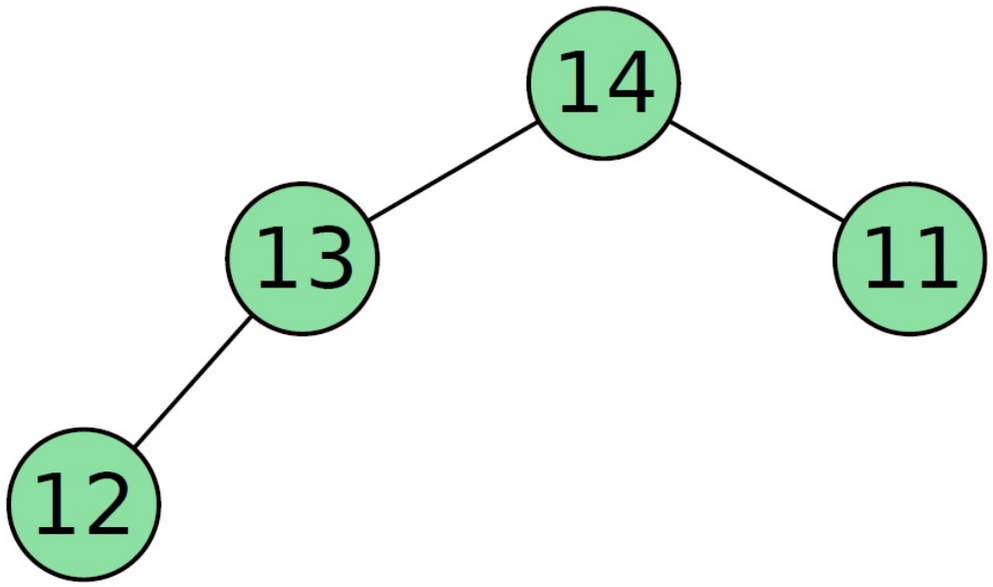
Сортировка кучей



HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY($A, 1$)

Сортировка кучей

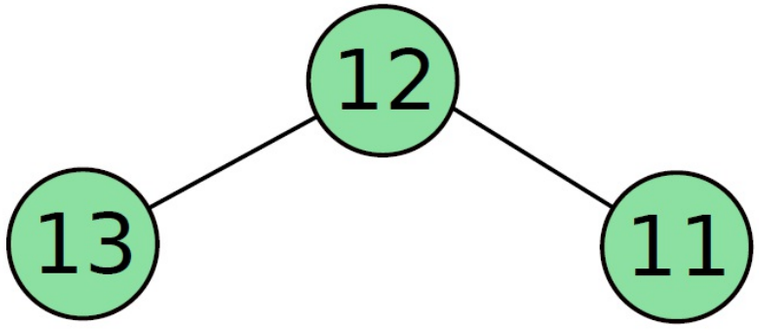


14	13	11	12	15	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY$(A, 1)$

Сортировка кучей

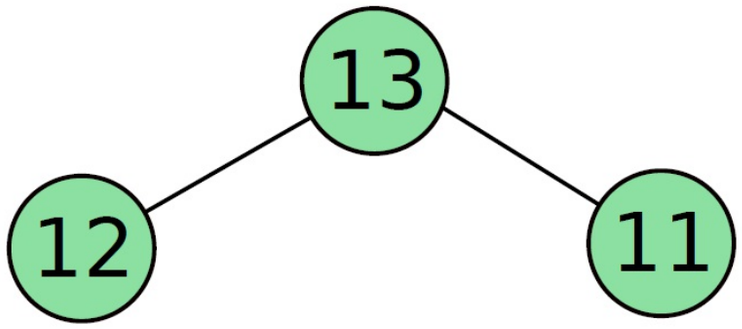


12	13	11	14	15	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY($A, 1$)

Сортировка кучей

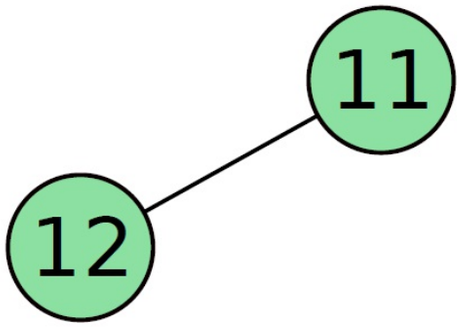


13	12	11	14	15	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY$(A, 1)$

Сортировка кучей

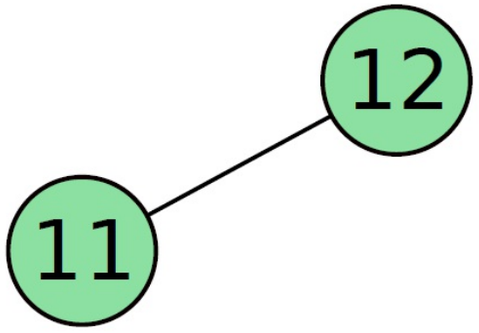


11	12	13	14	15	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY($A, 1$)

Сортировка кучей



12	11	13	14	15	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY$(A, 1)$

Сортировка кучей

11	12	13	14	15	17

HEAPSORT(A)

1 BUILD-MAX-HEAP(A)
2 **for** $i = A.length$ **downto** 2
3 Обменять $A[1]$ с $A[i]$
4 $A.heap\text{-}size = A.heap\text{-}size - 1$
5 MAX-HEAPIFY($A, 1$)

O(n)

n

log n