

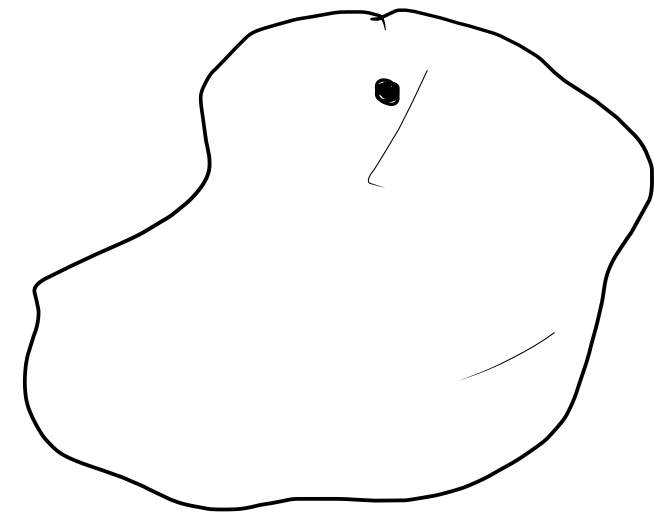
## Очередь с приоритетами на основе кучи

Очередь, которая просматривает (удаляет) самый лучший (с наивысшим приоритетом) элемент

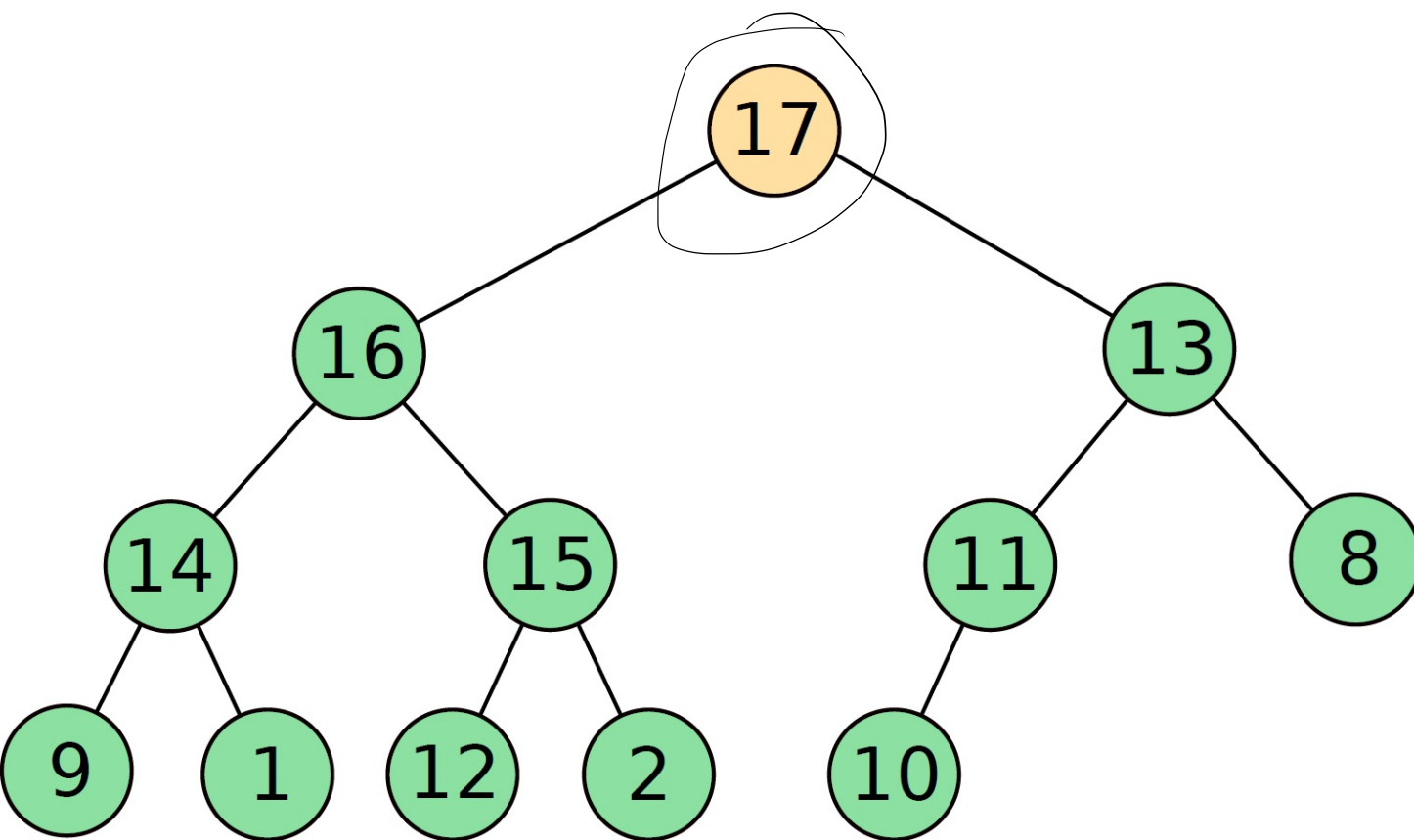
Каждый элемент содержит ключ, который определяет его приоритет

Операции:

- 1) **Heap\_Maximum(A)** - сообщает ключ элемента с наибольшим приоритетом
- 2) **Extract\_Max(A)** - извлекает из структуры элемент с наибольшим приоритетом
- 3) **Increase\_Key(A, i, key)** - меняет приоритет элемента A(i) на значение key
- 4) **Insert(A, key)** - добавляет в кучу новый элемент с приоритетом key



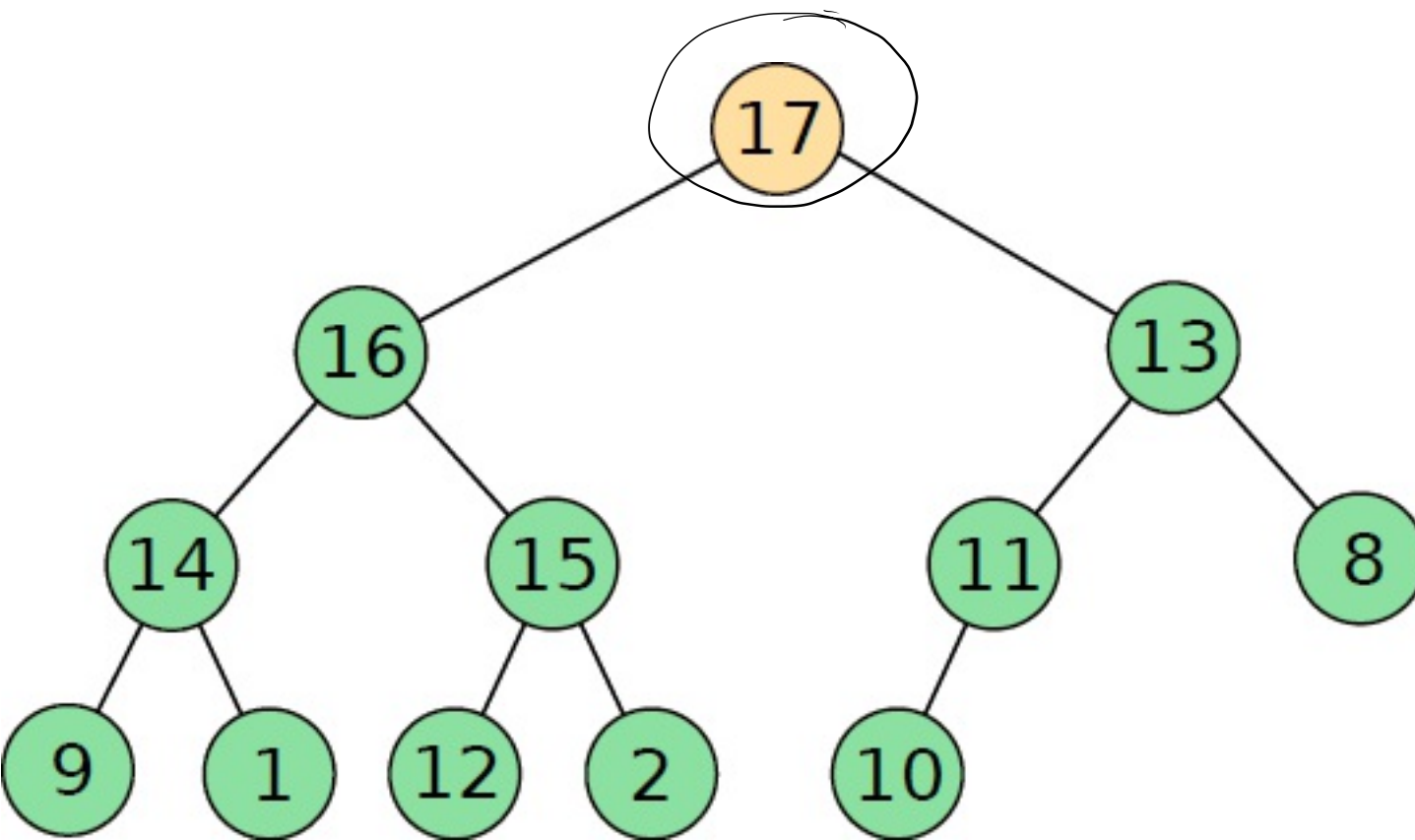
Очередь с приоритетами на основе кучи: Heap\_Maximum(A)



HEAP-MAXIMUM( $A$ )

1 **return**  $A[1]$

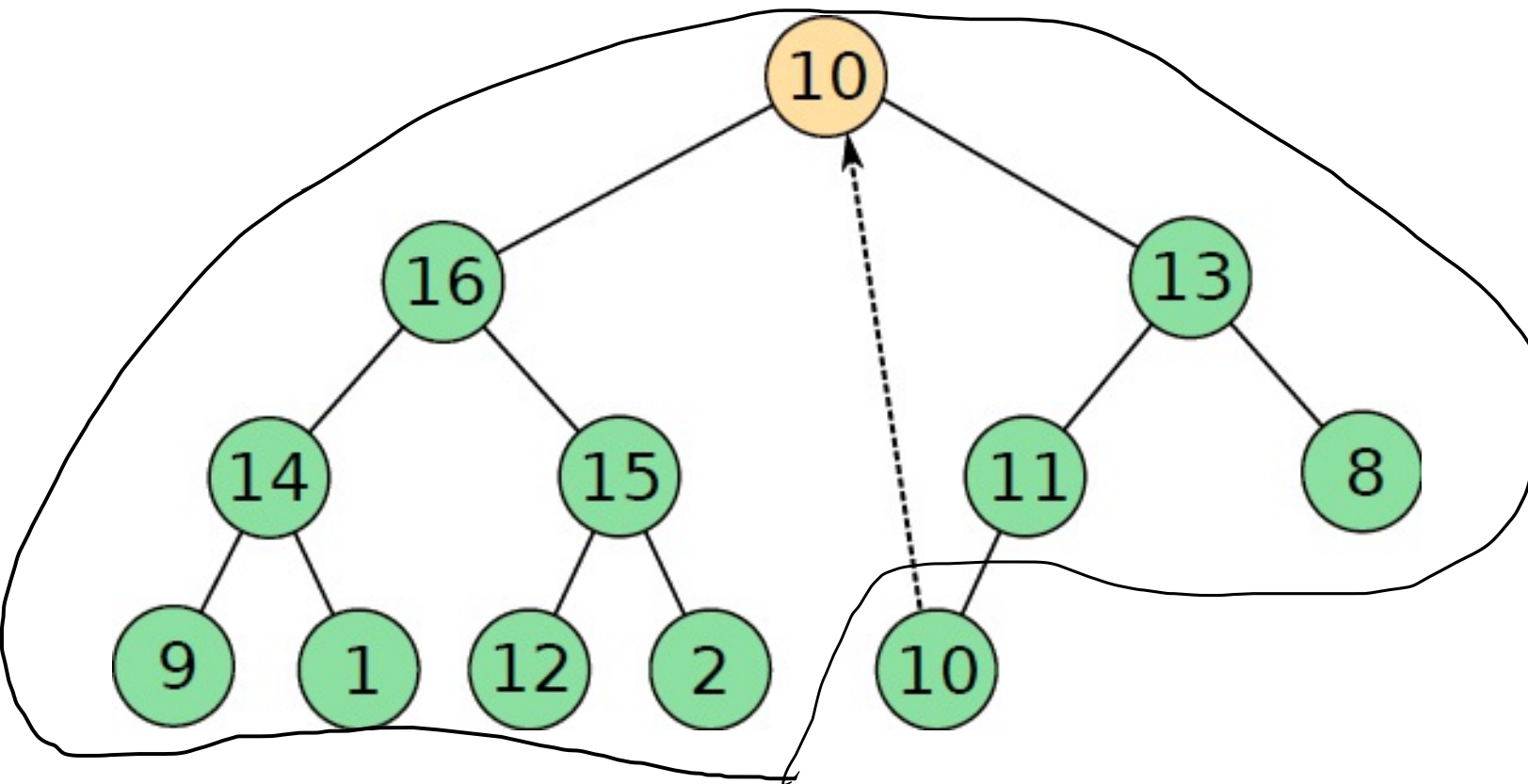
## Очередь с приоритетами на основе кучи: Extract\_Max(A)



HEAP-EXTRACT-MAX( $A$ )

```
1  if  $A.heap-size < 1$ 
2      error "Очередь пуста"
3   $max = A[1]$ 
4   $A[1] = A[A.heap-size]$ 
5   $A.heap-size = A.heap-size - 1$ 
6  MAX-HEAPIFY( $A, 1$ )
7  return  $max$ 
```

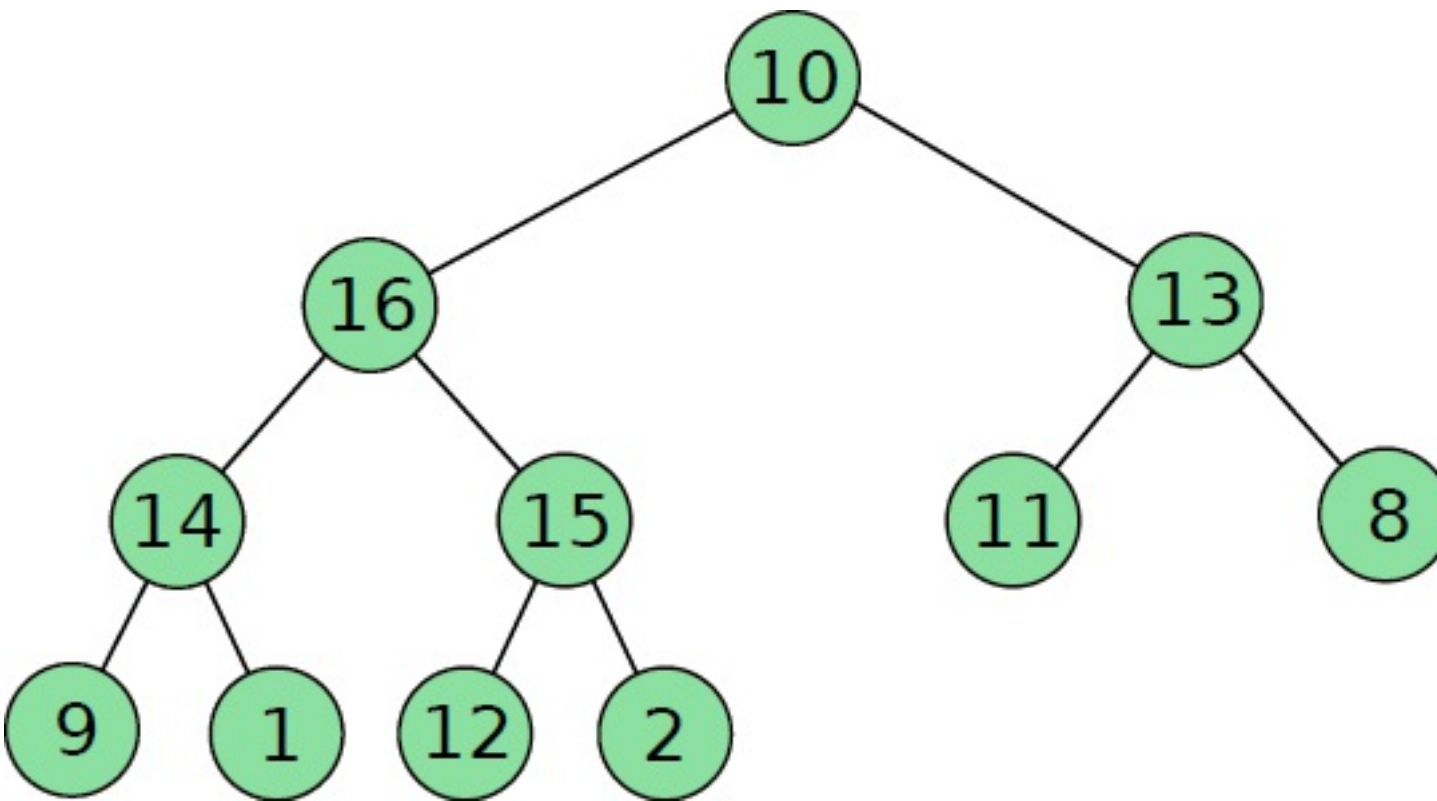
## Очередь с приоритетами на основе кучи: Extract\_Max(A)



HEAP-EXTRACT-MAX( $A$ )

```
1  if  $A.heap-size < 1$ 
2      error "Очередь пуста"
3   $max = A[1]$ 
4   $A[1] = A[A.heap-size]$ 
5   $A.heap-size = A.heap-size - 1$ 
6  MAX-HEAPIFY( $A, 1$ )
7  return  $max$ 
```

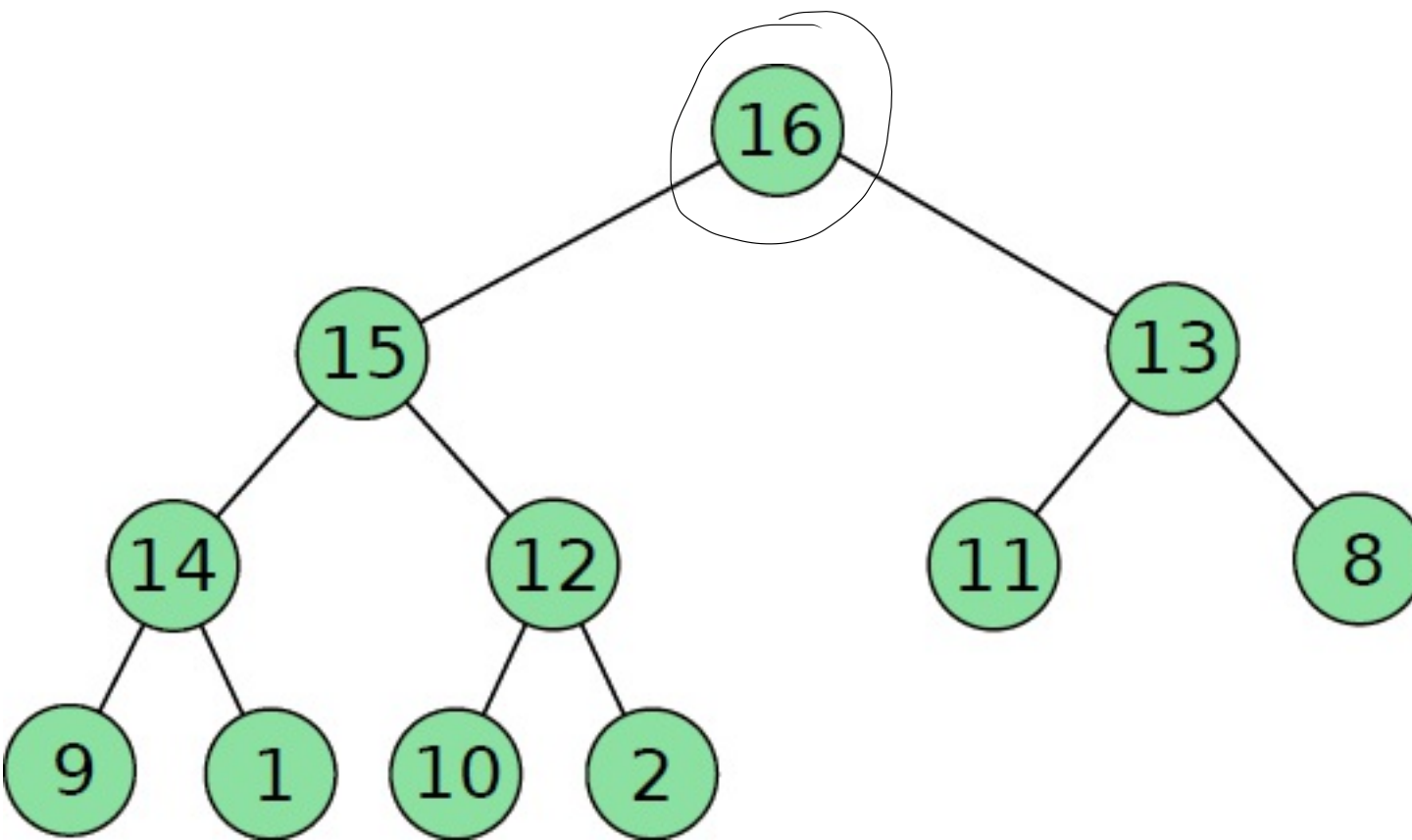
## Очередь с приоритетами на основе кучи: Extract\_Max(A)



HEAP-EXTRACT-MAX( $A$ )

```
1  if  $A.heap-size < 1$ 
2      error “Очередь пуста”
3   $max = A[1]$ 
4   $A[1] = A[A.heap-size]$ 
5   $A.heap-size = A.heap-size - 1$ 
6  MAX-HEAPIFY( $A, 1$ )
7  return  $max$ 
```

## Очередь с приоритетами на основе кучи: Extract\_Max(A)

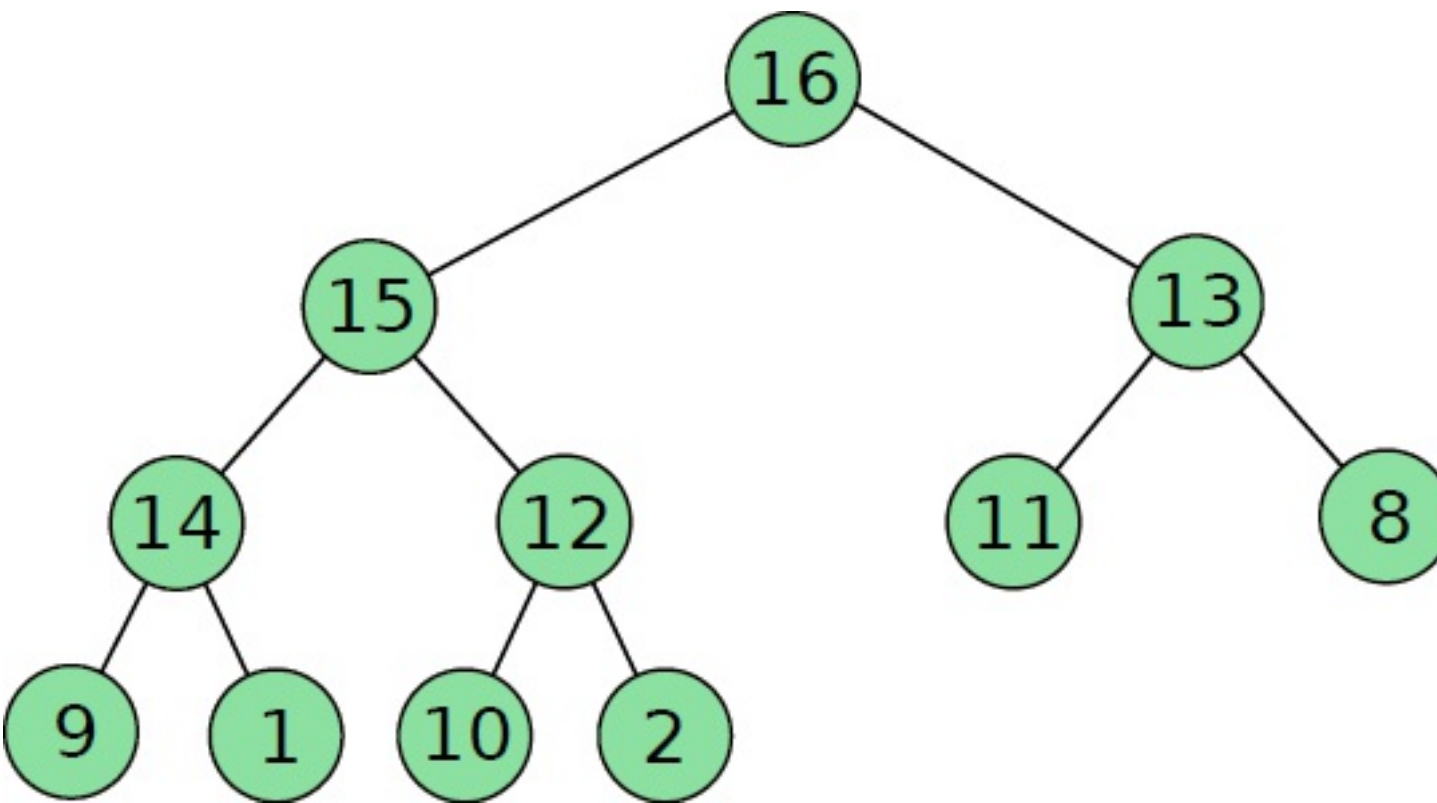


HEAP-EXTRACT-MAX(*A*)

```
1  if A.heap-size < 1
2      error "Очередь пуста"
3  max = A[1]
4  A[1] = A[A.heap-size]
5  A.heap-size = A.heap-size - 1
6  MAX-HEAPIFY(A, 1)
7  return max
```

$\log n$

## Очередь с приоритетами на основе кучи: Extract\_Max(A)

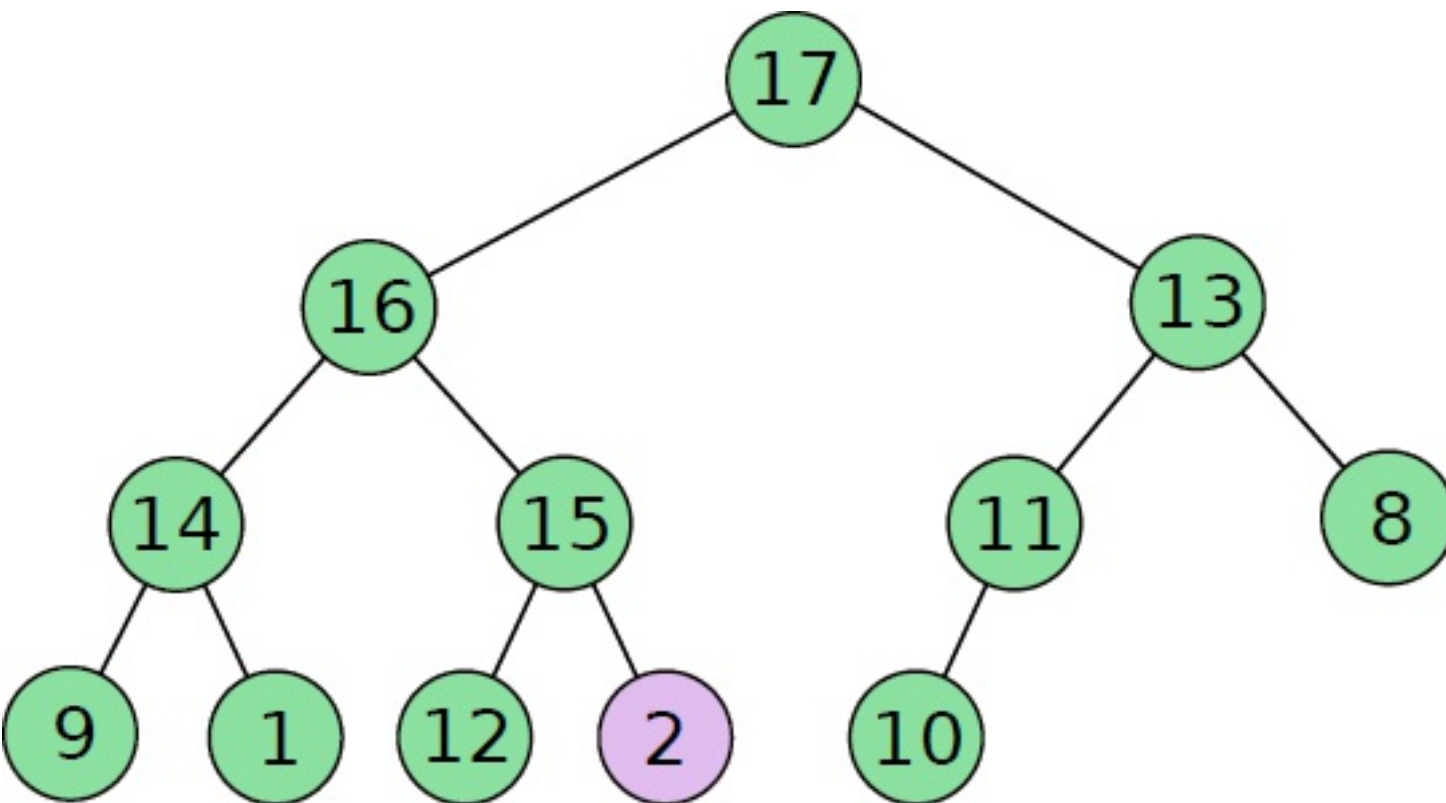


HEAP-EXTRACT-MAX( $A$ )

```
1  if  $A.heap-size < 1$   
2      error “Очередь пуста”  
3   $max = A[1]$   
4   $A[1] = A[A.heap-size]$   
5   $A.heap-size = A.heap-size - 1$   
6  MAX-HEAPIFY( $A, 1$ )  
7  return  $max$ 
```



## Очередь с приоритетами на основе кучи: Increase\_key(A, i, key)

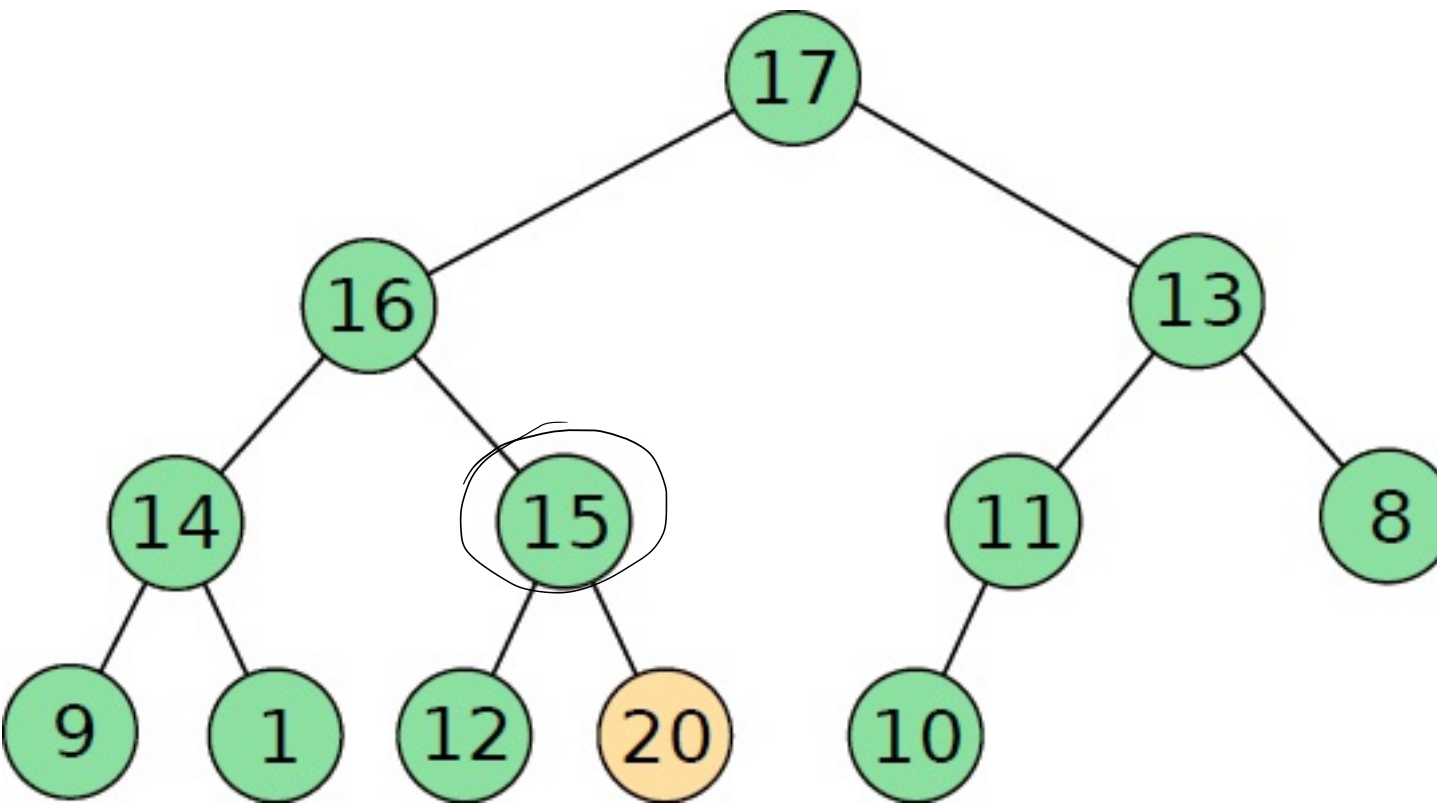


HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error “Новый ключ меньше текущего”
3   $A[i] = key$ 
4  while  $i > 1$  и  $A[\text{PARENT}(i)] < A[i]$ 
5      Обменять  $A[i]$  и  $A[\text{PARENT}(i)]$ 
6       $i = \text{PARENT}(i)$ 
```



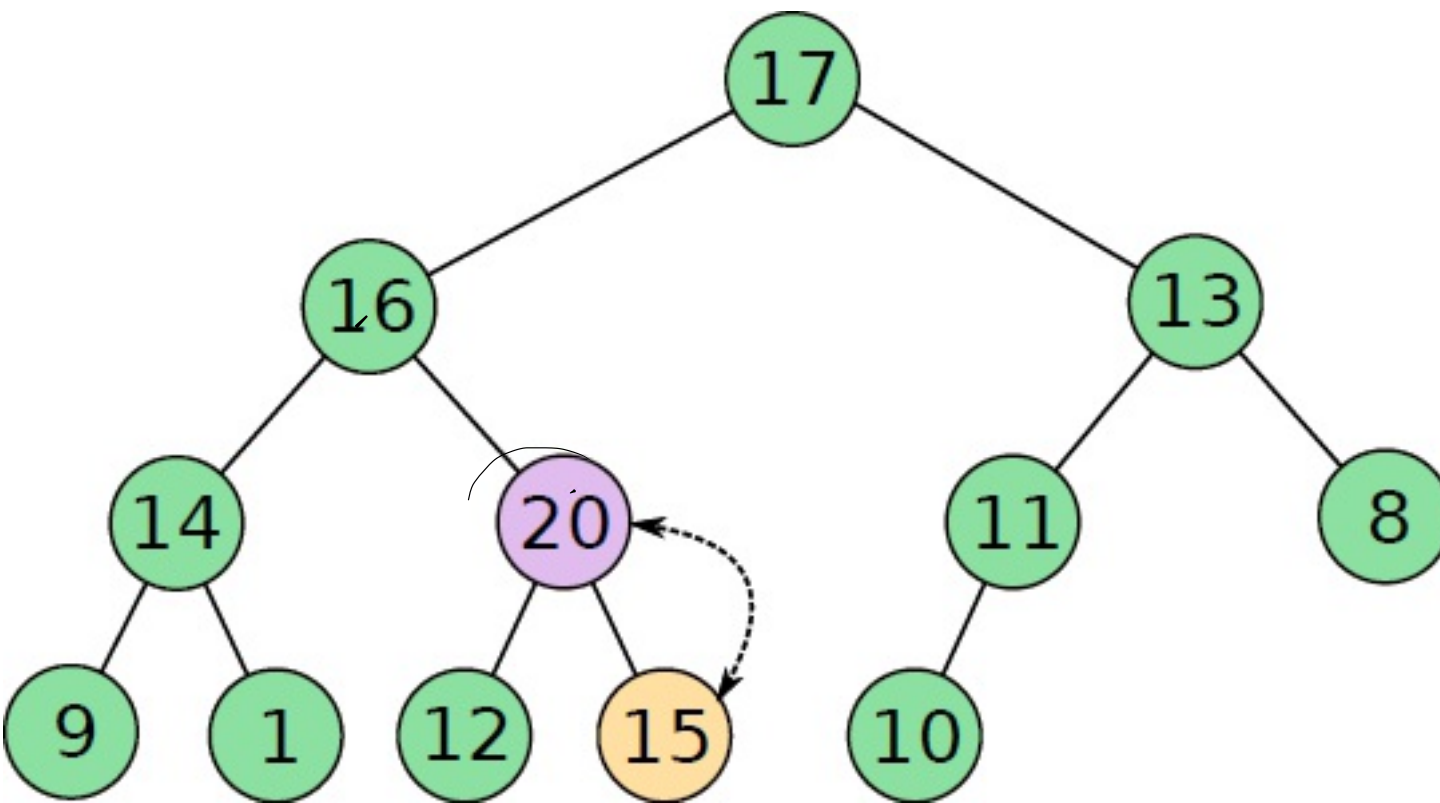
## Очередь с приоритетами на основе кучи: Increase\_key( $A, i, key$ )



HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error "Новый ключ меньше текущего"
3   $A[i] = key$ 
4  while  $i > 1$  и  $A[\text{PARENT}(i)] < A[i]$ 
5      Обменять  $A[i]$  и  $A[\text{PARENT}(i)]$ 
6       $i = \text{PARENT}(i)$ 
```

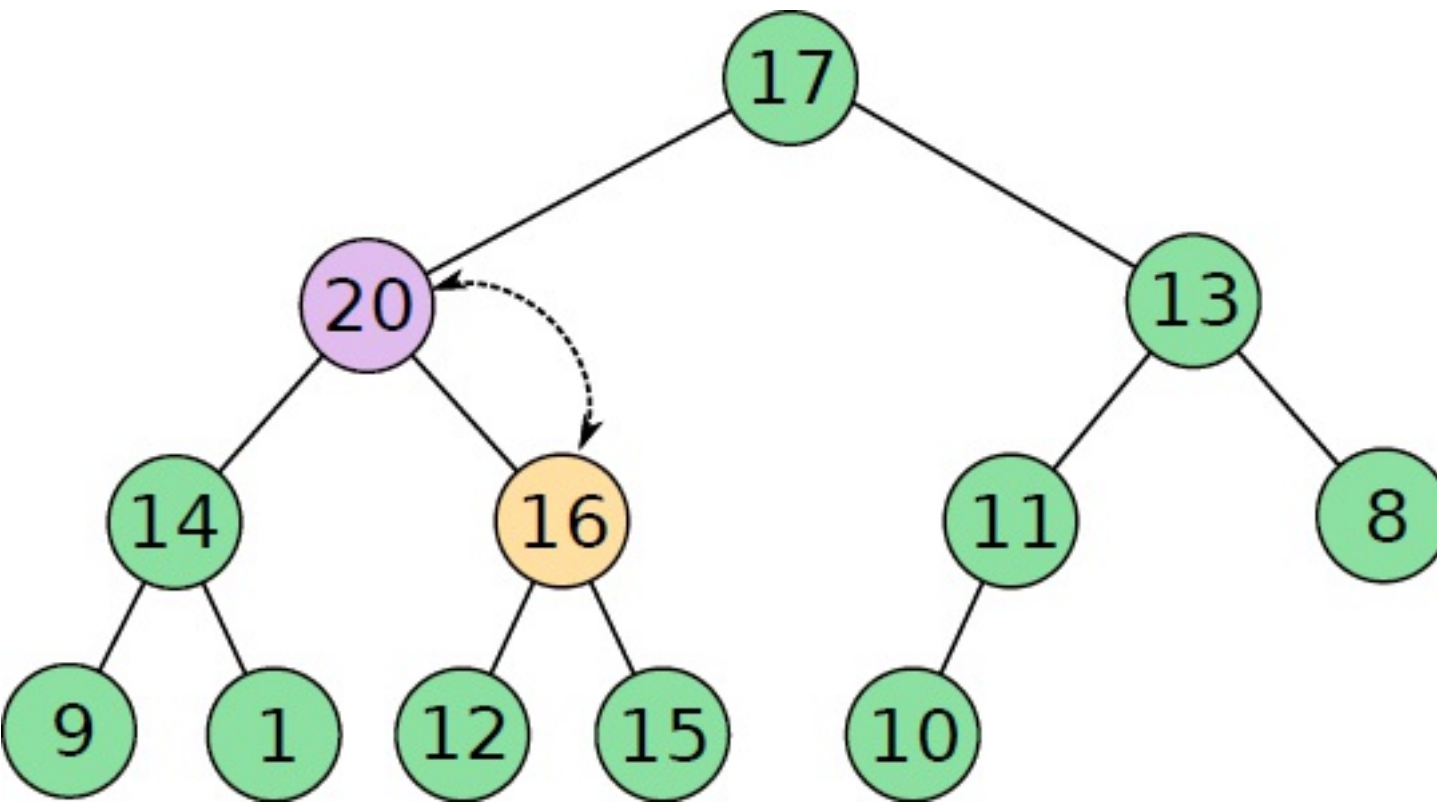
## Очередь с приоритетами на основе кучи: Increase\_key( $A, i, key$ )



HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error "Новый ключ меньше текущего"
3   $A[i] = key$ 
4  while  $i > 1$  и  $A[\text{PARENT}(i)] < A[i]$ 
5      Обменять  $A[i]$  и  $A[\text{PARENT}(i)]$ 
6       $i = \text{PARENT}(i)$ 
```

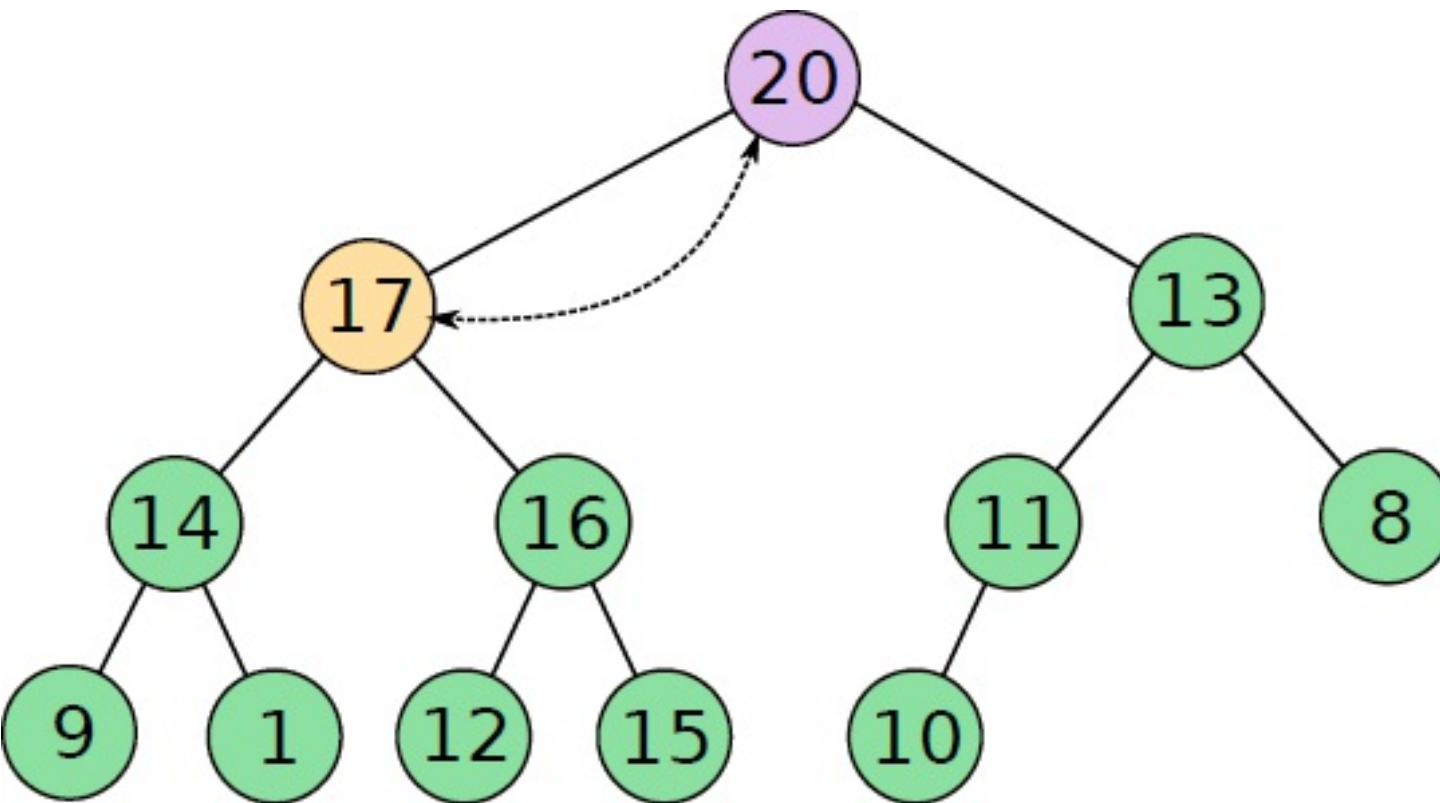
## Очередь с приоритетами на основе кучи: Increase\_key( $A, i, key$ )



HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error “Новый ключ меньше текущего”
3   $A[i] = key$ 
4  while  $i > 1$  и  $A[\text{PARENT}(i)] < A[i]$ 
5      Обменять  $A[i]$  и  $A[\text{PARENT}(i)]$ 
6       $i = \text{PARENT}(i)$ 
```

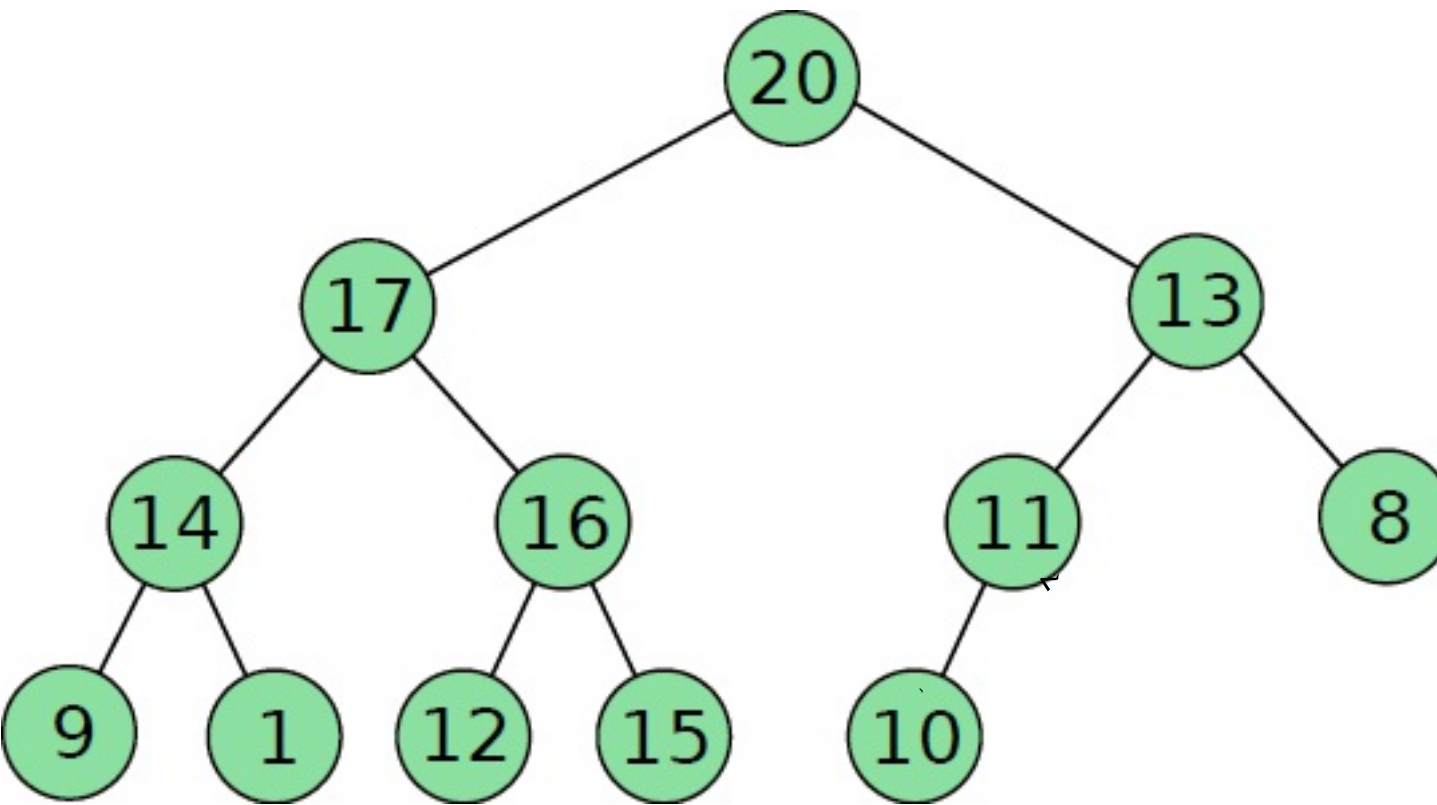
## Очередь с приоритетами на основе кучи: Increase\_key( $A, i, key$ )



HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error "Новый ключ меньше текущего"
3   $A[i] = key$ 
4  while  $i > 1$  и  $A[\text{PARENT}(i)] < A[i]$ 
5      Обменять  $A[i]$  и  $A[\text{PARENT}(i)]$ 
6       $i = \text{PARENT}(i)$ 
```

## Очередь с приоритетами на основе кучи: Increase\_key( $A, i, key$ )

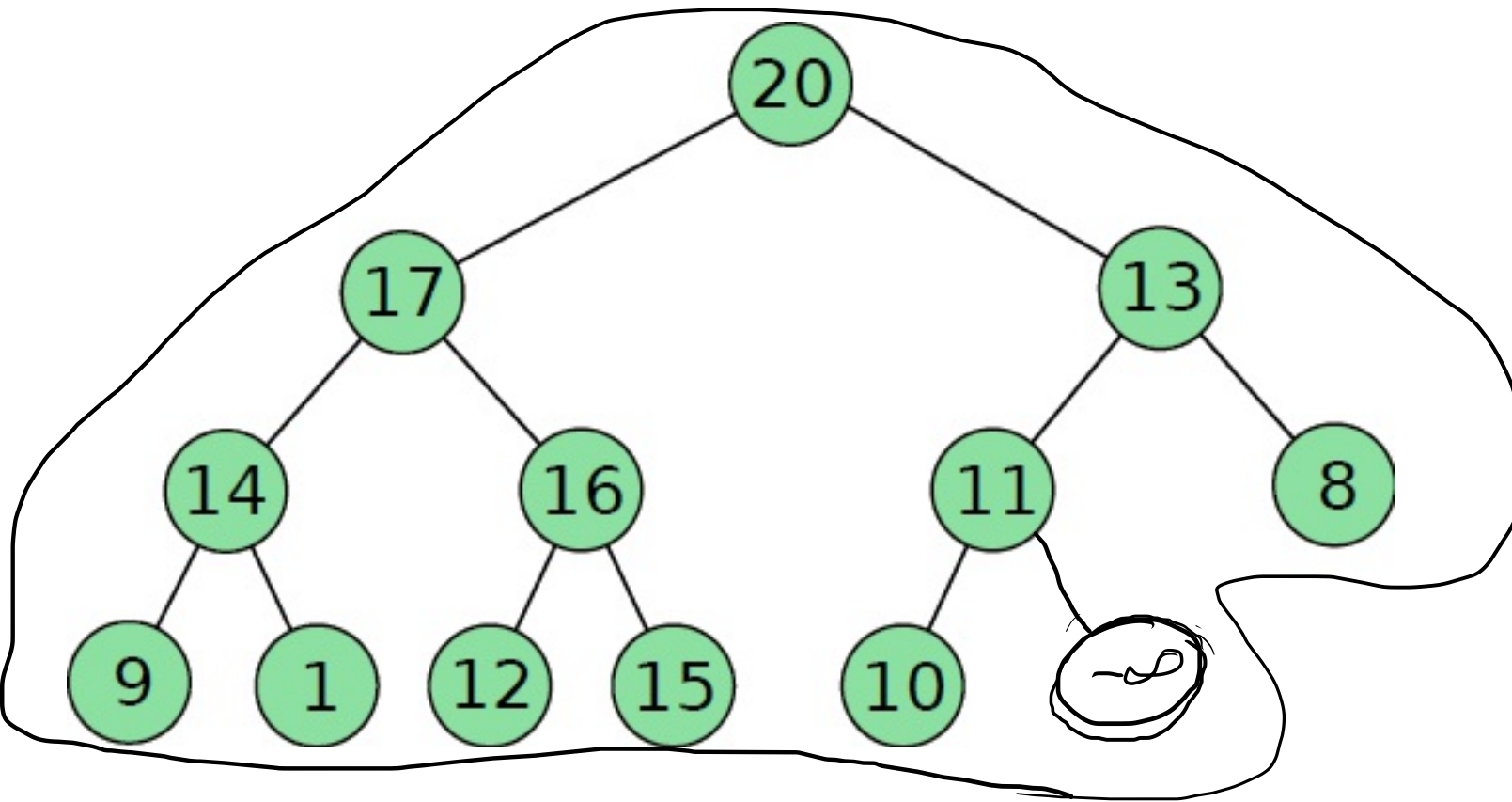


HEAP-INCREASE-KEY( $A, i, key$ )

```
1  if  $key < A[i]$ 
2      error "Новый ключ меньше текущего"
3   $A[i] = key$ 
4  while  $i > 1$  и  $A[PARENT(i)] < A[i]$ 
5      Обменять  $A[i]$  и  $A[PARENT(i)]$ 
6       $i = PARENT(i)$ 
```

$\log(n)$

## Очередь с приоритетами на основе кучи: Insert(A, key)



MAX-HEAP-INSERT( $A, key$ )

- 1  $A.heap-size = A.heap-size + 1$
- 2  $A[A.heap-size] = -\infty$
- 3 HEAP-INCREASE-KEY( $A, A.heap-size, \underline{key}$ )

$\log(n)$ .