

## Очереди с приоритетом

Абстрактный тип данных,  
использующий  
следующие операции:

- Insert( $p$ ) - добавить новый элемент с приоритетом  $p$
- ExtractMax() - извлечь из-за максимумом приоритетом
- Remove( $i$ ) - удалить
- GetMax() - вернуть макс. при-  
оритет
- ChangePriority( $i, p$ ) - изменить  
приоритет  $i$ -го на  $p$

- Алгоритм Дейкстры
- Алгоритм Прима
- Алгоритм Хаффмана
- Алгоритм сортировки кучей

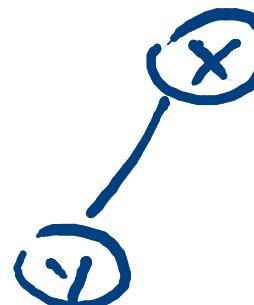
## Примитивные реализации

	Insert	Extract Max
Массив / стек	$O(1)$	$O(n)$
Вспомогательный массив / стек	$O(n)$	$O(1)$

## Двумерные куски

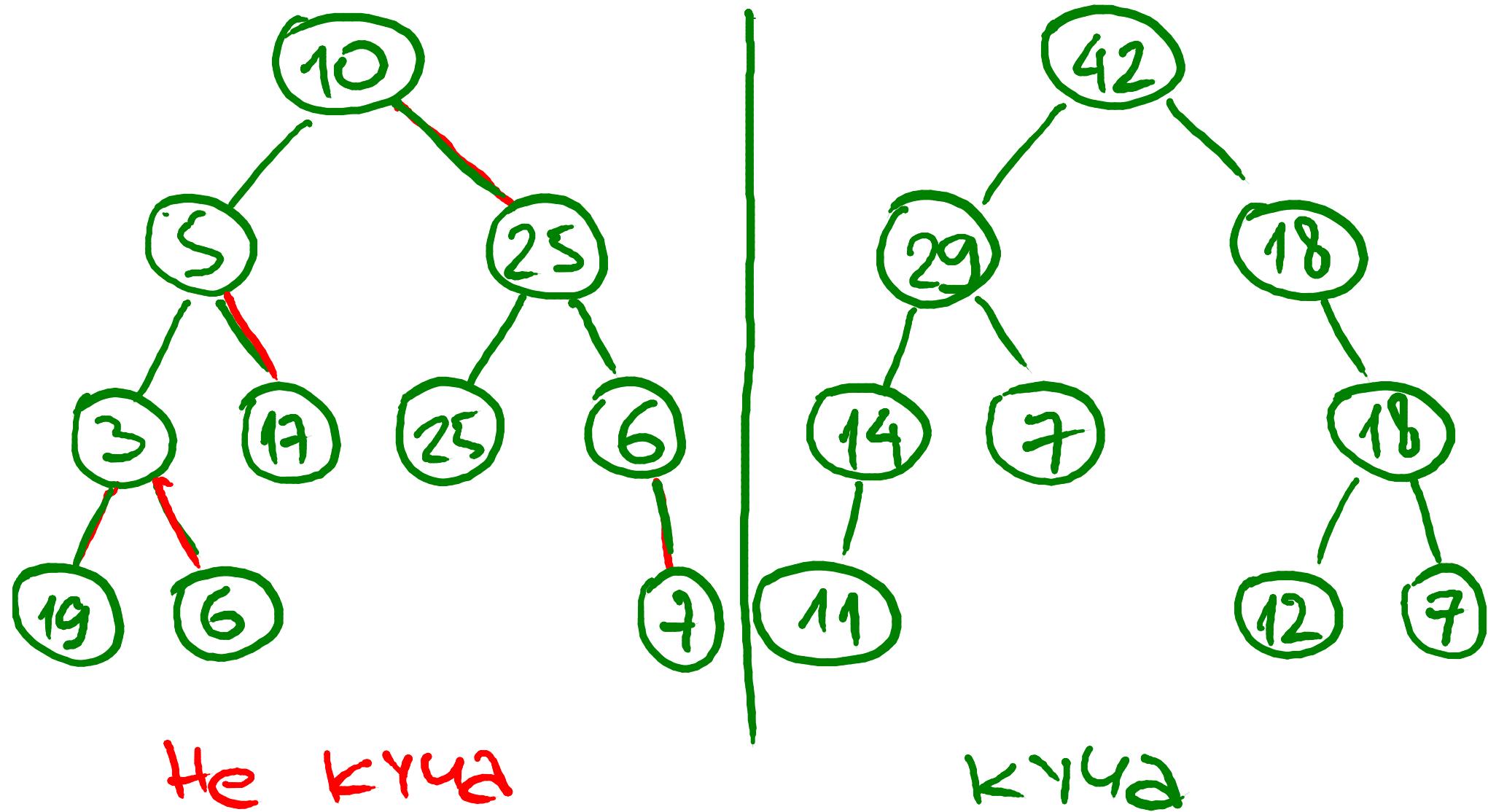
Определение: двумерный Max-knapsack-задача перевозки ('в каждой вершине не более двух связей'), в котором значение в каждой вершине не меньше значений в её супервузах.

Для любого ребра:

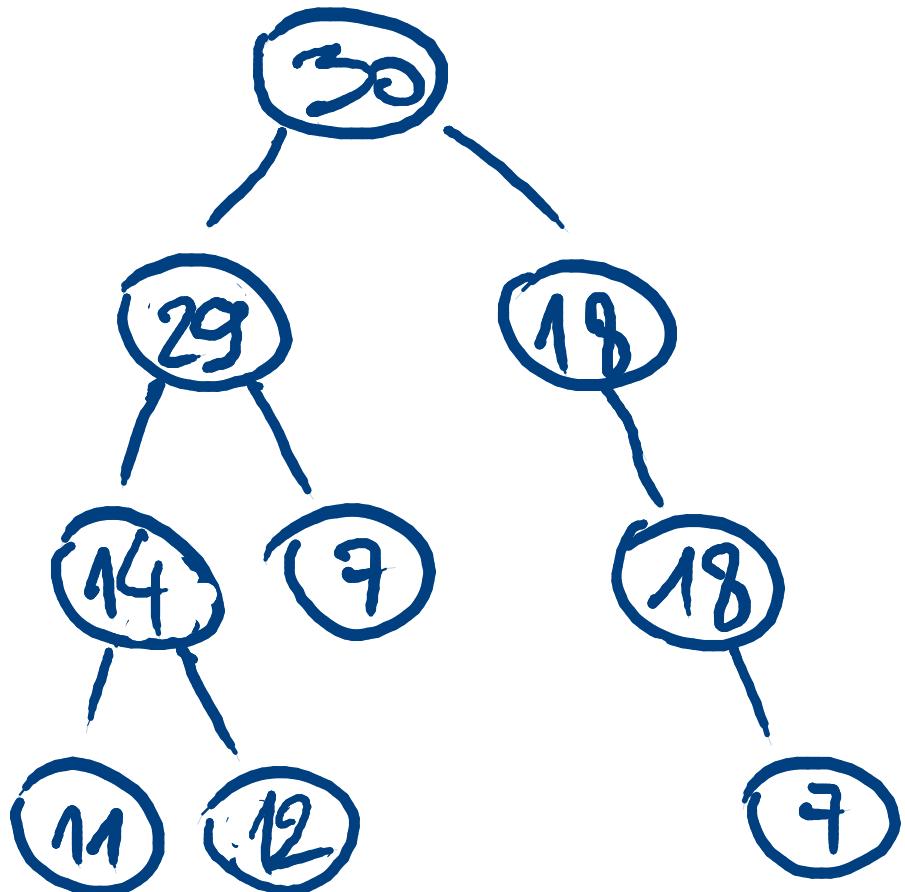


$$x \geq y$$

# Пример



# Операции



- Get Max : корень
- Insert : добавить в лист, просеять сверху
- Extract Max : обменять корень с листом, просеять вниз
- Change Priority : изменить приоритет, просеять сверху/вниз
- Remove : извлечь из приоритета на  $\infty$ , просеять вверх, извлечь максимум

# Резоне

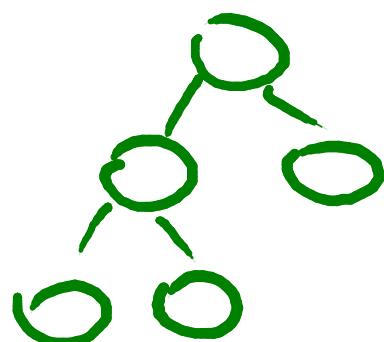
Все операции имеют вид  
Работы  $O(n^2)$ .

Определение хэши  
последовательно, дерево  
изменяется!

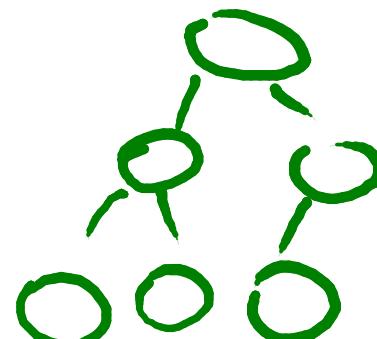
## Полностью заполненные двоичные деревья

Все узлы полнос~~ть~~то заполнены,  
кроме , возможно, последнего,  
который может оставаться пустым.

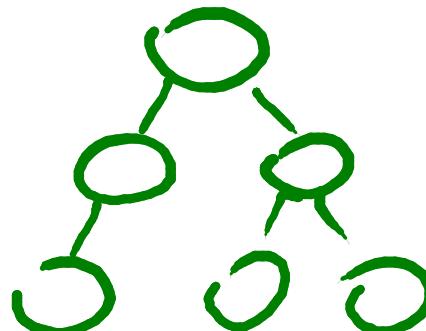
Пример:



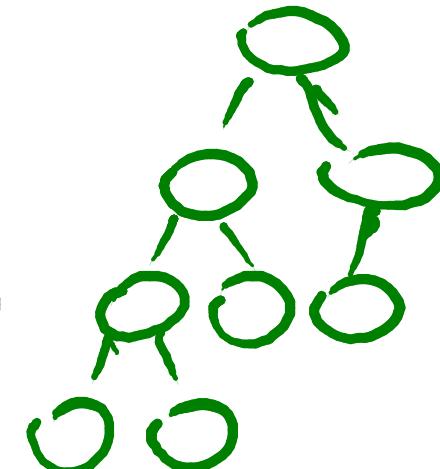
✓



✓



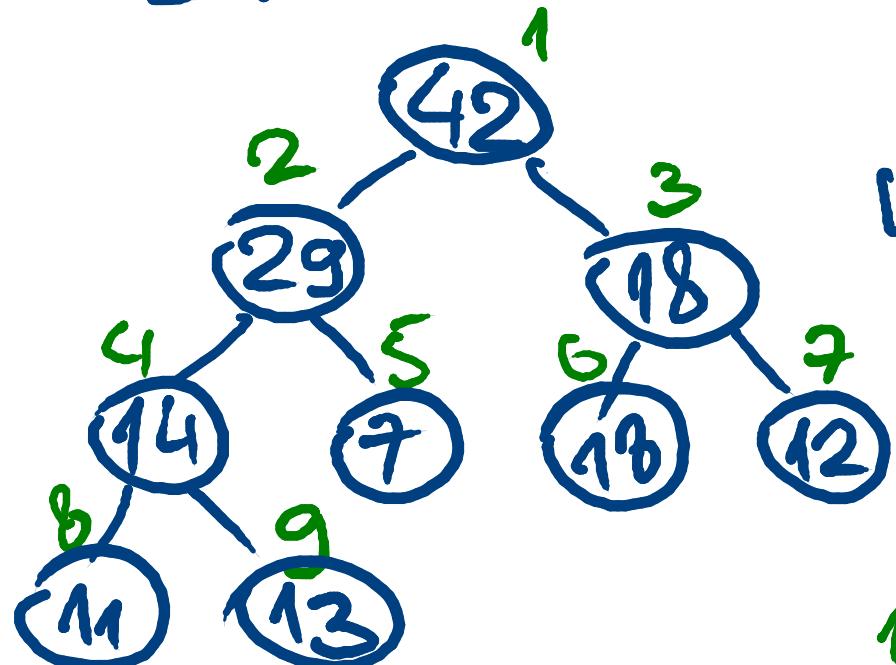
✗



✗

# Преимущества

1. Быстрая Р.З.Д.Д. с н. вспл. -  $O(\log n)$ .
2. Часто используется в кв. схеме:



$$\text{PARENT}(i) = \lfloor i/2 \rfloor$$

$$\text{LEFT CHILD}(i) = 2i$$

$$\text{RIGHT CHILD}(i) = 2i+1$$

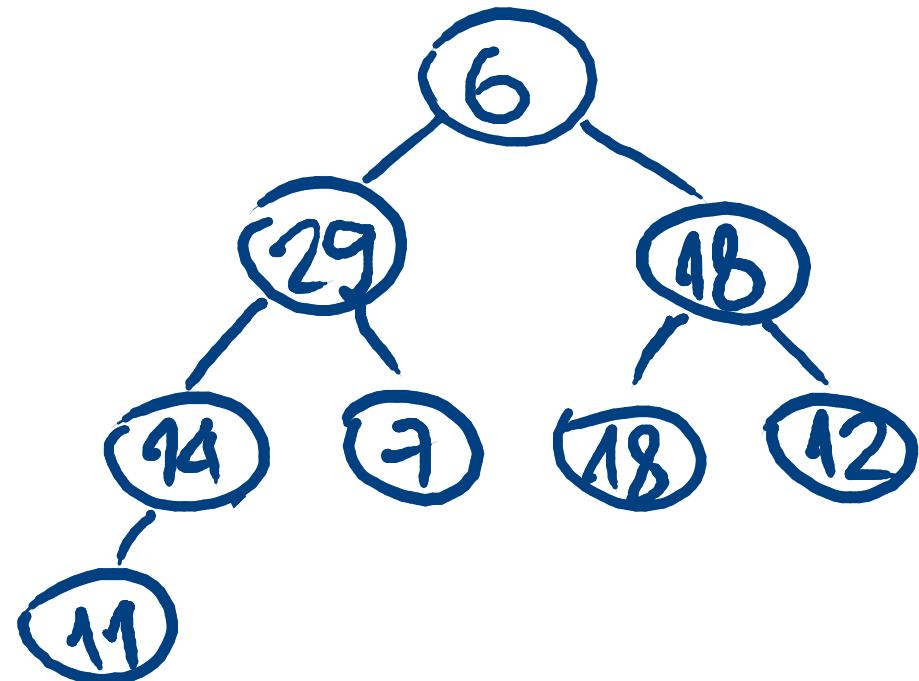
1	2	3	4	5	6	7	8	9
42	29	18	14	7	13	12	11	13

# Как поддерживать дерево, Полностью заинтересованным?

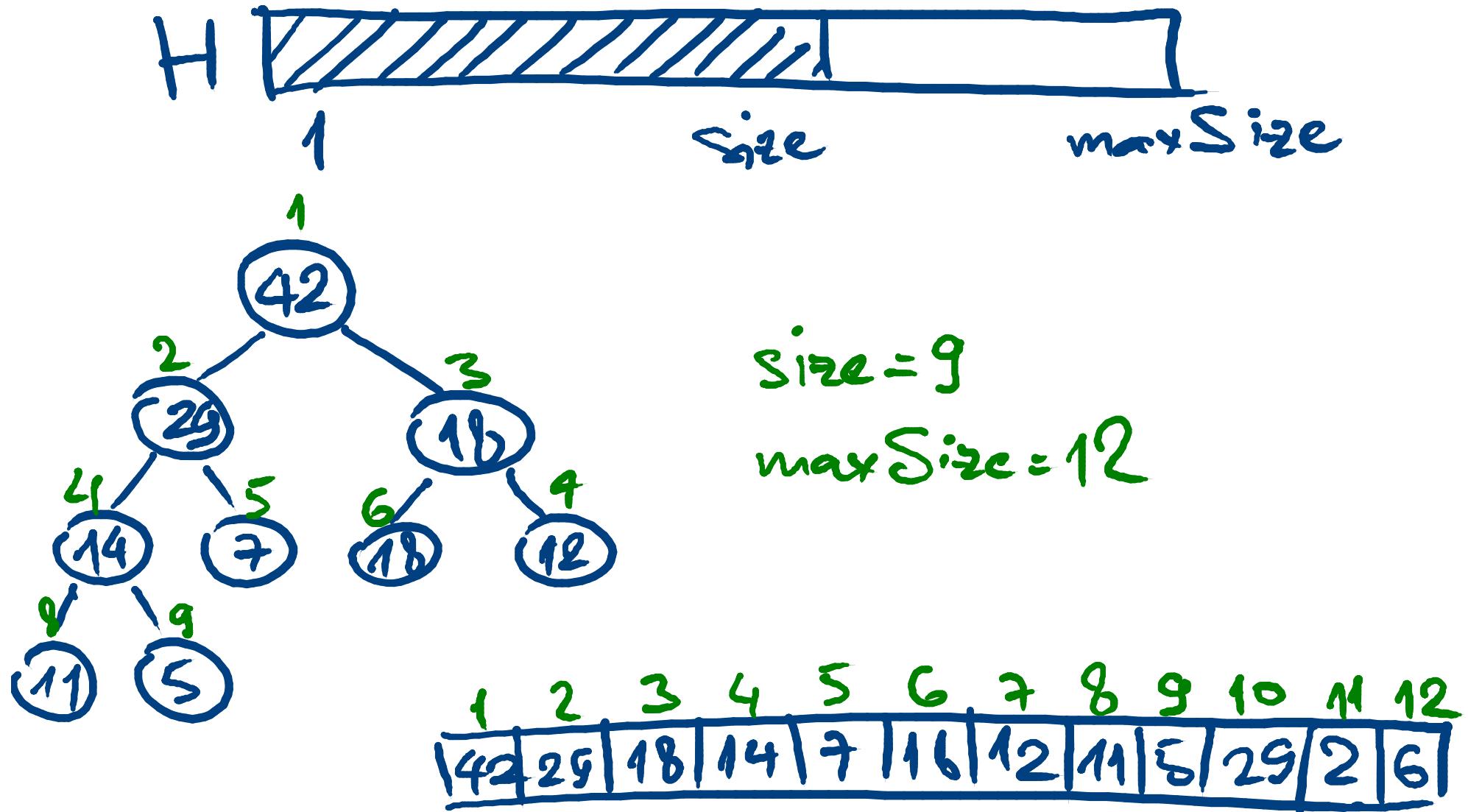
Если приложения, только  
Insert и ExtractMax изменяют  
форму дерева.

Insert: Всё время  
на первое  
свободное  
 место на последнем  
уровне

ExtractMax:  
наоборот



# TreeBOKOD



Parent(i):

return  $\lfloor i/2 \rfloor$

LeftChild(i):

return  $2i$

RightChild(i):

return  $2i+1$

Sift Up( $i$ ):

while  $i > 1$  and  $H[\text{Parent}(i)] < H[i]$ :

    swap  $H[\text{Parent}(i)]$  and  $H[i]$

$\leftarrow \text{Parent}(i)$

SiftDown(i):

maxIndex  $\leftarrow i$

$l \leftarrow \text{LeftChild}(i)$

if  $l \leq \text{size}$  and  $H[l] > H[\text{maxIndex}]$ :

    maxIndex  $\leftarrow l$

$r \leftarrow \text{RightChild}(i)$

if  $r \leq \text{size}$  and  $H[r] > H[\text{maxIndex}]$ :

    maxIndex  $\leftarrow r$

if  $i \neq \text{maxIndex}$ :

    swap  $H[i]$  and  $H[\text{maxIndex}]$

    SiftDown(maxIndex)

Insert(p):

If  $\text{size} = \text{maxSize}$ ;  
return ERROR

$\text{size} \leftarrow \text{size} + 1$

$H[\text{size}] \leftarrow p$

SiftUp(size)

## ExtractMax():

result  $\leftarrow H[1]$

$H[1] \leftarrow H[size]$

$size \leftarrow size - 1$

SiftDown(1)

return result

Remove(i):

$H[i] \leftarrow \infty$

SiftUp(i)

ExtractMax()

ChangePriority( $i, p$ ):

$oldp \leftarrow H[i]$

$H[i] \leftarrow p$

if  $p > oldp$ :

$SiftUp(i)$

else:

$SiftDown(i)$

# Задачи

- ✓ быстро
- ✓ компактно
- ✓ легко разобрать

## Сортировка кучей

HeapSort(A[1..n])

create an empty priority queue  
for i from 1 to n:

    Insert(A[i])

for i from n down to 1:  
    A[i]  $\leftarrow$  ExtractMax()

---

②  $\Theta(n \log n)$ , где  $n$  — количество элементов  
сортировки. Время сортировки не хуже

Как представить MaxHeap  
в куче?

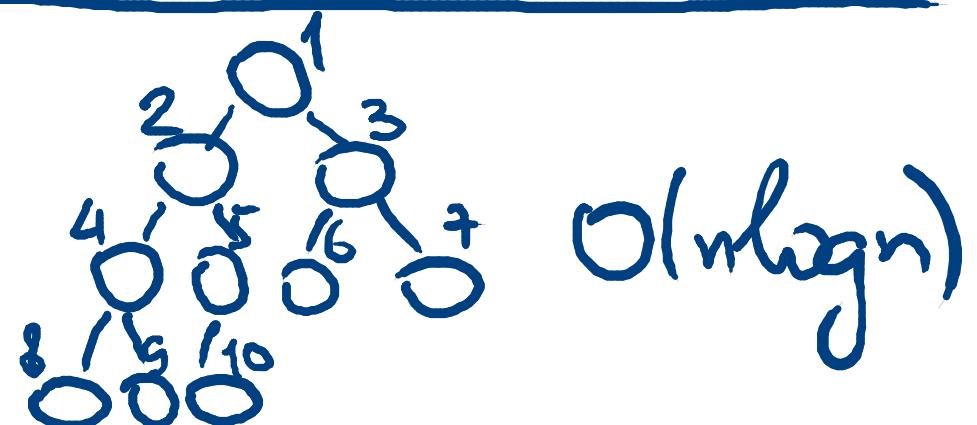
BuildHeap(A[1..n])

size  $\leftarrow n$

for i from  $\lfloor n/2 \rfloor$  down to 1:  
    SiftDown(i)

---

Чтобы уменьшить количество  
перемещений, нужно  
использовать



$O(n\lg n)$

# Сортировка кучей на Мерк

HeapSort(A[1..n])

BuildHeap(A)

{size = n}

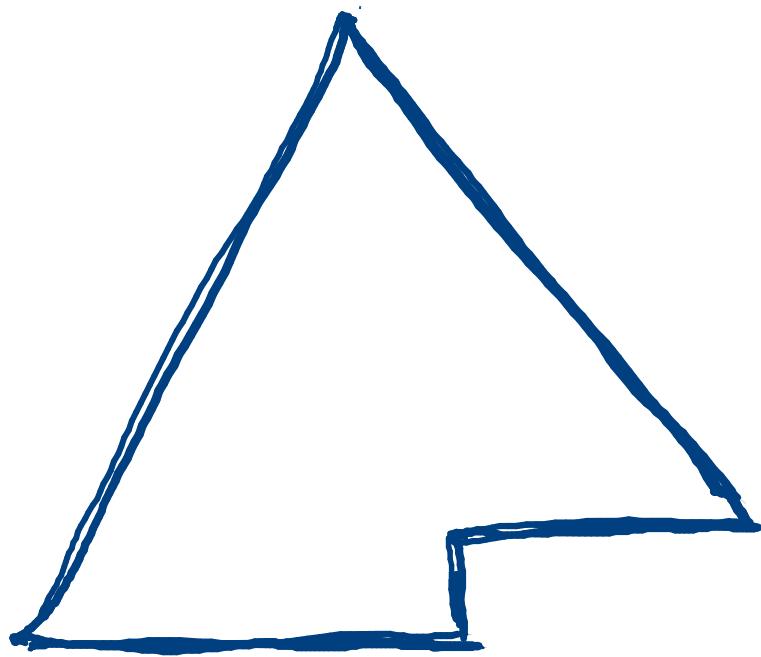
repeat (n-1) times:

swap A[1] and A[size]

size  $\leftarrow$  size - 1

SiftDown(1)

BPEM1 Doctoral Exam



#BPEM1H

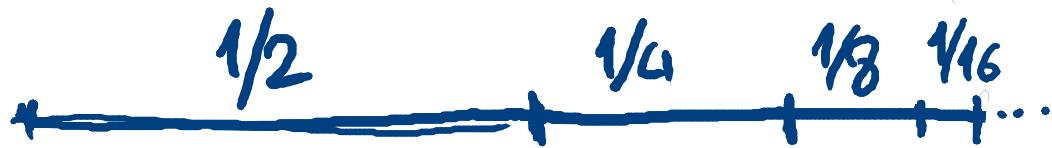
1 .  
2 .  
⋮  
 $\leq n/4$  .  
 $\leq n/2$  .

T(ShiftDown)

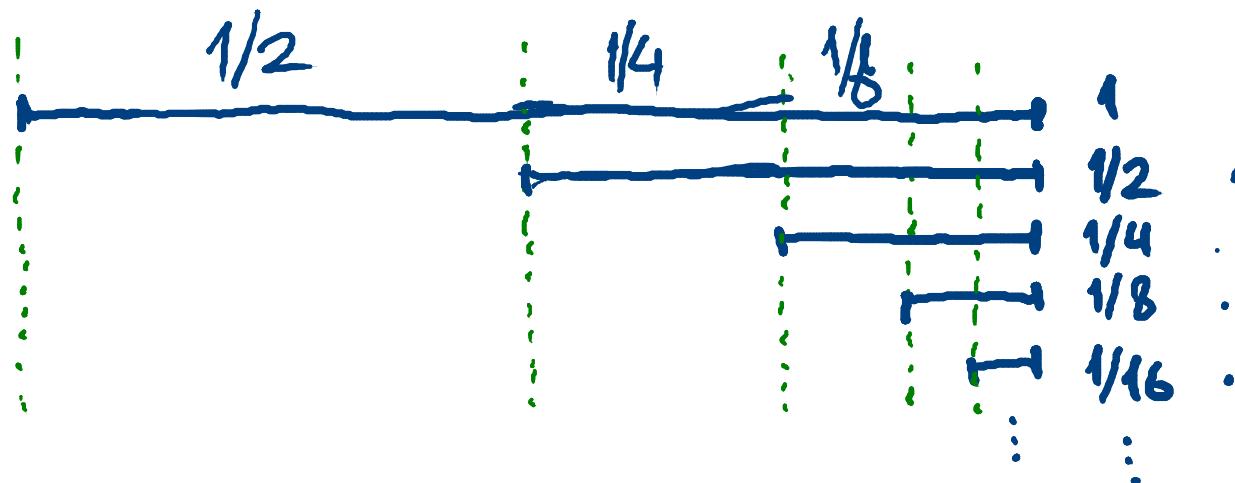
$\log_2 n$  .  
⋮  
2 .  
1

$$T(\text{BuildHeap}) \leq \frac{n}{2} \cdot 1 + \frac{n}{4} \cdot 2 + \frac{n}{8} \cdot 3 + \dots$$

$$\leq n \cdot \sum_{i=1}^{\infty} \frac{i}{2^i} = 2n$$



$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots = \sum_{k=1}^{\infty} \frac{1}{2^k} = 1$$



$$\frac{1}{2} + \frac{2}{4} + \frac{3}{8} + \frac{4}{16} + \dots = \sum_{k=1}^{\infty} \frac{k}{2^k} = 2$$

## Удаление из списка

Найти k максимальных элементов

PartialSorting(A[1..n], k)

BuildHeap(A)

for i from 1 to k:

ExtractMax()

$O(n + k \log n)$

## Задачи с кучами

1. Массивы с индексацией с 0 (0-based)

$$\text{parent}(i) = \lfloor \frac{i-1}{2} \rfloor, \text{left}(i) = 2i+1, \text{right}(i) = 2i+2$$

2. Мин-куча : аналогично

3. d-членая куча

Высота:  $\log_d n$

$$T(\text{siftUp}) = O(\log_d n)$$

$$T(\text{siftDown}) = O(d \log_d n)$$

