# Лекция по АиСД

Кучи. Сортировка кучей.

# Куча (heap)

Структура данных для быстрого поиска минимума

1. Добавление за O(log n)
2. ExtractMin (извлечение минимума) за O(log n)
3. GetMin (узнать минимум) за O(1)

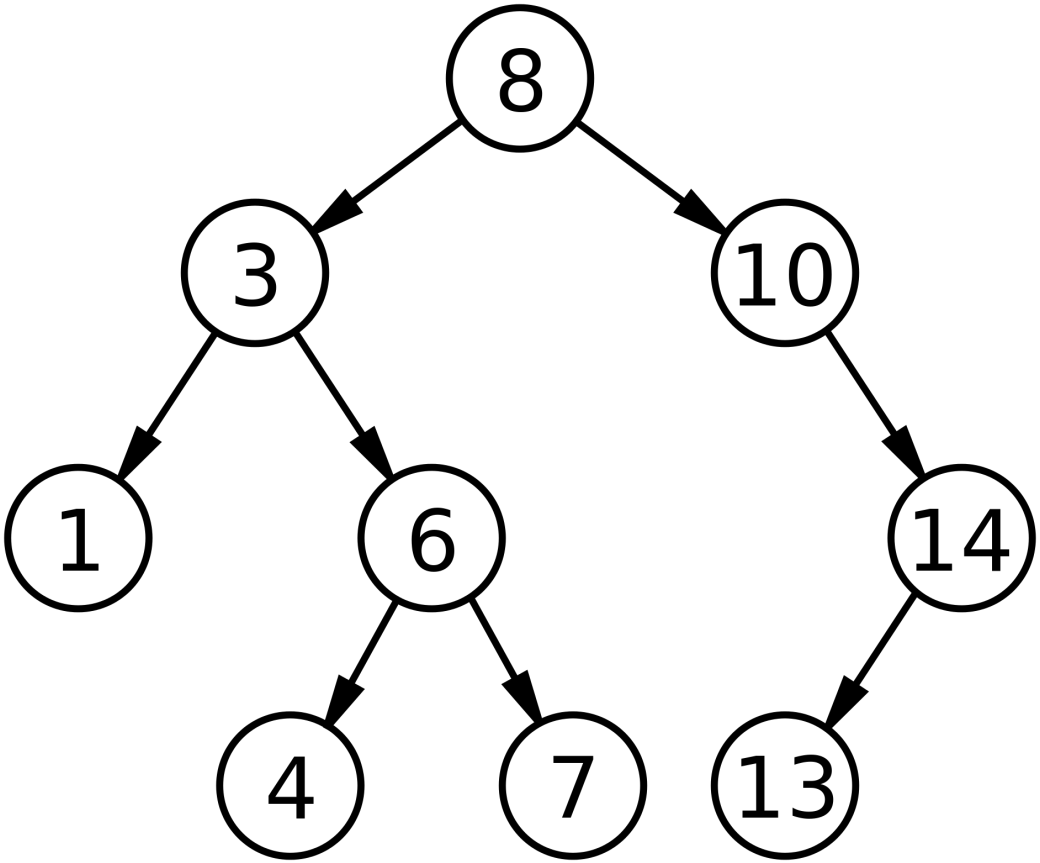
# Дерево

Граф – G = (V, E), где V – множество вершин, E – множество ребер

Дерево – связный граф без циклов

Бинарное дерево – у каждого узла не больше 2х потомков

# Бинарное дерево



# Куча (heap)

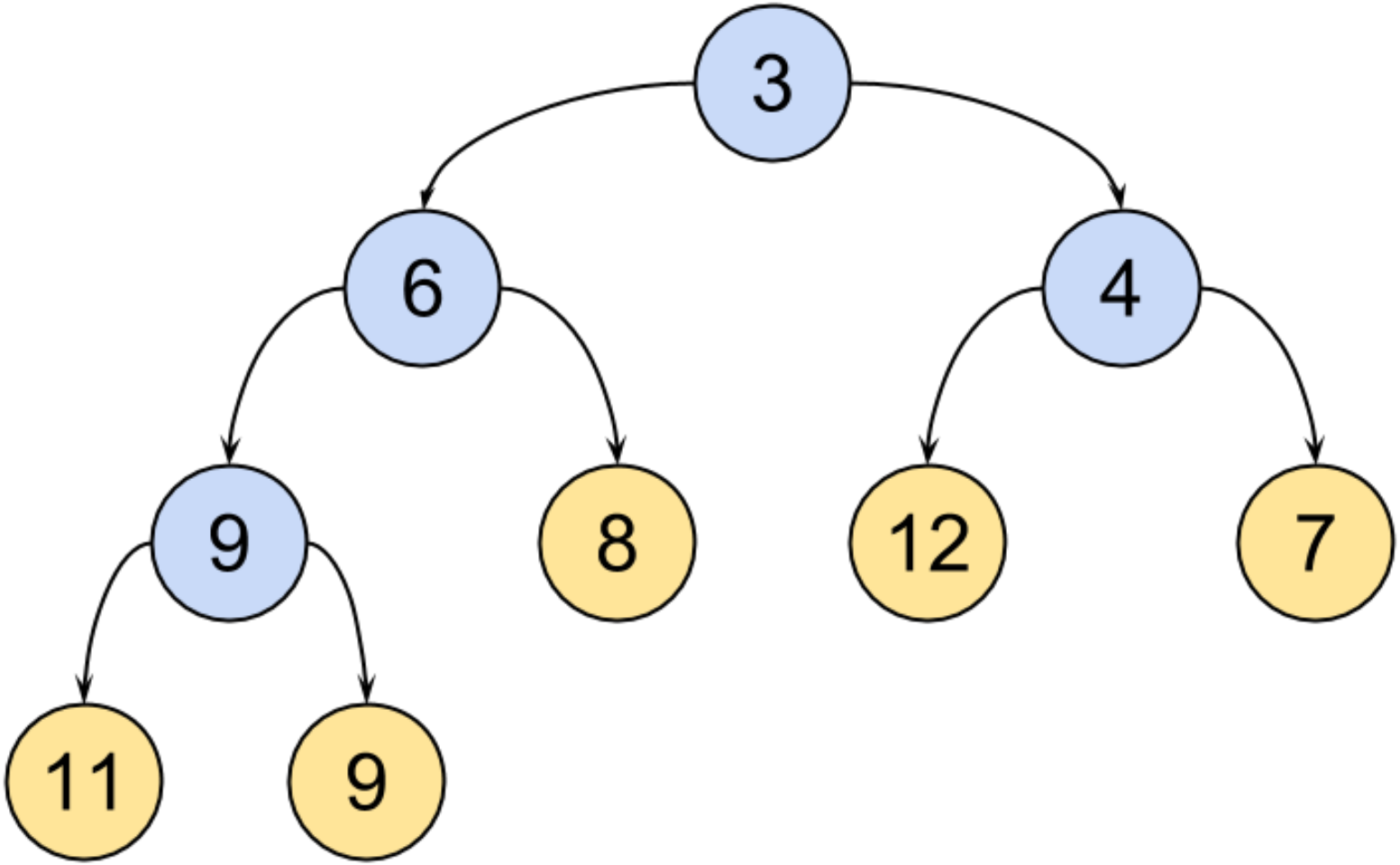
Свойства кучи

1. Значение в узле не превышает значений его потомков

2. Почти полное бинарное дерево (все уровни заполнены, кроме возможно последнего)

3. Последний слой заполнен слева направо

# Куча



# Куча

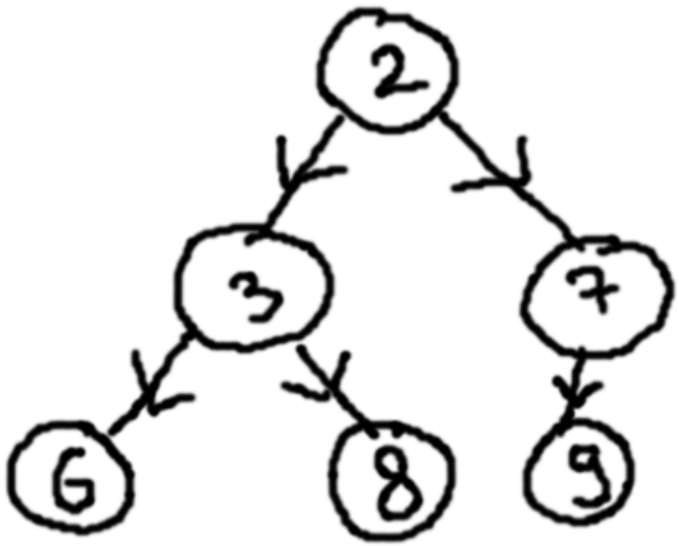
Кучу удобно хранить в виде массива

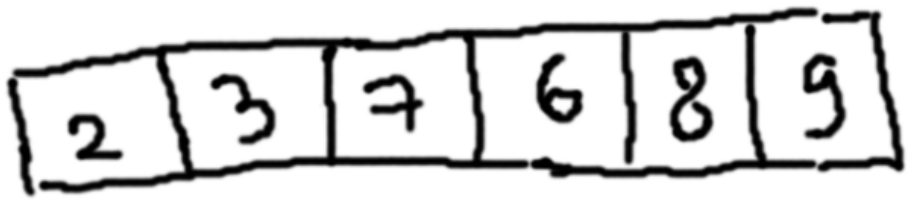
a[0] – корень

для i-го узла его потомки находятся в a[2 * i + 1] и a[2 * i + 2]

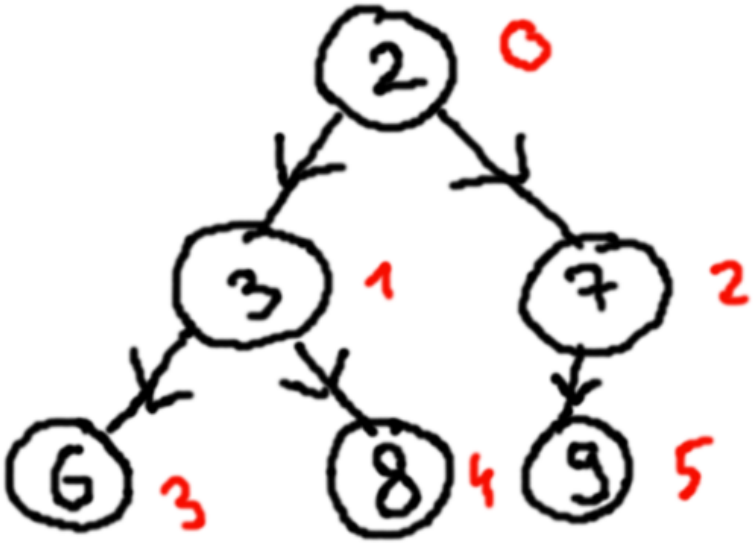
для i-го узла его родитель находится в a[(i - 1) // 2]

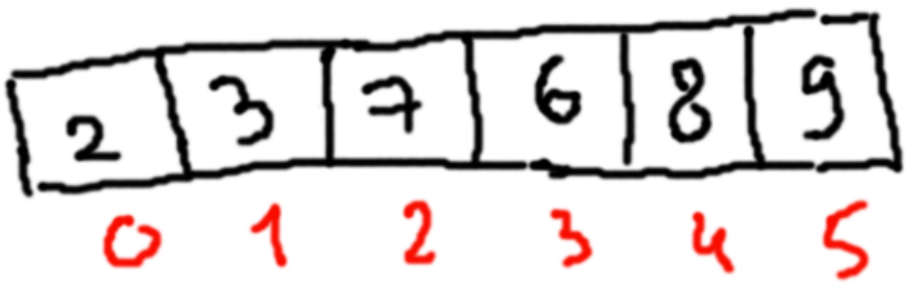
# Куча



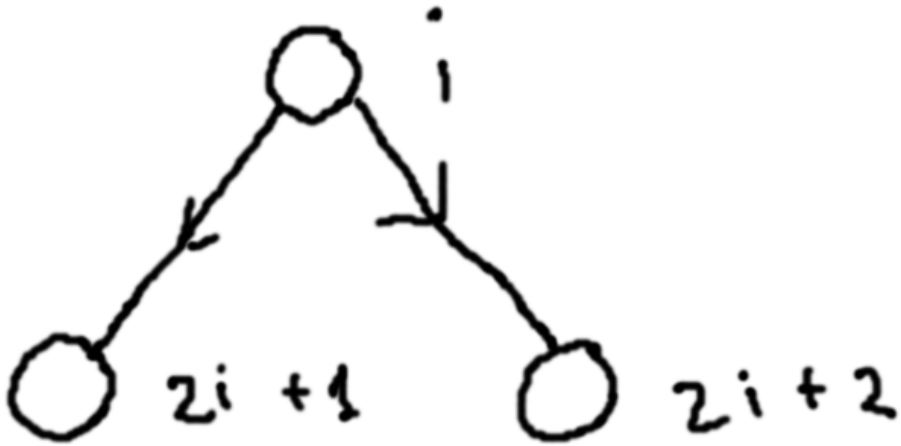



# Куча

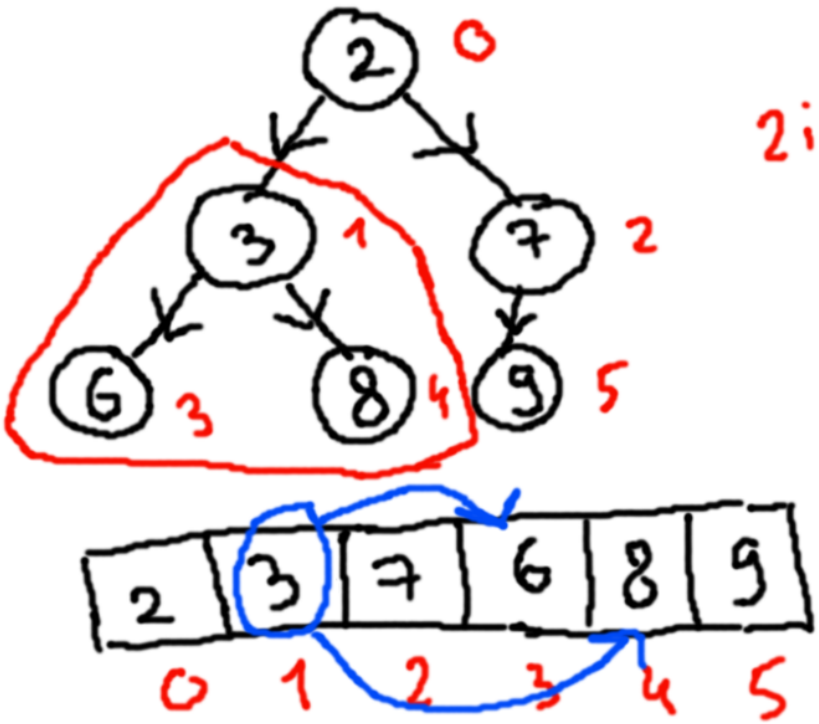


# Куча



# Куча



# Куча

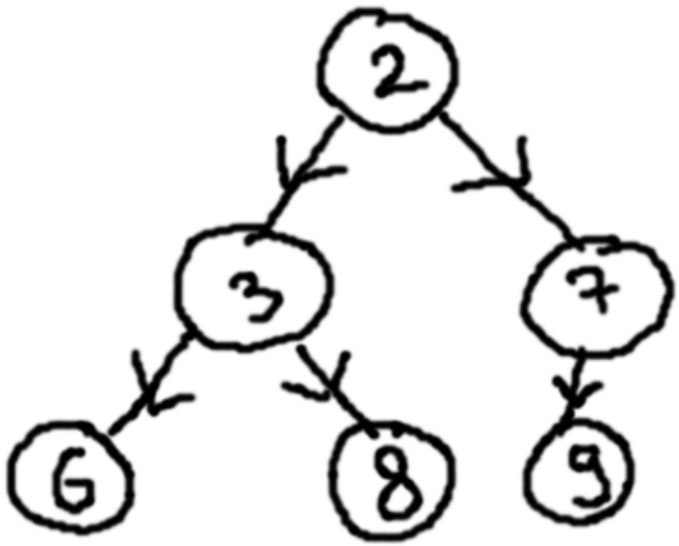
1. GetMin – корень кучи (a[0])
2. ExtractMin
3. Push

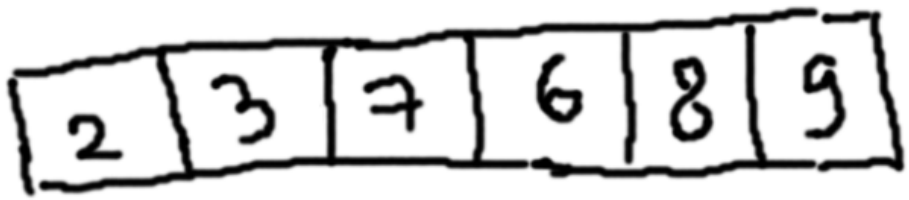
# Push

Добавим элемент в конец массива

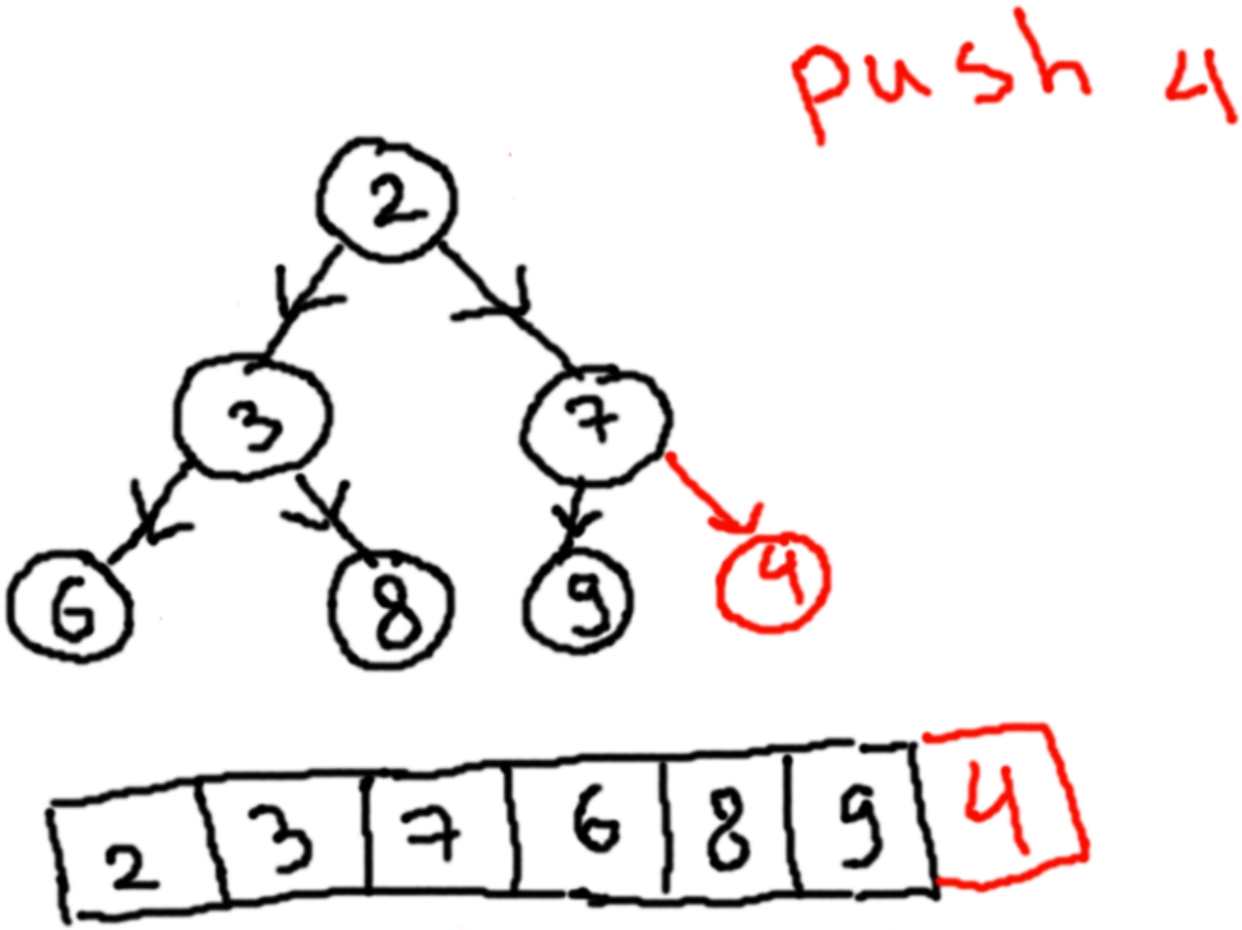
Это может поломать нам свойства кучи, их нужно будет восстановить

# Куча

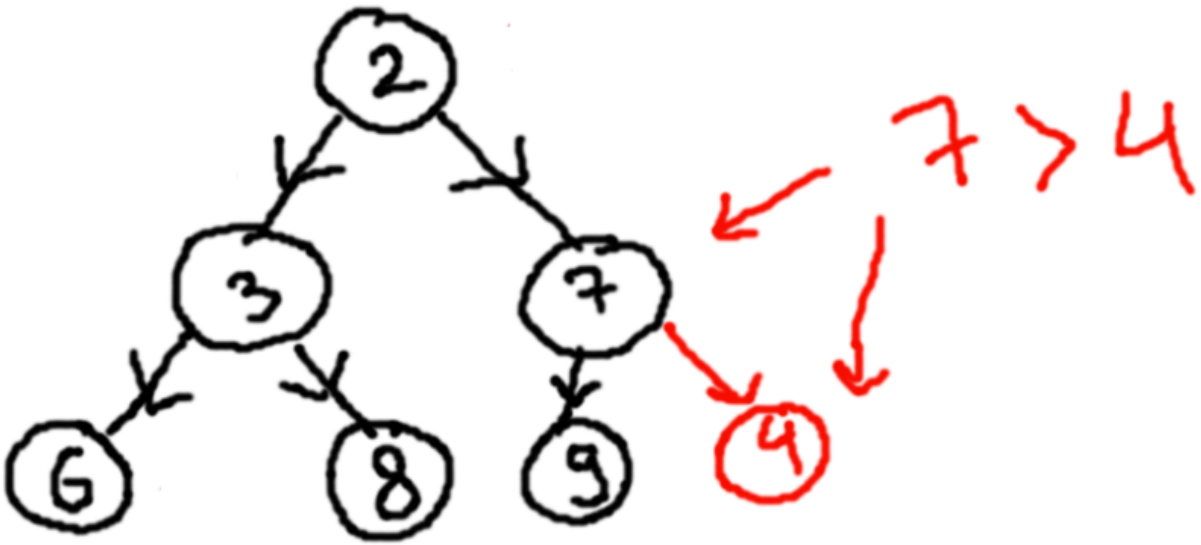


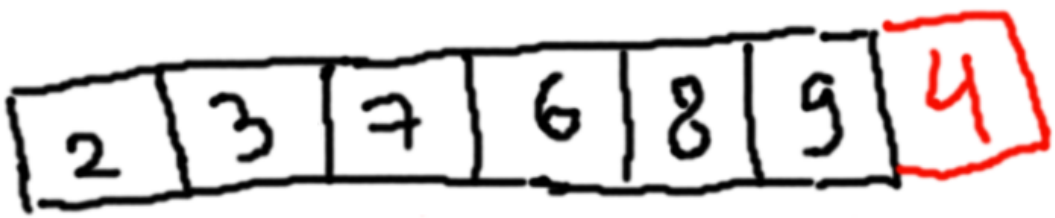



# Куча



# Куча



# SiftUp

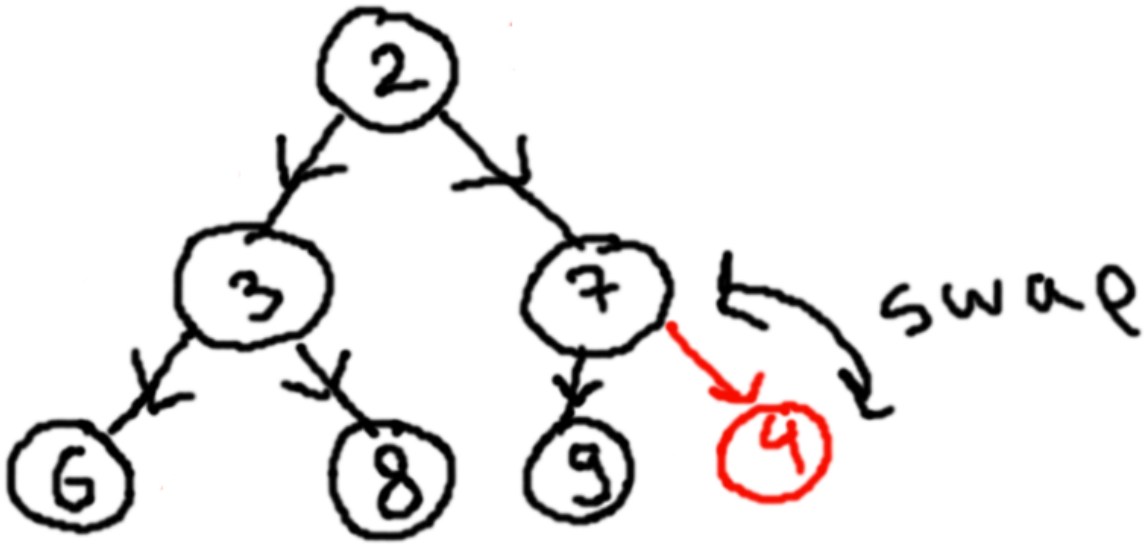
Для восстановления свойств кучи будем делать SiftUp:

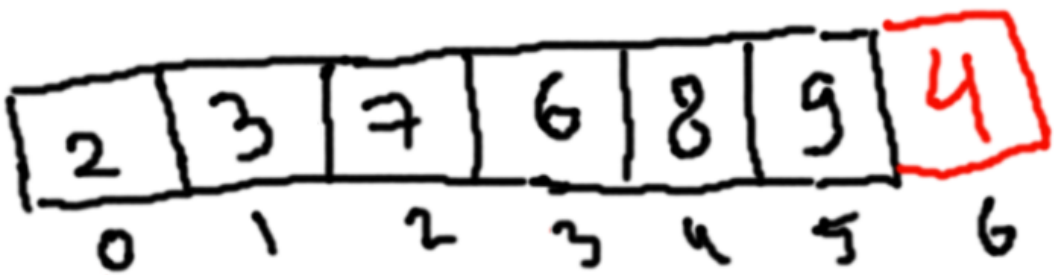
1. Если элемент больше своего родителя, то все хорошо и больше ничего делать не нужно.
2. Если элемент меньше своего родителя, меняем его с родителем местами, после чего выполняем SiftUp для родителя.

Иными словами, слишком маленький элемент всплывает наверх

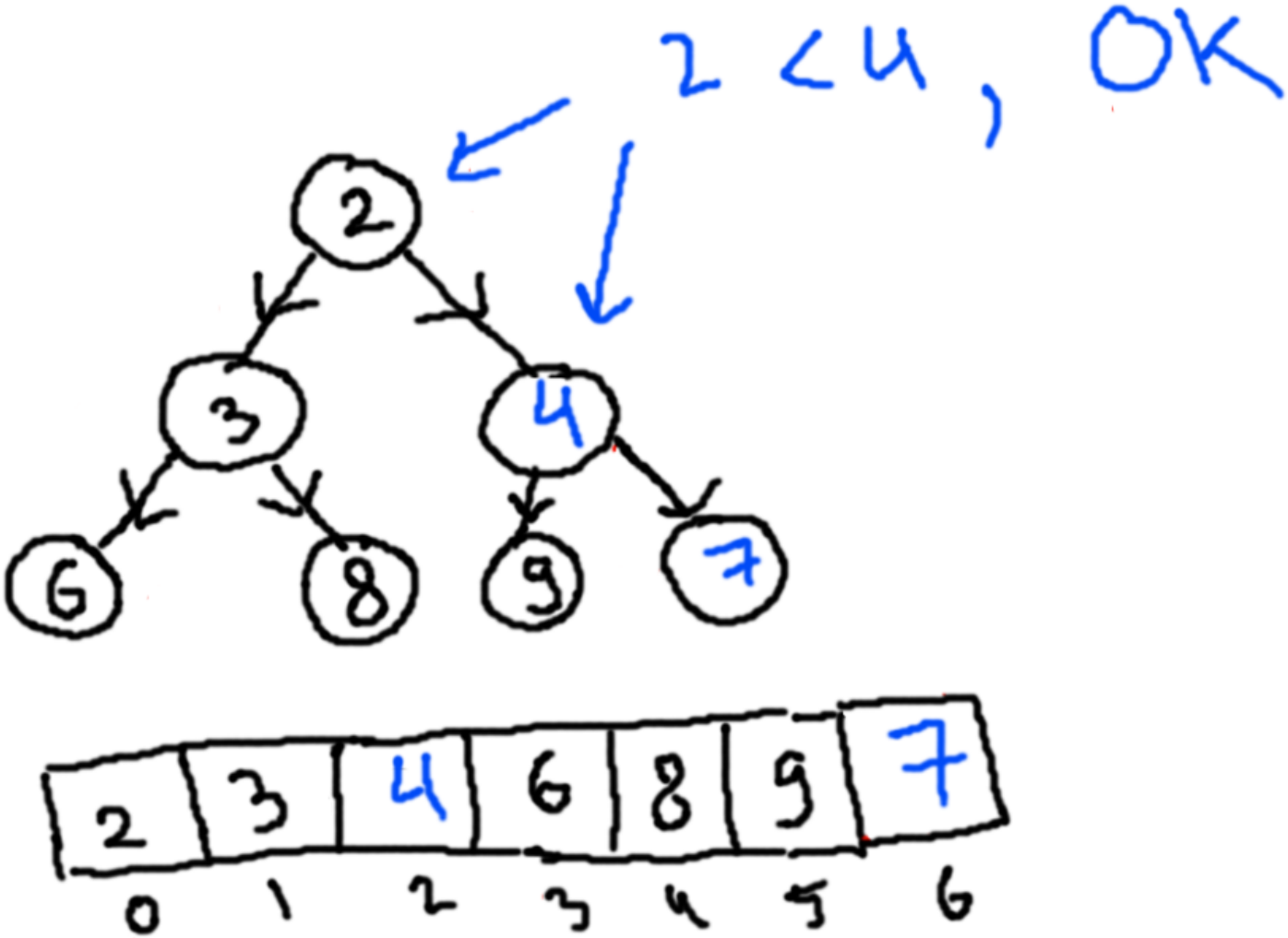
Сложность – O(log n)

# SiftUp



# SiftUp



# SiftUp

```
def SiftUp(self, child):
    parent = (child - 1) // 2
    while child > 0 and a[child] < a[parent]:
        a[child], a[parent] = a[parent], a[child]
        child = parent
        parent = (child - 1) // 2
```

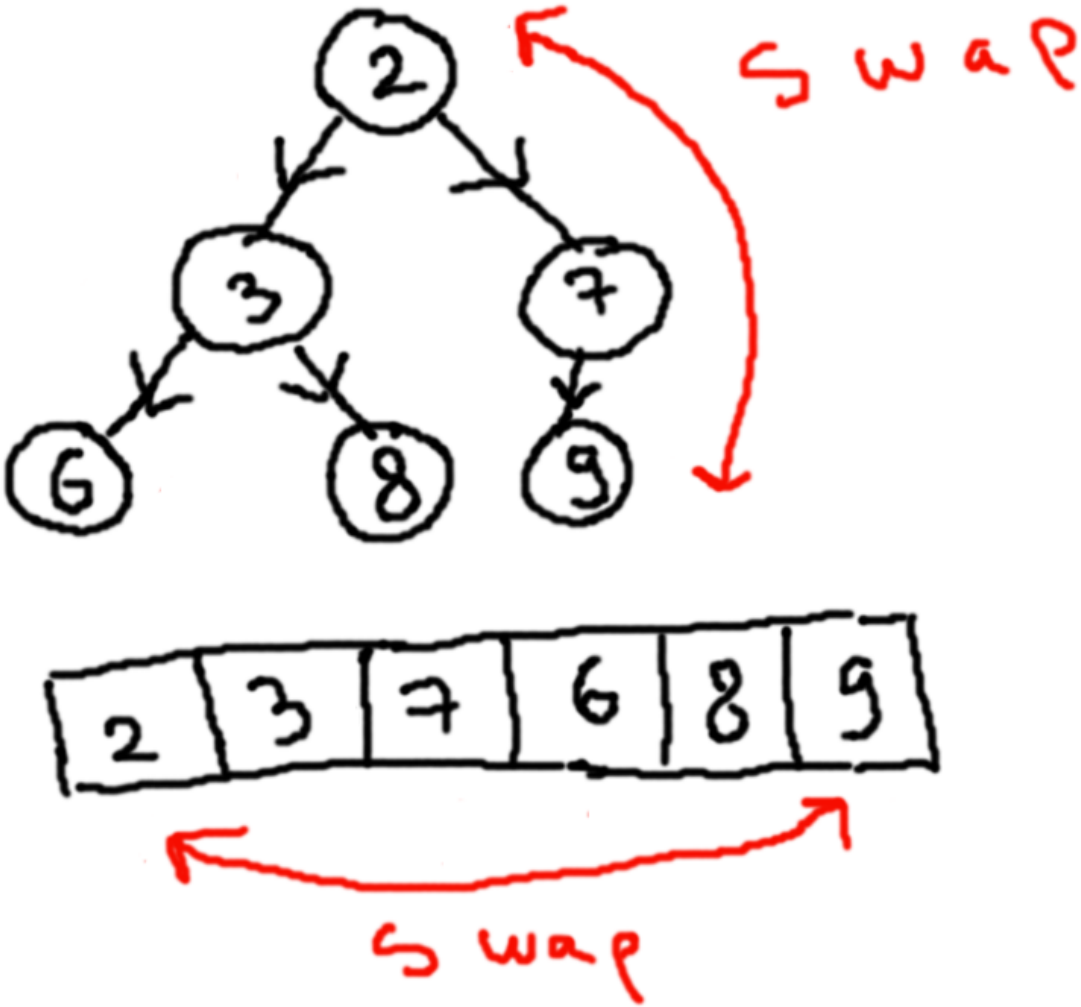
# ExtractMin

Нужно удалить минимальный элемент (то есть корень)

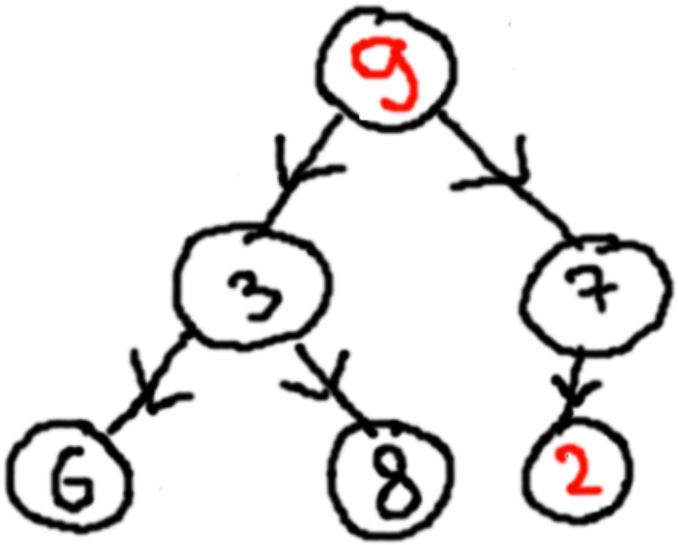
Поменяем его местами с последним элементом и удалим последний

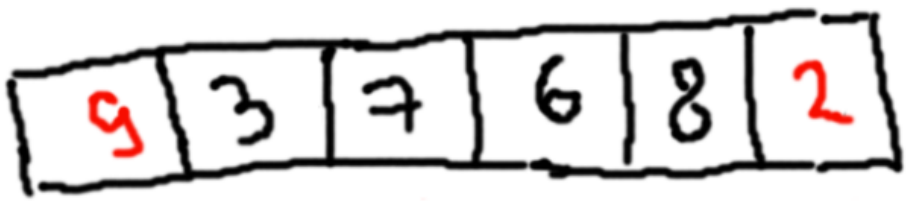
Это может поломать нам свойства кучи, их нужно будет восстановить

# ExtractMin

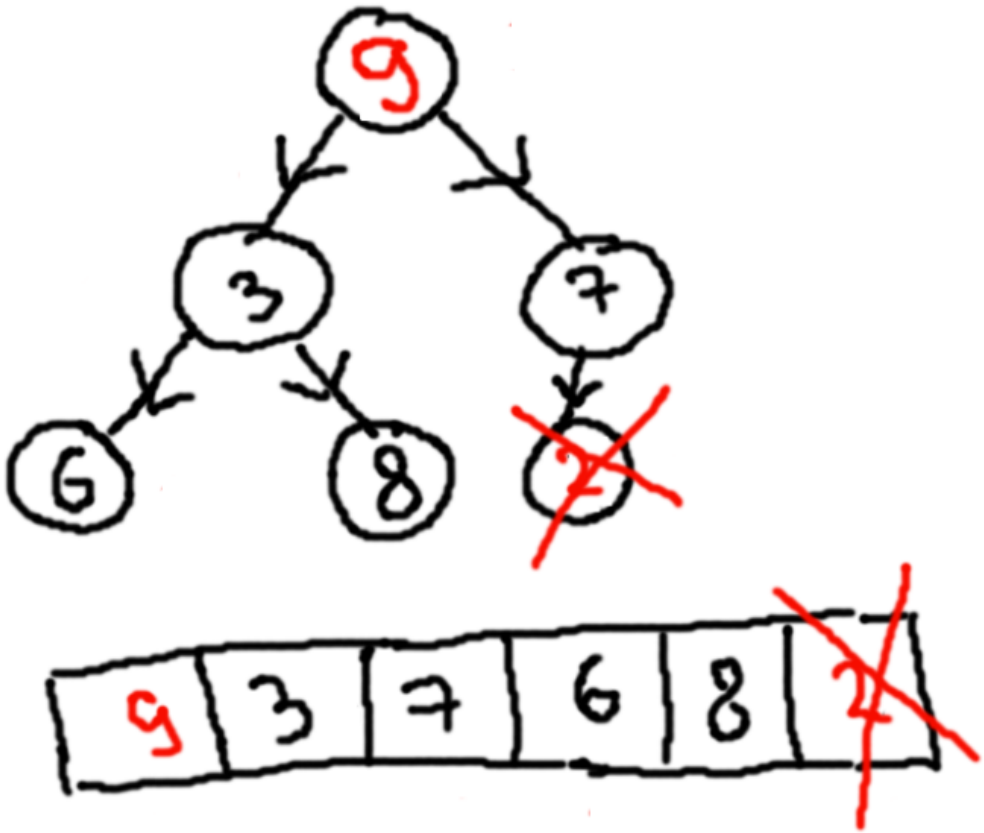


# ExtractMin

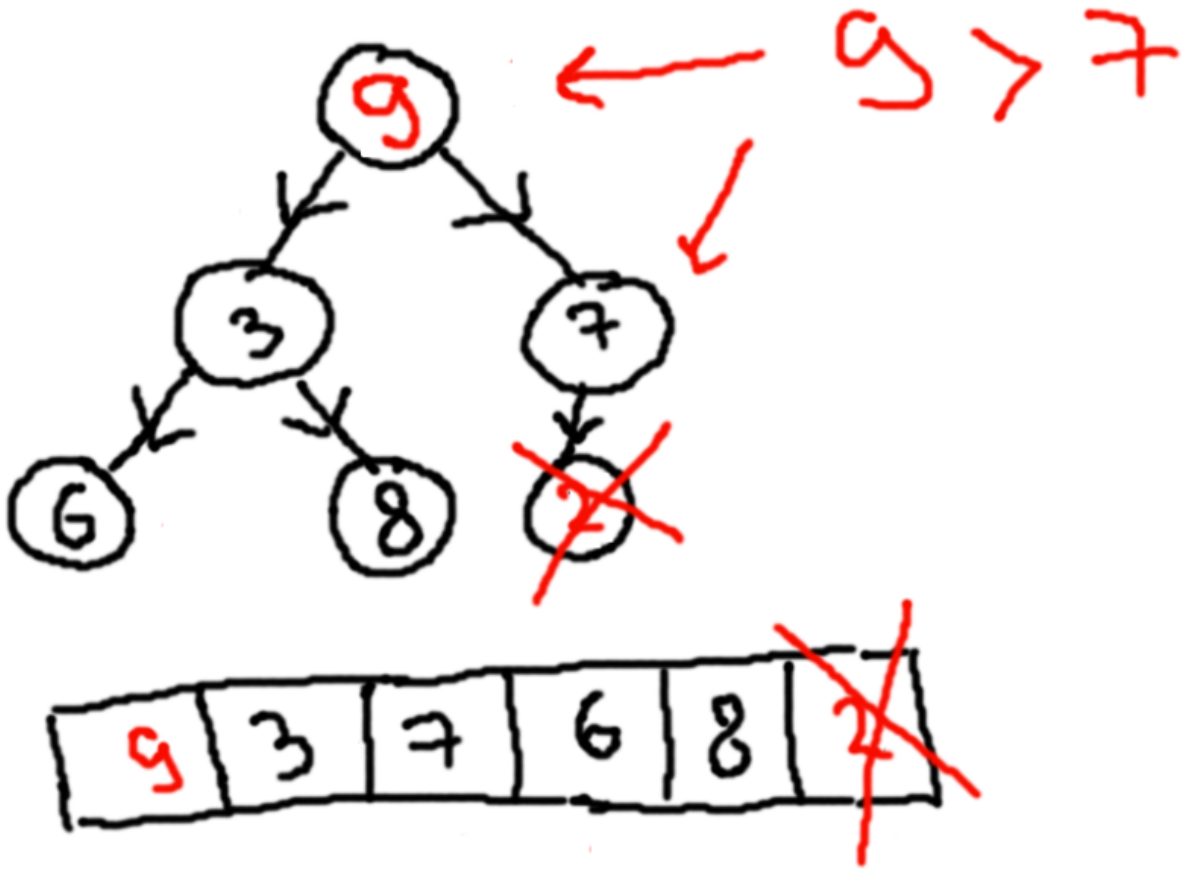


# ExtractMin



# ExtractMin



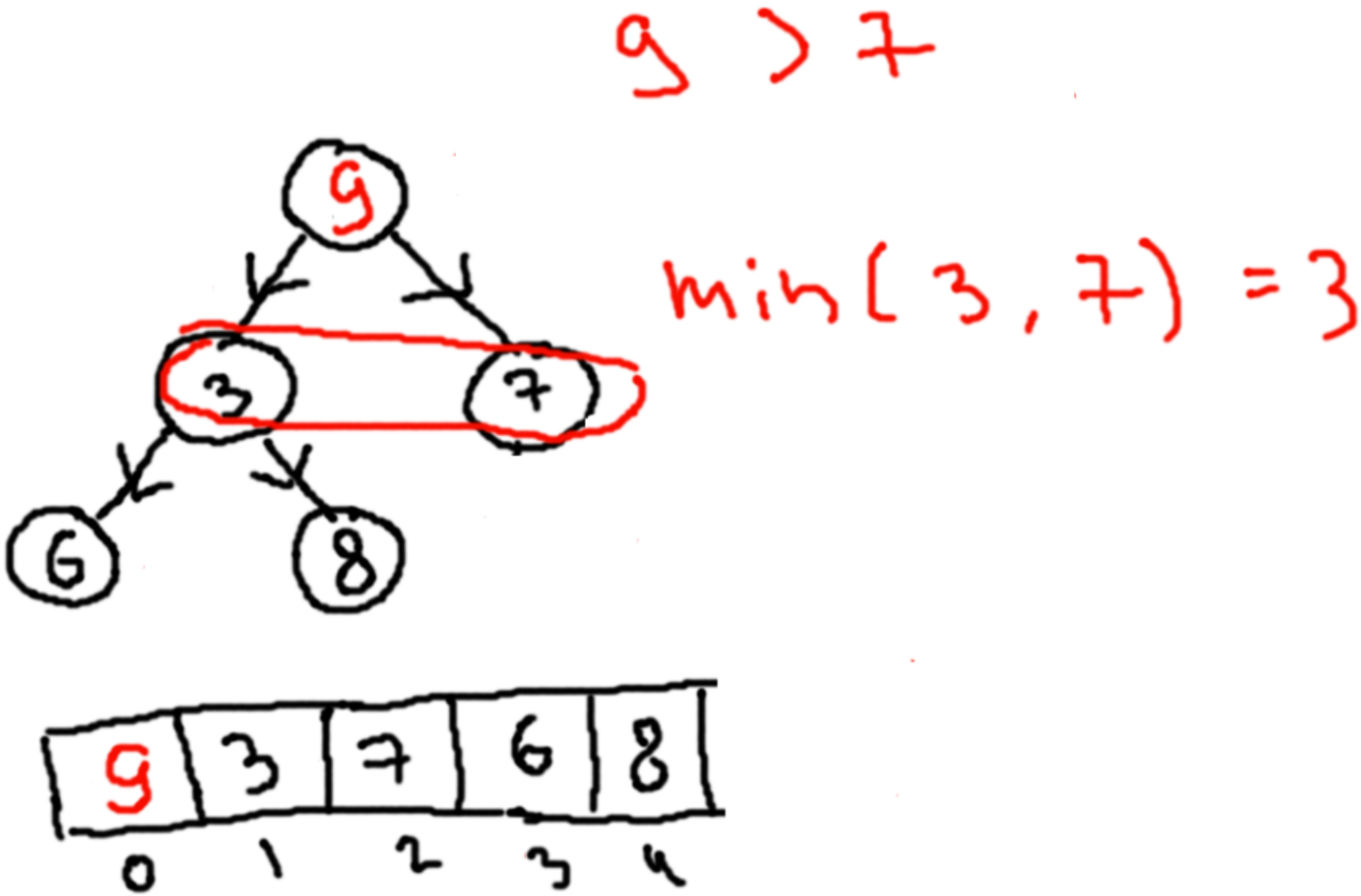
# SiftDown

Для восстановления свойств кучи будем делать SiftDown

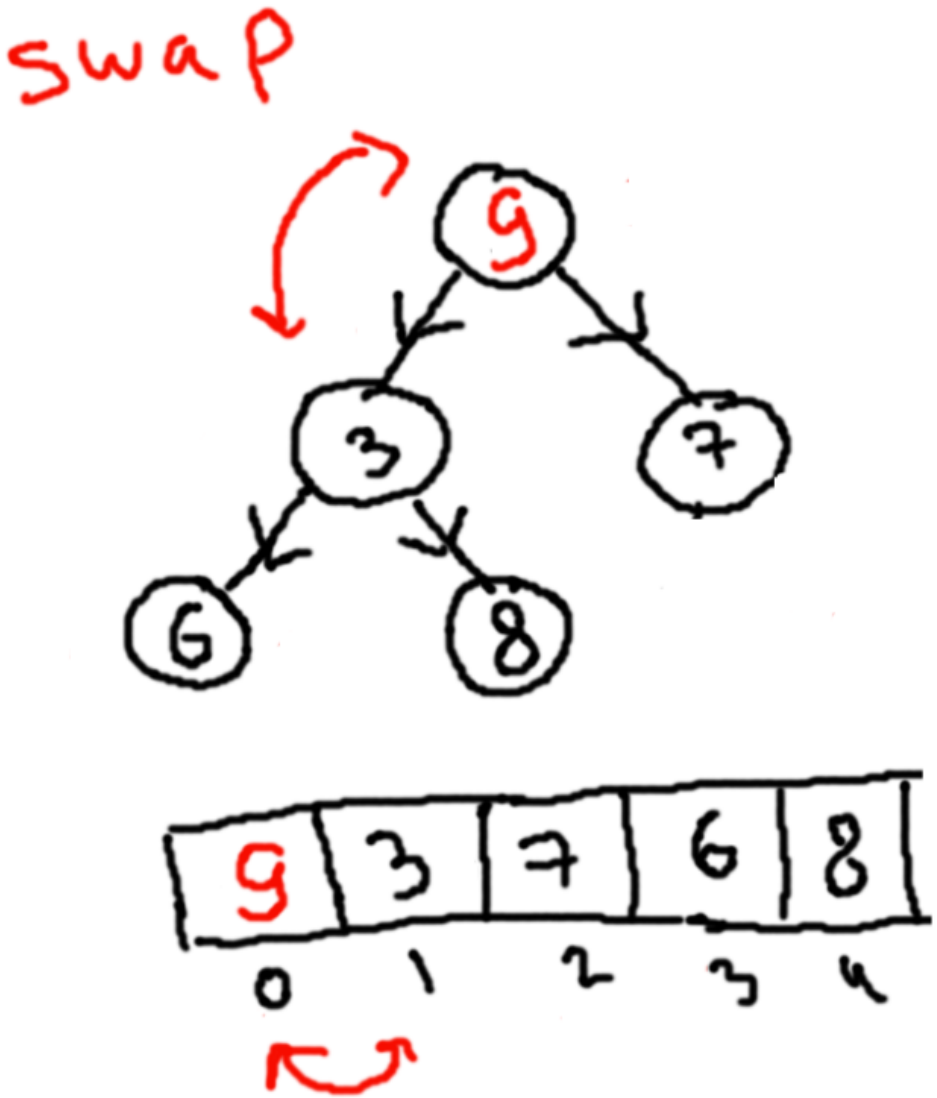
1. Если i-й элемент меньше, чем его сыновья, ничего делать не нужно.
2. Иначе меняем местами i-й элемент с наименьшим из его сыновей, после чего выполняем SiftDown для этого сына.

Сложность – O(log n)

# SiftDown

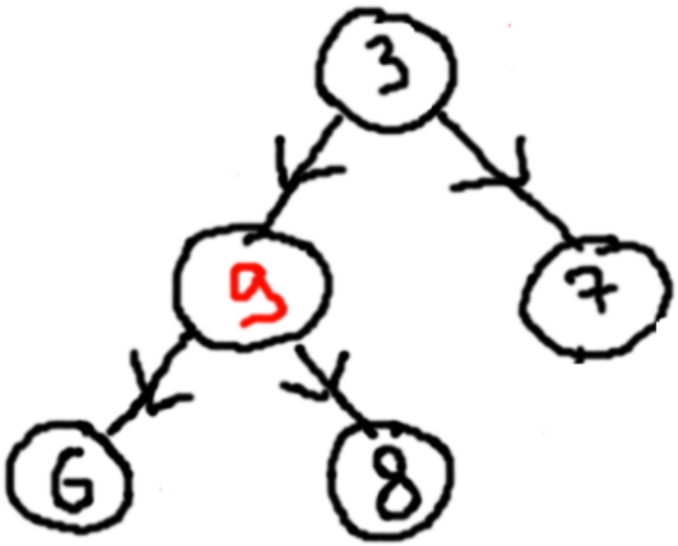


# SiftDown

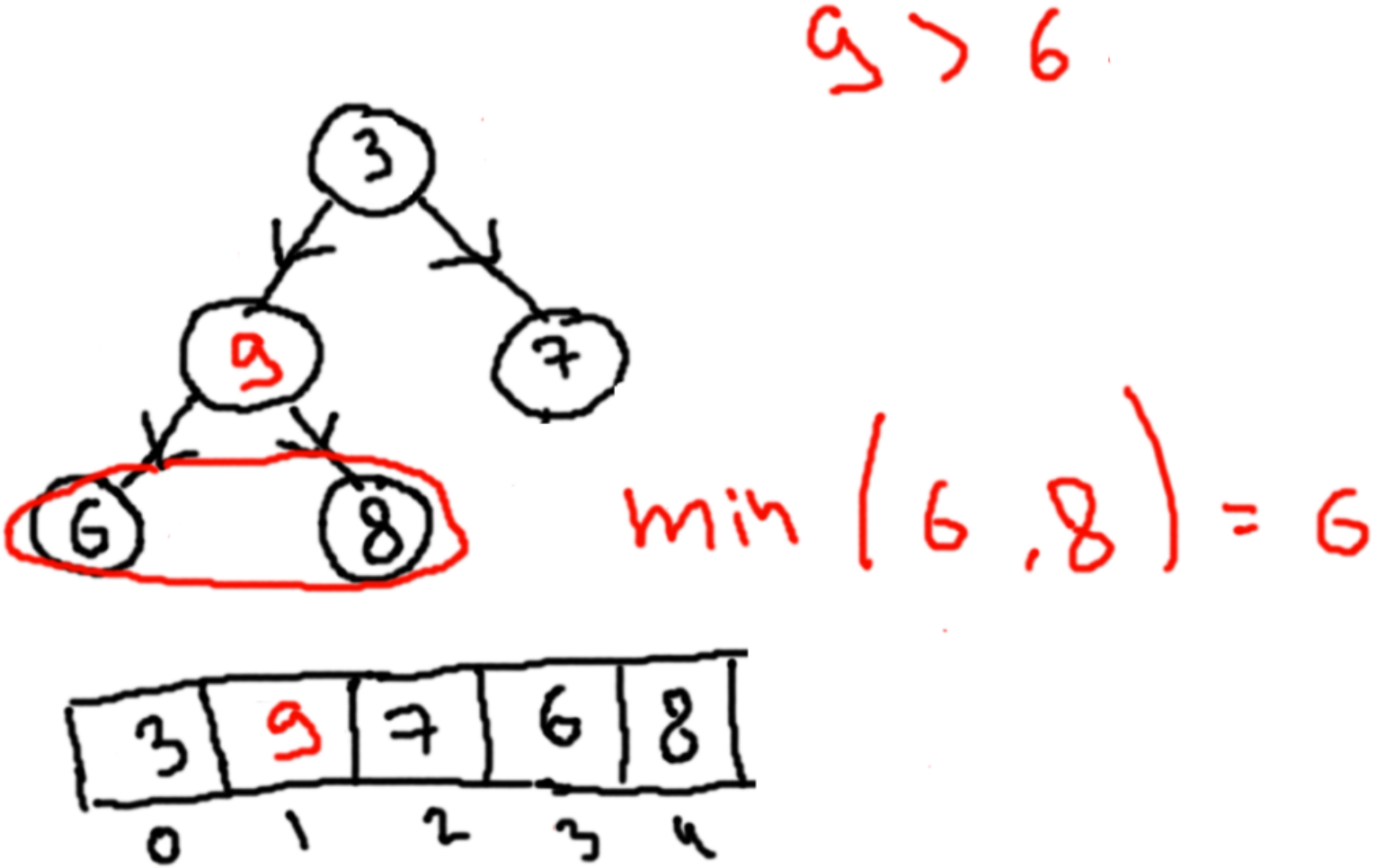


# SiftDown

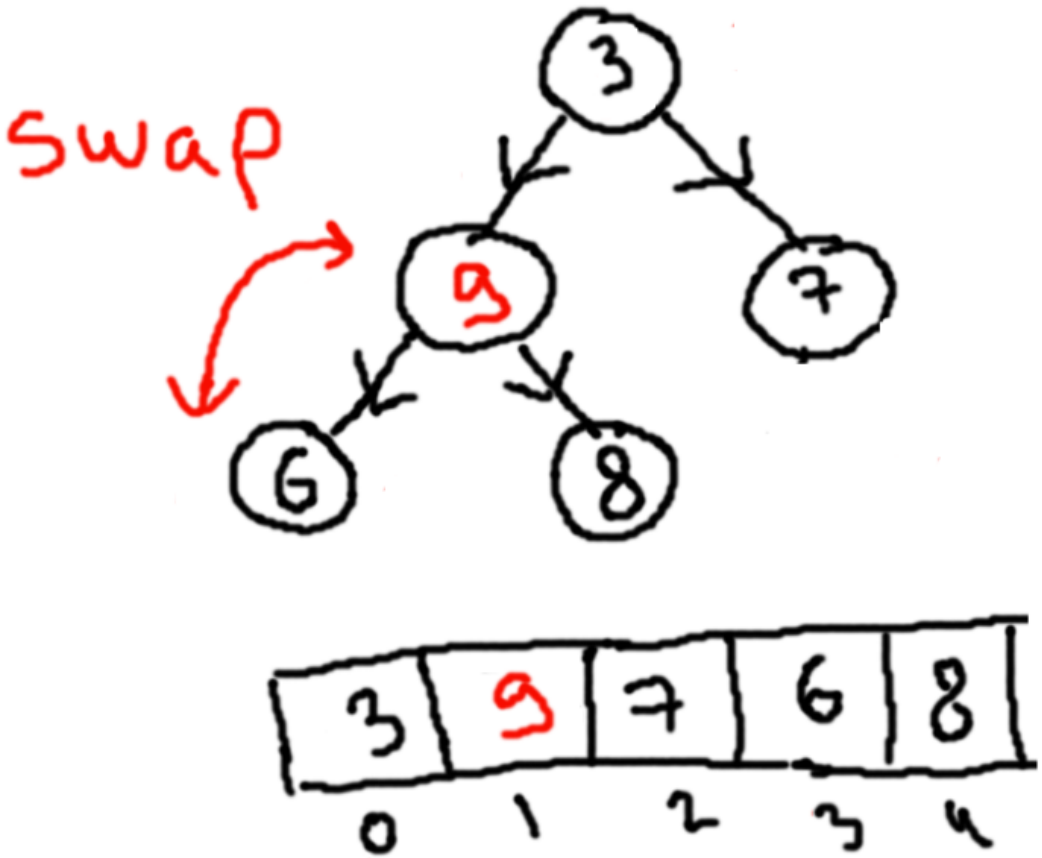
9 > 6



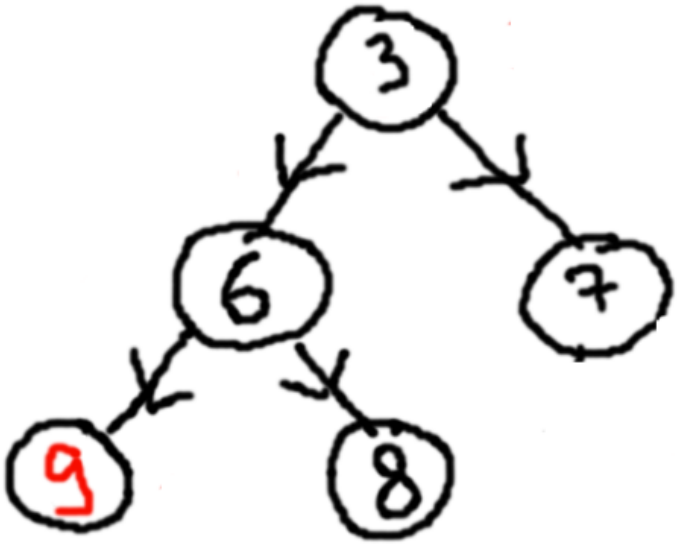

# SiftDown

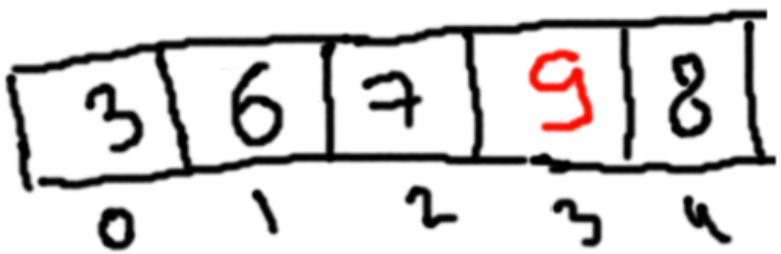


# SiftDown



# SiftDown



# SiftDown

```
def SiftDown(parent):
    while 2 * parent + 1 < len(a):
        left = 2 * parent + 1
        right = 2 * parent + 2

        minChild = left
        if right < len(a) and a[right] < a[left]:
            minChild = right

        if a[minChild] > a[parent]:
            break

        a[minChild], a[parent] = a[parent], a[minChild]
        parent = minChild
```