

**Восстановление свойств кучи (неубывающей)**

Если в куче изменяется один из элементов, то она может перестать удовлетворять свойству упорядоченности. Для восстановления этого свойства служат процедуры siftDown (просеивание вниз) и siftUp (просеивание вверх).

**siftDown**

Если значение измененного элемента увеличивается, то свойства кучи восстанавливаются функцией siftDownsiftDown.

Работа процедуры: если ii-й элемент меньше, чем его сыновья, всё поддерево уже является кучей, и делать ничего не надо. В противном случае меняем местами ii-й элемент с наименьшим из его сыновей, после чего выполняем siftDownsiftDown для этого сына. Процедура выполняется за время O(logn)

**function** siftDown(i : **int**):

**while** 2 \* i + 1 < a.heapSize // heapSize — количество элементов в куче

left = 2 \* i + 1 // left — левый сын

right = 2 \* i + 2 // right — правый сын

j = left

**if** right < a.heapSize **and** a[right] < a[left]

j = right

**if** a[i] <= a[j]

**break**

swap(a[i], a[j])

i = j

**siftUp**

Если значение измененного элемента уменьшается, то свойства кучи восстанавливаются функцией siftUpsiftUp.

Работа процедуры: если элемент больше своего отца, условие 1 соблюдено для всего дерева, и больше ничего делать не нужно. Иначе, мы меняем местами его с отцом. После чего выполняем siftUpsiftUp для этого отца. Иными словами, слишком маленький элемент всплывает наверх. Процедура выполняется за время O(logn)

**function** siftUp(i : **int**):

**while** a[i] < a[(i - 1) / 2] // i 0 — мы в корне

swap(a[i], a[(i - 1) / 2])

i = (i - 1) / 2





