двоичное дерево- Это структура данных, где у каждого узла есть 0–2 ребёнка. Само дерево хранит ссылку на общего предка — на корень дерева.

****

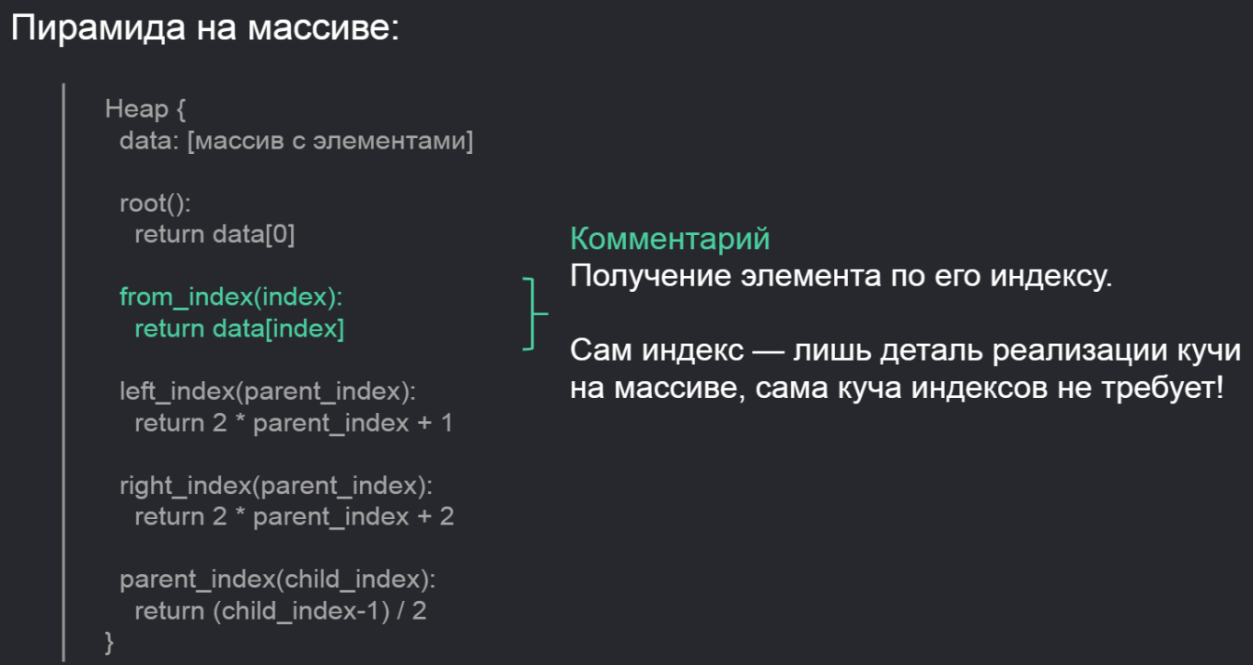
**полное двоичное дерево-** Это дерево, уровни которого мы заполняем элементами сверху вниз слева направо, причём слева идут сначала полнодетные, затем все неполнодетные.

Пирамида (куча) — полное двоичное дерево, для которого  
соблюдается правило: значение в каждом родителе не больше  
чем у детей. Такая пирамида называется пирамидой на минимум.   
Стоит отметить, что подойдут любые типы данных, для которых  
вы определите понятие сравнения. Если сравнение идёт  
по какой-то части элемента, эта часть называется ключом.

****

Структура для самой пирамиды

Заведём массив, в котором будем держать элементы нашей пирамиды  
Будем считать, что корень пирамиды хранится в ячейке  
с индексом 0  
Для любого узла пирамиды, хранящегося в ячейке под индексом i,  
будем считать, что его левый ребёнок хранится в 2i+1, а правый  
в 2i+2.  
Разберём пример: необходимо узнать, где находятся левый  
и правый дети для узла пирамиды, хранящегося в ячейке  
под номером 1

Левый ребёнок: 2 ✕ 1 + 1 = 3  
Правый ребёнок: 2 ✕ 1 + 2 = 4  
Переход к детям от родителя будет всё также за O(1)  
****

Номер родительской ячейки получается обратным действием к получению номера дочернего узла. Там умножали на 2, тут делим на 2

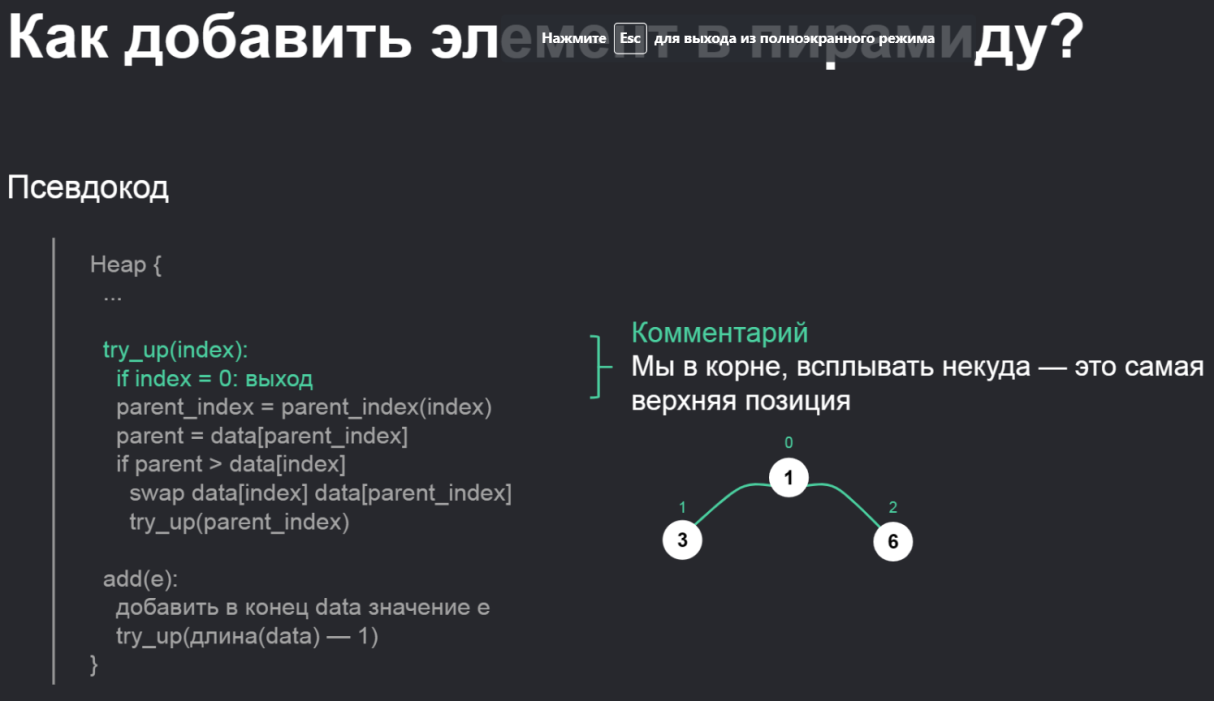
Левый ребёнок

Правый ребёнок

Корень пирамиды в 0-й ячейке

Массив значений из узлов пирамид

Если при добавлении нового элемента условие пирамиды нарушено, то запустим в этом месте операцию «всплывания» элемента, для этого  
Сравниваем значение нового узла со значением родителя, если родитель  
больше — повторяем операцию для позиции родителя.  
Время операции — O(высоты), а т. к. высота — это логарифм n, то O(log 2 n).

****

Добавляем элемент в конец массива

Добавляем элемент в пирамиду

Добавляем элемент в конец массива

Пытаемся всплыть от новой позиции добавляемого элемента

Меняем местами этот элемент с его родителем

Проверяем, нарушается ли свойство пирамиды

Находим родителя этого элемента

Детей у него нет. Он может нарушить свойство пирамиды только если слишком маленький по значению. Попробуем всплыть

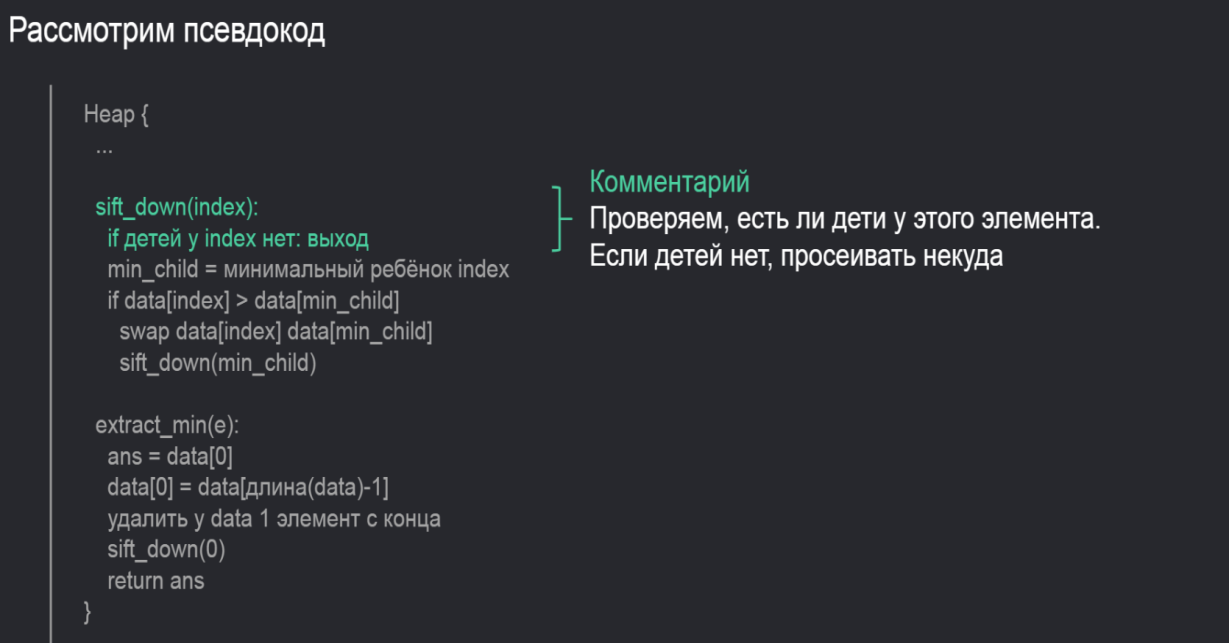
**Как удалить минимум в пирамиде?**  
Минимальный элемент в пирамиде на минимум лежит в корне, так что  
время его поиска — O(1). А вот с извлечением его из пирамиды — уже  
трудность.

**Реализация**:  
● вынем из корня элемент  
● заполним корень значением, лежащим в самом последнем узле пирамиды  
● удалим последний узел

Однако может нарушиться свойство пирамиды, если в корне будет значение большее, чем у его детей.

Если м видим, что свойство пирамиды нарушено, то сделаем операцию просеивания относительно нового корня пирамиды, пока свойство пирамидности  
не восстановится.

**Операция «просеивания»**Сравниванием значение родителя и минимального ребенка, если родитель  
больше — меняем местами.  
Время операции — O(высоты), а т. к. высота — это логарифм n, то O(log 2 n).

****

Помечаем ячейку, откуда переместили элемент в корень, как свободную

Помещаем в корень последний элемент массива

Минимум нам известен — он в корне

Извлекаем минимум из пирамиды

Поменяемся местами с этим ребёнком и

Попытаемся просеиваться дальше

Это индекс самого маленького ребёнка

Проверяем, нарушается ли свойство пирамиды, сравнивая с минимальным значением из детей этого элемента

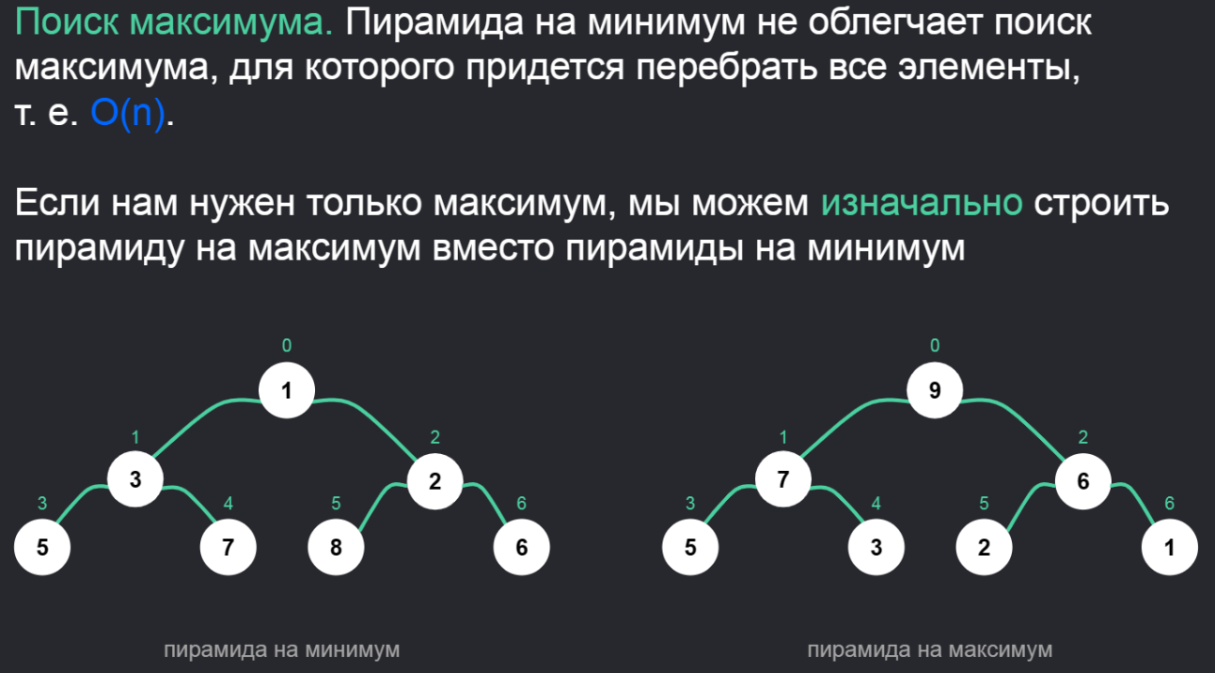
Пытаемся просеять новый корень.

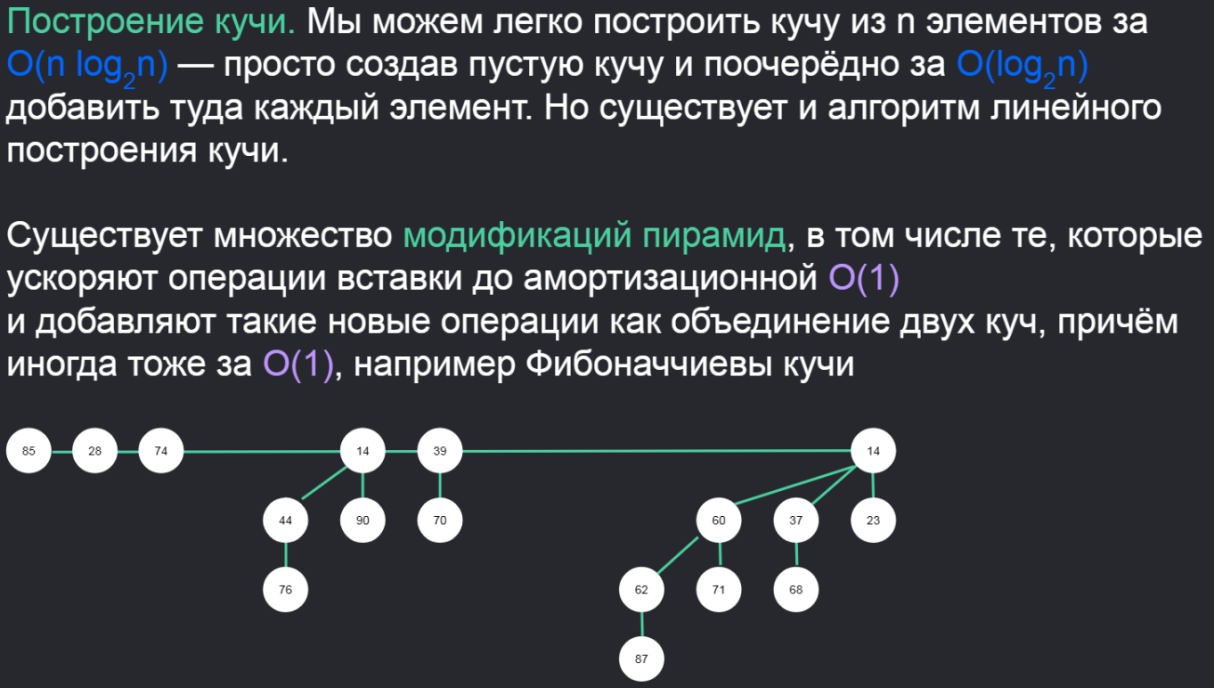
Теперь с пирамидой всё в порядке — возвращаем наш минимум.

**Другие операции с пирамидой**

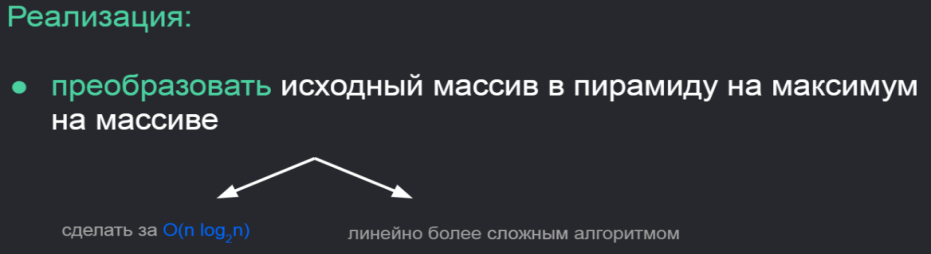
****

если мы уменьшаем значение элемента, после уменьшения необходимо провести всплывание. Это займёт O(log 2 n)

****

****

Пирамидальная сортировка  
Задача: необходимо отсортировать данный массив через  
использование пирамид

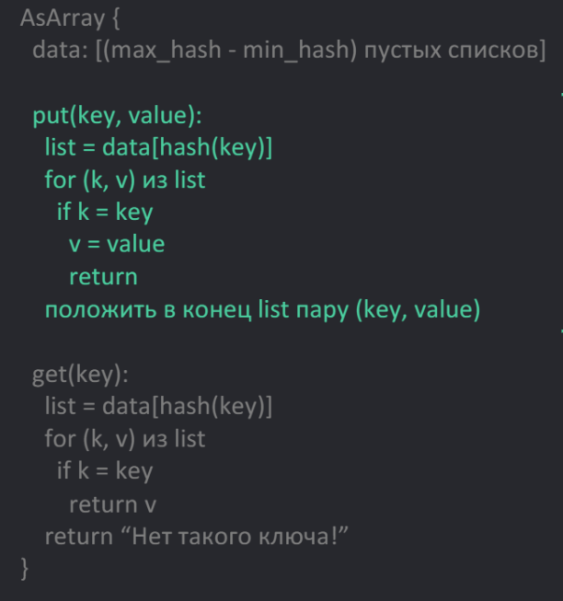
****

****

● извлекаем 8, перемещая его на последнее место просеиваем  
● извлекаем 6, перемещая его на последнее место просеиваем и т. д.

Выводы:  
● занимает времени O(n log 2  
n)  
● неустойчивая — не сохраняет порядок на одинаковых для сравнения  
элементов  
● на месте — не требует дополнительной памяти свыше O(1)  
Визуализация: https://visualgo.net/en/sorting → MER

**Хеш-функцией** объекта называется алгоритм генерации какого-либо  
числа (хеша), который гарантирует, что при применении к равным объектам даст одинаковое значение.  
при перестановки 2 букв в хэш-функции надо чтобы менялось её значение тогда это хорошая ф.

Массив списков размером в количество  
всевозможных хешей

Достаем список из ячейки с номером  
равным хешу ключа

Работаем с этим списком так же, как и в предыдущем наивном подходе

Достаем список из ячейки с номером равным хешу ключа

Работаем с этим списком так же, как и в предыдущем наивном подходе

Ужас с памятью