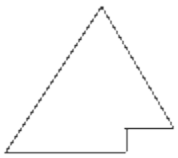
Пирамидальная сортировка

В основе пирамидальной сортировки лежит специальный тип бинарного дерева, называемый пирамидой. Сортировка начинается с построения пирамиды. Итак, назовем пирамидой бинарное дерево высоты k, в котором:

- все узлы имеют глубину k или k-1 - дерево сбалансированное.
- при этом уровень k-1 полностью заполнен, а уровень k заполнен слева направо, т.е форма пирамиды имеет приблизительно такой вид:



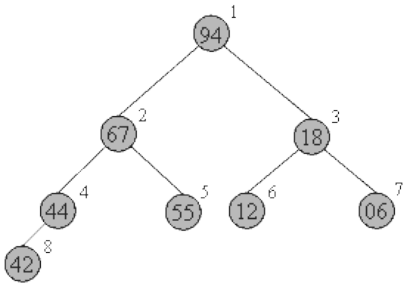
- выполняется "свойство пирамиды": каждый элемент меньше, либо равен родителю.

Как хранить пирамиду? Наименее хлопотно - поместить ее в массив.

Соответствие между геометрической структурой пирамиды как дерева и массивом устанавливается по следующей схеме:

- в a[0] хранится корень дерева
- левый и правый сыновья элемента a[i] хранятся, соответственно, в a[2i+1] и a[2i+2]

Таким образом, для массива, хранящего в себе пирамиду, выполняется следующее характеристическое свойство: a[i] >= a[2i+1] и a[i] >= a[2i+2].



Запишем в виде массива пирамиду, изображенную выше. Слева-направо, сверху-вниз: 94 67 18 44 55 12 06 42. На рисунке место элемента пирамиды в массиве обозначено цифрой справа-вверху от него. Восстановить пирамиду из массива как геометрический объект легко - достаточно вспомнить схему хранения и нарисовать, начиная от корня.

Фаза 1 сортировки: построение пирамиды

Начать построение пирамиды можно с a[k]...a[n], k = [size/2]. Эта часть массива удовлетворяет свойству пирамиды, так как не существует индексов i,j: i = 2i+1 (или j = 2i+2)... Просто потому, что такие i,j находятся за границей массива.

Следует заметить, что неправильно говорить о том, что a[k]..a[n] является пирамидой как самостоятельный массив. Это, вообще говоря, не верно: его элементы могут быть

любыми. Свойство пирамиды сохраняется лишь в рамках исходного, основного массива a[0]...a[n].

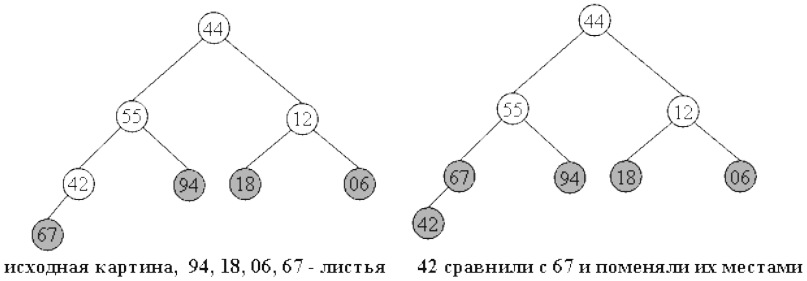
Далее будем расширять часть массива, обладающую столь полезным свойством, добавляя по одному элементу за шаг. Следующий элемент на каждом шаге добавления - тот, который стоит перед уже готовой частью.

Чтобы при добавлении элемента сохранялась пирамидальность, будем использовать следующую процедуру расширения пирамиды a[i+1]..a[n] на элемент a[i] влево:

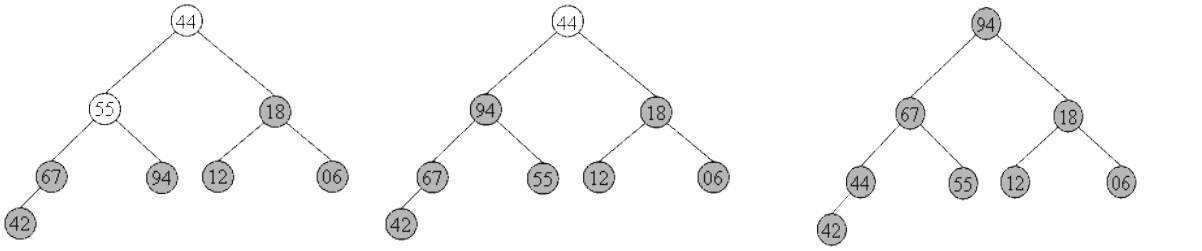
1. Смотрим на сыновей слева и справа - в массиве это a[2i+1] и a[2i+2] и выбираем наибольшего из них.
2. Если этот элемент больше a[i] - меняем его с a[i] местами и идем к шагу 2, имея в виду новое положение a[i] в массиве. Иначе конец процедуры.

Новый элемент "просеивается" сквозь пирамиду.

На рисунках ниже изображен процесс построения. Неготовая часть пирамиды (начало массива) окрашена в белый цвет, удовлетворяющий свойству пирамиды конец массива - в темный.



исходная картина, 94, 18, 06, 67 - листья 42 сравнили с 67 и поменяли их местами

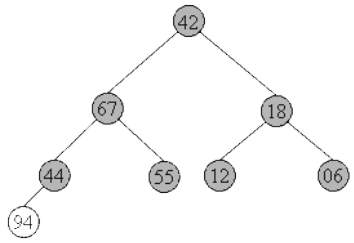


12 сравнили с max(18,6)=18 55 сравнили с max(67,94)=94 44 просеяли сквозь 94, 67, остановились на 42

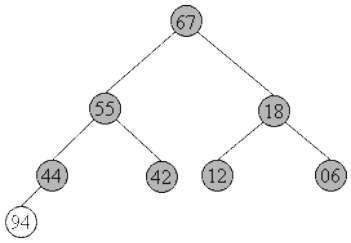
Фаза 2: собственно сортировка

Итак, задача построения пирамиды из массива успешно решена. Как видно из свойств пирамиды, в корне всегда находится максимальный элемент. Отсюда вытекает алгоритм фазы 2:

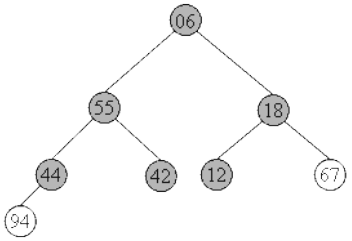
1. Берем верхний элемент пирамиды a[0]...a[n] (первый в массиве) и меняем с последним местами. Теперь "забываем" об этом элементе и далее рассматриваем массив a[0]...a[n-1]. Для превращения его в пирамиду достаточно просеять лишь новый первый элемент.
2. Повторяем шаг 1, пока обрабатываемая часть массива не уменьшится до одного элемента.



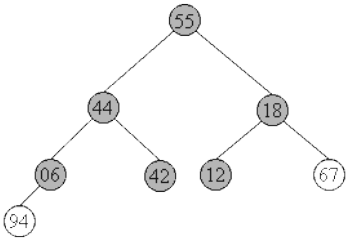
обменяли 94 и 42, забыли о 94



просеяли 42 сквозь 67, 55



обменяли 06 и 67, забыли о 67



просеяли 06 сквозь 55, 44…

Очевидно, в конец массива каждый раз попадает максимальный элемент из текущей пирамиды, поэтому в правой части постепенно возникает упорядоченная последовательность.

Итог

Построение пирамиды занимает O(n log n) операций, причем более точная оценка дает даже O(n) за счет того, что реальное время выполнения процедуры просеивания элементиа зависит от высоты уже созданной части пирамиды.

Вторая фаза занимает O(n log n) времени: O(n) раз берется максимум и происходит просеивание бывшего последнего элемента. Плюсом является стабильность метода: среднее число пересылок (n log n)/2, и отклонения от этого значения сравнительно малы.

Пирамидальная сортировка не использует дополнительной памяти. Метод не является устойчивым: по ходу работы массив так "перетряхивается", что исходный порядок элементов может измениться случайным образом. Поведение неестественно: частичная упорядоченность массива никак не учитывается.