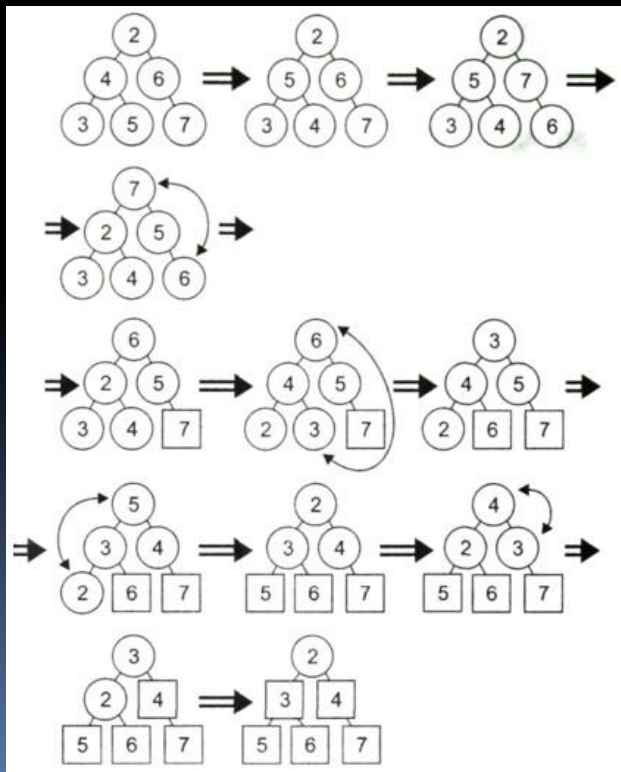
# МЕТОД СОРТИРОВКИ ПИРАМИДОЙ.

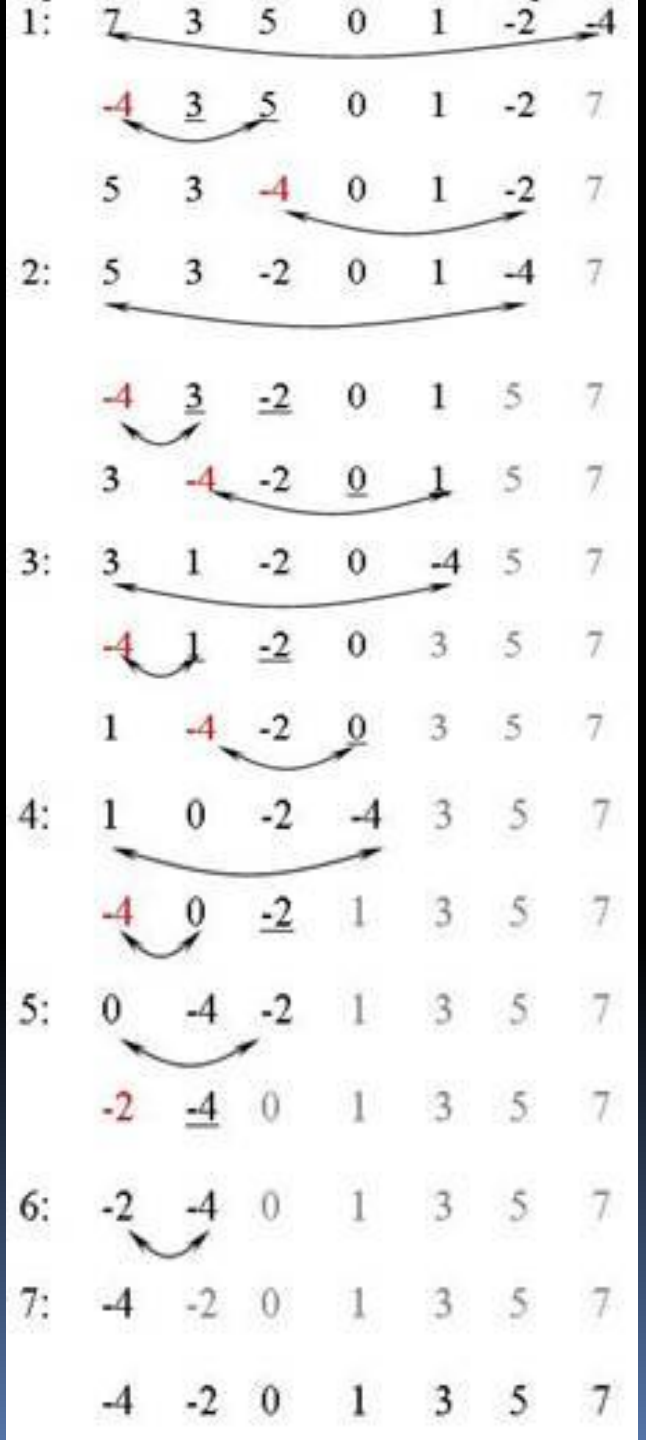
Идея метода довольно проста: найти наибольший элемент файла и поставить его на место N, найти следующий максимум и поставить его на место N-1 и т.д. до 2-го элемента.



- Сортировка пирамидой использует бинарное сортирующее дерево. Сортирующее дерево — это такое дерево, у которого выполнены условия:
- Каждый лист имеет глубину либо $d$, либо $d-1$, $d$ — максимальная глубина дерева.
- Значение в любой вершине не меньше (другой вариант — не больше) значения её потомков.
- Удобная структура данных для сортирующего дерева — такой массив Array, что Array[0] — элемент в корне, а потомки элемента Array[i] являются Array[2i+1] и Array[2i+2].

# Алгоритм сортировки будет состоять из двух основных шагов:

- 1. Выстраиваем элементы массива в виде сортирующего дерева
- Array[i]>=Array[2i+1]
- Array[i]>=Array[2i+2]
- При 0 <=i<n/2
- Этот шаг требует O(n) операций.
- 2. Будем удалять элементы из корня по одному за раз и перестраивать дерево. То есть на первом шаге обмениваем Array[1] и Array[n], преобразовываем Array[1], Array[2], … , Array[n-1] в сортирующее дерево. Затем переставляем Array[1] и Array[n-1], преобразовываем Array[1], Array[2], … , Array[n-2] в сортирующее дерево. Процесс продолжается до тех пор, пока в сортирующем дереве не останется один элемент. Тогда Array[1], Array[2], … , Array[n] — упорядоченная последовательность.
- Этот шаг требует O(n *log* n) операций.


1: 7 3 5 0 1 -2 -4
-4 3 5 0 1 -2 7
5 3 -4 0 1 -2 7
2: 5 3 -2 0 1 -4 7
-4 3 -2 0 1 5 7
3 -4 -2 0 1 5 7
3: 3 1 -2 0 -4 5 7
-4 1 -2 0 3 5 7
1 -4 -2 0 3 5 7
4: 1 0 -2 -4 3 5 7
-4 0 -2 1 3 5 7
5: 0 -4 -2 1 3 5 7
-2 -4 0 1 3 5 7
6: -2 -4 0 1 3 5 7
7: -4 -2 0 1 3 5 7
-4 -2 0 1 3 5 7

# Достоинства

- Имеет доказанную оценку худшего случая O(n*log n)
- Сортирует на месте, то есть требует всего *O*(1) дополнительной памяти.
- никаких дополнительных переменных, нужно лишь понимать схему.
- узлы хранятся от вершины и далее вниз, уровень за уровнем.
- узлы одного уровня хранятся в массиве слева направо.

# Недостатки

- Сложен в реализации.
- На почти отсортированных массивах работает столь же долго, как и на хаотических данных.
- На одном шаге выборку приходится делать хаотично по всей длине массива — поэтому алгоритм плохо сочетается с кэшированием и подкачкой памяти.
- Методу требуется «мгновенный» прямой доступ; не работает на связанных списках и других структурах памяти последовательного доступа.
- Из-за сложности алгоритма выигрыш получается только на больших *n*. На небольших *n* (до нескольких тысяч) быстрее сортировка Шелла.
- Поведение неестественно: частичная упорядоченность массива никак не учитывается.