**КФУ, Институт ИТИС, 1 курс, 2 семестр, 2021**

**Семестровая работа №1:**

**«Heapsort.**

**Treesort»**

**Выполнила:**

Галиева Илюза Ильдаровна

группа 11-203

**Преподаватель:**

Андреичев М.Д.

*Heapsort*

**Heapsort** или **Пирамидальная сортировка**(«Сортировка кучей») — алгоритм сортировки, работающий в худшем, в среднем и в лучшем случае (то есть гарантированно) за O(n\*log n) операций при сортировке n элементов. Количество применяемой служебной памяти не зависит от размера массива (то есть, O(1)).

*Историческая справка*

Пирамидальная сортировка была предложена [Дж. Уильямсом](https://ru.wikipedia.org/w/index.php?title=%D0%A3%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC%D1%81,_%D0%94%D0%B6%D0%BE%D0%BD_%D0%92%D0%B8%D0%BB%D1%8C%D1%8F%D0%BC_%D0%94%D0%B6%D0%BE%D1%81%D0%B5%D1%84&action=edit&redlink=1" \o "Уильямс, Джон Вильям Джосеф (страница отсутствует)) в 1964 году.

*Описание*

Heapsort похожа на сортировку выбором, где мы сначала ищем максимальный элемент и помещаем его в конец. Далее мы повторяем ту же операцию для оставшихся элементов.

# *Что такое [двоичная куча](https://www.geeksforgeeks.org/binary-heap/)?*

Давайте сначала определим законченное двоичное дерево. Законченное двоичное дерево — это двоичное дерево, в котором каждый уровень, за исключением, возможно, последнего, имеет полный набор узлов, и все листья расположены как можно левее.

Двоичная куча — это законченное двоичное дерево, в котором элементы хранятся в особом порядке: значение в родительском узле больше (или меньше) значений в его двух дочерних узлах. Первый вариант называется max-heap, а второй — min-heap. Куча может быть представлена двоичным деревом или массивом.

# *Почему для двоичной кучи используется представление на основе массива ?*

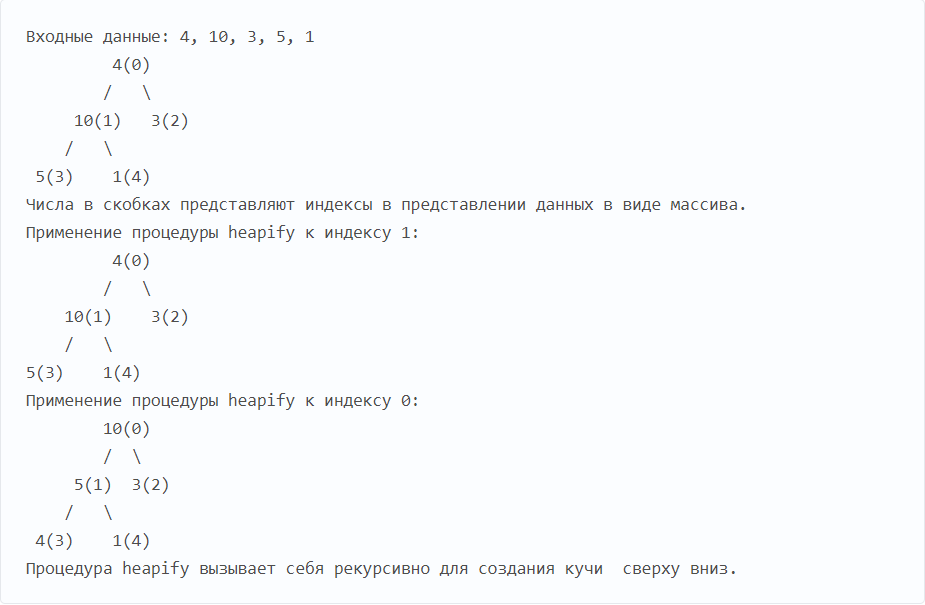
Поскольку двоичная куча — это законченное двоичное дерево, ее можно легко представить в виде массива, а представление на основе массива является эффективным с точки зрения расхода памяти. Если родительский узел хранится в индексе I, левый дочерний элемент может быть вычислен как 2\*I + 1, а правый дочерний элемент — как 2\*I + 2 (при условии, что индексирование начинается с 0).

***Алгоритм пирамидальной сортировки в порядке по возрастанию:***

1. Постройте max-heap из входных данных.
2. На данном этапе самый большой элемент хранится в корне кучи. Замените его на последний элемент кучи, а затем уменьшите ее размер на 1. Наконец, преобразуйте полученное дерево в max-heap с новым корнем.
3. Повторяйте вышеуказанные шаги, пока размер кучи больше 1.

# *Построение кучи*

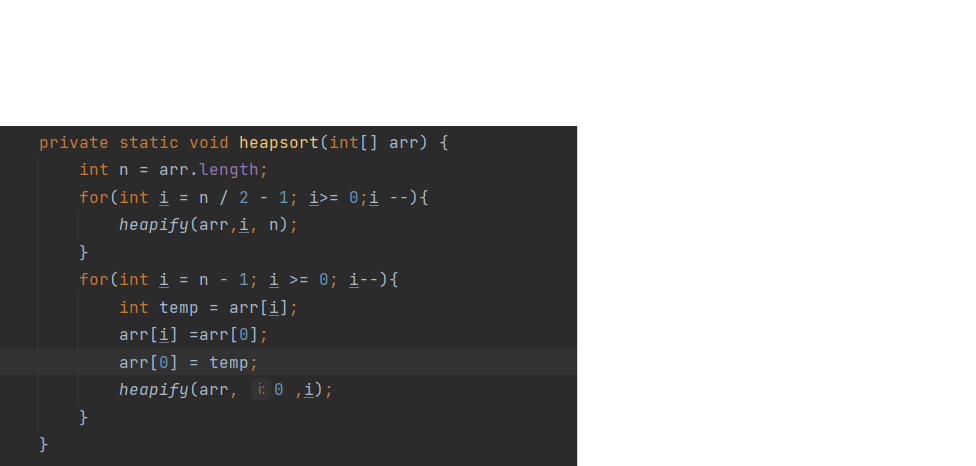
Процедура преобразования в кучу (далее процедура heapify) может быть применена к узлу, только если его дочерние узлы уже преобразованы. Таким образом, преобразование должно выполняться снизу вверх. Давайте разберемся с помощью примера:

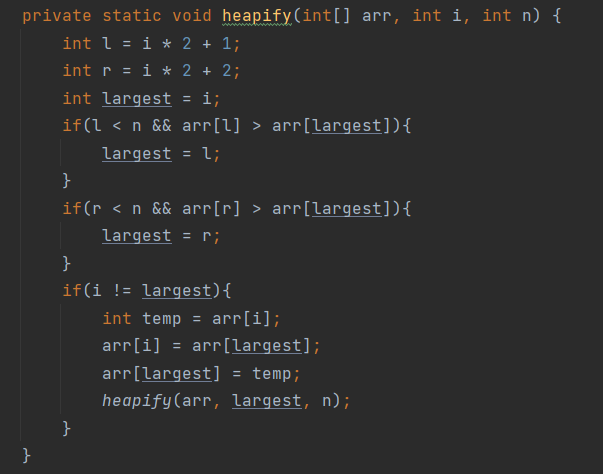


*Реализация*

*-*На вход функция heapsort() принимает массив чисел, а функция heapify() принимает массив и два индекса;

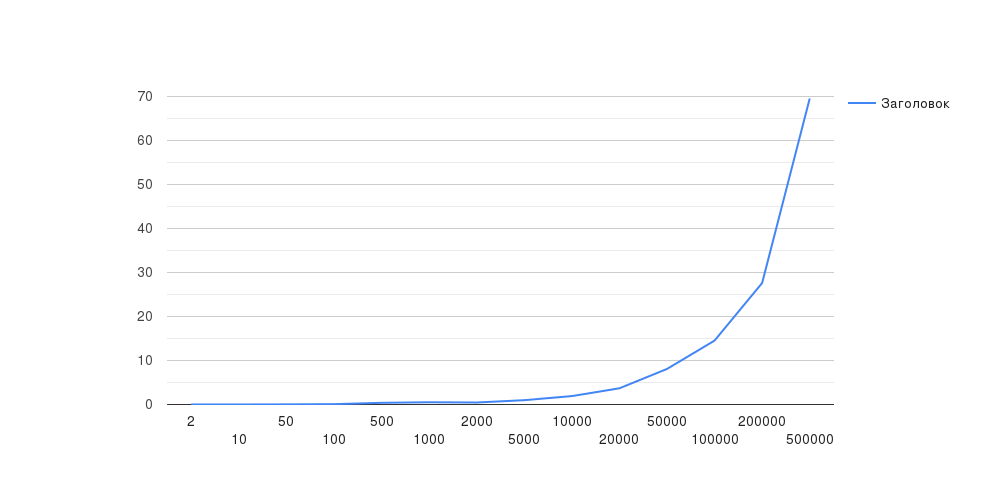
- Функция heapsort() сортирует массив, а функция heapify() преобразовывает в кучу.





*Оценка времени*

  Алгоритм сортировки, работающий в худшем, в среднем и в лучшем случае (то есть гарантированно) за O(n log n) операций при сортировке n элементов. Количество применяемой служебной памяти не зависит от размера массива (то есть, O(1)).



*Treesort*

*Tree sort или с****ортировка бинарным деревом*** – алгоритм сортировки, который заключается в построении двоичного дерева поиска по ключам массива, с последующим построением результирующего массива упорядоченных элементов путем обхода дерева.

*Описание*

### **Алгоритм:**

**Шаг 1:** Возьмите входные элементы в массив.

**Шаг 2.** Создайте двоичное дерево поиска, вставив элементы данных из массива в [двоичное дерево поиска](http://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/).

**Шаг 3.** Выполните обход дерева по порядку, чтобы получить элементы в отсортированном порядке.

## Преимущества:

-Основное преимущество алгоритма сортировки по дереву заключается в том, что мы можем очень легко вносить изменения, как в связанном списке.

-Сортировка в алгоритме сортировки по дереву выполняется так же быстро, как и алгоритм быстрой сортировки.

Недостатки:

-Наихудший случай возникает, когда элементы в массиве уже отсортированы.

-В худшем случае время выполнения алгоритма сортировки по дереву равно 0 (n^2).

Двоичным(бинарным) деревом назовем упорядоченную структуру данных,

в которой каждому элементу - предшественнику или корню (под)дерева -

поставлены в соответствие по крайней мере два других элемента (преемника).

Причем для каждого предшественника выполнено следующее правило:

левый преемник всегда меньше, а правый преемник всегда больше или равен

предшественнику.

Вместо “предшественник” и “преемник” также употребляют термины

“родитель” и “сын”. Все элементы дерева также называют “узлами”.

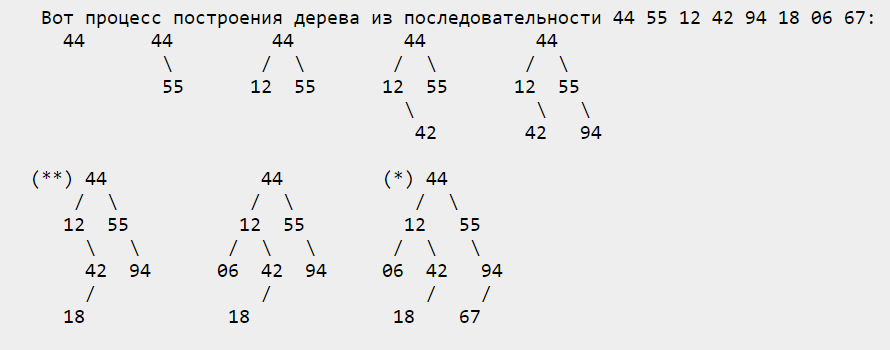
При добавлении в дерево нового элемента его последовательно сравнивают

с нижестоящими узлами, таким образом вставляя на место.

Если элемент >= корня - он идет в правое поддерево, сравниваем его уже

с правым сыном, иначе - он идет в левое поддерево, сравниваем с левым,

и так далее, пока есть сыновья, с которыми можно сравнить.



Дерево может быть и более-менее ровным, как на (\*), может и иметь всего

две основные ветви (\*\*), а если входная последовательность уже отсортирована,

то дерево выродится в линейный список.

*Реализация*

Чтобы написать Java-программу для сортировки по дереву, нужно:

1)Класс узлов, представляющий каждый узел в двоичном дереве поиска.

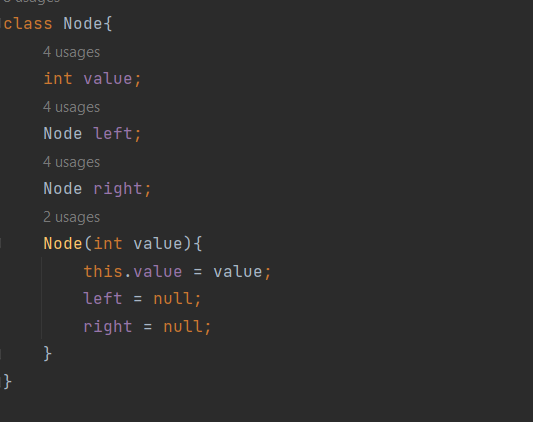
2)Метод вставки узлов в двоичное дерево поиска. Логика для вставки нового узла в двоичное дерево поиска выглядит так, как указано ниже.

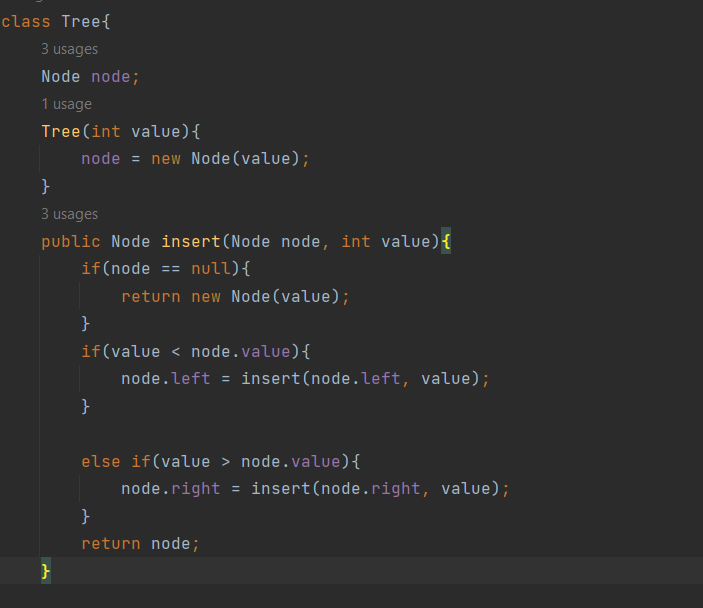
-Если значение нового узла меньше текущего узла, переместитесь влево.

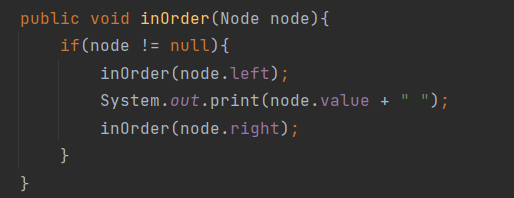
-Если значение нового узла больше, чем текущий узел, переместитесь вправо.

-Когда текущий узел равен нулю, это означает, что достигнут конечный узел. В эту позицию должен быть вставлен новый узел.

1. Метод обхода дерева для упорядочения элементов.







*Оценка времени*

Средняя временная сложность сортировки по дереву составляет O (nlogn), поскольку вставка элемента в двоичное дерево поиска занимает O (logn) времени, поэтому для n элементов это O (nlogn).

Пространственная сложность сортировки дерева равна O (n), так как нам нужно создать n узлов для n элементов.

