**УДК 004.021**

**АНАЛИЗ АЛГОРИТМОВ СО СРЕДНЕЙ АССИМПТОТИКОЙ NLOGN**

***А.А. Файтельсон***

*Бакалавр первого года обучения по направлению подготовки «Математическое обеспечение и администрирование информационных систем»*

*Курский государственный университет*

*e-mail: z0tedd@gmail.com*

*Научный руководитель:*

***В.А. Кудинов***

*Доктор педагогических наук, профессор, профессор кафедры программного обеспечения и администрирования информационных систем.*

*Курский государственный университет*

*Статья посвящена рассмотрению различных алгоритмов сортировки, а также проблеме выбора подходящего из них. Эффективность каждого алгоритма экспериментальна подтверждена с помощью уже разработанных программ для отслеживания процессорного времени выполнения.*

***Ключевые слова:*** *алгоритм сортировки, сравнение алгоритмов сортировки.*

**Введение**. В современном мире важностью алгоритмов обработки данных нельзя недооценивать. Данная область требует постоянного совершенствования методов анализа взаимодействия с информацией. С растущим каждую секунду объемом цифровых объектов задача выбора необходимого алгоритма становится все более актуальной. И именно по этой причине в данной работе будут рассмотрены алгоритмы со средним асимптотическим временем nlogn.

Прежде, чем углубиться в сами алгоритмы, стоит упомянуть, что такое асимптотическое время. Асимптотическое время - это оценка временной сложности алгоритма, которая указывает на поведение алгоритма при стремлении размера входных данных к бесконечности. Это позволяет оценить, насколько быстро будет работать алгоритм при увеличении объема данных. В научных статьях асимптотическое время часто используется для сравнения эффективности различных алгоритмов и определения их временной сложности. Временной сложностью обычно называют количество элементарных операций, совершенными алгоритмом. Время одной такой операции есть некоторая константа, в нотации «О» большое, время оценивается как О(1). В данной нотации учитывается только слагаемое самого высокого порядка, поэтому в независимости от самого значения константы, она считается, как 1. Время работы алгоритма отличается от самих входных данных, поэтому чаще всего используется время работы в худшем и среднем случае.

Алгоритмы сортировки со среднем временем О(nlogn) опираются либо на рекурсивный метод «разделяй и властвуй», либо на нестандартные структуры данных в виде кучи или двоичного дерева поиска. Перед рассмотрением алгоритмов необходимо разобраться с этими понятиями.

Метод «Разделяй и властвуй» заключается в разделении одной большой задачи на маленькие ее части, нахождения решения для каждой подзадачи и в конце объединение полученных результатов в финальное решение изначальной задачи.[1] Фактически сам метод можно представить в виде двух шагов[2]:

1. Определение простейшего случая(базового)
2. Дробление исходной задачи до тех пор, пока она не будет сведена до базового случая

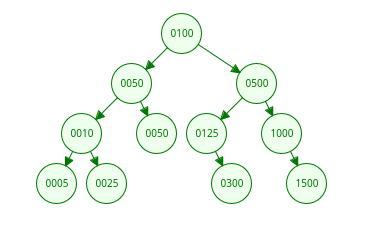
На основе данного метода основаны алгоритмы быстрой сортировки(quick sort) и сортировки слияния(merge sort).

Другие две сортировки, рассматриваемые в этой статье, опираются на определенную структуру данных – деревья. Один из алгоритмов основан на вставку элементов в двоичное дерево поиска, а другой на представлении исходного массива в виде кучи.

Дерево – структура, в которой у каждого узла(элементарной единицы структуры) есть 0 или более подузлом(«потомков»).[3]

Двоичное дерево поиска – дерево, для которого выполняется ряд условий:

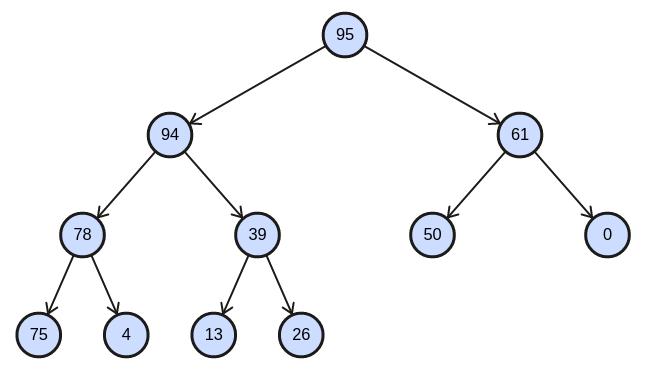
1. Каждый узел имеет не более двух потомков.
2. Любое поддерево любого узла является двоичным деревом.
3. Значения узла больше или равно значению подузла левого поддерева.
4. Значения узла меньше или равно значению подузла правого поддерева.



**Рис.1.** - Пример двоичного дерева поиска[4]

Куча – дерево, в котором значение любого узла больше или равно значению у его потомков. Конкретно в алгоритме пирамидальной сортировки используется двоичная куча(сортирующее дерево), поэтому к основному условию добавляются еще два:

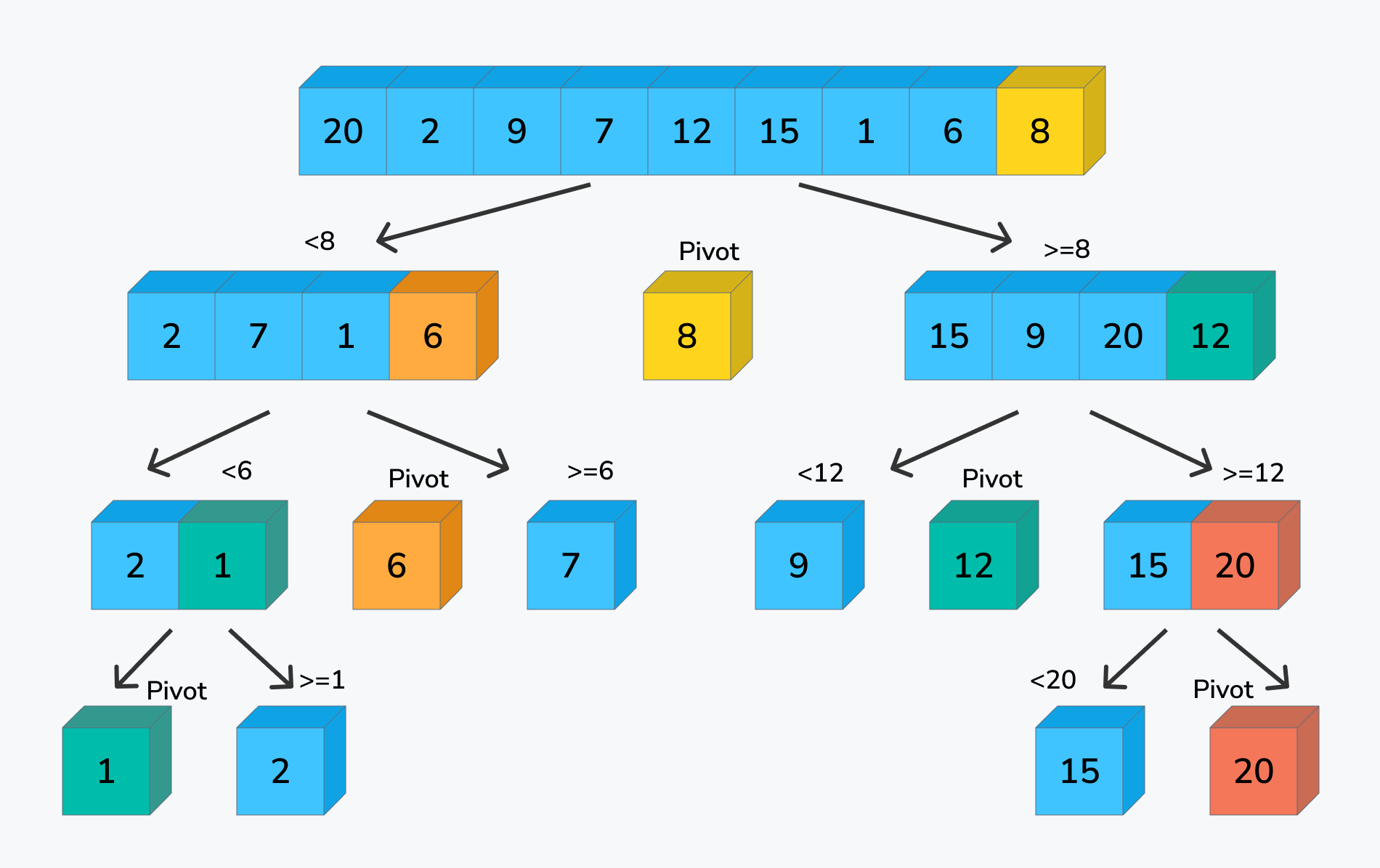
1. Каждый узел имеет не более двух потомков.
2. Слои заполняются последовательно сверху вниз и слева направо, без пробелов. [5]



**Рис.2.** – Пример двоичной кучи[6]

Данные структуры данных крайне эффективны из-за своих свойств, что и делает алгоритмы сортировки, основанные на них крайне быстрыми.

**Быстрая сортировка.** Алгоритм начинается с выбора опорного элемент, относительно которого массив делится на две части: больше опорного и меньше его. Эффективное деление массива реализует схема Хоара, она использует два указателя, которые идут с противоположных частей структуры. Данные указатели двигаются в сторону друг друга, пока не встретится пара элементов, где один из них меньше опорного и располагается после него, а второй располагается после и при этом больше опорного. После разделения массива на две половинки, алгоритм применяется заново для каждой из частей, пока мы не получим полностью отсортированный массив.

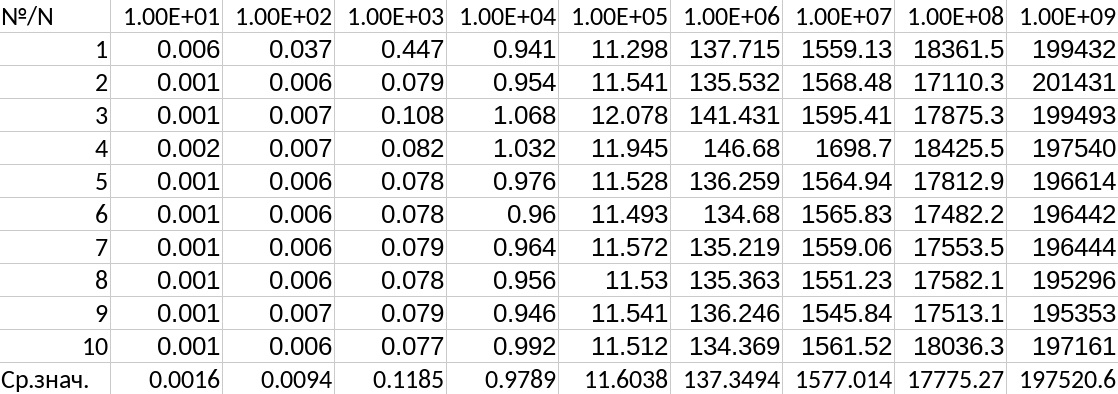


**Рис.3.** – Визуализация алгоритма быстрой сортировки[7]

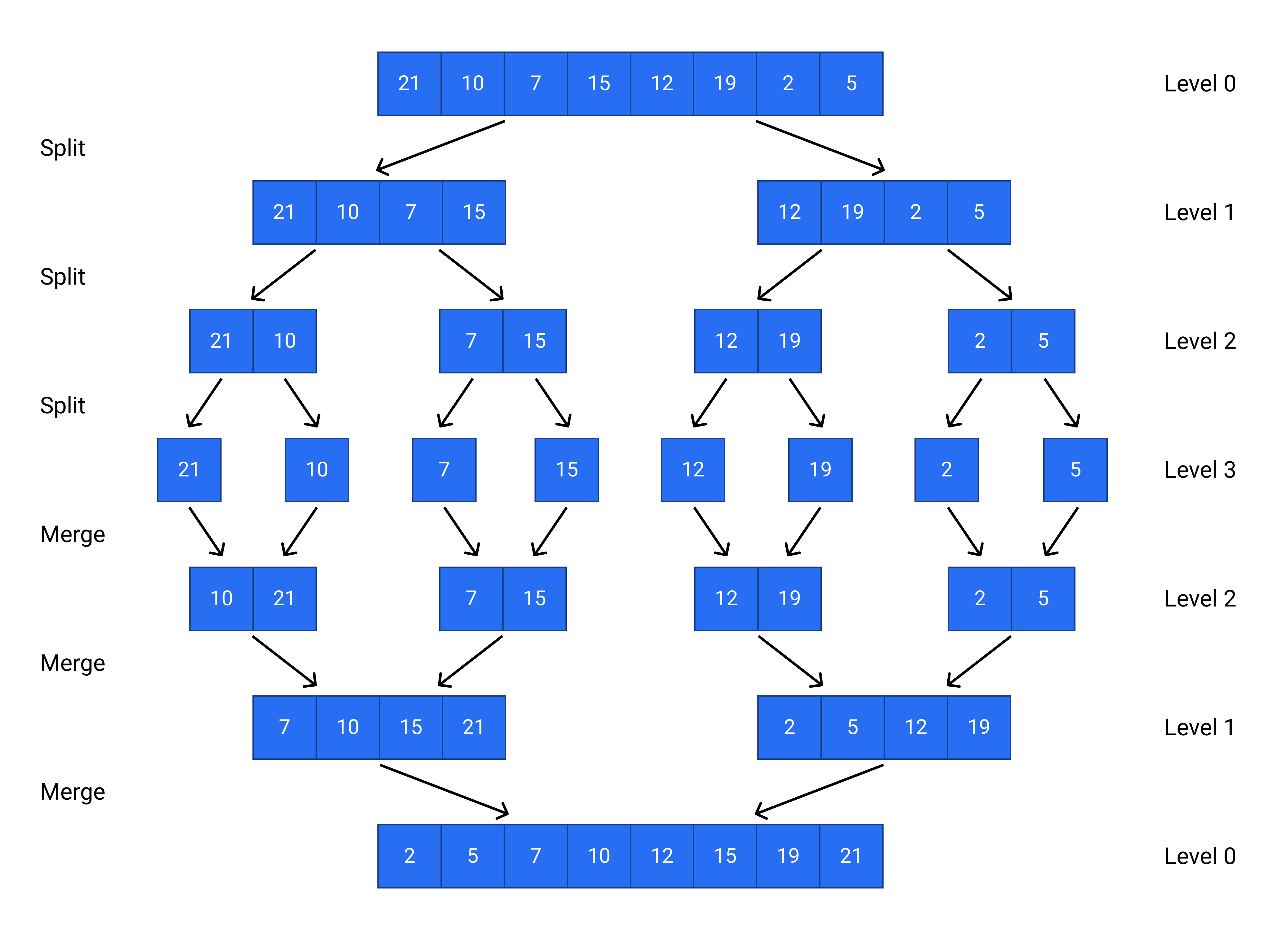
Характеристика алгоритма:

1. Асимптотическая оценка времени
   * Время в худшем случае – O()
   * Время в лучшем случае – O(nlogn)
   * Время в среднем случае – O(nlogn)
2. Оценка потребления памяти
   * Память в худшем случае – O(n)
   * Память в лучшем случае – O(logn)
   * Память в среднем случае – O(logn)
3. Реальная оценка времени

*Таблица 1 –* время работы алгоритма в сек. на переменных данных



**Сортировка слиянием.** Данный алгоритм также основан на парадигме «разделяй и властвуй», как и быстрая сортировка. Массив рекурсивно делится на две части, которые также делятся на две части, пока мы не придем к массивам длиной 1 или 2. Далее выполняется слияние полученных массивов в другой. Берутся два указателя на начала двух подмассивов, из этих указателей берется с минимальным элементом, и вставляется в новый массив, а указатель сдвигается вправо. Операция происходит до того момента, пока все подмассивы не будут слиты воедино. Само слияние выполняется за O(n), но так как мы имеем О(logn) подмассивов, то сама сортировка выполняется за О(nlogn).

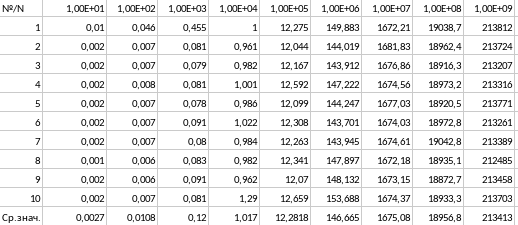


**Рис.4.** – Визуализация алгоритма сортировки слиянием[7]

Характеристика алгоритма:

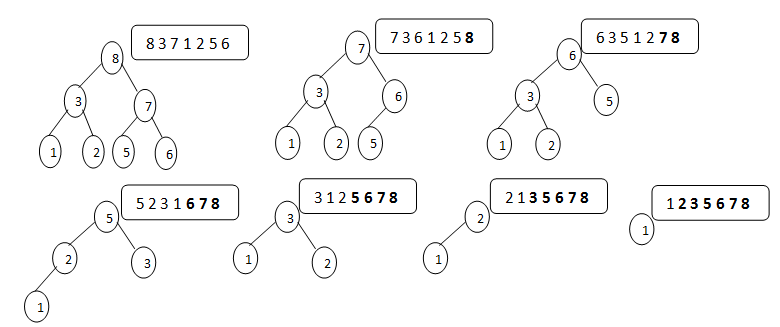
1. Асимптотическая оценка времени
   * Время в худшем случае – O(nlogn)
   * Время в лучшем случае – O(nlogn)
   * Время в среднем случае – O(nlogn)
2. Оценка потребления памяти
   * Память в худшем случае – O(n)
   * Память в лучшем случае – O(n)
   * Память в среднем случае – O(n)
3. Реальная оценка времени

*Таблица 2 –* время работы алгоритма в сек. на переменных данных



**Сортировка кучей**. В начале алгоритма из массива за О(n) строится куча.

Так как в корне кучи лежит максимальный(минимальный) элемент, то извлекая его мы можем построить новый отсортированный массив. При ивзлечении элемента мы «портим» кучу, поэтому, чтобы вернуть свойства данной структуры мы должны ее «просеить», делается это за O(logn). Последовательно извлекая и просеивая кучу из n элементов, получаем время О(nlogn).

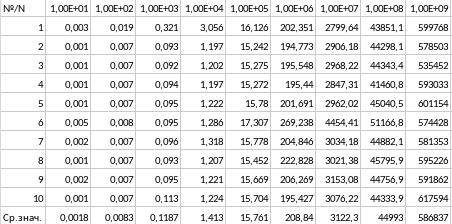


**Рис.5.** – Визуализация алгоритма сортировки кучей[8]

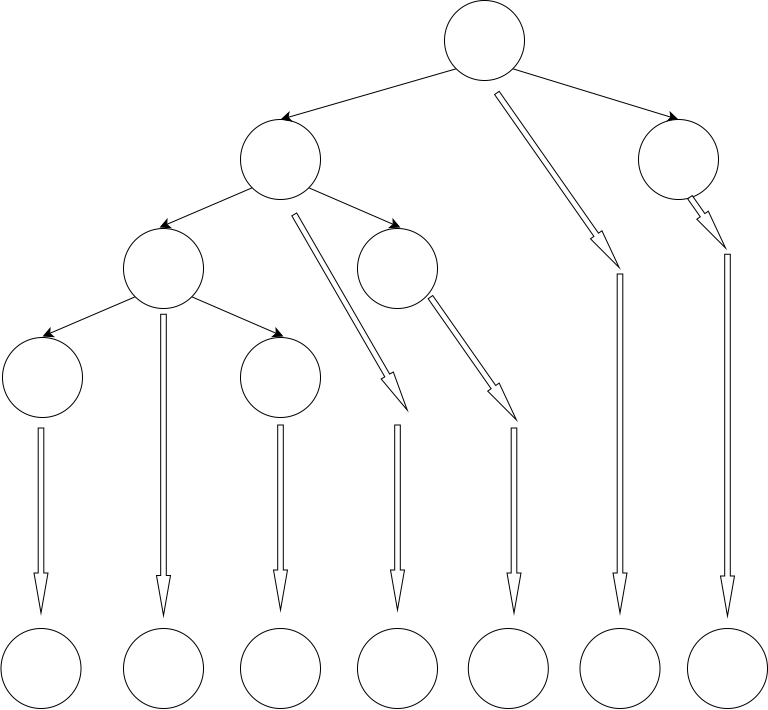
Характеристика алгоритма:

1. Асимптотическая оценка времени
   * Время в худшем случае – O(nlogn)
   * Время в лучшем случае – O(nlogn)
   * Время в среднем случае – O(nlogn)
2. Оценка потребления памяти
   * Память в худшем случае – O(1)
   * Память в лучшем случае – O(1)
   * Память в среднем случае – O(1)
3. Реальная оценка времени

*Таблица 3 –* время работы алгоритма в сек. на переменных данных



**Сортировка двоичным деревом поиска.** Данный алгоритм строит двоичное дерево поиска из элементов массива, а после этого обходит эту структуру и строит результирующий массив.

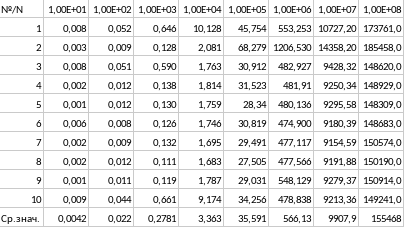


**Рис.6.** – Визуализация алгоритма сортировки деревом

Характеристика алгоритма:

1. Асимптотическая оценка времени
   * Время в худшем случае – O()
   * Время в лучшем случае – O(nlogn)
   * Время в среднем случае – O(nlogn)
2. Оценка потребления памяти
   * Память в худшем случае – O(n)
   * Память в лучшем случае – O(n)
   * Память в среднем случае – O(n)
3. Реальная оценка времени

*Таблица 4* – время работы алгоритма в сек. на переменных данных



**Заключение.** Из представленных таблиц можно сделать вывод, что не все сортировки с ассимптотически одинаковым временем имеют одинаковую производительности. Все сортировки, кроме сортировки двоичным деревом поиска эффективно показывают себя на больших размерах массива. На основе полученных данных составим 2 рейтинга алгоритмов по скорости выполнения на массиве из 100.000.000 чисел и по использованию памяти.

Рейтинг из рассмотренных алгоритмов по скорости:

1. Быстрая сортировка
2. Сортировка слиянием
3. Сортировка кучей
4. Сортировка двоичным деревом поиска

Рейтинг из рассмотренных алгоритмов по памяти:

1. Сортировка кучей
2. Быстрая сортировка
3. Сортировка слиянием, сортировка двоичным деревом поиска

Таким образом, можно сделать о вывод о необходимости выбора правильного алгоритма сортировки в зависимости от ограничений к разрабатываемой программы.

***Список используемой литературы***

[1]https://education.yandex.ru/handbook/algorithms/article/razdelyaj-i-vlastvuj

[2]Грокаем алгоритмы стр 88

[3]https://tproger.ru/translations/binary-search-tree-for-beginners

[4]https://www.cs.usfca.edu/~galles/visualization/BST.htm

[5]https://ru.algorithmica.org/cs/basic-structures/heap/

[6] <http://btv.melezinek.cz/binary-heap.html>

[7]https://workat.tech/problem-solving/tutorial/sorting-algorithms-quick-sort-merge-sort-dsa-tutorials-6j3h98lk6j2w

[8] https://www.programming-books.io/essential/algorithms/heap-sort-basic-information-3193e2927dbe4c03bcbc5645fa66cf21