Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Курсовая работа**

**по курсу «Языки и методы программирования»**

**II семестр**

**Задание №9**

**«Сортировка и поиск»**

|  |  |
| --- | --- |
| **Студент:** | Ватулин Валентин |
| **Группа:** | М8О-106Б-19 |
| **Преподаватель:** | Дубинин Алексей |
| **Оценка:** |  |
| **Дата:** |  |

Москва, 2020

**Содержание**

[Введение 3](#_Toc43690095)

[Сортировка Шелла 4](#_Toc43690096)

[Описание 4](#_Toc43690097)

[Принцип 4](#_Toc43690098)

[Время работы 4](#_Toc43690099)

[Реализация на СИ 5](#_Toc43690100)

[Бинарный поиск 7](#_Toc43690101)

[Заключение 8](#_Toc43690102)

[Список литературы 9](#_Toc43690103)

# Введение

Сортировки - один из самых известных и необходимых программисту разделов алгоритмов. Различные сортировки подходят для разных задач и условий, некоторые лучше работают на больших данных, некоторые на почти отсортированных. Зачем нужны сортировки? Ответ прост - для удобства работы с данными. С упорядоченной коллекцией работать гораздо проще, а некоторые алгоритмы, как бинарный поиск, работают только на отсортированных структурах данных. Но недостаточно просто отсортировать коллекцию. Нужно сделать это быстро и эффективно. Отсюда и вытекают основные характеристики сортировки - сложность по времени и по памяти.

# Пирамидальная сортировка

Max heap – такое бинарное дерево, что

1. Каждый лист имеет глубину либо *d*, либо *d-1*, *d* — максимальная глубина дерева.
2. Значение в любой вершине не меньше значения её потомков.

Общая идея пирамидальной сортировки заключается в том, что сначала строится max heap из элементов исходного массива, а затем осуществляется сортировка элементов.

Выполнение алгоритма разбивается на два этапа.

1. Выстраиваем элементы массива в виде сортирующего дерева

*Array[i] >= Array[i+1]*

*Array[i] >= Array[i+2]*

при 0 <= *i* <= *n*/2

Этот шаг требует *O(n)* операций.

1. Будем удалять элементы из корня по одному за раз и перестраивать дерево. То есть на первом шаге обмениваем *Array[0]* и *Array[n-1]*, преобразовываем *Array[0]*, *Array[1]*, … , *Array[n-2]* в сортирующее дерево. Затем переставляем *Array[0]* и *Array[n-2]*, преобразовываем *Array[0]*, *Array[1]*, … , *Array[n-3]* в сортирующее дерево. Процесс продолжается до тех пор, пока в сортирующем дереве не останется один элемент. Тогда *Array[0]*, *Array[1]*, … , *Array[n-1]* — упорядоченная последовательность.

Этот шаг требует *O(n logn)* операций.

Суммарно алгоритм требует *O(n) + O(n logn)* операций, что сводится к *O(n logn)*.

## Реализация на СИ

Нам понадобятся 3 функции для реализации пирамидальной сортировки:

void sift\_down(arr, n, i) – вычисляет наибольший элементы среди элемента I и его потомком и переставляет их местами, чтобы наибольший элемент оказался сверху.

void build\_heap(arr, n) – строит max heap из массива, является оберткой над sift\_down.

void heapsort(arr, n) – производит сортировку, сначала вызывая build\_head.

# Бинарный поиск

Двоичный (бинарный) поиск (также известен как метод деления пополам или дихотомия) — классический алгоритм поиска элемента в отсортированном массиве (векторе), использующий дробление массива на половины. Используется в информатике, вычислительной математике и математическом программировании. Алгоритм:

1. Определение значения элемента в середине структуры данных. Полученное значение сравнивается с ключом.

2. Если ключ меньше значения середины, то поиск осуществляется в первой половине элементов, иначе — во второй.

3. Поиск сводится к тому, что вновь определяется значение серединного элемента в выбранной половине и сравнивается с ключом.

4. Процесс продолжается до тех пор, пока не будет найден элемент со значением ключа или не станет пустым интервал для поиска.

# Заключение

Сортировки является одним из самых необходимых и важных алгоритмов в программировании. Для их эффективного использования необходимо знать контекст применения каждой из сортировок, их особенности, затраты и способы модификации. Знание бинарного поиска позволит быстро и эффективно искать необходимые значения в отсортированной коллекции.

# Список литературы

1. <http://algolist.ru/sort/shell_sort.php>
2. <https://ru.wikipedia.org/wiki/Двоичный_поиск>
3. Д. Хайнеман.​“Алгоритмы справочник”.
4. Б. Керниган, Д. Ритчи. “Язык программирования Си”