

	<p>Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)</p>
---	--

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по рубежному контролю № 1

По курсу "Анализ Алгоритмов"

Сравнение алгоритмов полного перебора и бинарного поиска в массиве

Студент:

Турсунов Жасурбек Рустамович

Группа: ИУ7-56Б

Преподаватели:

Волкова Лилия Леонидовна

Строганов Юрий Владимирович

Москва, 2020 г.

Содержание

1	Постановка задачи	2
2	Решение	2
3	Листинг кода	3
4	Тестирование функций	3
5	Примеры работы	4
6	Анализ полученных данных	5
	Вывод	6

1 Постановка задачи

Изучить алгоритмы полного перебора и бинарного поиска элемента в массиве. Провести анализ на полученных данных и определить самый быстрый алгоритм. Размер массива будет варьироваться от 1000 до 10000. Для алгоритма бинарного поиска необходимо подготовить данные, а именно отсортировать их заранее. Для сортировки будет использована внутренняя функция Python - `sort()`. Замер времени будет произведен функцией из библиотеки `time - perf-counter`.

2 Решение

Алгоритм полного перебора не обладает специфичными действиями. Этот алгоритм поэлементно проходится по элементам массива до тех пор, пока не найдет нужный элемент. В случае пустого массива или поиска несуществующего элемента, алгоритм вернет значение `None`.

Алгоритм двоичного поиска применяется к заранее упорядоченному словарю. Процесс двоичного поиска можно описать при помощи шагов:

1. получить значение по индексу, находящееся в середине массива, и сравнить его с данным;
2. в случае, если значение меньше (в контексте типа данных) данного, продолжить поиск в левой части словаря, в обратном случае – в правой;
3. на новом интервале получить значение по индексу из середины этого интервала и сравнить его с данным;
4. продолжать поиск до тех пор, пока найденное значение по индексу не будет равно данному.

Поиск в массиве с использованием данного алгоритма в худшем случае будет иметь трудоемкость $O(\log_2 N)$, что быстрее поиска при помощи алгоритма полного перебора. Но стоит учитывать тот факт, что данный алгоритм работает только для заранее упорядоченного массива. В случае большого объема данных и обратного порядка сортировки может произойти так, что алгоритм полного перебора будет эффективнее по времени, чем алгоритм двоичного поиска.

3 Листинг кода

На листингах 1, 2 представлена реализация алгоритмов поиска в словаре.

```
1 def full_iteration(data, val):
2     if len(emails) == 0:
3         return None
4     for i in range(len(data)):
5         if data[i] == val:
6             return i+1
7     return None
8
```

Листинг 1: Алгоритм полного перебора

```
1 def bin_search(data, val):
2     result = None
3     if len(data) == 0:
4         return result
5     mid, low, high = len(data) // 2, 0, len(data) - 1
6     while data[mid] != val and low <= high:
7         if val > data[mid]:
8             low = mid + 1
9         else:
10            high = mid - 1
11            mid = (low + high) // 2
12    if low < high:
13        result = mid
14    return result
```

Листинг 2: Алгоритм бинарного поиска

4 Тестирование функций

На рисунке 1 приведены результаты функциональных тестов для функций, реализующих алгоритмы поиска в массиве.

```

Начало тестирования....
Результаты проверки алгоритма полного перебора:
-----

1) Входной параметр: не пустой массив и Tina Henson
Ожидаемый результат: 730. Полученный результат: 730
-----

2) Входной параметр: пустой массив и Tina Henson
Ожидаемый результат: None. Полученный результат: None
-----

3) Входной параметр: не пустой массив и пустое значение
Ожидаемый результат: None. Полученный результат: None

Проверка алгоритма двоичного поиска:
-----

1) Входной параметр: не пустой массив и Tina Henson
Ожидаемый результат: 948. Полученный результат: 948
-----

2) Входной параметр: пустой массив и Tina Henson
Warning! Empty array.
Ожидаемый результат: None. Полученный результат: None
-----

3) Входной параметр: не пустой массив и пустое значение
Ожидаемый результат: None. Полученный результат: None

```

Рис. 1: Результаты функциональных тестов.

Все тесты прошли успешно.

5 Примеры работы

На рисунке 2 показан пример работы программы. На вход подается массив значений и значение которого требуется найти. На выходе получаем индекс элемента.

```
Value: Robert Rodriguez || Result: 806
Value: James Huang || Result: 745
Value: Judy Medina || Result: 1517
Value: Nicholas Lawrence || Result: 2980
Value: Joseph Ramirez || Result: 2370
Value: Patrick Williams || Result: 4568
Value: Alexis White || Result: 142
Value: Bridget Farley || Result: 1057
Value: Timothy Erickson || Result: 8496
```

Рис. 2: Пример работы программы.

6 Анализ полученных данных

На рисунке 3 показаны результаты замера времени для алгоритма полного перебора и бинарного поиска.



Рис. 3: Результаты проведенного анализа.

Из полученного графика можно сделать вывод, о том что бинарный поиск в разы быстрее алгоритма полного перебора. При замере времени для бинарного поиска, не учитывалось время сортировки, которое может быть довольно длительным.

Вывод

В ходе работы был реализован алгоритм полного перебора и бинарного поиска в массиве. Также проведен анализ замера времени этих алгоритмов.