Министерство науки и высшего образования 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)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
По курсу: "Анализ алгоритмов"

Студент Сукочева Алис
Группа ИУ7-53Б
Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7
Тема Поиск в словаре.

Студент:	_____	Сукочева А.
	подпись, дата	Фамилия, И.О.
Преподаватель:	_____	Волкова Л.Л.
	подпись, дата	Фамилия, И. О.
Преподаватель:	_____	Строганов Ю.В.
	подпись, дата	Фамилия, И. О.

Москва — 2020 г.

Оглавление

Введение

С появлением словарей появилась нужда в том, чтобы уметь быстро находить нужное значение по ключу. Со временем стали разрабатывать алгоритмы поиска в словаре. В данной лабораторной работе мы рассмотрим три алгоритма:

1. Поиск полным перебором;
2. Бинарный поиск;
3. Частичный анализ.

Целью данной работы является изучение и реализация трех алгоритмов.

В рамках выполнения работы необходимо решить следующие задачи:

1. Изучения трех алгоритмов поиска в словаре;
2. Применение изученных основ для реализации поиска значений в словаре по ключу;
3. Получения практических навыков;
4. Получение замеров времени;
5. Описание и обоснование полученных результатов;
6. Выбор и обоснование языка программирования, для решения данной задачи.

1 | Аналитическая часть

1.1 Описание словаря

В данной лабораторной работе использовался словарь VR-игр. Ключами в данном словаре были названия игр. Значениями были объекты, которые имели характеристики данной игры, а именно:

- Издатель;
- Год издания;
- Жанр;
- Возрастное ограничение;
- Цена.

1.2 Теоретические данные

Словари – объекты, записанные парой "ключ-значение". Отличным примером словаря является толковый словарь. В данном словаре ключи – это нужное нам слово, а значения – это значение искомого слова. Ключи в словаре должны быть уникальными.

Поиск необходимой информации в словаре – одна из фундаментальных задач программирования.

1.3 Поиск полным перебором

Поиск полным перебором может производиться в неотсортированном словаре. Весь алгоритм сводится к последовательному прохождению по

всему словарю и сравнению всех значений с ключом. Данный алгоритм имеет N + 1 случаев и требует N сравнений для худшего случая. Это простейший из алгоритмов поиска. Этот алгоритм не очень эффективен, однако он работает на произвольном списке.

1.4 Бинарный поиск

Бинарный поиск базируется на том, что словарь изначально отсортирован, что позволяет сравнивать ключ с средним элементом, и, если, он меньше, то продолжать искать в левой части, таким же методом, иначе в правой.

На рис. 1.1 показан пример бинарного поиска.

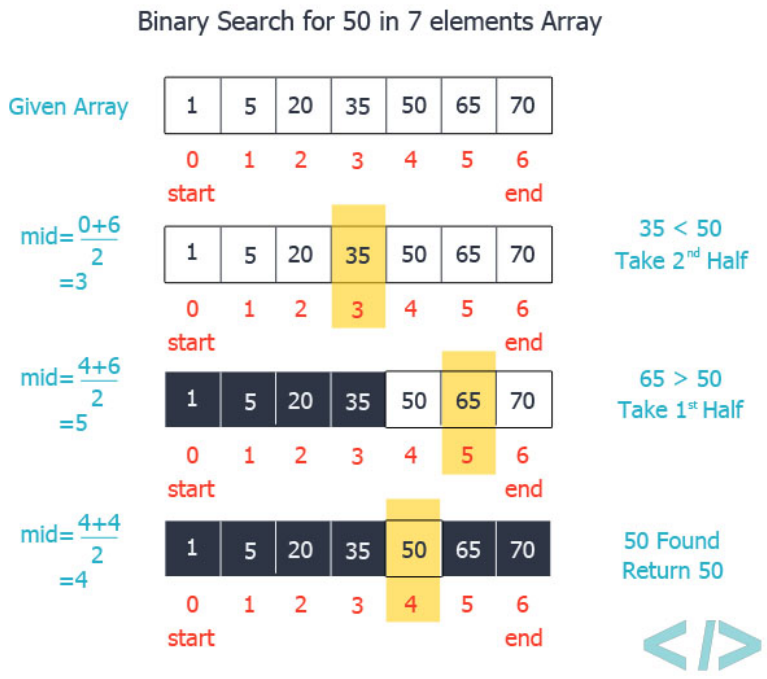

Рис. 1.1: Бинарный поиск

1.5 Частичный анализ

Идея частичного анализа состоит в том, изначальный словарь отсортирован по частоте первого символа. Также словарь имеет ключи равные буквам алфавита, а значениями являются словари с исходными данными, которые имеют первую букву схожую с ключом словаря. На деле получается, что это простой толковый словарь, который имеет в заголовке букву, а на странице все слова, которые начинаются на написанную в заголовке букву. Поиск изначально осуществляется по первой букве ключа. Далее, когда найден словарь, в котором нужно искать искомое слово производится поиск полным перебором.

1.6 Вывод

В данном разделе были рассмотрены основополагающие материалы, которые в дальнейшем потребуются при реализации алгоритмов поиска в словаре.

2 Конструкторская часть

2.1 Разработка алгоритмов

В данном разделе будут рассмотрены схемы.

На рис. 2.1 показана схема алгоритма поиска полным перебором.

На рис. 2.2 показана схема алгоритма бинарного поиска.

На рис. 2.3 показана схема алгоритма частичного анализа.

2.2 Вывод

В данном разделе были рассмотрены схемы алгоритмов.

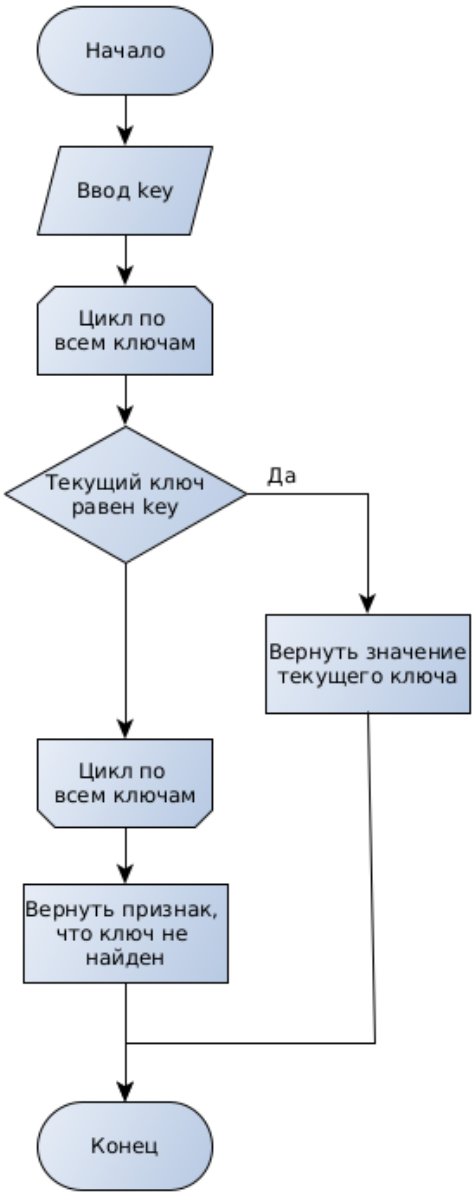

Рис. 2.1: Поиск полным перебором

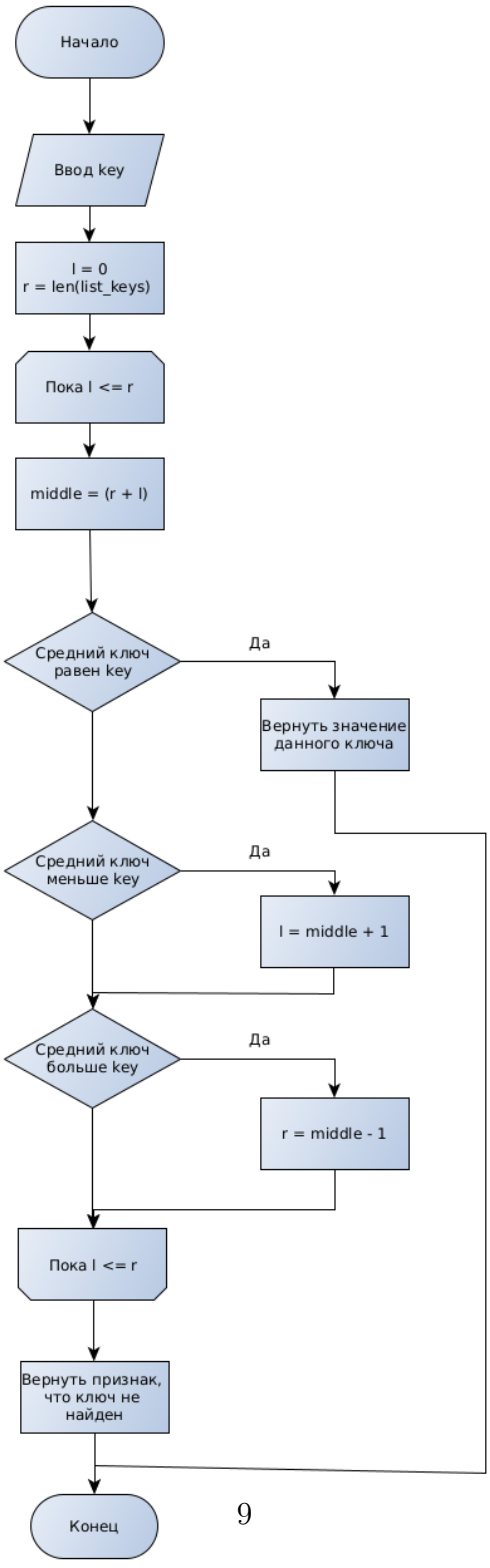

Рис. 2.2: Бинарный поиск

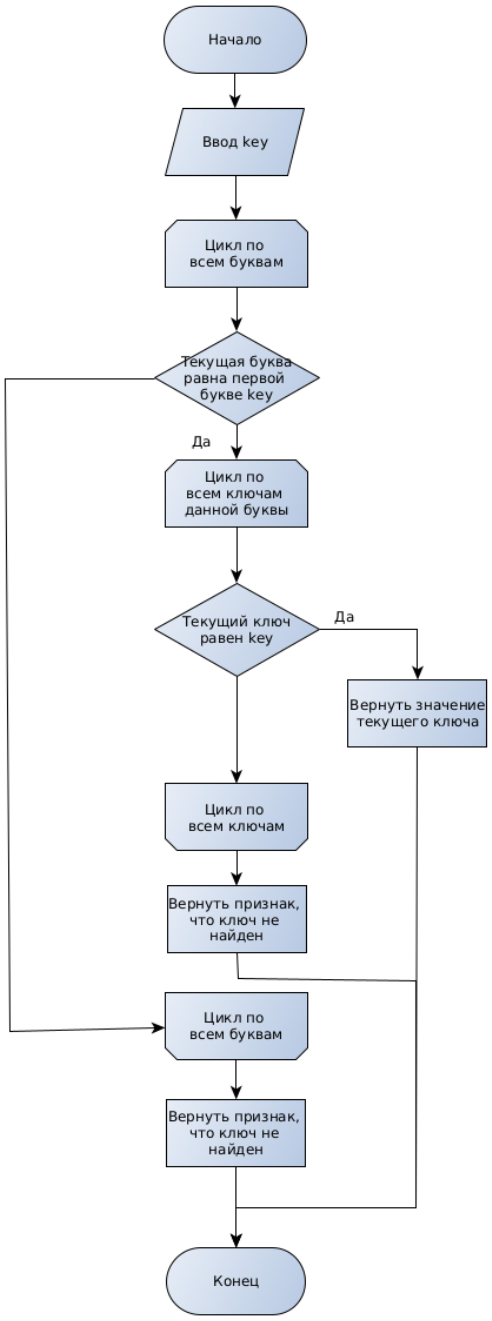

Рис. 2.3: Частичный анализ

3 | Технологическая часть

3.1 Выбор ЯП

В данной лабораторной работе использовался язык программирования - python. [4] Я знакома с ним. Поэтому данный язык был выбран. В качестве среды разработки я использовала Visual Studio Code [1]. Visual Studio Code подходит не только для Windows [2], но и для Linux [3], это причина, по которой я выбрала VS code, т.к. у меня установлена ОС Ubuntu 18.04.4 [5]. В моей архитектуре присутствует 8 ядер.

3.2 Сведения о модулях программы

Данная программа разбита на модули.

- main.py - главный файл, содержащий точку входа в программу.
- dictionary.py - файл, содержащий класс dictionary.
- world_info.py - файл, содержащий класс, который описывает информацию о игре.

На листингах 3.1-3.4 представлен основной код программы.

Листинг 3.1: Поиск полным перебором

```
1 def SequentialSearch(self, key):
2 for elem in self.data:
3   if key == elem:
4     return self.data[elem]
5 return -1
```

Листинг 3.2: Бинарный поиск

```
def BinarySearch(self, key, list_keys):
  l, r = 0, len(list_keys) - 1

  while l <= r:
    middle = (l + (r - l) // 2)

    if list_keys[middle] == key:
      return self.data[elem]

    elif list_keys[middle] < key:
      l = middle + 1

    else:
      r = middle - 1

  return -1
```

Листинг 3.3: Оптимизация бинарного поиска

```
def BinarySearch(self, key, list_keys, l, r):
  while l <= r:
    middle = (r + l) // 2 # Optimization.
    elem = list_keys[middle] # Optimization.

    if elem == key:
      return self.data[elem]

    elif elem < key:
      l = middle + 1

    else:
      r = middle - 1

  return -1
```

Листинг 3.4: Частичный анализ

```
def Search(self, key, new_dict):
  # Loop by letter
  for k in new_dict:
```

```
4       if key[0] == k:
5         for elem in new_dict[k]:
6           if elem == key:
7             return new_dict[k][elem]
8         return -1
9
10    return -1
```

3.3 Тестирование

В данном разделе будет приведена таблица с тестами (таблица 3.1).

Таблица 3.1: Таблица тестов

Входные данные	Пояснение	Результат
A Good Librarian Like a Good Shepherd	Первый элемент	Ответ верный
Championship Manager 2008	Средний элемент	Ответ верный
Patapon 3	Последний элемент	Ответ верный
Adrift	Произвольный элемент	Ответ верный
Adrift 333	Несуществующий элемент	Ответ верный

Все тесты пройдены.

3.4 Вывод

В данном разделе были разобраны листинги рис 3.1-3.4, показывающие работу каждого алгоритма и приведена таблица с тестами (таблица 3.1)..

4 | Экспериментальная часть

В данном разделе будет произведено сравнение трех алгоритмов.

4.1 Временные характеристики

Так как поиск в словаре считается короткой задачей, воспользуемся усреднением массового эксперимента. Для этого сложим результат работы алгоритма n раз (n >= 10), после чего поделим на n. Тем самым получим достаточно точные характеристики времени. Сравнение произведем при n = 1000.

На рис. 4.1 приведено сравнение времени выполнения трех алгоритмов для поиска последнего значения. По результатам эксперимента видно, что поиск полным перебором затрачивает больше всего времени, т.к. он последовательно обходит все элементы, в то время, как остальные два алгоритма выполняют поиск значительно быстрее.

На рис. 4.2 произведено аналогичное сравнение, только в качестве искомого ключа взят несуществующий. Аналогично поиск полным перебором затрачивает больше всего времени по вышеописанной причине.

Однако не всегда поиск полным перебором дает худший результат. На рис. 4.3 представлен результат поиска первого значения. Выигрыш алгоритма поиска полным перебором обосновывается тем, что он тратит лишь одно сравнение для того, чтобы найти первый ключ, в том время, когда бинарный поиск затрачивает гораздо больше сравнений. Частичный анализ работает чуть медленнее, так как ему нужно произвести дополнительное сравнение первых букв.

На рис. 4.4 представлен результат поиска произвольного ключа. Алгоритм полного перебора работает медленнее всех.

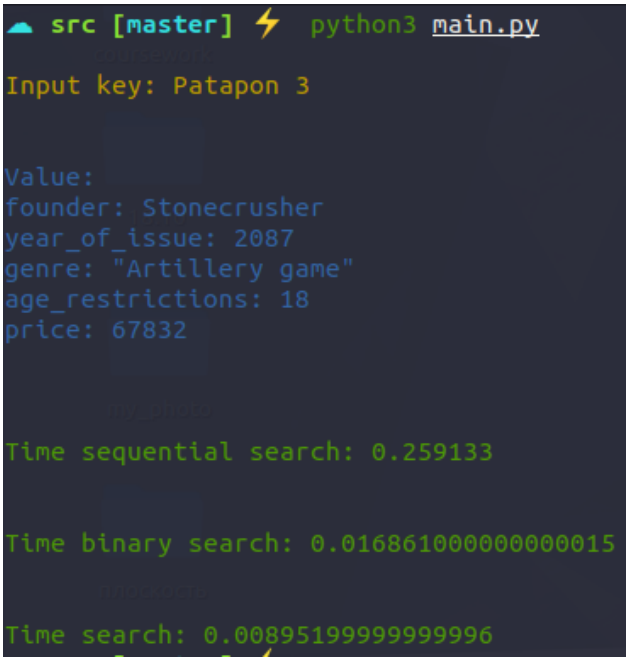

Рис. 4.1: Последний ключ

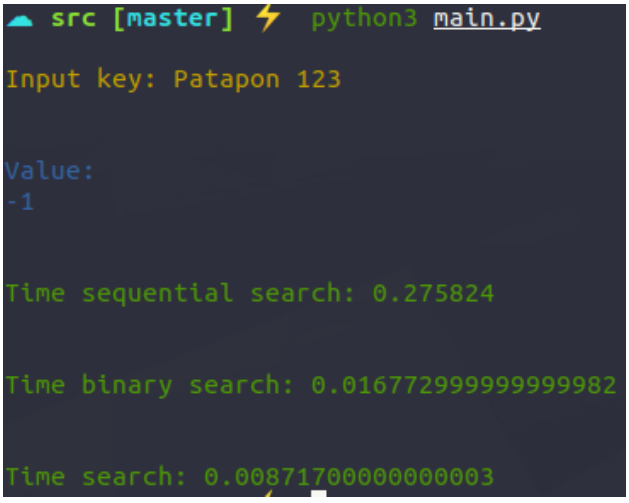

Рис. 4.2: Несуществующий ключ

4.2 Вывод

В данном разделе было произведено сравнение трех алгоритмов. По результатам исследования было доказано, что алгоритм полного перебора

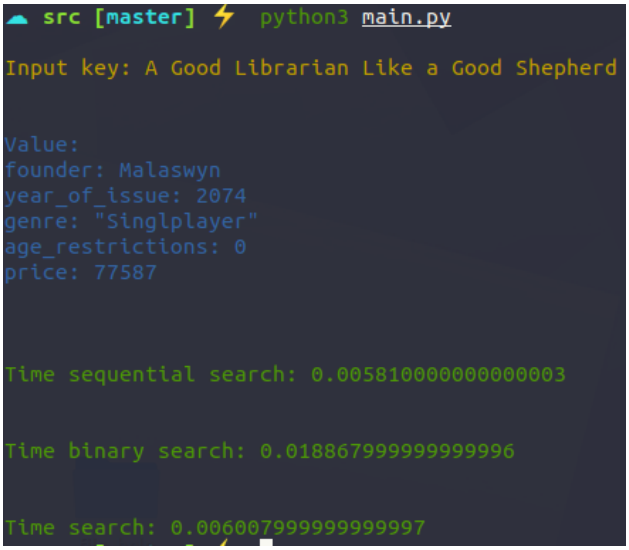

Рис. 4.3: Первый ключ

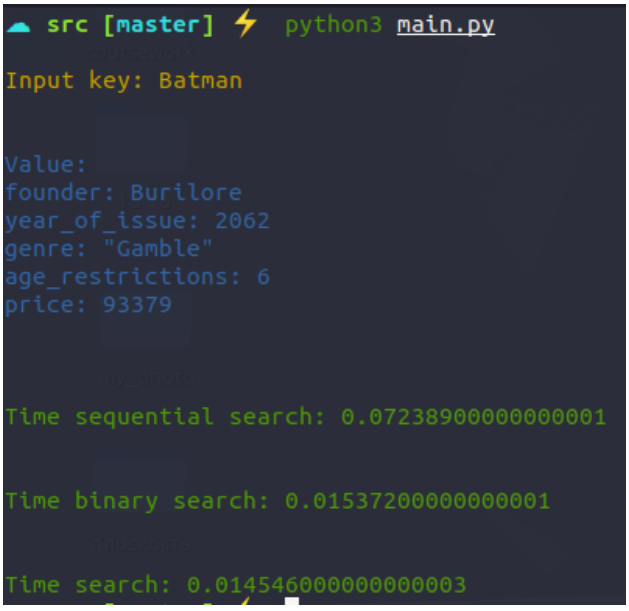

Рис. 4.4: Произвольный ключ

в основном работает медленнее всех, за исключением, когда ключ лежит достаточно близко к началу.

Заключение

В данной лабораторной работе были рассмотрены основополагающие материалы которые в дальнейшем потребовались при реализации трех алгоритмов поиска в словаре. Были рассмотрены схемы (рис. 2.1 - 2.3) показывающие работу каждого алгоритма. Также были разобраны листинги (рис 3.1-3.4), была приведена таблица с тестами (таблица 3.1). показывающая корректную работу программы. Был произведен сравнительный анализ (рис. 4.1 - 4.4) и объяснены результаты.

В рамках выполнения работы были решены следующие задачи:

1. Изучили три алгоритма поиска в словаре;
2. Применили изученные основы для реализации поиска значений в словаре по ключу;
3. Получили практические навыки;
4. Получили замеры времени;
5. Описали и обосновали полученные результаты;
6. Выбрали и обосновали язык программирования, для решения данной задачи.

Литература

[1] Visual Studio Code [Электронный ресурс], режим доступа: https://code.visualstudio.com/ (дата обращения: 02.10.2020)

[2] Windows [Электронный ресурс], режим доступа:https://www.microsoft.com/ru-ru/windows (дата обращения: 02.10.2020)

[3] Linux [Электронный ресурс], режим доступа:https://www.linux.org.ru/ (дата обращения: 02.10.2020)

[4] Руководство по языку python [Электронный ресурс], - режим доступа: https://pythonworld.ru/samouchitel-python (дата обращения: 26.11.2020)

[5] Ubuntu 18.04 [Электронный ресурс], режим доступа:https://releases.ubuntu.com/18.04/ (дата обращения: 02.10.2020)