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 №6 по курсу "Анализ алгоритмов"

Тема Поиск по словарю

Студент Кузнецова А.В.

Группа ИУ7-51Б

Оценка (баллы)

Преподаватель Волкова Л.Л.

Москва — 2022 г.

Оглавление

Введение

Словарь, как тип данных, применяется везде, где есть связь «ключ – значение» или «объект – данные». Поиск — основная задача при использовании словаря. Данная задача решается различными способами, которые дают различную скорость решения.

Цель данной работы: получить навык поиска по словарю при ограничении на значение признака, заданного при помощи лингвистической переменной.

Для достижения цели поставлены следующие задачи:

— формализовать объект и его признак;

— составить анкету для заполнения респондентом;

— провести анкетирование респондентов;

— описать структуру данных словаря;

— предложить и реализовать алгоритм поиска в словаре.

1 Аналитическая часть

В данном разделе рассмотрены словарь как структура данных и алгоритм полного перебора.

1.1 Словарь как структура данных

Словарь (или «ассоциативный массив») [1] - абстрактный тип данных (интерфейс к хранилищу данных), позволяющий хранить пары вида «(ключ, значение)» и поддерживающий операции добавления пары, а также поиска и удаления пары по ключу:

- INSERT(k, v);
- FIND(k);
- REMOVE(k).

В паре (k, v): v — значение, ассоциированное с ключом k. Семантика и названия вышеупомянутых операций в разных реализациях ассоциативного массива могут отличаться.

Операция ПОИСК(k) возвращает значение, ассоциированное с заданным ключом, или некоторый специальный объект, означающий, что значения, ассоциированного с заданным ключом, нет. Две другие операции ничего не возвращают (за исключением, возможно, информации о том, успешно ли была выполнена данная операция).

Ассоциативный массив с точки зрения интерфейса удобно рассматривать как обычный массив, в котором в качестве индексов можно использовать не только целые числа, но и значения других типов — например, строки.

1.2 Алгоритм полного перебора

Алгоритмом полного перебора [2] называют метод решения задачи, при котором по очереди рассматриваются все возможные варианты. В слу-

чае реализации алгоритма в рамках данной работы будут последовательно перебираться ключи словаря до тех пор, пока не будет найден нужный.

Трудоёмкость алгоритма зависит от того, присутствует ли искомый ключ в словаре, и, если присутствует – насколько он далеко от начала массива ключей.

Пусть на старте алгоритм затрагивает k_0 операций, а при сравнении k_1 операций.

Пусть алгоритм нашёл элемент на первом сравнении (лучший случай), тогда будет затрачено $k_0 + k_1$ операций, на втором — $k_0 + 2 \cdot k_1$, на последнем (худший случай) — $k_0 + N \cdot k_1$. Если ключа нет в массиве ключей, то мы сможем понять это, только перебрав все ключи, таким образом трудоёмкость такого случая равно трудоёмкости случая с ключом на последней позиции. Трудоёмкость в среднем может быть рассчитана как математическое ожидание по формуле (1.1), где Ω — множество всех возможных случаев.

$$\sum_{i \in \Omega} p_i \cdot f_i = k_0 + k_1 \cdot \left(1 + \frac{N}{2} - \frac{1}{N+1}\right) \tag{1.1}$$

Вывод

В данном разделе рассмотрены словарь как структура данных и алгоритм полного перебора.

2 Конструкторская часть

В этом разделе будут представлено описание используемых типов данных, а также схемы алгоритмов поиска в словаре.

2.1 Описание используемых типов данных

При реализации алгоритмов будут использованы следующие типы данных:

- словарь – встроенный тип dict [3] в Python[4] будет использован в созданном классе Dictionary;
- массив ключей – встроенный тип list [6] в Python[4];
- длина массива/словаря – целое число int.

2.2 Структура разрабатываемой программы

В данном ПО буде реализован метод структурного программирования, при этом также будет реализован класс Dictionary для работы со словарем.

Взаимодействие с пользователем будет через консоль, будет дана возможность ввода строки для поиска значений в словаре.

2.3 Схемы алгоритмов

На рисунке 2.1 представлена схема алгоритма поиска в словаре полным перебором.

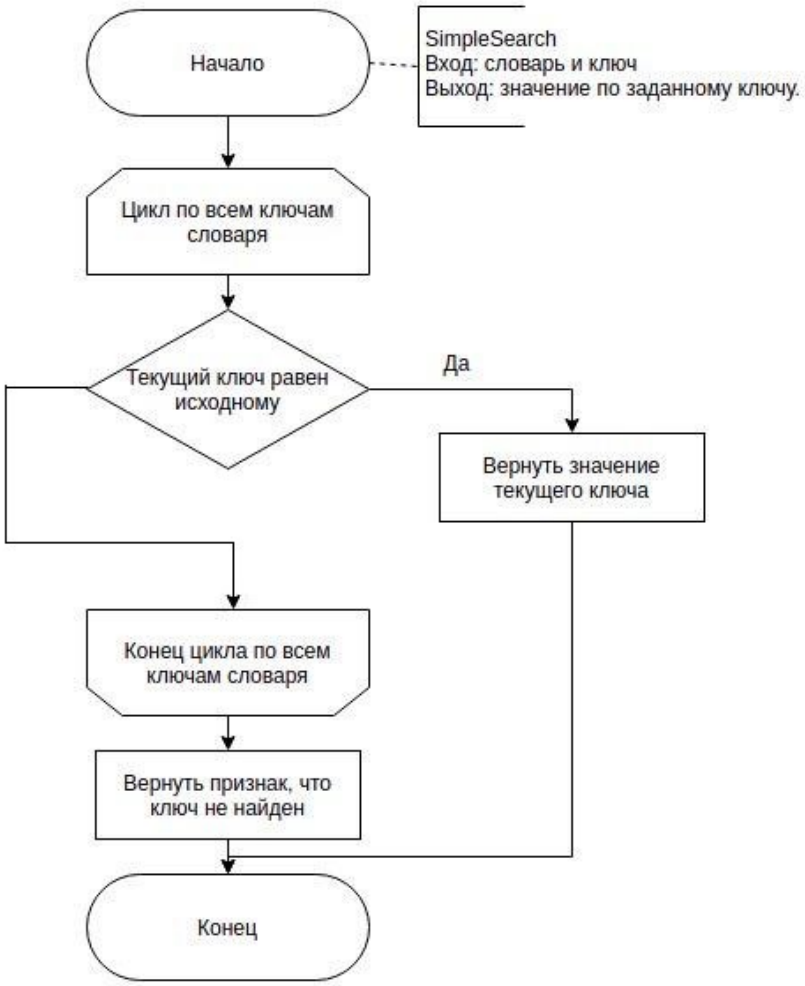

Рисунок 2.1 – Схема алгоритма поиска в словаре полным перебором

2.4 Классы эквивалентности при тестировании

Для тестирования выделены классы эквивалентности, представленные ниже.

- некорректный ввод ключа – пустая строка;
- корректный ввод, но ключа нет в словаре – вывод будет (-1), как знак того, что такого ключа нет словаре;
- корректный ввод и ключ есть в словаре – вывод верного значения.

Вывод

В данном разделе была построена схема алгоритма, рассматриваемого в лабораторной работе, были описаны классы эквивалентности для тестирования, структура программы.

3 Технологическая часть

В данном разделе будут приведены требования к программному обеспечению, средства реализации и листинги кода.

3.1 Требования к программе

К программе предъявляется ряд требований:

- на вход подаётся строка, на основании которой производится поиск;
- на выходе — результат поиска в словаре;
- программа не должна аварийно завершаться при отсутствии ключа в словаре.

3.2 Средства реализации

В качестве языка программирования для реализации данной лабораторной работы был выбран ЯП Python [4].

Данный язык достаточно удобен и гибок в использовании.

В качестве среды разработки выбор сделан в сторону Visual Studio Code [6]. Данная среда подходит как для Windows, так для Linux. и macOS

3.3 Сведения о модулях программы

Данная программа разбита на модули:

- main.py – файл, содержащий меню программы и основную функцию;
- info_for_prog.py – файл, содержащий класс, который описывает информацию о данных;
- dictionary.py – файл, содержащий класс ”Словарь”.

3.4 Реализация алгоритмов

В листинге 3.1 представлена реализация алгоритма поиска в словаре полным перебором.

Листинг 3.1 – Реализация алгоритма поиска полным перебором

```
        def Search(self, key):
            k = 0
            keys = list(self.data.keys())
            for elem in self.data:
                k += 1
                if key == elem:
                    // Log
                    self.f.write(f"{keys.index(key)},{key},{k}\n")
                    return self.data[elem]
            return -1
```

3.5 Функциональное тестирование

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

Таблица 3.1 – Таблица тестов

Входные данные	Результат
длинная река	Нил
короткая река	Москва
средняя река	Енисей
компот	Элемент не существует (-1)
8273	Элемент не существует (-1)

Все тесты пройдены успешно.

Вывод

В данном разделе был представлен листинг рассматриваемого алгоритма поиска в словаре, приведена информация о средствах реализации, сведения о модулях программы и было проведено функциональное тестирование.

4 Исследовательская часть

В данном разделе приведена постановка эксперимента.

4.1 Исследование

В данной работе, в качестве респондентов, принимали участие следующие студенты:

— Косарев А.А.;

— Никулина А.А.;

— Бурлаков И.А.;

— Баринов Н.Ю.;

— Каландадзе Д.В.

Исследуемый признак — длина рек.

Возможные значения признака (описывают длину):

— короткая;

— средняя;

— длинная;

— очень длинная;

— не очень длинная;

— не очень короткая.

По результатам опроса была сформирована и занесена в таблицу 4.1 обобщённая статистика.

Сокращения, используемые в таблице:

— К — короткая;

— С — длинная;

— Д — много;

— ОД — очень длинная;

— НОД — не очень длинная;

— НОК — не очень короткая.

Таблица 4.1 – Обобщенная статистика

Длина реки, км	К	С	Д	ОД	НОД	НОК
<1000	4	0	0	0	0	1
>1000 и <3000	1	0	0	0	3	1
>3000 и <4000	0	2	1	0	1	1
>4000 и <5000	0	1	2	0	0	2
>5000 и <6000	0	0	3	2	0	0
>6000	0	0	0	5	0	0

В данной таблице приведены количество голосов, отданных респондентами в пользу истинности разных утверждений. В узлах таблицы расположено количество голосов в пользу высказывания, формируемого по принципу «длина реки — тезис» (Например, за истинность высказывания «длина реки менее 1000 км — короткая» проголосовали четыре человека).

Таблица 4.2 содержит нормализованные значения из таблицы 4.1.

Таблица 4.2 – Нормализованные значения

Длина реки, км	М	С	МН	ОМН	НОМН	НОМ
<1000	0.8	0	0	0	0	0.2
>1000 и <3000	0.2	0	0	0	0.6	0.2
>3000 и <4000	0	0.4	0.2	0	0.2	0.2
>4000 и <5000	0	0.2	0.4	0	0	0.4
>5000 и <6000	0	0	0.6	0.4	0	0
>6000	0	0	0	1	0	0

4.2 Функции принадлежности

Построенные функции принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов, приведены на рисунке 4.1.

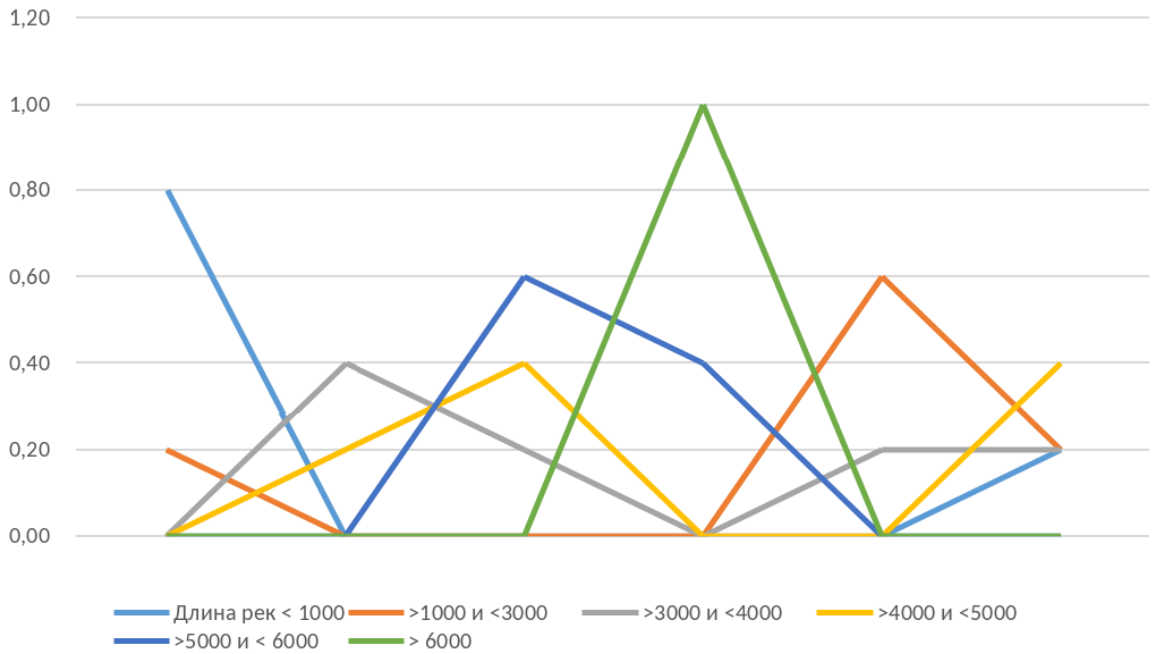

Рисунок 4.1 – Функции принадлежности

Вывод

Таким образом, на основании экспертной оценки, были точечно заданы функции принадлежности числового значения указанного признака некоторому термину. Благодаря данным функциям можно с некоторой точностью определить описанную принадлежность.

Заключение

В ходе выполнения лабораторной работы были решены следующие задачи:

— формализовать объект и его признак;

— сформировать анкету для заполнения респондентом;

— провести анкетирование респондентов;

— описать структуру данных словаря;

— предложить и реализовать алгоритм поиска в словаре.

Поставленная цель достигнута: получить навык поиска по словарю при ограничении на значение признака, заданного при помощи лингвистической переменной.

Список использованных источников

[1] National Institute of Standards and Technology [Электронный ресурс]. Режим доступа: https://xlinux.nist.gov/dads/HTML/assocarray.html (дата обращения 13.12.2022).

[2] Н. Нильсон. Искусственный интеллект. Методы поиска решений. М.: Мир, 1973. с. 273.

[3] dict Python [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/2to3.html?highlight=dict/to3fixer-dict (дата обращения: 04.09.2022).

[4] Welcome to Python [Электронный ресурс]. Режим доступа: https://www.python.org (дата обращения: 04.09.2022).

[5] list Python [Электронный ресурс]. Режим доступа: https://docs.python.org/3/library/pdb.html?highlight=list/pdbcommand-list (дата обращения: 04.09.2022).

[6] Visual Studio Code [Электронный ресурс]. Режим доступа: https://code.visualstudio.com/ (дата обращения: 30.09.2022).

Список использованных источников

[1] National Institute of Standards and Technology. — Режим доступа: https://xlinux.nist.gov/dads/HTML/assocarray.html (дата обращения: 19.12.2022)

[2] Искусственный интеллект. Методы поиска решений./ Нильсон Н. // 1973 – С.273

[3] dict Python [Электронный ресурс]. — Режим доступа urlhttps://docs.python.org/3/library/2to3.html?highlight=dict/to3fixer-dict (дата обращения: 19.12..2022)

[4] Welcome to Python [Электронный ресурс]. — Режим доступа https://www.python.org (дата обращения: 19.12..2022)

[5] Time access and conversions [Электронный ресурс]. — Режим доступа https://docs.python.org/3/library/time.html/functions (дата обращения: 19.12..2022)

[6] list Python [Электронный ресурс]. — Режим доступа https://docs.python.org/3/library/pdb.html?highlight=list/pdbcommand-list (дата обращения: 19.12..2022)

[7] time — Time access and conversions. — Режим доступа: https://docs.python.org/3/library/time.html/functions (дата обращения: 19.12.2022)