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 Калашков П. А.

Группа ИУ7-56Б

Оценка (баллы)

Преподаватели Волкова Л. Л., Строганов Ю. В.

Москва — 2022 г.

Содержание

Введение

В процессе развития компьютерных систем количество обрабатываемых данных увеличивалось, вледствие чего множество операций над наборами данных стали выполняться очень долго, поскольку чаще всего это был обычный перебор. Это вызвало необходимость создать новые алгоритмы, которые решают поставленную задачу на порядок быстрее стандартного решения прямого обхода. В том числе это касается и словарей, в которых одной из основных операций является операция поиска.

Цель работы: получить навык поиска по словарю при ограничении на значение признака, заданном при помощи лингвистической переменной. Для достижения поставленной цели необходимо выполнить следующие задачи:

1) формализовать объект по варианту и его признак;
2) составить анкета для заполнения респондентом;
3) провести анкетирование респондентов;
4) построить функцию принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов;
5) описать алгоритм поиска в словаре объектов;
6) описать структуру данных словаря;
7) реализовать описанный алгоритм поиска в словаре;
8) описать и обосновать результаты в виде отчёта о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

1 Аналитическая часть

В данном разделе будут рассмотрены словарь как структура данных и алгоритм полного перебора, а также представлены требования к разрабатываемой программе.

1.1 Словарь как структура данных

Словарь [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:

1) *insert(k, v)*;

2) *find(k)*;

3) *remove(k)*.

В паре *(k, v)*: v называется значением, ассоциированным с ключом k. Где k — это ключ, а v — значение. Семантика и названия вышеупомянутых операций в разных реализациях ассоциативного массива могут отличаться.

Операция поиска *find(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).

Словарь 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 (именно по этой причине словарь также иногда называют «ассоциативным массивом»).

1.2 Алгоритм полного перебора

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

ализации 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.

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

1.3 Требования к программе

К разрабатываемой в данной работе программе предъявляется ряд требований:

1) на вход будет подаваться строка, на основании которой производится поиск;

2) на выходе — результат поиска в словаре;

3) программа не должна аварийно завершаться при отсутствии ключа в словаре.

1.4 Вывод

В данном разделе были рассмотрены словарь как структура данных и алгоритм полного перебора, а также приведены требования к разрабатываемой программе.

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:

1) словарь — встроенный тип dict [3] в Python[4] будет использован в созданном классе Dictionary;

2) массив ключей — встроенный тип list [5] в Python[4];

3) длина массива/словаря — целое число int.

2.2 Структура разрабатываемого ПО

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.

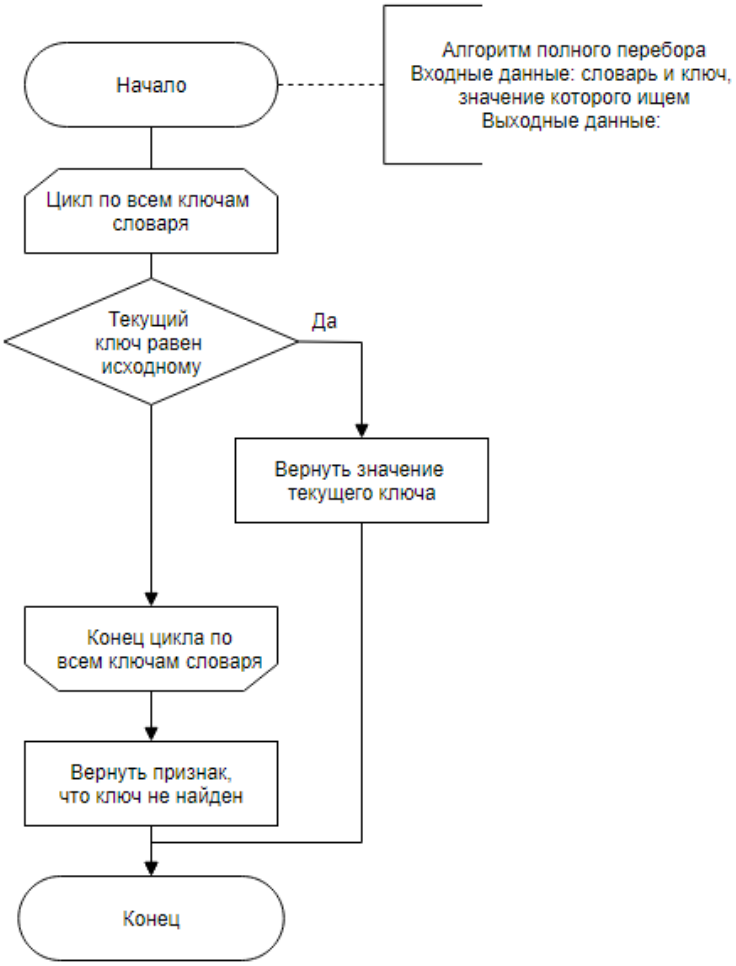

Рисунок 2.1 – Схема алгоритма поиска в словаре полным перебором

2.4 Классы эквивалентности при тестировании

Для тестирования выделены классы эквивалентности, представленные ниже.

1. Некорректный ввод ключа — пустая строка.
2. Корректный ввод, но ключа нет в словаре — вывод будет равен -1, как знак того, что такого ключа нет словаре.
3. Корректный ввод и ключ есть в словаре — вывод верного значения.

2.5 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 средства реализации и листинги реализованных алгоритмов.

3.1 Средства реализации

В данной работе для реализации был выбран язык программирования *Python* [4]. В текущей лабораторной работе требуется замерить процессорное время работы выполняемой программы и визуализировать результаты при помощи графиков. Инструменты для этого присутствуют в выбранном языке программирования.

3.2 Сведения о модулях программы

Программа состоит из четырёх модулей:

1) *main.py* — файл, содержащий точку входа;
2) *utils.py* — файл, содержащий служебные алгоритмы;
3) *constants.py* — файл, содержаший константы программы;
4) *humans.py* — файл, содержащий код класса *Human*, формализуемого рассматриваемый объект.

3.3 Реализация алгоритмов

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

Листинг 3.1 – Реализация алгоритма поиска полным перебором

```
1         def full_comb_search(self, key):
2             k = 0
3             keys = list(self.data.keys())
4             for elem in keys:
5                 if key == elem:
6                     return self.data[elem]
7             return -1
```

3.4 Функциональное тестирование

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

Входные данные	Пояснение	Результат
низкий	Средний элемент	Ответ верный
высокий	Первый элемент	Ответ верный
средний	Последний элемент	Ответ верный
упс	Несуществующий элемент	Ответ верный (-1)
123	Несуществующий элемент	Ответ верный (-1)

Таблица 3.1 – Таблица тестов

Все тесты пройдены *успешно*.

3.5 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 Формализация объекта и его признака

Согласно согласованному варианту, формализуем объект «человек» следующим образом: определим набор данных и признак объекта, на основании которого составим набор термов. Набор данных:

1) имя человека — строка;
2) пол человека — строка;
3) родная страна человека — строка;
4) профессия человека — строка.

Согласно варианту, признаком, по которому будет производиться поиск объектов, будет являться *рост* в сантиметрах — целое число.

Определим следующие термы, соответствующие признаку «рост»:

1) «Очень низкий»;
2) «Низкий»;
3) «Средний»;
4) «Высокий»;
5) «Очень высокий».

Также введём универсальное для данной задачи множество оцениваемой величины (роста) H:

$$H = \{80, 90, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190, 200, 210, 220\} \quad (4.1)$$

4.2 Анкетирование респондентов

Было проведено анкетирование следующих респондентов:

1) Соловов Юрий, группа ИУ7-56Б — Респондент 1;

2) Чепрасов Кирилл, группа ИУ7-56Б — Респондент 2;

3) Виноградов Алексей, группа ИУ7-56Б — Респондент 3;

4) Авсюнин Алексей, группа ИУ7-56Б — Респондент 4;

5) Ковель Александр, группа ИУ7-56Б — Респондент 5;

6) Комаров Никита, группа ИУ7-52Б — Респондент 6.

Респонденты, выступающие в качестве экспертов, для каждого из приведённых выше термов указали соответствующий промежуток, элементами которого являются числа из введённого для поставленной задачи множества оценимоемой величины.

Результаты анкетирования перечисленных респондентов продемонстрированы в таблице 4.1. В данной таблице Респ. — сокращение от «Респондент», термы 1 – 5 — термы, соответствующие обозначенным в п. 4.1 термам.

Таблица 4.1 – Результаты анкетирования

Терм	Респ. 1	Респ. 2	Респ. 3	Респ. 4	Респ. 5	Респ.6
1	[80; 110)	[80; 140]	[80; 150]	[80; 140]	[80; 130]	[80; 149]
2	[110; 150)	(140; 160]	(150; 165)	(140; 160]	(130; 150]	[150; 164]
3	[150; 180)	(160; 180]	[165; 190]	(160 ; 180]	(150; 170]	[165; 179]
4	[180; 190)	(180; 190]	[190; 200]	(180; 195]	(170; 190]	[180; 195]
5	[190; 220]	(190; 220]	(200; 220]	(195; 220]	(190; 220]	[196; 220]

4.2.1 Построение функции принадлежности термам

Построим графики функций принадлежности числовых значений переменной термам, описывающим группы значений лингвистической переменной.

Для этого для каждого значения из H для каждого терма из перечисленных найдём количество респондентов, согласно которым значение из H удоволетворяет сопоставляемому терму. Данное значение поделим на количество респондентов — это и будет значением функции μ для терма в точке. Графики функций принадлежности числовых значений роста термам, приведён на рисунке 4.1.

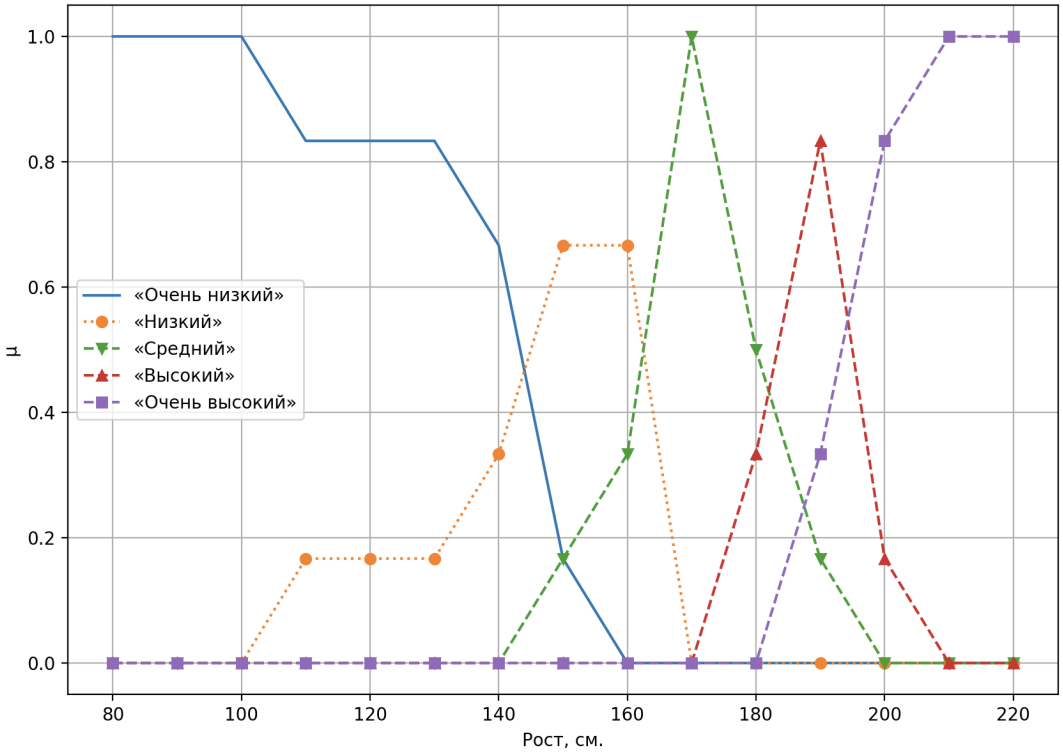

Рисунок 4.1 – Графики функций принадлежности числовых значений переменной термам, описывающим группы значений лингвистической переменной

В соответствии с полученным графиком будем считать людей:

1) очень низкого роста, если их значение их роста лежит в промежутке $[80; 143]$ сантиметров;

2) низкого роста, если их значение их роста лежит в промежутке $[144; 162]$ сантиметров;

3) среднего роста, если их значение их роста лежит в промежутке $[163; 182]$ сантиметров;

4) высокого роста, если их значение их роста лежит в промежутке $[183; 194]$ сантиметров;

5) очень высокого роста, если их значение их роста лежит в промежутке $[195; 220]$ сантиметров.

Заключение

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

В ходе выполнения лабораторной работы были решены все задачи:

1) формализован объект по варианту и его признак;
2) составлена анкета для заполнения респондентом;
3) проведено анкетирование респондентов;
4) построена функцию принадлежности термам числовых значений признака, описываемого лингвистической переменной, на основе статистической обработки мнений респондентов, выступающих в роли экспертов;
5) описан алгоритм поиска в словаре объектов;
6) описана структуру данных словаря;
7) реализоваан описанный алгоритм поиска в словаре;
8) полученные результаты описаны в виде отчёта о выполненной лабораторной работе, выполненном как расчётно-пояснительная записка к работе.

Список использованных источников

1. С. Шапошникова. Словари — Лаборатория линуксоида [Электронный ресурс]. 2022. URL: Режим доступа: `https://younglinux.info/python/dictionary` (дата обращения: 19.12.2022).

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).