Санкт-Петербургский Национальный Исследовательский Университет
Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №2

Выполнили:

Бабаев Р.С.

Садовая А.Р.

Абдулов И.А.

Проверил:

Мусаев А.А.

Санкт-Петербург,

2022

Оглавление

Введение

Данная работа представляет собой отчет о выполненных заданиях:

1. Написать программу для бинарного поиска, результатом которой является количество шагов, необходимых для нахождения требуемого числа;
2. Написать программу, которая по определенным характеристикам будет угадывать студента;
3. Составить граф для второй программы.

1 Решение задач

1.1 Программа для бинарного поиска

Бинарный поиск — тип поискового алгоритма, который последовательно делит пополам заранее отсортированный массив данных, чтобы обнаружить нужный элемент. Другие его названия — двоичный поиск, метод половинного деления, дихотомия.

Основная последовательность действий алгоритма выглядит так:

1. Сортируем массив данных;
2. Делим его пополам и находим середину;
3. Сравниваем срединный элемент с заданным искомым элементом;
4. Если искомое число больше среднего — продолжаем поиск в правой части массива (если он отсортирован по возрастанию): делим ее пополам, повторяя пункт 3. Если же заданное число меньше — алгоритм продолжит поиск в левой части массива, снова возвращаясь к пункту 3.

Схема действий данного алгоритма представлена на рисунках 1.1. и 1.2.

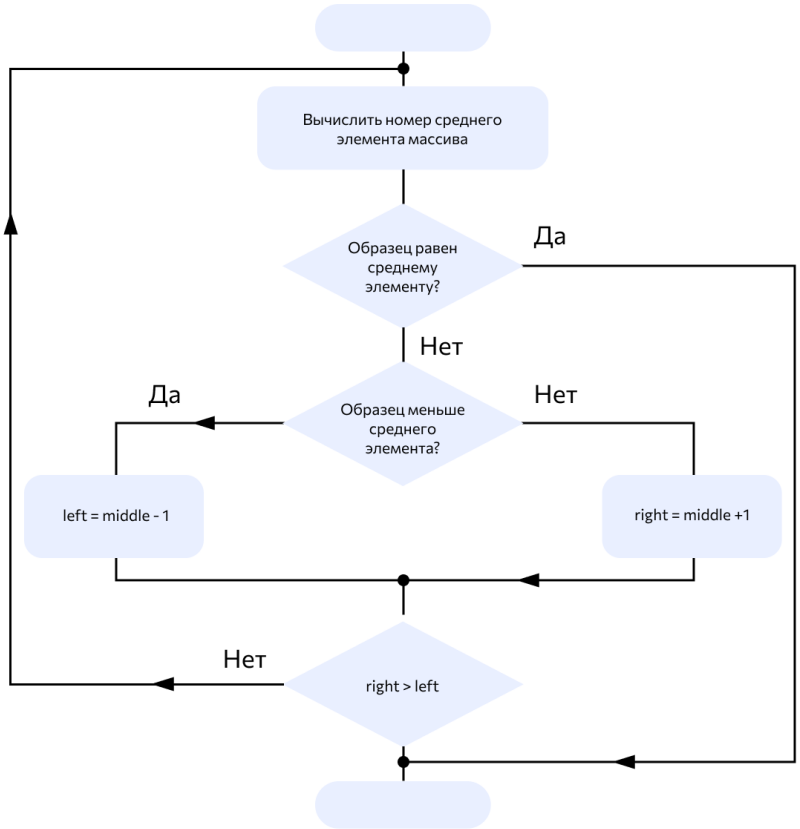

Рисунок 1.1 – Блок-схема бинарного поиска

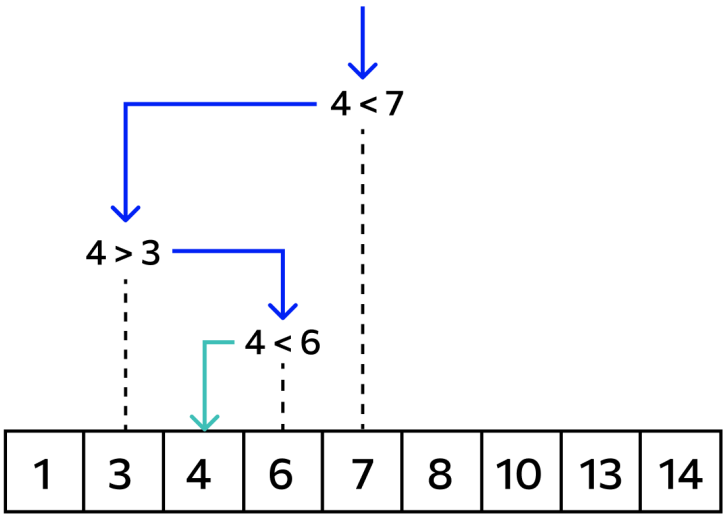

Рисунок 1.2 – Поиск позиции заданного элемента в списке

Существуют два способа реализации бинарного поиска: итерационный (с помощью цикла) и рекурсивный (функция, которая вызывает саму себя). В приведенной программе был реализован рекурсивный метод (см. рисунок 1.3). Функция binary_search(array, element, start, end), где array – исходный список, element – искомый элемент, start и end – границы поиска, сначала проверяет, не стала ли левая граница правее правой: такой случай означает, что искомого элемента нет в данном списке. Затем, если с границами всё в порядке, программа добавляет шаг (STEP) и индекс среднего элемента в списке. Если искомый элемент равен среднему, то функция выводит количество шагов, необходимых для нахождения данного элемента, иначе – вызывает саму себя с измененными границами.

На рисунке 1.4 представлен пример работы данной программы.

```
STEP = 0


def binary_search(array, element, start, end):
    """Рекурсивный вариант бинарного поиска"""
    if start > end:  # если элемент не существует в списке
        return 'Данного элемента нет в списке!'

    global STEP
    STEP += 1
    mid = (start + end) // 2
    if element == array[mid]:
        return f'Итоговое количество шагов: {STEP}'

    if element < array[mid]:
        return binary_search(array, element, start, mid - 1)
    else:
        return binary_search(array, element, mid + 1, end)


try:
    print('Доброго времени суток! Данная программа осуществляет алгоритм бинарного поиска и '
          'выводит количество шагов для нахождения искомого числа!')
    arr = [int(i) for i in input('Для начала запишите через пробел все целые числа '
                                 'в порядке возрастания, среди которых будет вестись поиск:\n').split()]
    el = int(input('Теперь введите искомое число: '))
    print(binary_search(sorted(arr), el, 0, len(arr) - 1))
except Exception:
    print('Некорректный формат ввода!')
```

Рисунок 1.3 – Код программы для бинарного поиска

```
C:\Users\rusba\PycharmProjects\algos\lab_2>python task_1.py
Доброго времени суток! Данная программа осуществляет алгоритм бинарного поиска и выводит количество шагов для нахождения искомого числа!
Для начала запишите через пробел все целые числа в порядке возрастания, среди которых будет вестись поиск:
1 2 3 4 5 6
Теперь введите искомое число: 7
Данного элемента нет в списке!

C:\Users\rusba\PycharmProjects\algos\lab_2>python task_1.py
Доброго времени суток! Данная программа осуществляет алгоритм бинарного поиска и выводит количество шагов для нахождения искомого числа!
Для начала запишите через пробел все целые числа в порядке возрастания, среди которых будет вестись поиск:
1 2 arr 2
Некорректный формат ввода!

C:\Users\rusba\PycharmProjects\algos\lab_2>python task_1.py
Доброго времени суток! Данная программа осуществляет алгоритм бинарного поиска и выводит количество шагов для нахождения искомого числа!
Для начала запишите через пробел все целые числа в порядке возрастания, среди которых будет вестись поиск:
1 2 4 5 6 9 11
Теперь введите искомое число: 3
Данного элемента нет в списке!

C:\Users\rusba\PycharmProjects\algos\lab_2>python task_1.py
Доброго времени суток! Данная программа осуществляет алгоритм бинарного поиска и выводит количество шагов для нахождения искомого числа!
Для начала запишите через пробел все целые числа в порядке возрастания, среди которых будет вестись поиск:
1 2 3 4 5 6 7 8 9 10
Теперь введите искомое число: 4
Итоговое количество шагов: 4

C:\Users\rusba\PycharmProjects\algos\lab_2>python task_1.py
Доброго времени суток! Данная программа осуществляет алгоритм бинарного поиска и выводит количество шагов для нахождения искомого числа!
Для начала запишите через пробел все целые числа в порядке возрастания, среди которых будет вестись поиск:
1 2
Теперь введите искомое число: 1
Итоговое количество шагов: 1
```

Рисунок 1.4 – Пример работы программы для бинарного поиска

1.2 Программа для угадывания студента

Перед написанием программы был проведен опрос среди студентов группы К3121, на основе которого были созданы список словарей студентов, где ключами выступают темы вопросов, а значениями – их ответы вида «да/нет» (см. рисунок 1.5), а также словарь вопросов с идентичной структурой (см. рисунок 1.6).

```
{
    "name": "Бабаев Руслан",
    "girl": False,
    "is18": True,
    "glasses": False,
    "brownEyes": True,
    "darkHair": True,
    "siblings": True,
    "pet": True,
    "fromSPb": False,
    "iphone": False,
    "tushavin": False,
    "coffe": False,
    "homefood": False,
    "watch": False,
    "nameVowel": False,
    "pizza": True,
    "smoke": False,
    "heightOver175": True,
    "partner": True,
    "winterOrSpring": True,
    "gaming": False,
    "campus": False,
    "movies": False,
    "anime": True,
},
```

Рисунок 1.5 – Пример словаря с информацией о студенте

```python
questions = {
    "girl": "Это студентка?",
    "is18": "Студенту 18 лет?",
    "glasses": "Студент носит очки?",
    "brownEyes": "У студента карие глаза?",
    "darkHair": "У студента тёмные волосы?",
    "siblings": "У студента есть братья/сёстры?",
    "pet": "У студента есть домашнее животное?",
    "fromSPb": "Студент из Санкт-Петербурга?",
    "iphone": "У студента iPhone?",
    "coffe": "Студент любит кофе больше, чем чай?",
    "pizza": "Студент любит пиццу больше, чем суши?",
    "smoke": "Студент курит?",
    "heightOver175": "Студент выше 175см?",
    "partner": "У студента есть вторая половинка?",
    "winterOrSpring": "У студента день рождения зимой или весной?",
    "gaming": "Студент любит видеоигры?",
    "campus": "Студент живёт в общежитии?",
    "movies": "Студент предпочитает больше фильмы, чем сериалы?",
    "anime": "Студент смотрит аниме?"
}
```

Рисунок 1.6 – Словарь с вопросами

Алгоритм угадывания студента достаточно прост: программа задает случайный вопрос из списка и в зависимости от ответа вычеркивает неподходящих студентов до тех пор, пока не останется единственный студент (в таком случае она выведет его имя) или вовсе не останется ни одного студента (в данной ситуации программа сообщит, что студент не найден). Код программы представлен на рисунке 1.7, а примеры ее работы – на рисунках 1.8 и 1.9.

```python
from random import randint
from answers import students
from questions import questions

print("Добро пожаловать в игру Акинатор!")
print("Загадайте студента из группы К3121 и Акинатор его угадает!")
print("Отвечайте на вопросы в формате: 'да' / 'нет'\n")

student = list(range(len(students)))  # список индексов студентов
question = [i for i in questions.keys()]  # список ключей вопросов
while len(student) != 1:
    n_question = randint(0, len(question) - 1)  # выбираем случайный вопрос
    print(questions[question[n_question]])
    answer = input()
    while answer.lower() != "да" and answer.lower() != "нет":
        print("Неккоректный ответ! Скажите 'да' или 'нет'!")
        answer = input()
    if answer.lower() == "да":
        answer = True
    else:
        answer = False
    for i in student:
        if students[i][question[n_question]] != answer:
            student.remove(i)  # удаляем студента
    if not student:  # если удалили всех студентов
        print("Такого студента нет в списке! Попробуйте сыграть ещё раз.")
        exit(0)
    del question[n_question]  # удаляем вопрос
print("Ваш студент:", students[student[0]]["name"])
```

Рисунок 1.7 – Пример работы программы для угадывания студента

```
C:\Users\rusba\PycharmProjects\algos\lab_2>python task_2.py
Добро пожаловать в игру Акинатор!
Загадайте студента из группы К3121 и Акинатор его угадает!
Отвечайте на вопросы в формате: 'да' / 'нет'

У студента карие глаза?
нет
У студента есть домашнее животное?
да
Студент носит очки?
нет
У студента есть братья/сёстры?
да
Студент любит пиццу больше, чем суши?
нет
У студента день рождения зимой или весной?
да
Студент смотрит аниме?
нет
Студент выше 175см?
нет
У студента есть вторая половинка?
нет
Студент любит видеоигры?
нет
Ваш студент: Садовая Анастасия Романовна
```

Рисунок 1.8 – Пример работы программы №2 (1)

```
C:\Users\rusba\PycharmProjects\algos\lab_2>python task_2.py
Добро пожаловать в игру Акинатор!
Загадайте студента из группы К3121 и Акинатор его угадает!
Отвечайте на вопросы в формате: 'да' / 'нет'

Студент смотрит аниме?
нет
Студент любит видеоигры?
да
У студента день рождения зимой или весной?
нет
У студента карие глаза?
нет
Студент носит очки?
да
У студента iPhone?
да
Студенту 18 лет?
да
У студента есть вторая половинка?
нет
Студент любит кофе больше, чем чай?
нет
Ваш студент: Абдулов Илья Александрович
```

Рисунок 1.9 – Пример работы программы №2 (2)

1.3 Граф для второй программы

Для создания графа была использована библиотека networkx, которая предоставляет довольно удобные инструменты для генерации больших графов. Программа всё так же использует сформированные словари с вопросами и информацией о студентах и с помощью функции add_edge(f_item, s_item, graph) связывает студентов (f_item) с соответствующими характеристиками (s_item). Таким образом она проходится по всем выбранным студентам и рисует граф.

С кодом программы можно ознакомиться на рисунке 1.10, а со сформированными ею графами – на рисунках 1.11–1.15.

```python
import itertools
import networkx as nx
import numpy as np
import matplotlib.pyplot as plt
from answers import students
from questions import for_graph

graph = nx.Graph()
ixes = list(range(len(students)))  # индексы студентов
for ix in ixes:
    node_name = students[ix]["name"].replace(' ', '\n')
    graph.add_node(node_name)

for i in for_graph.keys():
    graph.add_node(for_graph[i])


def add_edge(f_item, s_item, graph=None):
    graph.add_edge(f_item, s_item)
    graph.add_edge(s_item, f_item)


for i in for_graph.keys():
    for ix in ixes:
        node_name = students[ix]["name"].replace(' ', '\n')
        if students[ix][i]:
            add_edge(node_name, for_graph[i], graph=graph)

nx.draw_shell(graph, node_color='white', node_size=300, with_labels=True)  # draw_circular/draw_shell
```

Рисунок 1.10 – Код программы для генерации графов

Рисунок 1.11 – Граф №1

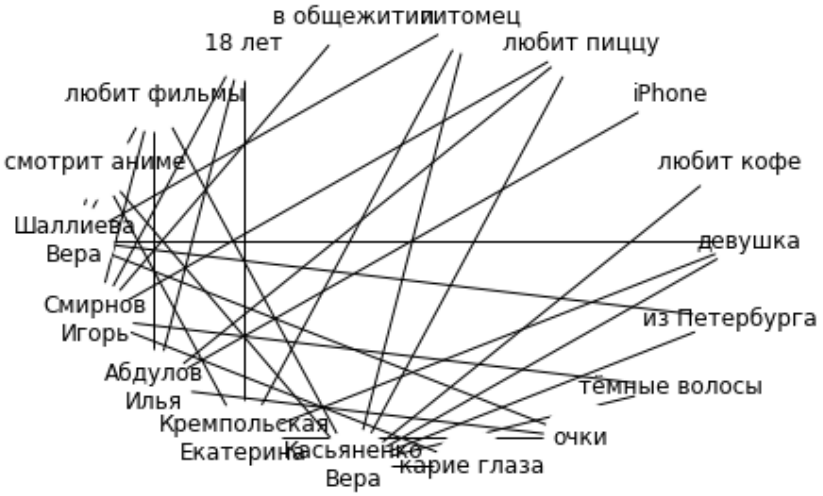

Рисунок 1.12 – Граф №2

Рисунок 1.13 – Граф №3

Рисунок 1.14 – Граф №4

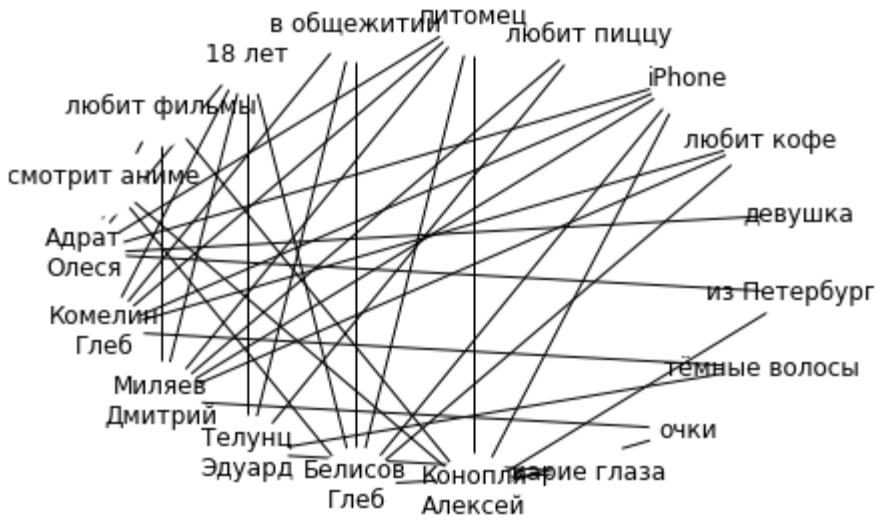

Рисунок 1.15 – Граф №5

Заключение

Таким образом, для каждой задачи была написана программа на языке программирования Python. Был изучен алгоритм бинарного поиска, реализована функция, осуществляющая рекурсивный двоичный поиск элемента из списка (целые числа). Также написана программа, угадывающая студента группы К3121 по ответам пользователя в формате «да/нет». Кроме того, составлены графы для данной программы с помощью библиотеки networkx. Для каждого решения приведены примеры с тестами.

Все программы можно найти на Github [2].

Список литературы

1. SkillFactory.blog [Электронный ресурс] – https://blog.skillfactory.ru/glossary/binarnyj-poisk/ (Дата обращения – 20.10.2022);
2. Github [Электронный ресурс] – https://github.com/HeraldOfWar/algos/tree/master/lab_2 (Дата обращения – 20.10.2022).