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

Факультет инфокоммуникационных технологий

Лабораторная работа №2
Вариант №1

Выполнил:
Бацанова Е. А.
Проверил
Мусаев А.А.

Санкт-Петербург,
2024

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Целью данной работы является ознакомление с графами в рамках программирования на Python. В ходе работы были реализованы следующие задачи:

1) Написание программы для бинарного поиска в отсортированном массиве и ее тестирование – определение количества шагов, необходимых для нахождения требуемого числа;
2) Работа со словарями, а именно создание словаря с характеристиками студентов, по которым можно однозначно определить описываемого человека, и разработка программы для угадывания студента по заданным параметрам;
3) Построение графа, отражающего работу программы из прошлого пункта, а именно связи студентов с их характеристиками.

Задание 1

Задание: Написать программу для бинарного поиска. Результатом должно быть количество шагов, которое потребуется, чтобы найти требуемое число.

> **Двоичный (бинарный) поиск** (также известен как **метод деления пополам** или **дихотомия**) — классический алгоритм поиска элемента в отсортированном массиве (векторе), использующий дробление массива на половины.

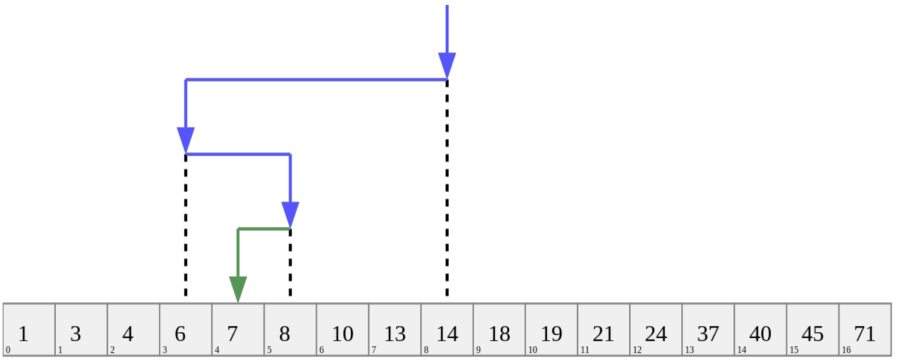

Рисунок 1 – Описание алгоритма бинарного поиска

Решение:

На рис. 2 приведена написанная программа, реализующая алгоритм бинарного поиска в отсортированном массиве. Бинарный поиск реализуется посредством функции *binary_search()*, на вход которой подается отсортированный массив *array* и целевое значение *target*, которое нужно найти в массиве.

Программа устанавливает начальные значения левого и правого указателей на границы массива и переменную *cnt* для подсчета итераций алгоритма. Пока левый указатель меньше правого, цикл продолжается.

Внутри цикла переменная *m* определяет середину массива путем нахождения среднего индекса между *left* и *right*. Затем происходит сравнение *target* с элементом в середине массива *m*. Если *target* больше *m*, то левый указатель *left* заменяется на *m + 1*, иначе – правый указатель *right* на *m*.

В конце функция возвращает количество итераций *cnt*, которые понадобились для поиска значения *target* в массиве.

Для проверки кода создадим отсортированный массив чисел от 0 до 99 и назначим целевым значение 42. Результат вызова функции *binary_search()* (количество итераций) выведем на экран. Вывод программы – 7.

```python
def binary_search(array, target):
    left, right = 0, len(array) - 1
    cnt = 0
    while left < right:
        cnt += 1
        m = (left + right) // 2
        if target > m:
            left = m + 1
        else:
            right = m
    return cnt


array, target = [i for i in range(100)], 42
print(binary_search(array, target))
```

Рисунок 2 – Реализация программы для бинарного поиска

Задание 2

Задание: Для своей группы составить словарь, который будет описывать характеристики каждого из студентов. Реализовать программу, которая по определенным характеристикам будет угадывать студента.

Пример выполнения:

>> Студент курит?

>> Нет.

>> Студент блондин?

>> Да.

>> …

>> …

>> Вы загадали Ивана Иванова.

Рисунок 3 – Пример работы программы

Решение:

Для начала попробуем понять, сколько нужно вопросов, чтобы мы могли однозначно идентифицировать студента. При условии, что ситуаций с полным совпадением ответов не реализуется, всего в группе 28 студентов, а на любой вопрос каждый из них может ответить только «да» или «нет», для формирования индивидуального двоичного номера студента необходимо по крайней мере 5 двоичных разрядов ($2^5 = 32$), то есть 5 вопросов. Сформулируем 6 вопросов, чтобы снизить вероятность совпадения ключей.

Создадим таблицу Excel с именем "students.xlsx", содержащую ФИО студентов, вопросы и ответы студентов на них (рис. 4).

ФИО	Студент парень?	Студент блондин(ка)?	Студент родился зимой?	У студента есть домашнее животное?	Умеете ли студент играть на музыкальном инструменте?	Студент кореной петербуржец?
Абдрахманов Мартин Маратович	да	нет	нет	да	нет	да
Бацанова Екатерина Александровна	нет	да	нет	нет	да	нет
Белоус Ярослав Романович	да	да	да	нет	да	да
Бордюг Владислав Юрьевич	да	нет	нет	да	да	нет
Буров Глеб Максимович	да	да	да	да	нет	да
Ветошкин Ростислав Владимирович	да	нет	нет	нет	да	да
Влазнев Данила Владимирович	да	нет	да	да	нет	нет
Власов Александр Алексеевич	да	да	нет	нет	да	да
Даниленко Дмитрий Александрович	да	да	да	да	нет	нет
Денисов Илья Алексеевич	да	нет	нет	нет	да	нет
Егорова Валерия Игоревна	нет	да	да	нет	да	нет
Жбейл Жеорж	да	нет	да	нет	да	да
Зенкин Даниил Николаевич	да	нет	нет	нет	нет	нет
Зименков Никита Вячеславович	да	нет	да	да	да	да
Иванова Екатерина Андреевна	нет	нет	нет	да	да	нет
Ишанова Надежда Романовна	нет	нет	да	да	да	да
Калачева Вера Викторовна	нет	да	нет	нет	нет	да
Клопов Михаил Павлович	да	да	да	да	да	нет
Колсанов Ярослав Иванович	да	нет	нет	нет	нет	да
Крашенинникова Дарья Олеговна	нет	да	нет	да	да	нет
Куцак Анастасия Михайловна	нет	нет	нет	нет	нет	нет
Маноменов Иван Андреевич	да	нет	нет	да	да	да
Марков Даниил Всеволодович	да	нет	да	нет	нет	нет
Панаёт Виктор Тудорович	да	нет	да	нет	да	нет
Париш Павел Игоревич	да	да	да	да	да	да
Петрова Виктория Владимировна	нет	да	нет	да	нет	нет
Соколова Дарья Максимовна	нет	да	да	да	да	да
Федоров Владислав Дмитриевич	да	нет	нет	да	нет	нет

Рисунок 4 – Созданная таблица, содержащая ФИО студентов, вопросы и их ответы

С помощью кода (рис. 5) считаем данные из файла "students.xlsx" в формате таблицы данных и затем сохраняет эту таблицу данных в переменную *df* с использованием библиотеки *pandas*. Затем для удобства последующей обработки извлечем значения из таблицы в массив *data_array*. Далее создадим пустой словарь *characteristics*, который будет хранить ответы на вопросы студентов в бинарном виде в качестве ключей, а в качестве значений, соответствующих этим ключам – ФИО студентов. Перебирая строки в массиве данных *data_array* преобразуем текстовый формат ответов «да» / «нет» в двоичный ключ. Значение ключа формируется в переменной *s*. После этого проверяем, если строка *s* уже существует в словаре *characteristics* как ключ, то выводим сообщение о совпадении ключей для студентов. В противном случае добавляем строку *s* в словарь *characteristics* с ключом *s* и значением, которое соответствует ФИО студента *i[0]*.

Для проверки корректности заполнения словаря *characteristics* выведем в консоль пары ключ – значение из словаря (рис. 6).

```
import pandas as pd


df = pd.read_excel('students.xlsx')
data_array = df.values
characteristics = {}
for i in data_array:
    s = ''
    for j in range(1, 7):
        if i[j] == 'да':
            s += '1'
        else:
            s += '0'
    if s in characteristics.keys():
        print(f"Студент {characteristics[s]} и {i[0]} имеют совпадающие ключи!")
    else:
        characteristics[s] = i[0]

for i in characteristics:
    print(f"{i} --> {characteristics[i]}")
```

Рисунок 5 – Реализация создания словаря (бинарный ключ – ФИО) и вывода его значений

```
100101 --> Абдрахманов Мартин Маратович
010010 --> Бацанова Екатерина Александровна
111011 --> Белоус Ярослав Романович
100110 --> Бордюг Владислав Юрьевич
111101 --> Буров Глеб Максимович
100011 --> Ветошкин Ростислав Владимирович
101100 --> Влазнев Данила Владимирович
110011 --> Власов Александр Алексеевич
111100 --> Даниленко Дмитрий Александрович
100010 --> Денисов Илья Алексеевич
011010 --> Егорова Валерия Игоревна
101011 --> Жбейл Жеорж
100000 --> Зенкин Даниил Николаевич
101111 --> Зименков Никита Вячеславович
000110 --> Иванова Екатерина Андреевна
001111 --> Ишанова Надежда Романовна
010001 --> Калачева Вера Викторовна
111110 --> Клопов Михаил Павлович
100001 --> Колсанов Ярослав Иванович
010110 --> Крашенинникова Дарья Олеговна
000000 --> Куцак Анастасия Михайловна
100111 --> Маноменов Иван Андреевич
101000 --> Марков Даниил Всеволодович
101010 --> Панаёт Виктор Тудорович
111111 --> Париш Павел Игоревич
010100 --> Петрова Виктория Владимировна
011111 --> Соколова Дарья Максимовна
100100 --> Федоров Владислав Дмитриевич
```

Рисунок 6 – Вывод значений словаря (бинарный ключ – ФИО)

Видим, что словарь заполнен корректно – ни разу не появилось сообщение о совпадении ключей для студентов.Теперь реализуем программу, угадывающую студента.

Код программы приведен на рисунке 7. Сначала был создан список *mas*, содержащий ключи из созданного ранее словаря *characteristics*. После каждого вопроса из массива удаляются ключи не подходящих студентов. Это позволяет сократить количество итераций, так как ответ будет выведен на экран как только массив начнет содержать всего один элемент (то есть единственный ключ, подошедший под все данные ранее ответы). При этом определить человека мы можем менее чем за 6 вопросов. В конце выводится информация о загаданном студенте.

```
mas = [i for i in characteristics.keys()]
cnt = 0
while len(mas) > 1:
    mas_temp = []
    answer = '1' if 'да' in input(f">> {questions[cnt]}\n>> ").lower() else '0'
    for i in mas:
        if i[cnt] == answer:
            mas_temp.append(i)
    mas = mas_temp
    cnt += 1

print(f"Загаданный студент: {characteristics[mas[0]]}.")
```

Рисунок 7 – Реализация программы, угадывающей студента

```
>> Студент парень?
>> да
>> Студент блондин(ка)?
>> да
>> Студент родился зимой?
>> ект
Загаданный студент: Власов Александр Алексеевич.
```

Рисунок 8 – Пример работы программы, угадывающей студента

Задание 3

Задание: Составьте граф для задания 2.

Решение:

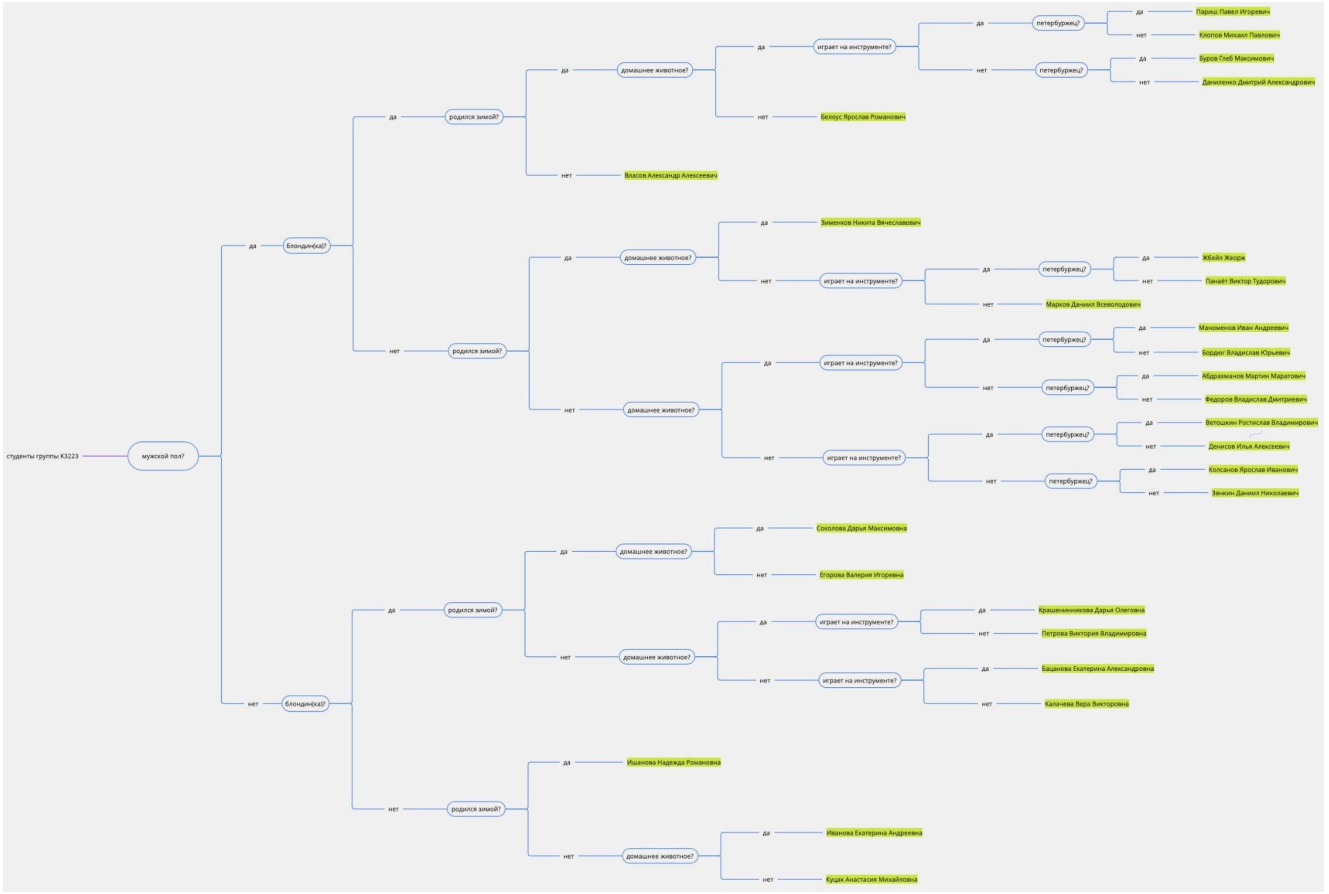

Рисунок 9 – Граф для задания 2

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была проведена работы с графами в рамках программирования на Python.

Была создана и протестирована программа для бинарного поиска. Данный алгоритм оказался удобен для поиска значения в массиве – в среднем он работает быстрее линейного поиска. В линейном поиске может понадобится от 1 до n шагов при поиске элемента в массиве из n чисел. В бинарном поиске количество шагов варьируется от $[log_2 n] - 1$ до $[log_2 n]$, где [...] – операция взятия целой части от числа (округление в большую сторону).

Далее была реализована программа для угадывания студента по заданным параметрам. Для этого был разработан словарь с характеристиками студентов, однозначно идентифицирующими каждого из них. После этого был построен граф, иллюстрирующий работу данной программы. Минимальное количество шагов, за которое можно было угадать студента, оказалось равно 3.

СПИСОК ЛИТЕРАТУРЫ

1) Habr. Решение задач с использованием алгоритма бинарного поиска. [Электронный ресурс] – (Дата последнего обращения 19.05.2024).

ПРИЛОЖЕНИЕ

Для удобства все файлы выгружены на GitHub: https://github.com/kathykkKk/Algorithms-and-Data-Structures-ICT.git