

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

Факультет инфокоммуникационных технологий

Лабораторная работа №1

Выполнили:

Стафеев И.А., Лапшина Ю.С., Килебе Нтангу Дьевина

Проверил

Мусаев А.А.

Санкт-Петербург,

2023

ВВЕДЕНИЕ	3
1 ОСНОВНАЯ ЧАСТЬ	4
1.1 Создание программы для бинарного поиска	4
1.2 Программа для угадывания студентов	5
1.3 Построение графа.....	6
ЗАКЛЮЧЕНИЕ	7
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ	8

ВВЕДЕНИЕ

Цель работы: научиться применять алгоритм бинарного поиска и теорию графов для выполнения практических задач.

Для достижения цели были поставлены следующие задачи:

- изучить принцип работы алгоритма бинарного поиска;
- написать программу для бинарного поиска, которая возвращает количество шагов для нахождения заданного числа;
- собрать данные о студентах учебной группы для последующего создания программы по угадыванию студентов;
- придумать способ реализации программы по угадыванию студентов
- написать программу для угадывания студентов, используя придуманный ранее способ реализации и используя данные, полученные о студентах учебной группы;
- составить граф, отображающий работу программы по угадыванию студентов.

1 ОСНОВНАЯ ЧАСТЬ

1.1 Создание программы для бинарного поиска

Для создания программы был применен стандартный алгоритм бинарного поиска, работающий по принципу «разделяй и властвуй» [1]. Код программы доступен по следующей ссылке: https://github.com/staffeev/itmo_algos_labs/blob/main/Lab1/task1.py.

Имплементация самого алгоритма бинарного поиска представлена в функции *binary_search*, которая принимает в качестве первого аргумента список, а в качестве второго – искомое число. Сортировка списка для работы бинарного поиска осуществлена с помощью функции *sorted()*, преобразование искомого числа в вещественное – с помощью функции *float()*. Функция возвращает целое положительное число, равное количеству итераций цикла, то есть количество шагов, необходимых для нахождения числа, или выводит на экран или в консоль строку «введённого числа нет в заданном списке», если заданное число в списке отсутствует. В теле функции задаются переменные для нижней и верхней границы поиска и переменная для хранения количества шагов (*low*, *high* и *steps*, соответственно). Затем в цикле берется средний из области поиска элемент и сравнивается с искомым числом. Формула для нахождения индекса среднего элемента представлена ниже:

$$mid = (low + high) // 2$$

Если средний элемент в списке меньше искомого, поиск продолжается в правой половине элементов области поиска, если средний элемент больше искомого – в левой. В первом случае нижняя граница обновляется до значения $mid + 1$, во втором – верхняя граница обновляется до значения $mid - 1$. Цикл выполняется до тех пор, пока средний элемент не будет равен искомому или пока левая границы поиска не превосходит правую (это, в свою очередь, свидетельствует о том, что искомое число отсутствует в заданном списке). На каждой итерации цикла количество шагов увеличивается на единицу, и если

средний элемент равен искомому, то функция возвращает значение переменной *steps*, в которой и хранится количество шагов. Иначе после цикла функция выводит в консоль или на экран текст «введённого числа нет в заданном списке».

1.2 Программа для угадывания студентов

Для получения данных о студентах была создана google-форма, содержащая 11 вопросов формата да/нет, результаты которой сохранены в csv-таблице в папке с кодом программы. Код программы доступен по ссылке: https://github.com/staffeev/itmo_algos_labs/blob/main/Lab1/task2.py. Функция *get_questions_and_answers* возвращает список вопросов и словарь, где ключами являются имена студентов, а значениями – списки из нулей и единиц, где каждой единице соответствует ответ «да» студента на вопрос и нулю соответствует ответ «нет».

Реализация программы требует применения структуры данных, с помощью которой можно было бы удобно представить связь студентов с данными ими ответами на вопросы. Один из вариантов такой структуры данных – бинарное дерево [2]. Пусть узлами дерева будут вопросы, метками ребер ответы «Да» и «Нет», а листьями – имена студентов. Тогда для нахождения конкретного студента нужно пройти по ветви дерева, ребра которой соответствуют ответам студента на вопросы. Имплементация построения бинарного дерева происходит в рекурсивной функции *create_tree*, принимающей индекс вопроса, по которому будут разделены студенты на две группы и словарь с ответами студентов. В теле функции происходит деление студентов на тех, кто ответил на текущий вопрос «да» и «нет», затем функция возвращает результат выполнения себя же, но уже с первой группой и со второй. Выход из функции происходит тогда, когда в словаре студентов остается не более одного человека. Итог выполнения функции – кортеж из двух

Угадывание студента происходит в функции *guessing_game*, принимающей список вопросов и дерево ответов. В зависимости от ответа пользователя на вопрос поиск продолжается в поддереве исходного дерева, соответствующему ответу «да» или «нет» на текущий вопрос. Когда дерево вырождается в строку, значит, программа нашла студента или определила, что студента с такими характеристиками не существует.

Поскольку код программы по угадыванию студентов уже создает граф (бинарное дерево), то остается его только наглядно представить. Для представления достаточно послойно рассматривать построенное дерево, в вершину записывать соответствующий слою вопрос (или лист, если ветвь дерева заканчивается) и между вершинами соседних слоев помещать ребра с метками «Да» и «Нет». На рисунке 1 представлено изображение графа, получаемого при работе программы №2. Изображение в высоком качестве доступно [по ссылке:](#)

```

graph TD
    Q1{Вы мужчина?} -- Да --> Q2{Вам есть 18 лет?}
    Q1 -- Нет --> Q3{Вам есть 18 лет?}
    
    Q2 -- Да --> Q4{Вы местный? (Из Санкт-Петербурга)}
    Q2 -- Нет --> Q5{Вы местный? (Из Санкт-Петербурга)}
    
    Q4 -- Да --> Q6{Ваш уровень английского - B2 или выше?}
    Q4 -- Нет --> Q7{Ваш уровень английского - B2 или выше?}
    
    Q6 -- Да --> A1[Вы - Роман Корчагин]
    Q6 -- Нет --> A2[Вы - Сергей Наустов]
    
    Q7 -- Да --> A3[Вы - Алексей Мозаук]
    Q7 -- Нет --> A4[Вы - Максим Ольхович]
    
    Q5 -- Да --> A5[Вы - Александр Лазуренко]
    Q5 -- Нет --> Q8{Вы местный? (Из Санкт-Петербурга)}
    
    Q8 -- Да --> Q9{Ваш уровень английского - B2 или выше?}
    Q8 -- Нет --> Q10{Ваш уровень английского - B2 или выше?}
    
    Q9 -- Да --> A6[Вы - Юлия Платина]
    Q9 -- Нет --> A7[Вы - Екатерина Давыдова]
    
    Q10 -- Да --> A8[Вы - Шмидтова Ольга]
    Q10 -- Нет --> A9[Вы - Елизавета Лаптюкова]
    
    Q3 -- Да --> Q11{Вы местный? (Из Санкт-Петербурга)}
    Q3 -- Нет --> Q12{Вы местный? (Из Санкт-Петербурга)}
    
    Q11 -- Да --> Q13{Ваш уровень английского - B2 или выше?}
    Q11 -- Нет --> Q14{Ваш уровень английского - B2 или выше?}
    
    Q13 -- Да --> A10[Вы - Иван Стафеев]
    Q13 -- Нет --> Q15{Вы окончили школу с медалью?}
    
    Q15 -- Да --> Q16{Вы умеете готовить?}
    Q15 -- Нет --> A11[Вы - Даниил Букинцев]
    
    Q16 -- Да --> A12[Вы - Михаил Румянцев]
    Q16 -- Нет --> A13[Вы - Максим Дармолов]
    
    Q14 -- Да --> A14[Извините, я вас не знаю!]
    Q14 -- Нет --> Q17{Вы окончили школу с медалью?}
    
    Q17 -- Да --> A15[Вы - Максим Фельдман]
    Q17 -- Нет --> A16[Вы - Алёша Могаук]

```

6

ЗАКЛЮЧЕНИЕ

Результатом выполнения лабораторной работы стало повышение навыков по применению алгоритма бинарного поиска и теории графов для решения практических задач. В ходе работы были написаны: программа, реализующая алгоритм бинарного поиска; программа, угадывающая студентов по их характеристикам и использующая структуру данных «бинарное дерево»; программа, строящая граф, отображающий построенное бинарное дерево. Таким образом, полученные навыки позволят оптимизировать и упростить работу над рядом практических задач, которые можно решить с использованием бинарного дерева и бинарного поиска.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Habr. Binary search algorithm [Электронный ресурс] – URL: <https://habr.com/ru/articles/563652/> (дата обращения 09.09.2023)
2. Ахо А.В., Хопкрофт Д.Э., Ульман Дж.Д. Алгоритмы и структуры данных [Электронный ресурс] – URL: https://vk.com/wall-114485185_260 (дата обращения 09.09.2023)
3. NetworkX. Документация библиотеки network для Python [Электронный ресурс] – URL: <https://networkx.org/documentation/stable/reference/index.html> (дата обращения 12.09.2023)