Министерство цифрового развития и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

федеральное государственное образовательное бюджетное

учреждение высшего профессионального образования

«Московский технический университет связи и информатики»

**Кафедра «Математическая кибернетика и**

**информационные технологии»**

**Отчет по лабораторной работе №2**

по дисциплине«Структуры и алгоритмы обработки данных»

на тему: «**Методы поиска**»

Выполнил: студент

группы БСТ2001

Ибодуллоев У.Х.

Проверил:

Ст. преп. Чайка А.Д.

Москва, 2022

Оглавление

[**1. Цель работы 3**](#_Toc102090679)

[**2. Задания 3**](#_Toc102090680)

[**Задание №1: 3**](#_Toc102090681)

[**Задание №2: 3**](#_Toc102090682)

[**Задание №3: 3**](#_Toc102090683)

[**3. Ход работы 3**](#_Toc102090684)

[**Задание №1: 3**](#_Toc102090685)

[**Бинарный поиск: 4**](#_Toc102090686)

[**Бинарное дерево: 4**](#_Toc102090687)

[**Поиск фибоначчи: 4**](#_Toc102090688)

[**Интерполяционный поиск: 4**](#_Toc102090689)

[**Задание №2: 4**](#_Toc102090690)

[**Задание №3: 4**](#_Toc102090691)

[**4. Вывод 4**](#_Toc102090692)

[**5. Ссылка на удалённый репозиторий 4**](#_Toc102090693)

[**6. Список использованных источников 4**](#_Toc102090694)

# Цель работы

Научится использовать различные способы поиска и хэширования на языке программирования python.

# Задания

## Задание №1:

Реализовать методы поиска в соответствии с заданием. Организовать генерацию начального набора случайных данных. Для всех вариантов добавить реализацию добавления, поиска и удаления элементов. Оценить время работы каждого алгоритма поиска и сравнить его со временем работы стандартной функции поиска, используемой в выбранном языке программирования.

* Бинарный поиск
* Бинарное дерево
* Фибоначчиев
* Интерполяционный

## Задание №2:

Реализовать следующие методы в выбранном языке программирования:

* Простое рехэширование
* Рехэширование с помощью псевдослучайных чисел
* Метод цепочек

## Задание №3:

Расставить на стандартной 64-клеточной шахматной доске 8 ферзей так, чтобы ни один из них не находился под боем другого». Подразумевается, что ферзь бьёт все клетки, расположенные по вертикалям, горизонталям и обеим диагоналям. Написать программу, которая находит хотя бы один способ решения задач.

# Ход работы

## Задание №1:

### Бинарный поиск:

Бинарный поиск работает по принципу [«разделяй и властвуй»](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D1%8F%D0%B9_%D0%B8_%D0%B2%D0%BB%D0%B0%D1%81%D1%82%D0%B2%D1%83%D0%B9_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)).

Предполагая, что мы ищем значение val в отсортированном массиве, алгоритм сравнивает val со значением среднего элемента массива, который мы будем называть mid.

Если mid — это тот элемент, который мы ищем (в лучшем случае), мы возвращаем его индекс.

Если нет, мы определяем, в какой половине массива мы будем искать val дальше, основываясь на том, меньше или больше значение val значения mid, и отбрасываем вторую половину массива.

Затем мы рекурсивно или итеративно выполняем те же шаги, выбирая новое значение для mid, сравнивая его с val и отбрасывая половину массива на каждой итерации алгоритма.

### Бинарное дерево:

Бинарное дерево — иерархическая структура данных, в которой каждый узел имеет не более двух потомков (детей). Как правило, первый называется родительским узлом, а дети называются левым и правым наследниками. Двоичное дерево является упорядоченным ориентированным.

### Поиск фибоначчи:

Поиск Фибоначчи — это еще один алгоритм «разделяй и властвуй», который имеет сходство как с бинарным поиском, так и с jump search. Он получил свое название потому, что использует числа Фибоначчи для вычисления размера блока или диапазона поиска на каждом шаге.

### Интерполяционный поиск:

Интерполяционный поиск — это еще один алгоритм «разделяй и властвуй», аналогичный бинарному поиску. В отличие от бинарного поиска, он не всегда начинает поиск с середины. Интерполяционный поиск вычисляет вероятную позицию искомого элемента по формуле:

index = low + [(val-lys[low])\*(high-low) / (lys[high]-lys[low])]

В этой формуле используются следующие переменные:

lys — наш входной массив.

val — искомый элемент.

index — вероятный индекс искомого элемента. Он вычисляется как более высокое значение, когда значение val ближе по значению к элементу в конце массива (lys[high]), и более низкое, когда значение val ближе по значению к элементу в начале массива (lys[low]).

low — начальный индекс массива.

high — последний индекс массива.

Алгоритм осуществляет поиск путем вычисления значения индекса:

Если значение найдено (когда lys[index] == val), возвращается индекс.

Если значение val меньше lys[index], то значение индекса пересчитывается по формуле для левого подмассива.

Если значение val больше lys[index], то значение индекса пересчитывается по формуле для правого подмассива.

## Задание №2:

Хэш-функция — это функция, которая принимает на вход какие-либо данные (например, строки) и возвращает число по некоторому заданному алгоритму. Если эта функция является идеальной - то для каждого переданного на вход функции значения будет возвращено число, отличное от ранее полученных. В противном случае, будут возникать [коллизии](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BB%D0%BB%D0%B8%D0%B7%D0%B8%D1%8F_%D1%85%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8), когда для различных входных данных возвращается одно и то же число.

Назначением хэш-функций является возможность помещения некоторого элемента (например, строки) в хэш-таблицу, на основе которых реализованы, например, словари и множества в Python.

## Задание №3:

Создаем двумерный массив 8 на 8 (шахматная доска), далее снизу вверх и слева направо ставим по одному ферзю. Пересечения вертикалей, горизонталей и диагоналей с первым ферзем мы закрашиваем, в оставшихся неокрашенных клетках второй строки мы по порядку и рекурсивно ставим ферзя и вычисляем дальнейшие действия. Если в конце мы достигли последней строки и смогли поставить ферзя, то количество решения увеличивается на 1. В конце выводится количество найденных решений.

# Вывод

Таким образом, я научился использовать различные способы поиска на языке программирования python.

# Ссылка на удалённый репозиторий

<https://github.com/1Double/MTUCI/tree/main/Term_4/SAOD/Lab2>

# Список использованных источников

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006.
2. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.
3. ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления