

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №9**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Алгоритмы поиска в таблице (массиве)

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-06-22 |  | Кликушин В.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

ОГЛАВЛЕНИЕ

[1 ОТЧЁТ ПО ЗАДАНИЮ 1 3](#_Toc136873661)

[1.1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc136873662)

[1.2 СТРУКТУРА ЗАПИСИ 3](#_Toc136873663)

[1.3 КОДЫ ФУНКЦИЙ, РЕАЛИЗУЮЩИХ АЛГОРИТМЫ 3](#_Toc136873664)

[1.4 ТЕОРЕТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМОВ 4](#_Toc136873665)

[1.5 ТАБЛИЦЫ РЕЗУЛЬТАТОВ 5](#_Toc136873666)

[1.6 ГРАФИКИ ЗАВИСИМОСТИ ВРЕМЕНИ ОТ ВХОДНЫХ ДАННЫХ 6](#_Toc136873667)

[1.7 ВЫВОДЫ 7](#_Toc136873668)

[2 ОТЧЁТ ПО ЗАДАНИЮ 2 8](#_Toc136873669)

[2.1 ПОСТАНОВКА ЗАДАЧИ 8](#_Toc136873670)

[2.2 АЛГОРИТМ ПОИСКА ФИБОНАЧЧИ 8](#_Toc136873671)

[2.3 КОДЫ ФУНКЦИЙ, РЕАЛИЗУЮЩИХ АЛГОРИТМЫ 9](#_Toc136873672)

[2.4 ТАБЛИЦЫ РЕЗУЛЬТАТОВ 10](#_Toc136873673)

[2.5 ГРАФИКИ ЗАВИСИМОСТИ ВРЕМЕНИ ОТ ВХОДНЫХ ДАННЫХ 11](#_Toc136873674)

[2.6 ВЫВОДЫ 12](#_Toc136873675)

[3 ВЫВОДЫ 13](#_Toc136873676)

[4 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ 14](#_Toc136873677)

# 1 ОТЧЁТ ПО ЗАДАНИЮ 1

## 1.1 ПОСТАНОВКА ЗАДАЧИ

Разработать программу поиска записи по ключу в таблице записей с применение двух алгоритмов линейного поиска.

Структура записи: Страховой полис: номер полиса, компания, фамилия владельца.

## 1.2 СТРУКТУРА ЗАПИСИ

struct Data

{

int key;

string company;

string surname;

};

Если считать, что название компании и фамилия владельца занимают до 15 символов, а номер полиса – 8-значное число, то размер одной записи: 8 + 15 + 15 = 38 Байт.

## 1.3 КОДЫ ФУНКЦИЙ, РЕАЛИЗУЮЩИХ АЛГОРИТМЫ

Линейный поиск:

Предусловие: элемент arr типа Table, ключ поиска key типа long long int.

Постусловие: индекс позиции найденного элемента (-1, если элемента не найдено).

int simple\_search(Table arr, long long int key)

{

for (int i = 0; i < arr.n; i++)

{

if (arr.table[i].key == key)

{

cout << " Компания: " << arr.table[i].company << " \n Фамилия владельца: " << arr.table[i].surname << endl;

return i;

}

}

return -1;

}

Барьерный поиск:

Предусловие: элемент arr типа Table, ключ поиска key типа long long int.

Постусловие: индекс позиции найденного элемента (-1, если элемента не найдено).

int bareer\_search(Table arr, long long int key)

{

int i = 0;

arr.table[arr.n + 1].key = key;

while (key != arr.table[i].key)

i++;

if (i <= arr.n)

{

cout << " Компания: " << arr.table[i].company << " \n Фамилия владельца: " << arr.table[i].surname << endl;

return i;

}

else

return -1;

}

## 1.4 ТЕОРЕТИЧЕСКАЯ СЛОЖНОСТЬ АЛГОРИТМОВ

Таблица 1. Оценка сложности линейного поиска (в худшем случае)

|  |  |
| --- | --- |
| Оператор | Количество выполнений оператора |
| for i ← 0 to n-1 do | 2\*(n + 1) |
| if table[i].key = key then | n |
| return i | 1 |
| od |  |
| return -1 | 1 |

Таблица 2. Оценка сложности линейного поиска с барьером (в худшем случае)

|  |  |
| --- | --- |
| Оператор | Количество выполнений оператора |
| table[n].key ← key | 1 |
| i ← 0 | 1 |
| while table[i].key != key do | n + 1 |
| i ← i + 1 | n |
| od |  |
| return i | 1 |

## 1.5 ТАБЛИЦЫ РЕЗУЛЬТАТОВ

Таблица 3. Анализ линейного поиска

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) - время | Tэт = f(C + M) =  3 \* n + 4 | Tэп = Cф + Mф |
| 100 | 0.1861 мс | 304 | Перемещений: 10  Сравнений: 200  Общее: 300 |
| 1000 | 0.454 мс | 3004 | Перемещений: 1000  Сравнений: 2000  Общее: 3000 |
| 10000 | 0.6854 мс | 30004 | Перемещений: 10000  Сравнений: 20000  Общее: 30000 |

Таблица 4. Анализ поиска с барьером

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) - время | Tэт = f(C + M) =  2 \* n + 4 | Tэп = Cф + Mф |
| 100 | 0.18 мс | 204 | Перемещений: 101  Сравнений: 101  Общее: 202 |
| 1000 | 0.189 мс | 2004 | Перемещений: 1001  Сравнений: 1001  Общее: 2002 |
| 10000 | 0.251 мс | 20004 | Перемещений: 10001  Сравнений: 10001  Общее: 20002 |

## 1.6 ГРАФИКИ ЗАВИСИМОСТИ ВРЕМЕНИ ОТ ВХОДНЫХ ДАННЫХ

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 1 – Графики линейного поиска и поиска с барьером

Изображение выглядит как текст, снимок экрана, линия, График

Автоматически созданное описание

Рисунок 2 – Асимптотическая оценка порядка роста

## 1.7 ВЫВОДЫ

Поиск с барьером является более эффективным, чем простой линейный поиск за счёт уменьшения количества операций сравнения, в теории давая преимущество примерно на 30%, однако на практике оно куда меньше (неидеальные условия замера времени выполнения). Оба алгоритма имеют линейный порядок роста и не требуют выделения дополнительной памяти.

# 2 ОТЧЁТ ПО ЗАДАНИЮ 2

## 2.1 ПОСТАНОВКА ЗАДАЧИ

Разработать программу поиска записи по ключу в таблице записей с применением алгоритма поиска Фибоначчи.

Подход к решению:

Для реализации данного алгоритма не требуются дополнительные структуры, но важно, чтобы ключи в таблице были отсортированы по возрастанию.

Пример: ключи в таблице = [1, 2, 3, **4**, 5, 6]

Нужно найти ключ = 4 (жирным выделены рассматриваемые промежутки)

|  |
| --- |
| [**1**, **2**, 3, 4, 5, 6] |
| [1, **2**, 3**,** 4, 5, 6] |
| [1, **2**, **3,** 4, 5, 6]  [1, 2, **3, 4**, 5, 6]  [1, 2, 3**, 4**, **5**, **6**] |

## 2.2 АЛГОРИТМ ПОИСКА ФИБОНАЧЧИ

fibbonachi\_search(arr, key)

fib2 ← 0

fib1 ← 0

fibM ← fib1 + fib2

offset ← -1

while (fibM << arr.n) do

fib2 ← fib1

fib1 ← fibM

fibM ← fib2 + fib1

od

while (fibM > 1) do

i ← min(offset + fib2, arr.n – 1)

if (arr.table[i].key < key) then

fibM ← fib1

fib1 ← fib2

fib2 ← fibM - fib1

offset ← i

else if (arr.table[i].key > key)

fibM ← fib2

fib1 ← fib1 – fib2

fib2 ← fibM - fib1

else

Возврат i

Endif

if ((fib1 && arr.table[offset + 1].key == key) then

Возврат offset + 1

Возврат -1

## КОДЫ ФУНКЦИЙ, РЕАЛИЗУЮЩИХ АЛГОРИТМЫ

Фибоначчи поиск:

Предусловие: элемент arr типа Table - таблица, где ключи отсортированы по возрастанию, целочисленный key – ключ поиска

Постусловие: позиция найденного элемента (-1, если элемент не найден)

int fibbonachi\_search(Table arr, long long int key)

{

long long int fib2 = 0;

long long int fib1 = 1;

long long int fibM = fib2 + fib1;

while (fibM < arr.n)

{

fib2 = fib1;

fib1 = fibM;

fibM = fib2 + fib1;

}

int offset = -1;

while (fibM > 1)

{

int i = min(offset + fib2, arr.n - 1);

if (arr.table[i].key < key)

{

fibM = fib1;

fib1 = fib2;

fib2 = fibM - fib1;

offset = i;

}

else if (arr.table[i].key > key)

{

fibM = fib2;

fib1 = fib1 - fib2;

fib2 = fibM - fib1;

}

else

return i;

}

if (fib1 && arr.table[offset + 1].key == key)

return offset + 1;

return -1;

}

## 2.4 ТАБЛИЦЫ РЕЗУЛЬТАТОВ

Таблица 5. Анализ Фибоначчи поиска

|  |  |  |  |
| --- | --- | --- | --- |
| n | T(n) - время | Tэт = f(C + M) =  logn | Tэп = Cф + Mф |
| 100 | 0.0005 мс | 7 | Перемещений: 45  Сравнений: 16  Общее: 61 |
| 1000 | 0.0008 мс | 10 | Перемещений: 60  Сравнений: 21  Общее: 81 |
| 10000 | 0.001 мс | 13 | Перемещений: 87  Сравнений: 31  Общее: 118 |

## 2.5 ГРАФИКИ ЗАВИСИМОСТИ ВРЕМЕНИ ОТ ВХОДНЫХ ДАННЫХ

Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание

Рисунок 3 – График поиска Фибоначчи

Изображение выглядит как линия, диаграмма, График

Автоматически созданное описание

Рисунок 4 – Асимптотическая оценка порядка роста

## 2.6 ВЫВОДЫ

Поиск Фибоначчи является куда более эффективным алгоритмом поиска по сравнению с двумя линейными ранее рассматриваемыми алгоритмами, однако за такую эффективность стоит платить дополнительной сортировкой в порядке неубывания ключей.

# 3 ВЫВОДЫ

В то время, как линейный поиск и поиск с барьером являются довольно медленными, они подходят для любых таблиц, вне зависимости от расположения ключей. Поиск Фибоначчи справляется с задачей поиска лучше, но в некоторых случаях, требуемая дополнительная сортировка будет лишь «абузой».

# 4 ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Учебно-методическое пособие СиАОД (часть 1)

2. Приложение к практическим работам – СДО (online-edu.mirea.ru)