ФГБОУ ВО “Чувашский государственный университет им.

И.Н. Ульянова”

Кафедра вычислительной техники

Лабораторная работа №4

по дисциплине

«Структуры и алгоритмы обработки данных»

Вариант 8.

Выполнил:

Студент гр. ИВТ-41-22

Иванов Владимир Сергеевич

Проверил:

Павлов Леонид Александрович

Чебоксары 2024

Цель работы: изучение основных методов быстрого поиска, получение навыков программирования задач быстрого поиска

Подготовка к работе:

**Реализуемые в моем варианте алгоритмы:**

1. Последовательный поиск в упорядоченной таблице.

2. Обычный бинарный поиск.

3. Поиск Фибоначчи с проверкой условия i < 0.

4. Поиск Фибоначчи без проверки условия i < 0.

5. Поиск в обычном дереве бинарного поиска (без балансировки).

6. Поиск в RB-дереве (узел содержит поле для указателя на отца).

7. Цифровой поиск.

8. Поиск с хешированием.

Алгоритмы на псевдокоде для каждого метода поиска:

**1. Последовательный поиск в упорядоченной таблице**

function sequentialSearch(table, target):

for i from 0 to length(table) - 1:

if table[i] == target:

return i // Элемент найден

else if table[i] > target:

return -1 // Элемент не может быть найден, так как таблица упорядочена

return -1 // Элемент не найден

**2. Бинарный поиск в упорядоченной таблице**

function binarySearch(table, target):

low = 0

high = length(table) - 1

while low <= high:

mid = low + (high - low) / 2

if table[mid] == target:

return mid // Элемент найден

else if table[mid] < target:

low = mid + 1

else:

high = mid - 1

return -1 // Элемент не найден

**3,4. Поиск Фибоначчи в упорядоченной таблице**

function fibonacciSearch(table, target):

fibM\_minus\_2 = 0

fibM\_minus\_1 = 1

fibM = fibM\_minus\_1 + fibM\_minus\_2

while fibM < length(table):

fibM\_minus\_2 = fibM\_minus\_1

fibM\_minus\_1 = fibM

fibM = fibM\_minus\_1 + fibM\_minus\_2

offset = 0

while fibM\_minus\_2 > 0:

i = min(offset + fibM\_minus\_2, length(table) - 1)

if table[i] == target:

return i // Элемент найден

else if table[i] < target:

fibM = fibM\_minus\_1

fibM\_minus\_1 = fibM\_minus\_2

fibM\_minus\_2 = fibM - fibM\_minus\_1

offset = i

else:

fibM = fibM\_minus\_2

fibM\_minus\_1 = fibM\_minus\_1 - fibM\_minus\_2

fibM\_minus\_2 = fibM - fibM\_minus\_1

return -1 // Элемент не найден

**5,6. Поиск в обычном и RB-дереве**

function RBTreeSearch(root, target):

while root is not nil:

if root.key == target:

return root // Найден узел с искомым ключом

else if root.key < target:

root = root.right

else:

root = root.left

return nil // Узел с искомым ключом не найден

**7. Поиск в хеш-таблице**

function hashTableSearch(hashTable, target):

return hashTable.contains(target);

**8. Цифровой поиск в упорядоченной таблице**

function digitalSearch(table, target):

low = 0

high = length(table) - 1

while low <= high:

mid = (low + high) / 2

if table[mid] == target:

return true // Элемент найден

else if table[mid] < target:

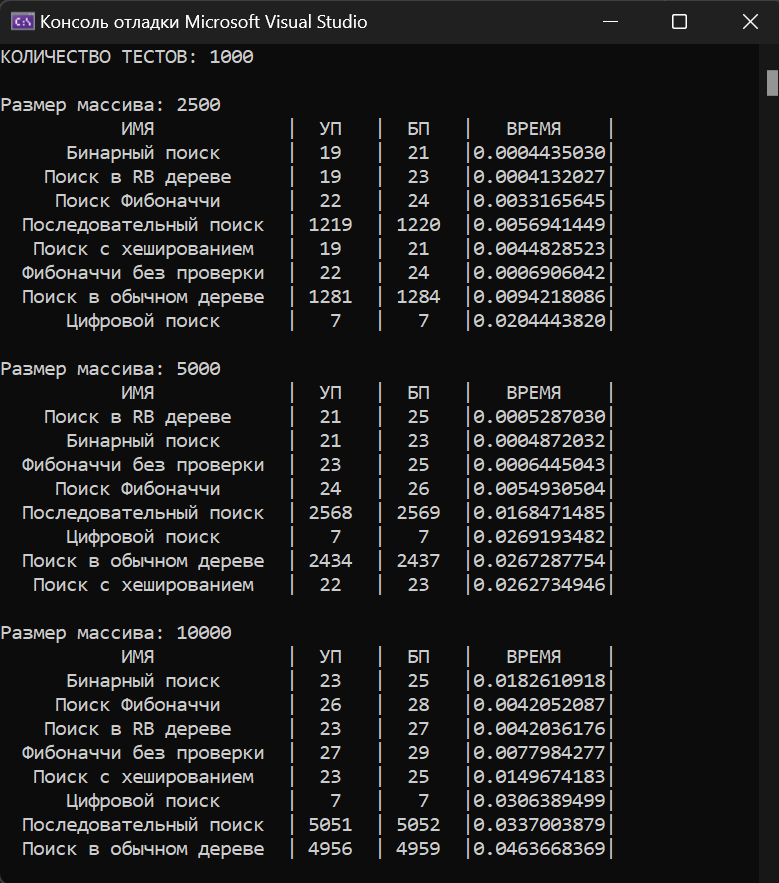
low = mid + 1

else:

high = mid - 1

return false // Элемент не найден

Скриншоты результатов выполнения программы:



Вывод: изучил основные методы быстрого поиска, получил навыки программирования задач быстрого поиска