МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(национальный исследовательский университет)»

Институт №3 «Системы управления, информатика и электроэнергетика»

Кафедра № 304 «Вычислительные машины, системы и сети»

Информатика

Отчет по лабораторной работе №2

Выполнила бригада группы М3О-211Б-21

Багиров Э. Р.

Нуриев Н. Н.

Оглавление

[Задание 3](#_Toc116234339)

[Код программы 4](#_Toc116234340)

[Таблицы зависимости времени работы (в мкс) алгоритмов поиска от количества элементов массива 7](#_Toc116234341)

[При отсутствии искомого элемента в массиве (минимальное значение за три итерации) 7](#_Toc116234342)

[Искомый элемент 64001 в первой половине массива 7](#_Toc116234343)

[Искомый элемент 157 в первой половине массива 8](#_Toc116234344)

[Искомый элемент 64001 во второй половин массива 8](#_Toc116234345)

[Искомый элемент 157 во второй половине массива 8](#_Toc116234346)

# Задание

Для алгоритмов **BLS** и **SLS** в качестве входного массива использовать одну и ту же последовательность значений (функция **rand**( )).

Для алгоритмов **Т** и **В** – значения массива должны быть отсортированы по неубыванию, одна и та же последовательность чисел (можно использовать соответствующую функцию из первой лабораторной работы).

Оценить длительность поиска для различных значений размеров последовательностей (начиная с 10000 до 200000 элементов массива, провести измерения не менее, чем для 10 разных размерностей).

Для каждой размерности рассматриваются случаи нахождения ключа поиска в начале, в середине и в конце массива.

Для алгоритмов **BLS** и **SLS** кроме подсчета **времени,** необходимого для поиска, требуется определить сколько раз выполняются операции **сравнения** (сравнение ключа с элементом массива, а также в одном из этих двух алгоритмов добавляется подсчет сравнений при анализе индекса элемента массива в цикле… ).

Все результаты оформить в виде таблиц и графиков. На графиках **- только временные характеристики** поиска.

По результатам сделать выводы об эффективности того или иного алгоритма поиска.

# Код программы

#include <iostream>

#include <ctime>

#include <chrono>

using namespace std;

void InsSort();

int BLS();

int SLS();

int T();

int B();

void InsSort(int\* arr, int length) {

int count = 0;

int temp;

for (int i = 1; i < length; i++)

for (int j = i; j > 0 && arr[j - 1] > arr[j]; j--) {

count++;

temp = arr[j - 1];

arr[j - 1] = arr[j];

arr[j] = temp;

}

}

int BLS(int\* arr, int length, int key) {

for (int i = 0; i < length; i++)

if (arr[i] == key)

return i;

return -1; // NOT FOUND

}

int SLS(int\* arr, int length, int key) {

int last = arr[length-1];

arr[length - 1] = key;

int i = 0;

while (arr[i] != key)

i++;

arr[length - 1] = last;

if (i < length-1 || arr[length - 1] == key)

return i;

return -1; // NOT FOUND

}

int T(int\* arr, int length, int key) {

arr[length] = 2147483647;

int i = 0;

while (key > arr[i]) i++;

if (arr[i] == key) return i;

return -1; // NOT FOUND

}

int B(int\* arr, int length, int key) {

int p = 0;

int r = length-1;

int q = 0;

while (true) {

int q = (p + r) / 2;

if (key < arr[q]) r = q - 1;

else if (key > arr[q]) p = q + 1;

else return q;

if (p > r) return -1; // NOT FOUND

}

}

int main() {

srand(0);

setlocale(LC\_ALL, "rus");

int length;

cout << "Введите длину массива: ";

cin >> length;

cout << endl;

int\* arr = new int[length];

for (int i = 0; i < length; i++) arr[i] = rand();

int key;

cout << "Введите ключ поиска: ";

cin >> key;

int pick;

cout << endl << "Хотите вставить ключ в массив?\n1.Да\n2.Нет\nВаш выбор: ";

cin >> pick;

if (pick == 1) {

int indexForKey;

cout << endl << "Введите индекс, в который вставить ключ поиска: ";

cin >> indexForKey;

cout << endl;

arr[indexForKey] = key;

}

int indexToFound;

auto startTime = chrono::steady\_clock::now();

indexToFound = BLS(arr, length, key);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "\nЭлемент не найден в поиске BLS.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

else {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Результат работы алгоритма поиска BLS: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

startTime = chrono::steady\_clock::now();

indexToFound = SLS(arr, length, key);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Элемент не найден в поиске SLS.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

else {

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Результат работы алгоритма поиска SLS: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

startTime = chrono::steady\_clock::now();

InsSort(arr, length);

auto endTime = chrono::steady\_clock::now();

auto time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Время сортировки: " << time.count() << " мкс.\n";

startTime = chrono::steady\_clock::now();

indexToFound = T(arr, length, key);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Элемент не найден в поиске T.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

else {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Результат работы алгоритма поиска T: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

startTime = chrono::steady\_clock::now();

indexToFound = B(arr, length, key);

if (indexToFound == -1) {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Элемент не найден в поиске B.\n" << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

else {

auto endTime = chrono::steady\_clock::now();

time = chrono::duration\_cast<chrono::microseconds>(endTime-startTime);

cout << "Результат работы алгоритма поиска B: " << indexToFound << endl << "\tАлгоритм выполнился за " << time.count() << " мкс.\n";

}

}

# Таблицы зависимости времени работы (в мкс) алгоритмов поиска от количества элементов массива

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BLS | SLS | T | B |
| 10000 | 10 мкс | 7 мкс | 9 мкс | 0 мкс |
| 15000 | 16 мкс | 10 мкс | 13 мкс | 0 мкс |
| 25000 | 27 мкс | 16 мкс | 22 мкс | 0 мкс |
| 40000 | 41 мкс | 25 мкс | 36 мкс | 0 мкс |
| 65000 | 67 мкс | 42 мкс | 58 мкс | 0 мкс |
| 90000 | 95 мкс | 57 мкс | 83 мкс | 0 мкс |
| 110000 | 112 мкс | 70 мкс | 98 мкс | 0 мкс |
| 140000 | 142 мкс | 98 мкс | 124 мкс | 0 мкс |
| 170000 | 174 мкс | 108 мкс | 151 мкс | 1 мкс |
| 200000 | 206 мкс | 131 мкс | 183 мкс | 1 мкс |

## При отсутствии искомого элемента в массиве (минимальное значение за три итерации)

## Искомый элемент 64001 в первой половине массива

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BLS | SLS | T | B |
| 10000 | 1 мкс | 1 мкс | 9 мкс | 0 мкс |
| 15000 | 3 мкс | 1 мкс | 12 мкс | 0 мкс |
| 25000 | 6 мкс | 3 мкс | 15 мкс | 0 мкс |
| 40000 | 11 мкс | 7 мкс | 14 мкс | 1 мкс |
| 65000 | 21 мкс | 13 мкс | 58 мкс | 0 мкс |
| 90000 | 31 мкс | 19 мкс | 79 мкс | 1 мкс |
| 110000 | 37 мкс | 23 мкс | 98 мкс | 1 мкс |
| 140000 | 47 мкс | 28 мкс | 128 мкс | 1 мкс |
| 170000 | 75 мкс | 45 мкс | 147 мкс | 1 мкс |
| 200000 | 85 мкс | 54 мкс | 178 мкс | 0 мкс |

## Искомый элемент 157 в первой половине массива

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BLS | SLS | T | B |
| 10000 | 1 мкс | 1 мкс | 0 мкс | 0 мкс |
| 15000 | 3 мкс | 1 мкс | 1 мкс | 0 мкс |
| 25000 | 6 мкс | 2 мкс | 2 мкс | 0 мкс |
| 40000 | 9 мкс | 7 мкс | 1 мкс | 1 мкс |
| 65000 | 18 мкс | 11 мкс | 2 мкс | 0 мкс |
| 90000 | 31 мкс | 19 мкс | 3 мкс | 1 мкс |
| 110000 | 37 мкс | 22 мкс | 1 мкс | 1 мкс |
| 140000 | 42 мкс | 29 мкс | 1 мкс | 1 мкс |
| 170000 | 73 мкс | 40 мкс | 3 мкс | 1 мкс |
| 200000 | 81 мкс | 52 мкс | 2 мкс | 0 мкс |

## Искомый элемент 64001 во второй половин массива

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BLS | SLS | T | B |
| 10000 | 9 мкс | 5 мкс | 21 мкс | 0 мкс |
| 15000 | 28 мкс | 8 мкс | 30 мкс | 0 мкс |
| 25000 | 21 мкс | 13 мкс | 23 мкс | 0 мкс |
| 40000 | 38 мкс | 23 мкс | 36 мкс | 1 мкс |
| 65000 | 61 мкс | 43 мкс | 57 мкс | 1 мкс |
| 90000 | 86 мкс | 56 мкс | 80 мкс | 1 мкс |
| 110000 | 101 мкс | 65 мкс | 103 мкс | 0 мкс |
| 140000 | 144 мкс | 102 мкс | 132 мкс | 1 мкс |
| 170000 | 166 мкс | 105 мкс | 151 мкс | 1 мкс |
| 200000 | 246 мкс | 124 мкс | 180 мкс | 0 мкс |

## Искомый элемент 157 во второй половине массива

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | BLS | SLS | T | B |
| 10000 | 8 мкс | 5 мкс | 0 мкс | 0 мкс |
| 15000 | 14 мкс | 9 мкс | 0 мкс | 0 мкс |
| 25000 | 26 мкс | 16 мкс | 0 мкс | 0 мкс |
| 40000 | 34 мкс | 22 мкс | 0 мкс | 0 мкс |
| 65000 | 62 мкс | 41 мк | 1 мкс | 0 мкс |
| 90000 | 83 мкс | 57 мкс | 0 мкс | 1 мкс |
| 110000 | 99 мкс | 63 мкс | 1 мкс | 0 мкс |
| 140000 | 142 мкс | 98 мкс | 1 мкс | 0 мкс |
| 170000 | 157 мкс | 104 мкс | 1 мкс | 1 мкс |
| 200000 | 234 мкс | 123 мкс | 1 мкс | 1 мкс |

# Вывод

Я изучил основные принципы работы алгоритмов поиска BLS – последовательный поиск, SLS – быстрый последовательный поиск, T – последовательный поиск в упорядоченном массиве и B – бинарный поиск, а также исследовал их свойства. Работа ведется с целочисленным массивом. Для вычисления времени используется библиотека chrono.