

МИНИСТЕРСТВО науки и высшего ОБРАЗОВАНИЯ РОссИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ 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»

Программирование

Отчет по лабораторной работе №2

по теме: «Алгоритмы поиска»

Выполнили студенты группы М3О-211Б-21

Мороз А.И.

Гордеева М.Д.

Проверила:

доцент каф. 304, к.т.н. Дмитриева Е.А.

Москва 2023 г.

Содержание

[Задание 3](#_Toc151156255)

[Код программы 3](#_Toc151156256)

[Результаты работы программы 6](#_Toc151156257)

[Графики временных характеристик поиска 7](#_Toc151156258)

[Вывод 12](#_Toc151156259)

# Задание

Оценить длительность работы алгоритмов поиска (BLS, SLS, OAS и BS). Для десяти разных размерностей рассмотреть случаи нахождения ключа поиска в начале, середине и конце массива. Также необходимо определить сколько раз выполняются операции сравнения.

# Код программы

#include <iostream>

#include <fstream>

#include <stdlib.h>

#include <chrono>

#include <climits>

using namespace std;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Г Л О Б А Л Ь Н Ы Е К О Н С Т А Н Т Ы \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

const char\* int\_array\_filename[2] = { "array1", "array2" }; // массив имен файлов, в которые будут записаны массивы

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* П Р О Т О Т И П Ы Ф У Н К Ц И Й \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//запись элементов последовательностей в файлы

void Output(int\* array, int size, int index);

// последовательный поиск (Better\_Linear\_Search)

int Better\_Linear\_Search(int\* array1, int size, int x);

// быстрый последовательный поиск (Sentinel\_ Linear\_Search)

int Sentinel\_Linear\_Search(int\* array1, int size, int x);

// последовательный поиск в упорядоченном массиве (Ordered\_Array\_Search)

int Ordered\_Array\_Search(int\* array2, int size, int x);

// бинарный поиск ( Binary Search)

int Binary\_Search(int\* array2, int size, int x);

int main(void)

{

setlocale(LC\_ALL, "Russian"); //подключение русского языка

system("color F0");

string FuncName[6] = { "Better\_Linear\_Search", "Sentinel\_Linear\_Search",

"Ordered\_Array\_Search", "Binary\_Search" }; //массив названий функций

int size; // размерность массивов

cout << "Введите размерность для массивов: ";

cin >> size;

//первый массив - случайная последовательность (для "Better\_Linear\_Search", "Sentinel\_Linear\_Search")

int\* array1 = new int[size];

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

{

if (i == size / 2)

{

array1[i] = -1;

}

else if (i == size - 1)

{

array1[i] = -2;

}

else

{

array1[i] = 2 + rand() % 200000;

}

}//for

Output(array1, size, 0); // запись массива 1 в файл

// второй массив - возрастающая последовательность (для "Ordered\_Array\_Search", "Binary\_Search")

int\* array2 = new int[size];

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

{

array2[i] = 2 + ((200002 - 2) / size) \* i; // создание последовательности (минимальное значение - 2, максимальное - 200000)

}

Output(array2, size, 1); // запись массива 2 в файл

int (\*int\_func[])(int\*, int, int) = { Better\_Linear\_Search,

Sentinel\_Linear\_Search, Ordered\_Array\_Search,

Binary\_Search }; // массив указателей на функции

string IndexName[6] = { "начале", "середине", "конце" }; //массив местонахождения индексов

int indexes\_of\_x[3] = { 0, size / 2, size - 1 }; // массив с индексами для ключа поиска - x

cout << endl;

cout << " Поиск элемента" << endl;

cout << endl;

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*BLS и SLS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

for (int i = 0; i < 2; i++) // для "Better\_Linear\_Search", "Sentinel\_Linear\_Search",

{

for (int j = 0; j < 3; j++)

{

cout << "В " << IndexName[j] << " массива" << endl;

//cout << "X = " << array1[indexes\_of\_x[j]] << endl;

auto begin = chrono::steady\_clock::now(); // фиксирование начального времени работы алгоритма

int result = int\_func[i](array1, size, array1[indexes\_of\_x[j]]); // обращение к i-й функции из массива указателей

auto end = chrono::steady\_clock::now(); // фиксирование конечного времени

auto elapsed\_ms = chrono::duration\_cast<chrono::microseconds>(end - begin); // получение времени работы в микросекундах

cout << "Время работы алгоритма " << FuncName[i] << ": " << elapsed\_ms.count() << " (мкС)" << endl; //вывод времени работы

cout << endl;

}//for

}//for

delete[] array1; // очистка памяти

cout << "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OAS и BS\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*" << endl;

for (int i = 2; i < 4; i++) // для "Ordered\_Array\_Search", "Binary\_Search"

{

for (int j = 0; j < 3; j++)

{

cout << "В " << IndexName[j] << " массива" << endl;

//cout << "X = " << array2[indexes\_of\_x[j]] << endl;

auto begin = chrono::steady\_clock::now(); // фиксирование начального времени работы алгоритма

int result = int\_func[i](array2, size, array2[indexes\_of\_x[j]]); // обращение к i-й функции из массива указателей

auto end = chrono::steady\_clock::now(); // фиксирование конечного времени

auto elapsed\_ms = chrono::duration\_cast<chrono::microseconds>(end - begin); // получение времени работы в микросекундах

cout << "Время работы алгоритма " << FuncName[i] << ": " << elapsed\_ms.count() << " (мкС)" << endl; //вывод времени работы

cout << endl;

}//for

}//for

delete[] array2; // очистка памяти

return 0;

}//main

void Output(int\* array, int size, int index) // запись элементов последовательностей в файлы

{

ofstream File; // инициализация файловой переменной для записи

File.open(int\_array\_filename[index]); // открытие файла по имени из массива имён

// запись элементов последовательности в файлы

for (int j = 0; j < size; j++)

{

File << array[j] << endl;

}//for j

File.close(); // закрытие файла

}//Output

int Better\_Linear\_Search(int\* array1, int size, int x)

{

int comparisons1 = 0; // счетчик сравнений ключа с элементом массива

int comp\_i = 0; // счетчик сравнений в цикле

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

{

comparisons1++;

comp\_i++; // сравнения в цикле

if (array1[i] == x)

{

comparisons1++;

cout << "Индекс искомого элемента = " << i << endl;

cout << "Кол-во сравнений в цикле в Better\_Linear\_Search: " << comp\_i << endl;

cout << "Кол-во сравнений ключа с элементом массива в Better\_Linear\_Search: " << comparisons1 << endl;

return i; // элемент найден, возвращение его индекса

}//if

}//for

cout << "Кол-во сравнений в цикле в Better\_Linear\_Search: " << comp\_i << endl;

cout << "Кол-во сравнений ключа с элементом массива в Better\_Linear\_Search: " << comparisons1 << endl;

return -1; // элемент не найден

}//Better\_Linear\_Search

int Sentinel\_Linear\_Search(int\* array1, int size, int x)

{

int comparisons2 = 0; // счетчик сравнений ключа с элементом массива

int last = array1[size]; // сохранение последнего э-та в last

array1[size] = x; // помещение x

while (array1[i] != x)

{

i++;

comparisons2++;

if (array1[i] == x)

{

array1[size] = last;

cout << "Индекс искомого элемента = " << i << endl;

cout << "Кол-во сравнений ключа с элементом массива в Sentinel\_Linear\_Search: " << comparisons2 << endl;

return i; // элемент найден, возвращение его индекса

}//if

}//for

cout << "Кол-во сравнений ключа с элементом массива в Sentinel\_Linear\_Search: " << comparisons2 << endl;

return -1; // элемент не найден

}//Sentinel\_Linear\_Search

int Ordered\_Array\_Search(int\* array2, int size, int x)

{

int last = array2[size]; // сохранение последнего э-та в last

array2[size] = LONG\_MAX; // заведомо большое значение

int i = 0;

while (x > array2[i]) {

i++;

}

array2[size] = last; // восстановление последнего элемента

if (x == array2[i]) {

cout << "Индекс искомого элемента = " << i << endl;

return i; // элемент найден, возвращение его индекса

}

if (x == array2[size]) {

cout << "Индекс искомого элемента = " << size << endl;

return size; // возвращение размерности массива

}

return -1; // элемент не найден

}//Ordered\_Array\_Search

int Binary\_Search(int\* array2, int size, int x)

{

int p = 0;

int r = size - 1;

while (p <= r) {

int q = (p + r) / 2; // середина интервала

if (array2[q] == x) {

cout << "Индекс искомого элемента = " << q << endl;

return q; // элемент найден, возвращение его индекса

}

else if (array2[q] > x) {

r = q - 1; // сужение интервала справа

}

else {

p = q + 1; // сужение интервала слева

}

}//while

return -1; // элемент не найден

}//Binary\_Search

# Результаты работы программы

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Местоположение** | **Размерность** | **Сравнения** | | | **Время (мкС)** | | | |
| **в цикле в BLS** | **ключа с эл-ом массива в BLS** | **ключа с эл-ом массива в SLS** | **BLS** | **SLS** | **OAS** | **BS** |
| Начало | 10000 | 1 | 2 | 1 | 2388 | 1536 | 657 | 665 |
| Середина | 5001 | 10002 | 5001 | 2546 | 2481 | 729 | 727 |
| Конец | 10000 | 20000 | 10000 | 3410 | 2317 | 770 | 725 |
| Начало | 25000 | 1 | 2 | 1 | 2173 | 1516 | 463 | 504 |
| Середина | 12501 | 25002 | 12501 | 3642 | 1739 | 600 | 591 |
| Конец | 25000 | 50000 | 25000 | 2604 | 2741 | 641 | 583 |
| Начало | 50000 | 1 | 2 | 1 | 2504 | 1523 | 443 | 544 |
| Середина | 25001 | 50002 | 25001 | 2965 | 1762 | 598 | 569 |
| Конец | 50000 | 100000 | 50000 | 2737 | 2020 | 627 | 565 |
| Начало | 75000 | 1 | 2 | 1 | 2236 | 1534 | 497 | 455 |
| Середина | 37501 | 75002 | 37501 | 2905 | 2124 | 687 | 599 |
| Конец | 75000 | 150000 | 75000 | 2702 | 1830 | 780 | 567 |
| Начало | 90000 | 1 | 2 | 1 | 2655 | 1517 | 455 | 479 |
| Середина | 45001 | 90002 | 45001 | 3388 | 1883 | 823 | 690 |
| Конец | 90000 | 180000 | 90000 | 2777 | 1859 | 709 | 586 |
| Начало | 100000 | 1 | 2 | 1 | 2270 | 1531 | 462 | 464 |
| Середина | 50001 | 100002 | 50001 | 3297 | 1819 | 655 | 695 |
| Конец | 100000 | 200000 | 100000 | 2825 | 2009 | 824 | 597 |
| Начало | 125000 | 1 | 2 | 1 | 2462 | 1524 | 486 | 476 |
| Середина | 62501 | 125002 | 62501 | 2774 | 1845 | 662 | 586 |
| Конец | 125000 | 250000 | 125000 | 3540 | 1941 | 828 | 614 |
| Начало | 150000 | 1 | 2 | 1 | 2645 | 1553 | 453 | 558 |
| Середина | 75001 | 150002 | 75001 | 3432 | 1847 | 858 | 582 |
| Конец | 150000 | 300000 | 150000 | 3583 | 2375 | 799 | 613 |
| Начало | 175000 | 1 | 2 | 1 | 2632 | 1526 | 484 | 529 |
| Середина | 87501 | 175002 | 87501 | 2895 | 2063 | 706 | 638 |
| Конец | 175000 | 350000 | 175000 | 3473 | 2031 | 839 | 673 |
| Начало | 200000 | 1 | 2 | 1 | 2462 | 1545 | 496 | 445 |
| Середина | 100001 | 200002 | 100001 | 3097 | 2160 | 794 | 599 |
| Конец | 200000 | 400000 | 200000 | 3046 | 2171 | 886 | 662 |

Таблица 1 – результаты выполнения программы

# Графики временных характеристик поиска

Рисунок 1. График зависимости времени от размерности случайной целочисленной последовательности для функции BLS по поиску ключа в начале последовательности

Рисунок 2. График зависимости времени от размерности случайной целочисленной последовательности для функции BLS по поиску ключа в середине последовательности

Рисунок 3. График зависимости времени от размерности случайной целочисленной последовательности для функции BLS по поиску ключа в конце последовательности

Рисунок 4. График зависимости времени от размерности случайной целочисленной последовательности для функции SLS по поиску ключа в начале последовательности

Рисунок 5. График зависимости времени от размерности случайной целочисленной последовательности для функции SLS по поиску ключа в середине последовательности

Рисунок 6. График зависимости времени от размерности случайной целочисленной последовательности для функции SLS по поиску ключа в конце последовательности

Рисунок 7. График зависимости времени от размерности возрастающей целочисленной последовательности для функции OAS по поиску ключа в начале последовательности

Рисунок 8. График зависимости времени от размерности возрастающей целочисленной последовательности для функции OAS по поиску ключа в середине последовательности

Рисунок 9. График зависимости времени от размерности возрастающей целочисленной последовательности для функции OAS по поиску ключа в конце последовательности

Рисунок 10. График зависимости времени от размерности возрастающей целочисленной последовательности для функции BS по поиску ключа в начале последовательности

Рисунок 11. График зависимости времени от размерности возрастающей целочисленной последовательности для функции BS по поиску ключа в середине последовательности

Рисунок 12. График зависимости времени от размерности возрастающей целочисленной последовательности для функции BS по поиску ключа в конце последовательности

# Вывод

Программа работает корректно на основе проделанных тестов. Были изучены алгоритмы поиска такие, как последовательный поиск – BLS, быстрый последовательный поиск – SLS, последовательный поиск в упорядоченном массиве – OAS и бинарный поиск – BS. Для подсчета времени были использованы функции, позволяющие производить оценку длительности времени выполнения алгоритма, с помощью библиотеки <chrono>.