ОТЧЁТ

О ЛАБОРАТОРНОЙ РАБОТЕ №5  
по теме: Алгоритмы поиска

по дисциплине: Разработка программных модулей

09.02.07 Информационные системы и программирование

Выполнил:

студент группы Y2333

\_\_\_\_\_\_\_\_\_\_\_Харченко Д.И.

Проверил:

\_\_\_\_\_\_\_\_ Антонов М.Б.

Дата: «\_\_» \_\_\_\_\_ 20\_\_ г.

Оценка: \_\_\_\_\_\_\_\_\_\_

УНИВЕРСИТЕТ ИТМО

Факультет среднего профессионального образования

Санкт-Петербург 2019

1. ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Цель работы: Реализовать и замерить скорость работы алгоритмов поиска на больших структурах данных.

Общее ограничение на лабораторную работу:

* использовать разбитие на файлы основной программы (в main.cpp только функция main.cpp);
* документировать весь код используя аннотации Doxygen.

Требования к исходной программе:

1. Изначально в программе размер массива задан как 1.000.000 элементов.
2. Все элементы должны быть от [0, 6.000.000).
3. Массив для поиска генерируется динамически случайно.
4. Генерируется массив из 40 элементов (или предоставить ввод этих элементов пользователем). Элементы должны быть от [0, 6.000.000).
5. Вывод программы: алгоритм поиска и сколько времени занял поиск в секундах.

Требования к лабораторной:

1. Выбрать один алгоритм модификации метода перебора и один алгоритм поиска в сортированном массиве и реализовать его в исходной программе.
2. За один запуск программа должна выполнить поиск всех 40 элементов и вывести результаты поиска в консоль.
3. Замеряется время поиска массиве всех 40 элементов и результат выводится в конце поиска.
4. Замерять только время выполнения сортировки, а не само время выполнения программы.
5. Проводится оценка сложности реализованных алгоритмов сортировки.

Код main.cpp:

#include <iostream>

#include <fstream>

#include <time.h>

#include "utils.h"

#include "algoth.h"

#include "longQSort.h"

#define SZ 1000000 ///< размер массива, в котором будем искать

#define tSZ 40 ///< размер массива, элементы которого нужно найти

using namespace std;

int main(int argc, const char \* argv[]) {

cout << "Выберите операцию: 1 – Cгенерировать массив в файл и завершить, 0 – Выполнить поиск в файле" << endl;

int ch = 1;

cin >> ch;

if (ch == 1) { ///< только генерация большого массива и запись в файл

ofstream arrayFileO;

arrayFileO.open("/Users/anponew/Desktop/ИТМО/Разработка программных модулей/Лаба 5/Лаба 5/Data.txt");

if (!arrayFileO.is\_open()) {

cout << "ttt" << endl;

return 32;

}

long \*dat = genArray(SZ);

fillArray(dat, SZ);

toFile(&arrayFileO, dat, SZ);

arrayFileO.close();

}

if (ch == 0) { ///< осуществление двух поисков

ifstream arrayFileI;

arrayFileI.open("/Users/anponew/Desktop/ИТМО/Разработка программных модулей/Лаба 5/Лаба 5/Data.txt");

if (!arrayFileI.is\_open()) {

cout << "ttt" << endl;

return 32;

}

long \*array = genArray(SZ);

fromFile(&arrayFileI, array, SZ);

long \*targetArray = genArray(tSZ);

fillArray(targetArray, tSZ);

cout << " – Поиск методом перемещения в начало – " << endl; ///< первое задание

clock\_t stat = clock(); ///< старт таймера

for (long i = 0; i < tSZ; i++) {

long place = uSearch(array, SZ, targetArray[i]); ///< применение к каждому элементу

if (place != -1) {

cout << targetArray[i] << " найден | Индекс: " << place << endl;

} else {

cout << targetArray[i] << " не найден" << endl;

}

}

clock\_t fin = clock(); ///< финиш таймера

cout << "Время выполнения: "<< (fin - stat) << " clock" << endl << endl;

cout << " – Поиск методом половинного деления – " << endl; ///< второе задание

clock\_t statSort = clock(); ///< старт таймера с учётом сортировки

quickSort(array, 0, SZ - 1);

stat = clock(); ///< старт таймера

for (long i = 0; i < tSZ; i++) {

long place = bSearch(array, 0, SZ - 1, targetArray[i]);

if (place != -1) {

cout << targetArray[i] << " найден | Индекс: " << place << endl;

} else {

cout << targetArray[i] << " не найден" << endl;

}

}

fin = clock(); ///< финиш таймера

cout << "Время выполнения без учёта сортировки: "<< (fin - stat) << " clock" << endl;

cout << "Время выполнения с учётом сортировки: "<< (fin - statSort) << " clock" << endl << endl;

arrayFileI.close();

}

return 0;

}

Код utils.h:

#ifndef utils\_h

#define utils\_h

#define rng 6000000 ///< область значений случайно сгенерированных чисел

/\*\*

Функция генерации массива

@param size необходимый размер массива

@return готовый массив

\*/

long\* genArray(long size) {

long\* array = new long[size];

return array;

}

/\*\*

Функция заполнения массива случаныйми числами

@param array массив

@param size размер массива

\*/

void fillArray(long \*array, long size) {

srand(unsigned(time(0)));

for(long i = 0; i < size; i++) {

array[i] = rand() % rng;

}

}

/\*\*

Функция записи массива в указанный файл

@param destination поток для выходного файла

@param array массив

@param size размер массива

\*/

void toFile(std::ofstream \*destination, long \*array, long size) {

for (long i = 0; i < size; i++) {

(\*destination) << array[i] << ' ';

}

}

/\*\*

Функция чтения массива из файла

@param destination поток файла, из которого осуществляется чтение

@param array массив

@param size размер массива

\*/

void fromFile(std::ifstream \*destination, long \*array, long size) {

for (long i = 0; i < size; i++) {

(\*destination) >> array[i];

}

}

/\*\*

Функция вывода массива в консоль

@param array массив

@param size размер массива

\*/

void toConsele(long \*array, long size) {

std::cout << std::endl;

for(long i = 0; i < size; i++) {

std::cout << array[i] << ' ';

}

std::cout << std::endl;

}

#endif /\* utils\_h \*/

Код algoth.h:

#ifndef algoth\_h

#define algoth\_h

/\*\*

Функция поиска элемента в неотсортированном массиве

Метод перемещения в начало

@param array массив, в котором ищем

@param size размер массива

@param target элемент, который ищем

@return индекс найденного элемента

@details вернёт -1, если элемент не найден

\*/

long uSearch(long \*array, long size, long target) {

long pos = -1;

for (long i = 0; i < size; i++) {

if (array[i] == target) {

std::swap(array[0], array[i]);

pos = i;

break;

}

}

return pos;

}

/\*\*

Функция бинарного поиска в массиве

Метод половинного деления

@param array массив, в котором ищем

@param left левый индекс

@param right правый индекс

@param target элемент, который хотим найти

@return возврат индекса элемента

@details вернет -1, если элемента нет в массиве

\*/

long bSearch(long \*array, long left, long right, long target) {

if (right >= left) {

long pivot = (left + right) / 2;

if (array[pivot] == target) {

return pivot;

}

if (array[pivot] > target) {

return bSearch(array, left, pivot - 1, target);

}

return bSearch(array, pivot + 1, right, target);

}

return -1;

}

#endif /\* algoth\_h \*/

Код longQSort.h:

#ifndef longQSort\_h

#define longQSort\_h

#include <math.h>

/\*\*

Функция быстрой сортировки по возрастанию элементов массива.

Алгоритм состоит из трёх шагов:

Выбрать элемент из массива. Назовём его ключевым.

Разбиение: перераспределение элементов в массиве таким образом, что элементы меньше ключевого помещаются перед ним, а больше или равные после.

Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента.

@param array массив данных

@param low левый индекс

@param high правый индекс

\*/

void quickSort(long \*array, long low, long high) {

long i = low, j = high;

long pivot = array[(low + high) / 2];

/// перемещение элементов относительно ключевого элемента

while (i <= j) {

while (array[i] < pivot)

i++;

while (array[j] > pivot)

j--;

if (i <= j) {

std::swap(array[i], array[j]);

i++;

j--;

}

}

/// рекурсия функции

if (low < j)

quickSort(array, low, j);

if (i < high)

quickSort(array, i, high);

}

#endif /\* longQSort\_h \*/

Ниже представлена таблица сравнения алгоритмов поиска. Время работы указывается в clock из-за высокой скорости работы.

Таблица 1 – Сравнение времени работы алгоритмов поиска

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | O(f) |
| Метод  перемещения  в начало | | 74382 | 72787 | 76808 | 72002 | 75684 | 71297 | 74611 | 70913 | 73844 | 74852 |  |
| Бинарный поиск | Без  учета сорт. | 223 | 215 | 286 | 213 | 195 | 219 | 204 | 256 | 219 | 278 |  |
| С учётом | 146968 | 152104 | 151220 | 151230 | 156348 | 158275 | 152037 | 158176 | 148320 | 154090 |