**Министерство образования и науки Украины**

**Национальный технический университет Украины "Киевский политехнический институт имени Игоря Сикорского"**

**Факультет информатики и вычислительной техники**

**Кафедра автоматизированных систем обработки**

**информации и управления**

**ОТЧЕТ**

по лабораторной работе № 1 по дисциплине

«Проектирование и анализ вычислительных алгоритмов»

„ **Проектирование и анализ алгоритмов сортировки** ”

**Выполнил**

(шифр, фамилия, имя, отчество)

*ІП-*

**Проверил**

(фамилия, имя, отчество )

*Головченко М.Н.*

Киев 2018

Содержание

[1 Цель лабораторной работы 3](#_Toc509037345)

[2 Задание 4](#_Toc509037346)

[3 Выполнение 6](#_Toc509037347)

[3.1 Анализ алгоритма на соответствие свойствам 6](#_Toc509037348)

[3.2 Псевдокод алгоритма 6](#_Toc509037349)

[3.3 Анализ временной сложности 6](#_Toc509037350)

[3.4 Программная реализация алгоритма 6](#_Toc509037351)

[3.4.1 Исходный код 6](#_Toc509037352)

[3.4.2 Примеры работы 7](#_Toc509037353)

[3.5 Испытания алгоритма 8](#_Toc509037354)

[3.5.1 Временные оценочные характеристики 8](#_Toc509037355)

[3.5.2 Графики зависимости временных оценочных характеристик от размерности массива 10](#_Toc509037356)

[Выводы 11](#_Toc509037357)

[Критерии оценивания 12](#_Toc509037358)

# Цель лабораторной работы

Цель работы – изучить основные подходы к анализу вычислительной сложности алгоритмов внутренней сортировки и оценить порог их эффективности.

# Задание

Согласно варианту (таблица 2.1), выполнить анализ алгоритма внутренней сортировки на соответствие следующим свойствам:

* устойчивость;
* естественность поведения;
* основанность на сравнениях;
* потребности в дополнительной памяти (объем);
* потребности в знаниях о структуре данных.

Записать алгоритм внутренней сортировки при помощи псевдокода (или другого способа по выбору).

Провести анализ временной сложности в худшем, лучшем и среднем случае и записать временную сложность в асимптотических оценках.

Выполнить программную реализацию алгоритма на любом языке программирования с фиксацией временных оценочных характеристик (количество сравнений, количество перестановок, глубина рекурсивного углубления и пр. в зависимости от алгоритма).

Провести ряд испытаний алгоритма на массивах разной размерности (10, 100, 1000, 5000, 10000, 20000, 50000 элементов) и разных наборах входных данных (упорядоченный массив, обратно упорядоченный массив, массив случайных чисел) и построить графики зависимости временных оценочных характеристик от размерности массива, нанести на график асимптотические оценку худшего и лучшего случаев для сравнения.

Сделать обобщенный вывод по лабораторной работе.

Таблица 2.1 – Варианты алгоритмов

|  |  |
| --- | --- |
| **№** | **Алгоритм сортировки** |
| 1 | Сортировка перемешиванием |
| 2 | Сортировка расчёской |
| 3 | Сортировка выбором |
| 4 | Сортировка Шелла (классическая) |
| 5 | Сортировка Шелла (*d* = последовательность Хиббарда) |
| 6 | Сортировка Шелла (*d* = последовательность Седжвика) |
| 7 | Сортировка Шелла (*d* = последовательность Пратта) |
| 8 | Сортировка Шелла (*d* = последовательность Марцина Циура) |
| 9 | Сортировка Шелла (*d* = последовательность Фибоначчи) |
| 10 | Сортировка Шелла (*d* = (3j−1)≤n, j ∈ N) |
| 11 | Быстрая сортировка (разбиение Ломуто) |
| 12 | Быстрая сортировка (разбиение Хоара) |
| 13 | Быстрая сортировка (повторяющиеся элементы) |
| 14 | Пирамидальная сортировка |
| 15 | Плавная сортировка |
| 16 | Интроспективная сортировка |
| 17 | Сортировка с помощью двоичного дерева |
| 18 | Сортировка перемешиванием |
| 19 | Сортировка Шелла (классическая) |
| 20 | Сортировка Шелла (*d* = последовательность Хиббарда) |
| 21 | Сортировка Шелла (*d* = последовательность Седжвика) |
| 22 | Сортировка Шелла (*d* = последовательность Пратта) |
| 23 | Сортировка Шелла (*d* = последовательность Марцина Циура) |
| 24 | Сортировка Шелла (*d* = последовательность Фибоначчи) |
| 25 | Сортировка с помощью двоичного дерева |
| 26 | Быстрая сортировка (разбиение Ломуто) |
| 27 | Быстрая сортировка (разбиение Хоара) |
| 28 | Быстрая сортировка (повторяющиеся элементы) |
| 29 | Пирамидальная сортировка |
| 30 | Плавная сортировка |

# Выполнение

## Анализ алгоритма на соответствие свойствам

Анализ алгоритма сортировки вставками на соответствие свойствам приведен в таблице 3.1.

Таблица 3.1 – Анализ алгоритма на соответствие свойствам

|  |  |
| --- | --- |
| **Свойство** | **Сортировка вставкими** |
| Устойчивость |  |
| Естественность поведения |  |
| Основанность на сравнениях |  |
| Потребности в дополнительной памяти (объем) |  |
| Потребности в знаниях о структуре данных |  |

## Псевдокод алгоритма

**for** j = 2 **to** A.length **do**

key = A[j]

i = j-1

**while** (i > 0 and A[i] > key) **do**

A[i + 1] = A[i]

i = i - 1

**end while**

A[i+1] = key

**end for**

## Анализ временной сложности

…

## Программная реализация алгоритма

### Исходный код

#include "stdafx.h"

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

void insertionSort(int \*, int); // прототип функции сортировки вставками

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

{

…

    system("pause");

    return 0;

}

void insertionSort(int \*arrayPtr, int length) // сортировка вставками

{

    …

}

### Примеры работы

На рисунках 3.1 и 3.2 показаны примеры работы программы сортировки массивов на 100 и 1000 элементов соответственно.

Рисунок 3.1 – Сортировка массива на 100 элементов

Рисунок 3.2 – Сортировка массива на 1000 элементов

## Испытания алгоритма

### Временные оценочные характеристики

В таблице 3.2 приведены оценочные характеристики числа сравнений и числа перестановок алгоритма сортировки вставками для массивов разной размерности, когда массивы содержат упорядоченную последовательность элементов.

Таблица 3.2 – Оценочные характеристики алгоритма сортировки вставками для упорядоченной последовательности элементов в массиве

|  |  |  |
| --- | --- | --- |
| Размерность массива | Число сравнений | Число перестановок |
| 10 |  |  |
| 100 |  |  |
| 1000 |  |  |
| 5000 |  |  |
| 10000 |  |  |
| 20000 |  |  |
| 50000 |  |  |

В таблице 3.3 приведены оценочные характеристики числа сравнений и числа перестановок алгоритма сортировки вставками для массивов разной размерности, когда массивы содержат обратно упорядоченную последовательность элементов.

Таблица 3.3 – Оценочные характеристики алгоритма сортировки вставками для обратно упорядоченной последовательности элементов в массиве

|  |  |  |
| --- | --- | --- |
| Размерность массива | Число сравнений | Число перестановок |
| 10 |  |  |
| 100 |  |  |
| 1000 |  |  |
| 5000 |  |  |
| 10000 |  |  |
| 20000 |  |  |
| 50000 |  |  |

В таблице 3.4 приведены оценочные характеристики числа сравнений и числа перестановок алгоритма сортировки вставками для массивов разной размерности, массивы содержат случайную последовательность элементов.

Таблица 3.4 – Оценочные характеристики алгоритма сортировки вставками для случайной последовательности элементов в массиве

|  |  |  |
| --- | --- | --- |
| Размерность массива | Число сравнений | Число перестановок |
| 10 |  |  |
| 100 |  |  |
| 1000 |  |  |
| 5000 |  |  |
| 10000 |  |  |
| 20000 |  |  |
| 50000 |  |  |

### Графики зависимости временных оценочных характеристик от размерности массива

На рисунке 3.3 показаны графики зависимости временных оценочных характеристик от размерности массива для случаев, когда массивы содержат упорядоченную последовательность элементов (зеленый график), когда массивы содержат обратно упорядоченную последовательность элементов (красный график), когда массивы содержат случайную последовательность элементов (синий график), также показаны асимптотические оценки худшего (фиолетовый график) и лучшего (желтый график) случаев для сравнения.

Рисунок 3.3 – Графики зависимости временных оценочных характеристик

Выводы

В рамках данной лабораторной работы…

Критерии оценивания

(удалить при сдаче)

При условии сдачи лабораторной работы до 2.04.2018 включительно максимальный балл равен – 5. После 2.04.2018 максимальный балл равен – 1.

Критерии оценивания в процентах от максимального балла:

* анализ алгоритма на соответствие свойствам – 10%;
* псевдокод алгоритма – 15%;
* анализ временной сложности – 25%;
* программная реализация алгоритма – 25%;
* испытания алгоритма – 20%;
* выводы – 5%.