

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«МИРЭА – Российский технологический университет»**

**РТУ МИРЭА**

|  |
| --- |
|  |

Институт информационных технологий

Кафедра математического обеспечения и стандартизации

информационных технологий (МОСИТ)

**ОТЧЕТ**

**ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

**по дисциплине**

«Структуры и алгоритмы обработки данных»

Выполнил студент группы ИКБО-01-21 Хитров Н. С.

Принял доцент кафедры МОСИТ, к.п.н. Рысин. М. Л.

Практическая работа выполнена «\_\_»\_\_\_\_\_\_\_2022 г.

«Зачтено» «\_\_»\_\_\_\_\_\_\_2022 г.

Москва 2022

Цель работы

Приобретение практических навыков по определению:

* сложности алгоритмов эмпирическим способом
* эффективного алгоритма решения задачи из нескольких
* емкостной и временной сложностей алгоритма и их зависимости от объема входных данных

Ход работы

Задание 1

Формулировка задания

Оценить эффективность простого алгоритма сортировки на массиве, заполненном случайными числами (в среднем случае). Используемый алгоритм – Insertion Sort.

Описание математической модели

Проходимся по всем элементам массива. Перемещаем каждый из перебираемых элементов по массиву следующим образом: если элемент больше предыдущего ему, то меняем эти элементы местами, выполняем проверку и перемещение элементов, пока перебираемый элемент не станет первым элементом массива или элемент слева от перебираемого не будет больше.

Блок-схема алгоритма

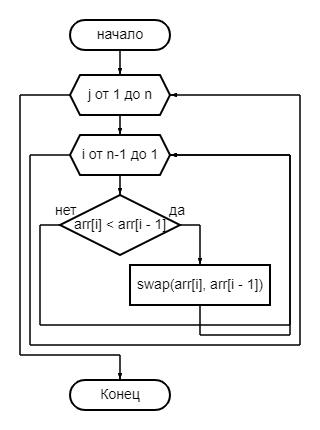


Рисунок 1 – блок-схема алгоритма сортировки

Код программы

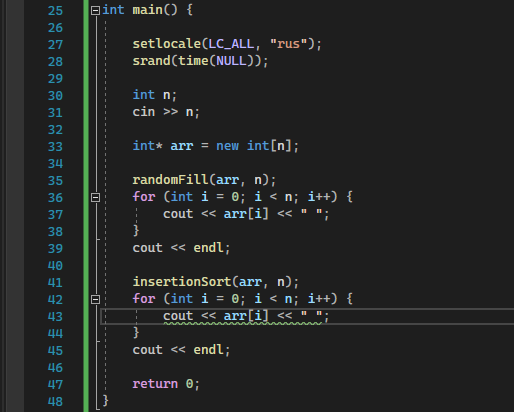


Рисунок 2 Функция main

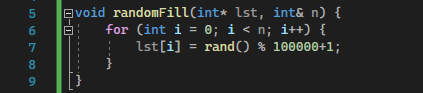


Рисунок 3 Функции заполнения массива

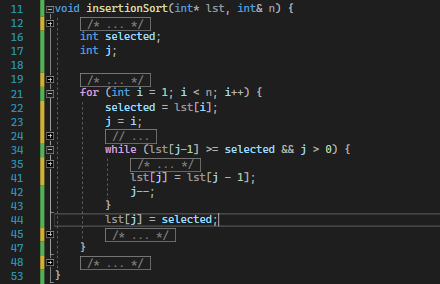


Рисунок 4 Алгоритм сортировки

Тестирование

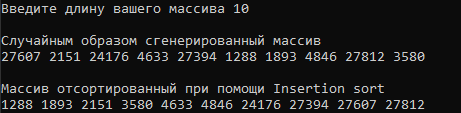


Рисунок 5 Тестирование и отладка программы на массиве из 10 элементов, сформированном значениями с клавиатуры

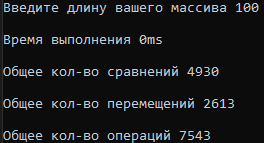


Рисунок 6 Тестирование при n = 100. Средний случай

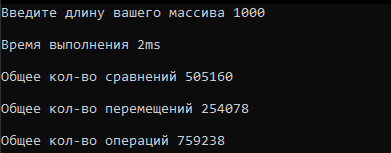


Рисунок 7 Тестирование при n = 1000. Средний случай

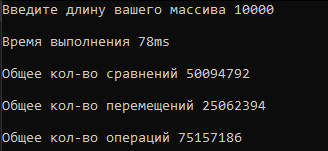


Рисунок 8 Тестирование при n = 10 000. Средний случай

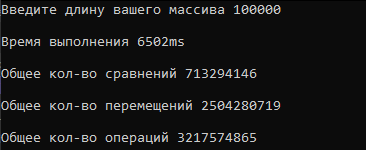


Рисунок 9 Тестирование при n = 100 000. Средний случай

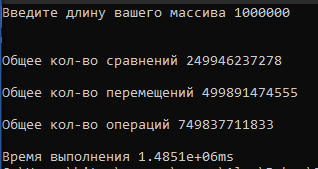


Рисунок 10 Тестирование при n = 1 000 000. Средний случай

Обработка результатов

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n) = Cф + Mф** |
| 100 | ≈0 | 7543 |
| 1 000 | 2 | 759 238 |
| 10 000 | 78 | 75 157 186 |
| 100 000 | 6 502 | 3 217 574 865 |
| 1 000 000 | 653 010 | 749 837 711 833 |

Таблица 1 – Insertion Sort, средний случай

Оценка емкостной сложности

Используется массив размером n для хранения элементов, а также 1 вспомогательная переменная, необходимая для реализации вставки. Емкостная сложность:

График зависимости сложности от n

Вывод

На основании данных, полученных в ходе работы, можно сделать следующий вывод: в среднем случае алгоритм обладает квадратичной вычислительной сложностью, что является показателем низкой эффективности, а также имеет линейную емкостную сложность, что для внутренней сортировки приемлемо.

Задание 2

Формулировка задания

Оценить эффективность алгоритма простой сортировки в случаях строгой упорядоченности по возрастанию и убыванию (условно лучший и худший случаи). Используемый алгоритм – Insertion Sort.

Вычислительная сложность алгоритма

|  |  |  |  |
| --- | --- | --- | --- |
| Номер оператора | Оператор | Кол-во  операций  в лучшем  случае | Кол-во  операций  в худшем  случае |
| 1 | insertionSort(int\* lst, int& n) { |  |  |
| 3 | for (int i = 1; i < n; i++) **do** | (n-1)+1 | (n-1)+1 |
| 4 | selected = lst[i]; | n-1 | n-1 |
| 5 | j = i; | n-1 | n-1 |
| 6 | while (lst[j-1] >= selected && j > 0) **do** | n-1 | n(n-1)/2 |
| 7 | lst[j] = lst[j - 1]; | 0 | n(n-1)/2 |
| 8 | j--; | 0 | n(n-1)/2 |
| 9 | **od** |  |  |
| 10 | lst[j] = selected; | n-1 | n-1 |
| 11 | **od** |  |  |
| 12 | } |  |  |

Общая вычислительная сложность алгоритма:

* В лучшем случае (массив отсортирован в возрастающем порядке): - 4
* В худшем случае (массив отсортирован в убывающем порядке):

Тестирование

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Обработка результатов

1. Лучший случай:

Таблица 2 – Insertion Sort, лучший случай

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n) = Cф + Mф** | **Тт(n) = С + М** |
| 100 | ≈0 |  |  |
| 1000 | ≈0 |  |  |
| 10 000 | ≈0 |  |  |
| 100 000 | ≈0 |  |  |
| 1 000 000 | 4 |  |  |

1. Худший случай:

Таблица 3 – Insertion Sort, худший случай

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n) = Cф + Mф** | **Тт(n) = С + М** |
| 100 | ≈0 |  |  |
| 1000 | 2 |  |  |
| 10 000 | 132 |  |  |
| 100 000 | 13 070 |  |  |
| 1 000 000 | 1 311 666 |  |  |

График зависимости сложности от n

График 2

Вывод по заданию 2

Полученные данные подтверждают, что сложность алгоритма сильно зависит от исходного состояния массива. Результаты теоретического расчета и практического выполнения совпали. В худшем случае сложность определяется формулой (квадратичная сложность), а в лучшем случае (линейная сложность). Средний случай по сложности ближе к худшему.

Задание 3

Формулировка задания

Сравнить эффективность двух алгоритмов простых сортировок. Второй используемый алгоритм – Selection Sort.

Описание математической модели

Проходим по всем индексам массива. На каждой итерации проходимся по всем элементам массива от данного до последнего и ищем максимальный. Меняем местами максимальный и данный элемент массива.

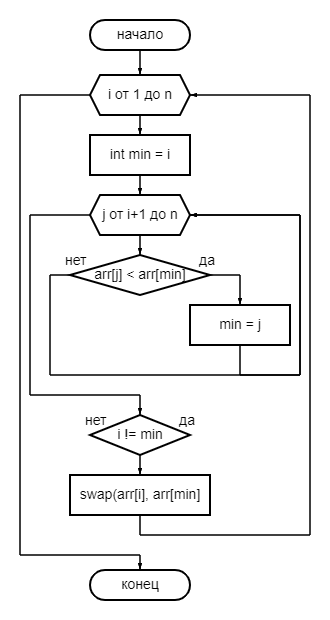
 Блок-схема алгоритма

Рисунок 25 Блок схема сортировки выбором

Реализация алгоритма

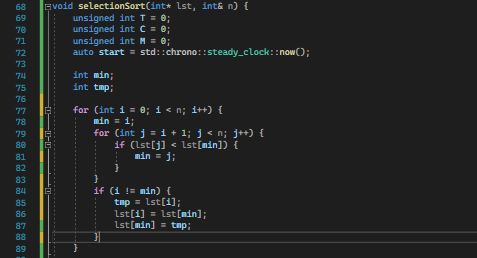
****

Рисунок 26 – Алгоритм сортировки

|  |  |
| --- | --- |
| Лучший случай | Худший случай |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Обработка результатов

Таблица 4 – Selection Sort, лучший случай

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n) = Cф + Mф** |
| 100 | 0,0114 | 10 301 |
| 1000 | 1,1089 | 1 003 001 |
| 10 000 | 100,391 | 100 030 001 |
| 100 000 | 10073,8 | 10 000 300 001 |
| 1 000 000 | 1.01408е+06 | 1 000 003 000 001 |

Таблица 5 – Selection Sort, худший случай

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Тп(n) = Cф + Mф** |
| 100 | 0,0174 | 12 951 |
| 1000 | 1,3003 | 1 254 501 |
| 10 000 | 114,827 | 125 045 001 |
| 100 000 | 11369 | 12 500 450 001 |
| 1 000 000 | 1.1362е+06 | 1 250 004 500 001 |

Оценка емкостной сложности

Используется массив размером n для хранения элементов, а также 1 вспомогательная переменная, необходимая для перестановок. Емкостная сложность – линейная:

Графики зависимости сложности от n

График 4

График 5

Вывод по заданию 3

Полученные данные подтверждают, что оба алгоритма обладают квадратичной вычислительной сложностью в худшем и среднем случаях, но на основе теоретического расчета можно установить, что функция роста первого алгоритма растет чуть быстрее. Алгоритм сортировки выбором в лучшем случае также имеет квадратичную сложность, а алгоритм сортировки вставками – линейную. Основываясь на графиках и таблицах, можно утверждать, что время выполнения зависит и от исходного состояния (упорядоченности) массива. Второй алгоритм уступает первому лишь в лучшем случае. Емкостная сложность у обоих алгоритмов – линейная.

Вывод

В ходе работы приобретены и отработаны практические навыки по:

* определению сложности алгоритмов сортировки на теоретическом и практическом уровнях;
* определению емкостной сложности алгоритмов сортировки;
* реализацией алгоритмов сортировки;
* определению зависимости (квадратичной/линейной) сложности алгоритма сортировки от объема входных данных;
* нахождению оптимального алгоритма сортировки с помощью данных, полученных в ходе теоретического расчета и практического выполнения.