

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

Отчет по выполнению практического задания № 2

**Тема:**

«Эмпирический анализ сложности простых алгоритмов сортировки»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Моисенко М. О.

Фамилия И.О.

Группа: ИКБО-00-22

Номер группы

Москва – 2023

СОДЕРЖАНИЕ

[1 ПОСТАНОВКА ЗАДАЧИ 3](#_Toc129854277)

[1.1 Условие задания 1 3](#_Toc129854278)

[1.2 Условие задания 2 4](#_Toc129854279)

[1.3 Условие задания 3 4](#_Toc129854280)

[2 РЕШЕНИЕ ЗАДАНИЯ 1 5](#_Toc129854281)

[2.1 Описание и реализация алгоритма Insertion Sort 5](#_Toc129854282)

[2.2 Анализ эффективности алгоритма 6](#_Toc129854283)

[2.3 Вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста 8](#_Toc129854284)

[3 РЕШЕНИЕ ЗАДАНИЯ 2 8](#_Toc129854285)

[4 РЕШЕНИЕ ЗАДАНИЯ 3 8](#_Toc129854286)

[4.1 Описание и реализация алгоритма Insertion Sort 8](#_Toc129854287)

[4.2 Анализ эффективности алгоритма 9](#_Toc129854288)

[4.3 Вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста 10](#_Toc129854289)

[5 ВЫВОД 13](#_Toc129854290)

[6 СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 13](#_Toc129854291)

# ПОСТАНОВКА ЗАДАЧИ

Цель:Актуализация знаний и приобретение практических умений по эмпирическому определению вычислительной̆ сложности алгоритмов.

## Условие задания 1

Оценить эмпирически вычислительную сложность алгоритма простой сортировки на массиве, заполненном случайными числами (средниӗ случай).

1. Составить функцию простой сортировки одномерного целочисленного массива A[n], используя алгоритм простого обмена («пузырек», Exchange sort). Провести тестирование программы на исходном массиве n=10.

2. Используя теоретический̆ подход, определить для алгоритма:
a. Что будет ситуациями лучшего, среднего и худшего случаев.
b. Функции роста времени работы алгоритма от объёма входа для лучшего и худшего случаев.

3. Провести контрольные прогоны программы массивов случайных чисел при n = 100, 1000, 10000, 100000 и 1000000 элементов с вычислением времени выполнения T(n) – (в миллисекундах/секундах). Полученные результаты свести в сводную таблицу 1.

4. Провести эмпирическую оценку вычислительной сложности алгоритма, для чего предусмотреть в программе подсчет фактического количества критических операций Тп как сумму сравнений Сп и перемещений Мп. Полученные результаты вставить в сводную таблицу 1.

5. Построить график функции роста Тп этого алгоритма от размера массива n. 6. Определить ёмкостную сложность алгоритма.

7. Сделать вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста.

Таблица 1 – Сводная таблица результатов

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **T(n)** | **TT = C + M** | **Tn = Cn + Mn** |
| 100 |  |  |  |
| 1000 |  |  |  |
| 10000 |  |  |  |
| 100000 |  |  |  |
| 1000000 |  |  |  |

## Условие задания 2

Оценить вычислительную сложность алгоритма простой сортировки в наихудшем и наилучшем случаях.

1. Провести дополнительные прогоны программы на массивах при n = 100, 1000, 10000, 100000 и 1000000 элементов, отсортированных:

a. строго в убывающем порядке значений, результаты представить в сводной̆ таблице по формату Таблицы 1;

b. строго в возрастающем порядке значений, результаты представить в сводной̆ таблице по формату Таблицы 1;

2. Сделать вывод о зависимости (или независимости) алгоритма сортировки от исходной упорядоченности массива.

## Условие задания 3

Сравнить эффективность алгоритмов простых сортировок

1. Выполнить разработку и программную реализацию второго алгоритма (Простая вставка – Insertion Sort).

2. Аналогично заданиям 1 и 2 сформировать таблицы с результатами эмпирического исследования второго алгоритма в среднем, лучшем и худшем случаях в соответствии с форматом Таблицы 1 (на тех же массивах, что и в заданиях 1 и 2).

3. Определить ёмкостную сложность алгоритма от n.

4. На одном сравнительном графике отобразить функции Тп(n) двух алгоритмов сортировки в худшем случае.

5. Аналогично на другом общем графике отобразить функции Тп(n) двух алгоритмов сортировки для лучшего случая.

6. Выполнить сравнительный̆ анализ полученных результатов для двух алгоритмов.

# РЕШЕНИЕ ЗАДАНИЯ 1

## Описание и реализация алгоритма Insertion Sort

Сортировка простого обмена (Exchange Sort) — это простой алгоритм сортировки, состоящий в повторяющихся проходах по сортируемому массиву. На каждой итерации последовательно сравниваются соседние элементы, и, если порядок в паре неверный, то элементы меняют местами.

Псевдокод:

N = length(A)

for i = 0 to N - 1 do

for j = 0 to N - 2 do

if (A[j] > A[j + 1])

 swap(A[j], A[j + 1])

 end

end

Инвариант цикла: после выполнения i-й операции все элементы массива, находящиеся в «верхней» части массива будут отсортированы. Размерность массива конечна, следовательно конечен и цикл.

Я реализовал алгоритм на языке C++, результат представлен на рис. 1.



Рисунок 1 – Реализация алгоритма «пузырьком» на C++

## Анализ эффективности алгоритма

Лучший случай: массив изначально отсортирован по возрастанию. Функция роста: T(n) = n - 1.

Средний случай: массив заполнен случайными числами. Функция роста: T(n) = 0,5n(n-1).

Худший случай: массив отсортирован по убыванию. Функция роста: T(n) = n(n-1).

В таблице 1 показаны теоретические и практические результаты вычисления количества совершённых операций и времени работы. На рис. 2 показана графическая визуализация зависимости количества операций от длины массива.

Таблица 2 – Сводная таблица результатов для алгоритма Exchange Sort

|  |  |  |
| --- | --- | --- |
| **n** | **T(n), мс** | **Tn = Cn + Mn** |
| 100 | 0,009 | 7132 |
| 1000 | 0,437 | 746144 |
| 10000 | 49,820 | 749214123 |
| 100000 | 10394,5 | 74964165183 |
| 1000000 | Невозможно измерить  | Невозможно измерить |

Рисунок 2 – Функция роста сортировки «пузырьком»

Ёмкостная сложность: O(n), так как программе необходимо хранить лишь сам массив из N элементов.

## Вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста

Сортировка простым обменом наиболее эффективна когда массив уже частично отсортирован и когда элементов массива не много. Судя по полученным результатам и графику функции роста, зависимость количества операций от длины массива – квадратичная.

# РЕШЕНИЕ ЗАДАНИЯ 2

Проверим работу алгоритма сортировки вставками на лучшем (T1) и худшем (T2) случае. В таблице 2 описаны полученные результаты.

Таблица 2 – Результаты работы алгоритма на разных входных данных

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **T1(n), мс** | **T2(n), мс** | **T2=С2+М2** | **T1=С1+М1** |
| 100 | 0,00002 | 0,003 | 9900 | 99 |
| 1000 | 0,0001 | 0,427 | 999000 | 999 |
| 10000 | 0,001 | 54,628 | 99990000 | 9999 |
| 100000 | 0,014 | 5260,59 | 9999900000 | 99999 |
| 1000000 | 0,302 | 537812 | 999999000000 | 999999 |

Чем меньше массив отсортирован, тем больше времени будет затрачено на сортировку данным алгоритмом.

# РЕШЕНИЕ ЗАДАНИЯ 3

## Описание и реализация алгоритма Insertion Sort

Сортировка вставками (Insertion Sort) — это простой алгоритм сортировки. Суть его заключается в том, что на каждом шаге алгоритма мы берем один из элементов массива, находим позицию для вставки и вставляем.

Псевдокод:

N = length(A)

for i = 1 to N - 1 do

j = i

while j > 0 and A[j-1] > A[j] do

 swap(A[j], A[j-1])

 j = j - 1

 end

end

Инвариант цикла: после выполнения i-й операции все элементы массива с позиции 0 до позиции i отсортированы. Размерность массива конечна, следовательно конечен и цикл.

Я реализовал алгоритм на языке C++, результат представлен на рис. 1.



Рисунок 3 – Реализация алгоритма сортировки вставками на C++

## Анализ эффективности алгоритма

Лучший случай: массив изначально отсортирован по возрастанию. Функция роста: T(n) = n - 1.

Средний случай: массив заполнен случайными числами. Функция роста: T(n) = 0,5n(n-1).

Худший случай: массив отсортирован по убыванию. Функция роста: T(n) = n(n-1).

В таблице 1 показаны теоретические и практические результаты вычисления количества совершённых операций и времени работы. На рис. 2 показана графическая визуализация зависимости количества операций от длины массива.

Таблица 3 – Сводная таблица результатов для алгоритма Insertion Sort

|  |  |  |
| --- | --- | --- |
| **n** | **T(n)** | **Tn = Cn + Mn** |
| 100 | 0,0016 | 6040 |
| 1000 | 0,107 | 545348 |
| 10000 | 10,187 | 6085127653 |
| 100000 | 1045,23 | 497743278328 |
| 1000000 | Невозможно измерить  | Невозможно измерить |

Ёмкостная сложность: O(n), так как программе необходимо хранить лишь сам массив.

## Вывод об эмпирической вычислительной сложности алгоритма на основе скорости роста функции роста

Судя по полученным результатам и графику функции роста, зависимость количества операций от длины массива – квадратичная.

Проверим работу алгоритма сортировки вставками на лучшем (T1) и худшем (T2) случае. В таблице 2 описаны полученные результаты.

Таблица 4 – Результаты работы алгоритма в лучшем и худшем случае

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **T1(n), мс** | **T2(n), мс** | **T2=С2+М2** | **T1=С1+М1** |
| 100 | 0,0000003 | 0,0015 | 9900 | 99 |
| 1000 | 0,000045 | 0,208 | 999000 | 999 |
| 10000 | 0,004 | 22,116 | 99990000 | 9999 |
| 100000 | 0,049 | 2146,94 | 9999900000 | 99999 |
| 1000000 | 0,53 | 251878 | 999999000000 | 999999 |

Чем меньше массив отсортирован, тем больше времени будет затрачено на сортировку данным алгоритмом.

Рисунок 4 – График сравнения худших случаев для сортировок

Рисунок 5 – График сравнения лучших случаев для сортировок

# ВЫВОД

В процессе выполнения работы мной были освоены навыки определения сложности алгоритма (емкостная и временная), реализации алгоритма на языке программирования и сравнение эффективности двух методов сортировок. Основываясь на рис. 4 и таблицах 2 и 4, я могу сделать вывод, что метод сортировки вставками является более эффективным, чем метод сортировки простого обмена.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Структуры данных и проектирование программ : Пер. с англ. / Р. Круз. — М.: БИНОМ. Лаборатория знаний, 2017. — 766 с.

2. Полный справочник по C++ : Пер. с англ. / Г. Шилдт. — М.: ООО "И.Д.Вильямс", 2016. — 796 с.: ил. — Предм. указ.: с. 787-796