МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФГБОУ ВО «Курганский государственный университет»

Кафедра «Программное обеспечение автоматизированных систем»

РФ КГУ 09.03.04.КР24.3600091 02

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Алгоритмы и структуры данных»**

**Пояснительная записка**

**АННОТАЦИЯ**

Проектируемая программная система предназначена для анализа качества алгоритмов сортировки данных, расположенных на внешнем носителе.

Основные задачи, решаемые разработчиком в процессе выполнения курсовой работы:

1. Программная реализация алгоритма сортировки данных, расположенных на внешнем носителе.
2. Проведение оценки качества реализованного алгоритма.
3. Программная реализация экспериментального исследования алгоритма и анализ его результатов.
4. Документирование проекта в соответствии с установленными требованиями.

Программа состоит из набора программных средств. Программа включает формы, описывающие программное и информационное обеспечение оптимальной сортировки массивов, сгенерированных случайно с помощью вероятностного распределительного закона Брэдфорда, методом Timsort.

Программный комплекс реализован с использованием языка программирования Microsoft Visual C++.

# **СОДЕРЖАНИЕ**

[**СОДЕРЖАНИЕ** 3](#_Toc183953776)

[**ВВЕДЕНИЕ** 4](#_Toc183953777)

[**1 АНАЛИТИЧЕСКИЙ ОБЗОР** 5](#_Toc183953778)

[**2 ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ** 6](#_Toc183953779)

# **ВВЕДЕНИЕ**

Необходимость сортировки данных возникает в программировании достаточно часто. Одной из важнейших задач является поиск эффективных методов сортировки для работы с большими массивами данных. В рамках данной курсовой работы рассматривается реализация программного приложения для сортировки массива методом Timsort.

Метод Timsort является адаптивным алгоритмом сортировки, который сочетает в себе элементы сортировки слиянием и сортировки вставками. Он был разработан для эффективной работы с реальными данными, в особенности с частично отсортированными массивами.

Для достижения цели необходимо решить следующие задачи:

1. Изучить теоретическое понятие сортировки и алгоритма сортировки.
2. Ознакомиться с критериями оценки эффективности алгоритмов сортировки
3. Изучить особенности алгоритма Timsort и его применения.
4. Разработать программу, реализующую сортировку массива данных методом Timsort.
5. Провести экспериментальное исследование и анализ эффективности алгоритма на различных данных, сгенерированных с помощью датчика случайных величин.

# **1 АНАЛИТИЧЕСКИЙ ОБЗОР**

Данное приложение формализует алгоритм сортировки массива данных методом Timsort. Программное приложение разработано с использованием технологии визуального проектирования и событийного программирования. Алгоритм сортировки, использующий метод Timsort, сочетает в себе элементы сортировки вставками и слиянием, что обеспечивает его высокую эффективность при работе с различными типами данных. В программе обеспечен ввод исходных данных, ввод коэффициентов вероятностного распределительного закона, сортировка данных с использованием алгоритма Timsort и вывод отсортированных данных в форме таблицы. Также реализовано оценивание коэффициентов уравнения связи между временем выполнения программы и объемом входных данных, а также анализ этих коэффициентов для оценки эффективности работы алгоритма.

Приложение написано на языке C++ в программной среде Microsoft Visual Studio 2022 Community

# **2 ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ**

Алгоритм сортировки Timsort был изобретен Тимом Петерсом в 2002 году. С тех пор стал стандартным алгоритмом сортировки в таких языках программирования, как Python и Java. Идея, лежащая в основе Timsort, заключается в том, что на реальной выборке данных часто встречаются уже упорядоченные участки. Осознавая это, Тим Петерс предложил использовать комбинацию нескольких алгоритмов сортировки, что позволяет эффективно работать с уже частично отсортированными массивами.

Основной принцип работы Timsort заключается в разбиении исходного массива на run, которые затем сливаются с помощью сортировки слиянием. Эти run - это небольшие отсортированные подмассивы, которые затем объединяются в один большой отсортированный массив.

В процессе разработки Timsort было введено понятие minrun - минимального размера отсортированных подмассивов, которые будут слияться. Минимальный размер run - это число, которое определяет размер подмассива, на котором будет работать сортировка вставками. minrun рассчитывается экспериментально и выбирается таким образом, чтобы обеспечить наилучшую производительность алгоритма. Как показали эксперименты, алгоритм работает наиболее эффективно, когда размер minrun находится в диапазоне от 32 до 64. Если значение minrun слишком маленькое, то потребуется больше времени на слияние, а если слишком большое - сортировка вставками будет слишком медленной из-за своей квадратичной сложности .

Для сбалансированного слияния количество run должно быть степенью двойки. Это важно, потому что алгоритм слияния наиболее эффективен, когда подмассивы имеют приблизительно одинаковый размер. Таким образом, минимальный размер каждого run определяется через minrun, которое в среднем должно быть в диапазоне от 32 до 64. Если размер массива меньше 64, то minrun устанавливается равным размеру всего массива, и алгоритм работает как простая сортировка вставками. В нашей сортировке минимальный размер подмассива minrun равен 64. Этот размер выбран исходя из предполагаемого диапазона размеров массива от 9000 до 50 000 элементов, что является оптимальным для данного случая.

Алгоритм сортировки методом Timsort:

Шаг 1. Разбиение массива на run и их сортировка вставками

1. Алгоритм начинается с установления указателя на начало массива. Далее начинается поиск упорядоченных подмассивов, называемых run. Эти подмассивы будут сортироваться с использованием алгоритма сортировки вставками, если их размер меньше minrun.
2. Каждый подмассив будет отсортирован по возрастанию, так как сортировка вставками работает именно с этим порядком.
3. Если найденный подмассив имеет размер меньше minrun, он сортируется с помощью сортировки вставками, которая инициализирует его как упорядоченный подмассив.
4. Алгоритм продолжает разбиение массива на run, пока весь массив не будет разделен на упорядоченные подмассивы. На каждом шаге размер подмассива будет увеличиваться, и подмассивы размером minrun или больше будут сортироваться вставками.

Шаг 2. Слияние подмассивов

1. Для слияния используется модификация сортировки слиянием, где два отсортированных подмассива сливаются в один. Слияние происходит по принципу «меньшее из двух» — элементы двух подмассивов сравниваются, и минимальный элемент добавляется в итоговый массив.
2. Слияние продолжается поочередно, пока все подмассивы не будут объединены в один отсортированный массив. Алгоритм увеличивает размер сливаемых подмассивов с каждым шагом, начиная с размера minrun, затем удваивая его: 64, 128, 256 и т. д.
3. Когда все подмассивы объединены в один массив, алгоритм завершает выполнение, и результатом является отсортированный массив.

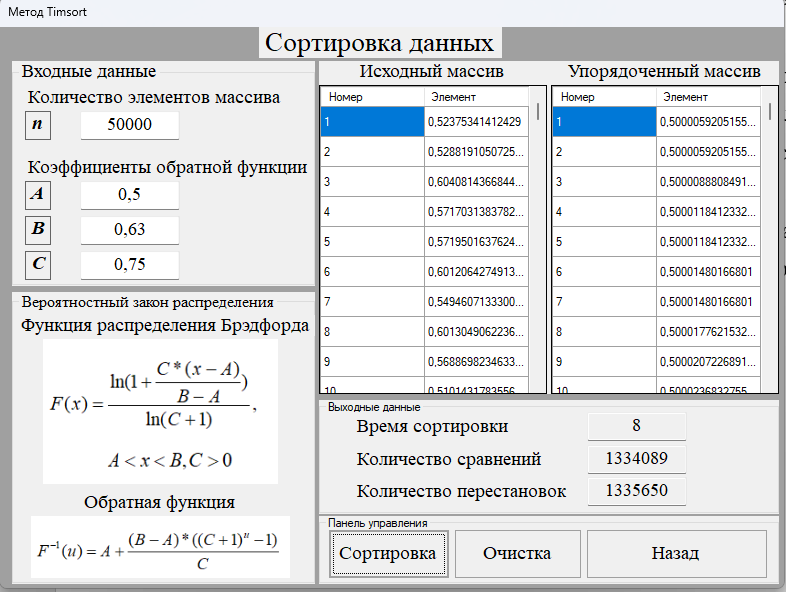
Пример использования приведен на рисунке 1.

Рисунок 1. – Сортировка массива с помощью Timsort

# **3 АНАЛИЗ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧА**

Анализ алгоритма – определение объема ресурсов (памяти и времени выполнения), требуемых алгоритмом для успешной обработки данных.

Для оценки качества алгоритма вводятся понятия сложность и эффективность алгоритма. Чем большее время и объем памяти требуются для реализации алгоритма, тем больше сложность и ниже эффективность.

## **Характеристики алгоритма**

Алгоритм обладает набором численных характеристик, сравнивая значения, которых можно произвести выбор наилучшего алгоритма. Сложность алгоритма делится на временную и ёмкостную, практическую и теоретическую.

Временная (вычислительная) сложность - критерий, характеризующий временные затраты на реализацию алгоритма. Практическая временная сложность оценивается во временных единицах (миллисекунды, секунды, количество временных тактов процессора, количество выполнения циклов).

Емкостная сложность **-** объем памяти компьютера, требуемый для реализации алгоритма. Практическая ёмкостная сложность выражается в битах, байтах, словах.

Вычислительная и емкостная сложности позволяют проводить анализ алгоритмов

## **Функции сложности алгоритмов**

Эффективность программы является важной характеристикой.

Пространственная эффективность измеряется количеством памяти, требуемой для выполнения программы. Временная эффективность программы определяется временем, необходимым для выполнения.

Сравнения алгоритмов производится сопоставлением порядков сложности. Порядок сложности – это функция, доминирующая над точечным выражением временной сложности. Функция  имеет порядок