МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное образование учреждение высшего образования

«Курганский государственный университет»

Кафедра «Программное обеспечение автоматизированных систем»

РФ КГУ 09.03.04. КР24.360091 02

**КУРСОВАЯ РАБОТА**

**по дисциплине**

**«Алгоритмы и структуры данных»**

**Пояснительная записка**

Листов 36

**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](#_Toc184360895)

[ВВЕДЕНИЕ 4](#_Toc184360896)

[1 АНАЛИТИЧЕСКИЙ ОБЗОР 5](#_Toc184360897)

[2 ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ 6](#_Toc184360898)

[3 АНАЛИЗ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧА 9](#_Toc184360899)

[3.1 Характеристики алгоритма 9](#_Toc184360900)

[3.2 Функции сложности алгоритмов 9](#_Toc184360901)

[3.3 Виды функции сложности алгоритмов 11](#_Toc184360902)

[3.4 Сравнение асимптотического поведения функций 14](#_Toc184360903)

[3.5 Базовое правило использования О большого 16](#_Toc184360904)

[3.6 Анализ функции сложности 16](#_Toc184360905)

[4 ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММНОГО КОМПЛЕКСА 17](#_Toc184360906)

[5 РАЗРАБОТКА ДИАГРАММЫ КЛАССОВ 18](#_Toc184360907)

[6 ОПИСАНИЕ СТРУКТУР ДАННЫХ 19](#_Toc184360908)

[7 ОПИСАНИЕ МЕТОДИКИ ПРОВЕДЕНИЯ ЭКСПЕРЕМЕНТАЛЬНОГО ИССЛЕДОВАНИЯ 21](#_Toc184360909)

[7.1 Математические методы оценивания времени выполнения алгоритмов 21](#_Toc184360910)

[7.2 Классическая регрессионная модель парной корреляции 21](#_Toc184360911)

[8 ОПИСАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ ПРОВЕДЕННОГО ИССЛЕДОВАНИЯ 30](#_Toc184360912)

[9 ВЫВОД ПО РЕЗУЛЬТАТАМ ПРОВЕДЕННОГО ИССЛЕДОВАНИЯ 31](#_Toc184360913)

# **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.

# **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

# **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.

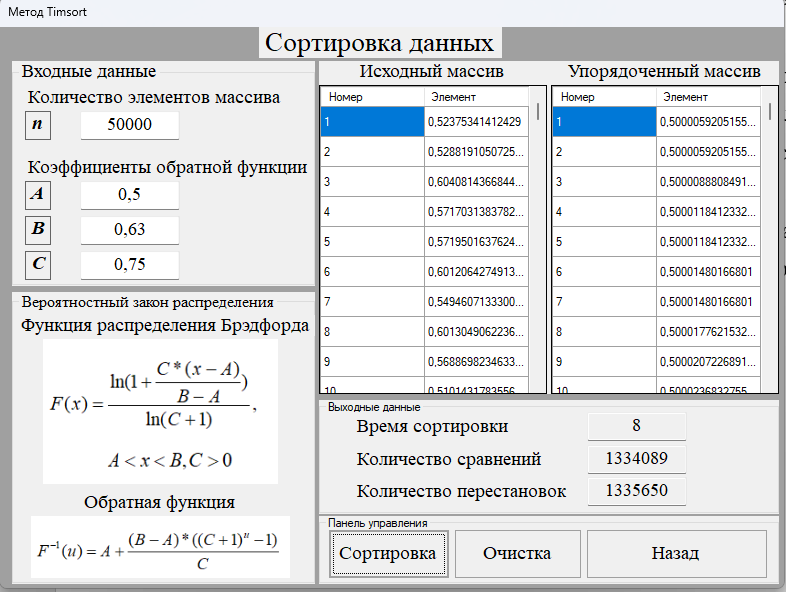
Пример использования приведен на рисунке 1.

Рисунок 1. – Сортировка массива с помощью Timsort

# **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  если имеются константа  и счетчик , такие, что выполняются условия  для любых . Например, точное время обработки массива определяется по формуле



где - размер массива.

Приближенное время определяется функцией

.

Функция сложности – это функция, которая определяет количество сравнений, перестановок, временные и объемные затраты на реализацию алгоритма.

Функция сложности О выражает относительную скорость алгоритма в зависимости от переменных. Для функции сложности сформулированы три правила:

1) ****. Постоянные множители не имеют значения для определения порядка сложности, например, **;**

2) **, **. Порядок сложности произведения двух функций равен произведению их сложностей, например, **;**

3) **** равна доминанте  и . Порядок сложности суммы функций определяется как порядок доминанты первого и второго слагаемых (выбирается наибольший порядок), например, **,** где k – константа, f и g – функции.

## **Виды функции сложности алгоритмов**

Время выполнения алгоритма T зависит от объема входных данных N (рисунок 2)



где - время выполнения алгоритма, мс;

N – объем входных данных.

Для оценивания трудоемкости алгоритмов введена специальная система обозначений – О-нотация.

N

Линерифмическая

Квадратичная

Кубическая

T

Линейная

Рисунок 2 – Графики функций сложности алгоритма

О-нотация позволяет учитывать в функции f (n) значимые элементы, отбрасывая второстепенные:

* 1. кубическая функция – функция f(n), старший член которой содержит . О-нотация имеет вид ;



* 1. квадратическая функция – функция f(n), старший член которой содержит . О-нотация имеет вид ;
  2. линейная функция – функция f(n), старший член которой содержит **.** О-нотация имеет вид ****;
  3. функция линерифмическая – функция, старший член которой равен  логарифмов . О-нотация имеет вид ****.

Например, в функции **** при больших **** компонента **** превосходит остальные слагаемые. Поведение функции определяется компонентой ****. Остальные компоненты отбрасываются. Функция **** имеет оценку поведения (скорость роста значений) ****

О-оценивание позволяет описывать характер поведения функции f(n) с ростом n: насколько быстро или медленно растет эта функция.

О-оценка разбивает функции сложности на группы в зависимости от скорости роста:

1. постоянные функции ****, которые с ростом  не растут;
2. функции с логарифмической скоростью роста **;**
3. функции с линейной скоростью роста ****;
4. функции с линейно-логарифмической скоростью роста ****;
5. функции с квадратичной скоростью роста ****;
6. функции со степенной скоростью роста **** при а>2;
7. функции с показательной или экспоненциальной скоростью роста ****
8. функции с факториальной степенью роста **** /9/.

Описание О-нотаций приведено в таблице 1.

|  |  |
| --- | --- |
| О-нотация | Описание |
| О (1) | Инструкции программы запускаются независимо от n. Время выполнения программы постоянно (помещение в стек). Операции в программе выполняются один или несколько раз. Алгоритм независимо от размера данных требует одно и тоже время. |
| О (n) | Время выполнения программы линейно и зависит от n. Входной элемент обрабатывается линейное число раз. |
| О (n2) | Время выполнения программы является квадратичным. Алгоритмы используются для небольших n (цикл двойного уровня вложенности, сортировки выбором, вставками) |
| О (n3) | Алгоритм программы имеет кубическое время выполнения (цикл тройного уровня вложенности). Применяется для небольших задач. (умножение матриц) |
| О (logn) | Логарифмическая зависимость. С ростом n программа работает медленнее. Время характерно для программ, которые сводят большую задачу к набору меньших подзадач, уменьшая на каждом шаге размер подзадачи на постоянный коэффициент. Общее решение находится в одной из подзадач (бинарный поиск) |
| О (n\*logn) | Линерифмическая зависимость. Время выполнения программы пропорционально n\*logn. Возникает, когда алгоритм решает задачу, разбивая ее на меньшие подзадачи, решает независимо и затем объединяет решения подзадач (комбинация, сортировки быстрая, слиянием) |

Продолжение таблицы 1.

|  |  |
| --- | --- |
| О (2n) | Экспоненциальная зависимость. Прямое решение задач (перебор и сравнение различных решений). Для комбинаторных задач нереализуемы. Сведение к приближенному алгоритму с приближенным значением. |

## **Сравнение асимптотического поведения функций**

Математические обозначения для сравнения асимптотического поведения функций имеют вид:

1. определение О большое. Функция  если имеются положительные константы  и  такие, что , когда ;
2. определение Омега большое. Функция  если имеются положительные константы  и  такие, что , когда ;
3. определение Тета большое. Функция  только и если только  и 
4. определение о малое. Функция  если и только  и 

Определение О большое устанавливает, что имеется такая точка , что для всех значений  за этой точкой значения  ограничены значениями , умноженными на константу. Например, время выполнения алгоритма . В некоторой точке время выполнения ограничивается квадратичной функцией. Константы игнорируются.

Определение Омега большое утверждает, что что скорость роста функции  больше или равна скорости функции . Например, алгоритм основанный на анализе подпоследовательности в задаче требует определения максимальной суммы непрерывной подпоследовательности требует время , потому что возможно наблюдение квадратичного числа подпоследовательностей. Это нижняя оценка, используемая в более точном анализе.

Определение Тета большое утверждает, что скорость роста функции  равна скорости функции . Например, алгоритм максимальной подпоследовательности выполняется за время . Время выполнения ограничено квадратичной функцией. Граница не может быть улучшена, потому что она ограничена снизу другой квадратичной функцией. Тета большое представляет верхнюю границу алгоритма и гарантирует точность анализа.

Определение о малое утверждает, что скорость роста функции  строго меньше, чем скорость роста функции . Функция о малое отлична от О большое, так как О большое допускает возможность одинаковых скоростей роста. Например, если время выполнения алгоритма составляет , оно гарантированно растет медленнее, чем квадратичная функция.  
Имеем подквадратичный алгоритм. Граница ниже, чем .

Сравнение асимптотического поведения функций приведено в таблице 2

Таблица 2 – Относительная скорость роста

|  |  |
| --- | --- |
| Математическое выражение | Относительные скорости роста |
|  | Скорость роста скорости роста |

Продолжение таблицы 2.

|  |  |
| --- | --- |
|  | Скорость роста скорости роста |
|  | Скорость роста скорости роста |
|  | Скорость роста скорости роста |

## **Базовое правило использования О большого**

Базовое правило формулируется следующим образом: время выполнения цикла не превышает времени выполнения операторов, входящих в цикл (включая операторы условия), умноженное на число итераций цикла.

Время выполнения операторов внутри группы вложенных циклов равно времени выполнения операторов, умноженному на число итераций во всех циклах. Время выполнения последовательности циклов, следующих друг за другом, равно времени выполнения доминантного цикла. Первый случай описывается квадратичной функцией. Второй случай линеен.

## **Анализ функции сложности**

Определение сложности алгоритма сводится к анализу циклов и рекурсивных вызовов. Алгоритмы без циклов и рекурсивных вызовов имеют константную сложность.

void insertionSort(double\* arr, int left, int right)

{

for (int i = left + 1; i <= right; i++)

{

double temp = arr[i];

W

int j = i - 1;

while (j >= left && arr[j] > temp)

{

S

arr[j + 1] = arr[j];

T

X

V

j--;

}

arr[j + 1] = temp;

U

}

}

void merge(double\* arr, int l, int m, int r)

{

int len1 = m - l + 1, len2 = r - m;

R

double\* left = new double[len1];

double\* right = new double[len2];

for (int i = 0; i < len1; i++)

QQ

left[i] = arr[l + i];

for (int i = 0; i < len2; i++)

P

right[i] = arr[m + 1 + i];

O

int i = 0, j = 0, k = l;

while (i < len1 && j < len2)

{

if (left[i] <= right[j])

{

arr[k] = left[i];

i++;

M

}

else {

N

arr[k] = right[j];

j++;

}

k++;

}

while (i < len1)

H

{

arr[k] = left[i];

k++;

L

K

i++;

}

while (j < len2)

{

arr[k] = right[j];

I

k++;

G

j++;

}

}

void TimSort(double\* arr, int n)

{

for (int i = 0; i < n; i += RUN)

F

insertionSort(arr, i, min((i + RUN - 1), (n - 1)));

G

for (int size = RUN; size < n; size = 2 \* size)

{

for (int left = 0; left < n; left += 2 \* size)

A

{

int mid = left + size - 1;

B

C

int right = min((left + 2 \* size - 1), (n - 1));

E

if (mid < right)

merge(arr, left, mid, right);

D

}

}

}

1. 









1. 





















1. 













Оценка функции сложности программного алгоритма метода.

Timsort – кубическая, следовательно при больших размерах массива, время упорядочивания производится крайне медленно. Для повышения быстродействия работы сортировки необходимо переработать алгоритм сортировки.

# **ОПИСАНИЕ СТРУКТУРЫ ПРОГРАММНОГО КОМПЛЕКСА**

На рисунке 3 приведена структура программного комплекса.

Диалоговые окна

Интерфейс пользователя

Меню

Ввод-вывод данных

Графическое представление

Бизнес-логика

Обработка данных

Оценивание коэффициентов уравнения связи

Расчет коэффициентов корреляции и детерминации

Расчет среднеквадратичной ошибки

Таблицы

Хранение данных

Графики

Рисунок 3 - Структура программного комплекса

# **РАЗРАБОТКА ДИАГРАММЫ КЛАССОВ**

На рисунке 4 приведена диаграмма классов.

**InfoForm**

buttonExit\_Click()

linkLabel1\_LinkClicked()

**MainForm**

buttonSort\_Click()

buttonEqConnection\_Click()

buttonInfo\_Click()

buttonExit\_Click()

**MHKForm**

buttonExit\_Click()

buttonExit\_Click()

buttonMHK\_Click()

buttonClear\_Click()

buttonCalc\_Click()

sample\_size\_TextChanged()

EqForm\_Load()

buttonExit\_Click()

buttonSort\_Click()

buttonClear\_Click()

textBoxN\_TextChanged()

SortForm\_Load()

**SortForm**

**EqForm**

Рисунок 4 – диаграмма классов.

# **ОПИСАНИЕ СТРУКТУР ДАННЫХ**

Таблица 3 – Описание структур данных

|  |  |  |
| --- | --- | --- |
| **MainForm Главное меню** | | |
| buttonExit\_Click() | Выход из программы. | |
| buttonInfo\_Click() | Открывает модальное окно с информацией о программе | |
| buttonSort\_Click() | Открывает модальное окно с сортировкой данных с помощью метода сортировки Timsort | |
| buttonEqConnection\_Click() | Открывает модальное окно с расчётом уравнения связи | |
| **InfoForm О программе** | | |
| buttonExit\_Click() | Возвращает в главное меню | |
| linkLabel1\_LinkClicked() | При нажатии открывает ссылку в браузере на репозиторий GitHub | |
| **SortForm Метод Timsort** | | |
| buttonExit\_Click() | | Возвращает в главное меню |
| buttonSort\_Click() | | Заполнение массива случайными числами и его сортировка |
| buttonClear\_Click() | | Очищает поля ввода и таблицы и выходные данные |
| textBoxN\_TextChanged() | | Делает кнопки buttonSort  и buttonClear активными при правильном вводе данных |

Продолжение таблицы 3.

|  |  |
| --- | --- |
| SortForm\_Load() | Отключает кнопки “Сортировка” и “Очистка” при запуске формы |
| **EqForm Уравнение связи. Расчетные данные и графики** | |
| buttonExit\_Click() | Возвращает в главное меню |
| buttonMHK\_Click() | Открывает модальное с теоретическим обоснованием |
| buttonClear\_Click() | Очищает все поля и таблицу и график |
| buttonCalc\_Click() | Провести анализ и рассчитать парный коэффициент корреляции Y и X, совокупный коэффициент детерминации , средняя квадратическая ошибка объема выборки X |
| sample\_size\_TextChanged() | Делает кнопки buttonSort  и buttonClear активными при правильном вводе данных |
| EqForm\_Load() | Отключает кнопку buttonSort при открытии модального окна |
| **MHKForm** | |
| buttonExit\_Click() | Возвращает к уравнению связи |

# **ОПИСАНИЕ МЕТОДИКИ ПРОВЕДЕНИЯ ЭКСПЕРЕМЕНТАЛЬНОГО ИССЛЕДОВАНИЯ**

## **Математические методы оценивания времени выполнения алгоритмов**

Цель эксперимента – определение уравнения связи между временем выполнения программного алгоритма (временной параметр T) и размером задачи (емкостной параметр V). В зависимости от постановки задачи разрабатывается регрессионная модель парной корреляции или регрессионная модель множественной корреляции. Например, в случае внешней сортировки данных для установления связи между временем выполнения программного алгоритма и количеством элементов исходного массива разрабатывается классическая регрессионная модель парной корреляции. Пусть оценки коэффициентов являются состоятельными, несмещенными и эффективными.

## **Классическая регрессионная модель парной корреляции**

Классическая регрессионная модель парной корреляции (КРМПК) устанавливает связь между эндогенной переменной и одной экзогенной переменной. Уравнение связи имеет вид

 (1)

Где  - эндогенная переменная;

 – экзогенная переменная, ;

 – свободный коэффициент;

 – выборочные коэффициенты;

Коррелияция – теснота (сила) связи. Регрессия – форма (уравнение) связи.

КРМПК используется в задачах сортировки массивов. Эндогенная переменная – времени сортировки, экзогенная переменная размер массива (количество элементов).

Экспериментальное исследование программного алгоритма решения задачи включает этапы:

Этап 1. *Получение выборки статистических данных в ходе проведения эксперимента на ЭВМ.*

Количество наблюдений m = 10. Результаты проведения эксперимента представляются в таблице 4

Таблица 4 – Исходные статистические данные

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Время, *у* | Размер массива, *х* |  |  |  |
| 1 | 4 | 22377 | 500730129 | 89508 | 16 |
| 2 | 4 | 26819 | 719258761 | 107276 | 16 |
| 3 | 2 | 9908 | 98168464 | 19816 | 4 |
| 4 | 9 | 38448 | 1478248704 | 346032 | 81 |
| 5 | 6 | 33776 | 1140818176 | 202656 | 36 |
| 6 | 6 | 35264 | 1243549696 | 211584 | 36 |
| 7 | 2 | 12004 | 144096016 | 24008 | 4 |
| 8 | 6 | 31731 | 1006856361 | 190386 | 36 |
| 9 | 3 | 19283 | 371834089 | 57849 | 9 |
| 10 | 6 | 34597 | 1196952409 | 207582 | 36 |

Продолжение таблицы 4.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Итого | 48 | 264207 | 7900512805 | 1456697 | 274 |

Этап 2. *Определение уравнения связи в общем виде.*

По результатам проведения эксперимента на ЭВМ строится и анализируется точечный график зависимости, .

На рисунке 5 предоставлен точечный график и график уравнения связи  зависимости времени  выполнения сортировки от количества  элементов массива.

Анализ точечного графика позволяет сделать вывод о линейной тенденции. Уравнение связи в общем виде имеет вид (1).

где  – время выполнения программного алгоритма;

 – количество элементов;

 – свободный коэффициент;

– выборочные коэффициенты.

Этап 3. *Оценивание коэффициентов уравнения связи.*

Оценивание коэффициентов уравнения связи – приближенный расчет значений коэффициентов. Оценивание коэффициентов  и  производится методом наименьших квадратов (МНК).

*Определение.* Сумма квадратов разности между наблюденным и расчетным значениями эндогенной переменной минимальная.

, (2)

где - наблюденное значение эндогенной переменной;

 - расчетное значение эндогенной переменной;

 - номер наблюдения, ,

- объем выборки.

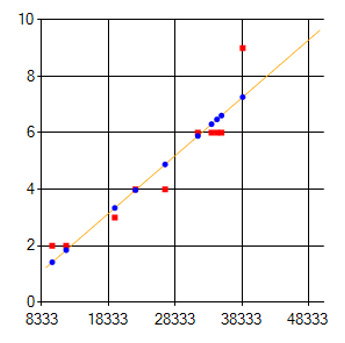
 Оценивание коэффициентов  и  производится по шагам.

Рисунок 5 – Точечный график и график уравнения связи зависимости времени сортировки от количества элементов массива

Шаг 1. Подставим в математическое описание (2) МНК вместо правую часть уравнения связи в общем виде (1). Математическое описание МНК будет иметь вид



Шаг 2. Определим систему нормальных уравнений для оценивания коэффициентов , уравнения связи.

 (4)

 (5)

 (5)

Выполним преобразования: сократим на (-2) левую и правую части, раскроем скобки, сгруппируем слагаемые, заменим аддитивную операцию  на мультипликативную операцию  СНУ имеет вид

 (6) (7)

 (7)

Система нормальный уравнений решается методом Крамера.

Вычисляем по формулам  определители 2×2 матрицы, необходимый для решения системы уравнений методом Крамера.

 (8)

 (9)

 (10)

С помощью определителей вычисляем свободный член  и выборочный коэффициент 

 (11)

 (12)

Этап 4. *Анализ уравнения связи и расчет доверительного интервала.*

Анализ уравнения связи включает расчет линейного коэффициента корреляции, совокупного коэффициента детерминации, частных коэффициентов корреляции, частных коэффициентов детерминации, частных коэффициентов эластичности и частных бэта-коэффициентов.

Коэффициент корреляции между и  рассчитывается по формуле

 (13)

Частные коэффициенты корреляции позволяют провести анализ тесноты связи между эндогенной переменной и одной из экзогенных переменных при неизменных других.

Коэффициент детерминации рассчитывается как

 (14)

Уравнение регрессии примет вид

 (15)

Коэффициент эластичности показывает на сколько процентов в среднем изменится эндогенная переменная  от своего среднего значения при изменении экзогенной переменной  на 1 процент от своей средней величины.

Коэффициент эластичности рассчитывается по формуле

 (16)

где - выборочный коэффициент;

 - среднее арифметическое значение объема выборки X;

 - среднее арифметическое значение объема выборки Y.

Среднее арифметическое значение объема выборки X рассчитывается по формуле

 (17)

Среднее арифметическое значение объема выборки Y рассчитывается по формуле

 (18)

Бета-коэффициент показывает на какую часть величины своего среднего квадратического отклонения изменится в среднем значение эндогенной переменной  при изменении экзогенной переменной  на величину среднеквадратического отклонения. Бета-коэффициент рассчитывается по формуле

, (19)

где  - выборочный коэффициент

 - средняя квадратическая ошибка (отклонение) объема выборки X;

- средняя квадратическая ошибка (отклонение) объема выборки Y.

Средняя квадратическая ошибка объема выборки X рассчитывается по формуле

, (20)

где  - экзогенная переменная;

- среднее арифметическое значение объема выборки Х;

 - объем выборки.

Средняя квадратическая ошибка объема выборки Y рассчитывается по формуле

, (21)

где  - экзогенная переменная;

- среднее арифметическое значение объема выборки Х;

 - объем выборки.

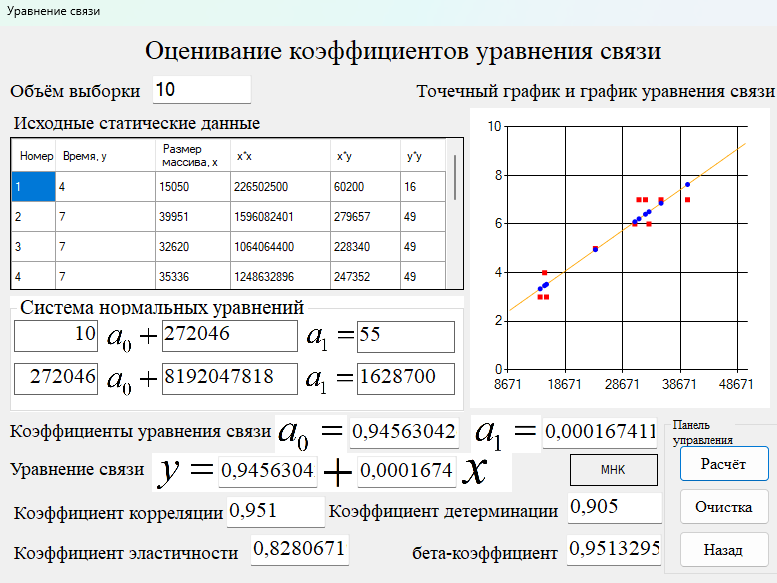
Пример анализа алгоритма приведен на рисунке 6.

Рисунок 6 – Окно программы (уравнение связи).

# **ОПИСАНИЕ И АНАЛИЗ РЕЗУЛЬТАТОВ ПРОВЕДЕННОГО ИССЛЕДОВАНИЯ**

По результатам проведения экспериментального исследования можно сделать вывод о том, что коэффициент парной корреляции равен 0,951. Это означает, что связь между переменными y и x – очень тесная. Коэффициент детерминации . Это означает, что входная переменная значима и весома. Уравнение связи имеет вид



# **ВЫВОД ПО РЕЗУЛЬТАТАМ ПРОВЕДЕННОГО ИССЛЕДОВАНИЯ**

В ходе выполнения курсовой работы был разработан алгоритм для решения задачи. Выполнены: формализация алгоритма метода Timsort, оценка сложности программного алгоритма с помощью функции , оценены коэффициенты уравнения связи между временным и емкостным параметрами и анализ уравнения связи.

В результате выполнения курсовой работы повышены теоретические знания и приобретены практические навыки в разработке и анализе алгоритмов решения задачи.

# **СПИСОК ЛИТЕРАТУРЫ**

1. Ахо Альфред В., Хопкрофт Джон, Ульман Джеффри Д. Структуры данных и алгоритмы: Учебное пособие/Пер. с англ. - М.: Издательский дом «Вильямс», 2000. – 432 с.
2. Давыдов В.Г. Программирование и основы алгоритмизации: Учеб. Пособие. – М.: Высш. шк., 2003. – 447 с.
3. Давыдов В.Г.Технологии программирования. С++. – СПб.: БХВ – Петербург, 2005. – 672 с.
4. Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы. Анализ и построение. - М.: «БИНОМ», 2000. - 960 с.
5. Кубенский А.А. Структуры и алгоритмы обработки данных. Объектно-ориентированный подход и реализация на С++. – Санкт-Петербург: БХВ-Петербург, 2004. - 466 с.
6. Семахин А. М. Алгоритмы и структуры данных: методические указания к выполнению лабораторных и курсовых работ для студентов. - Курган: Издательство Курганского государственного университета, 2012. - 63 с.
7. Семахин А. М. Алгоритмы и структуры данных: методические указания к выполнению курсовых работ для студентов. – Курган: Издательство Курганского государственного университета, 2014. – 73 с.
8. Шаг 113. Microsoft Visual C++ 2010. Начала. Справочная информация. Microsoft HTML Help Workshop //

https://it.kgsu.ru/VC++2010/vc2010\_113.html

1. Шаг 119.Microsoft Visual C++ 2010. Язык С/С++.Приложения Windows Forms. Организация работы с множеством форм // https://it.kgsu.ru/C++MSVS2010/c++vs2010\_119.html
2. Шаг 120. Microsoft Visual C++ 2010. Язык С/С++. Приложения Windows Forms. Вызов формы на выполнение// https://it.kgsu.ru/C++MSVS2010/c++vs2010\_120.html