https://lh5.googleusercontent.com/kzTYZUXBnjoKLBf6x-4PL1InpxhwgjzkO3dYRdfJXSmagQbCUtSnixbraQN8t6p-bUMx1v3H3UiKYg1jcED7TTS19iEZkUf3o4KLJBA7W8f0AeZIm2hS5Qf4G9YShlvNgyzNpxoSTwlmUSRCAg

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«Дальневосточный федеральный университет»**

ШКОЛА ЕСТЕСТВЕННЫХ НАУК  
 **Кафедра прикладной математики, механики, управления и программного обеспечения**

Бомко Виктор Федорович

ЭКСПЕРЕМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ АЛГОРИТМОВ ИНТРОСПЕКТИВНОЙ И БЫСТРОЙ СОРТИРОВОК

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Структуры и алгоритмы компьютерной обработки данных» по образовательной программе подготовки бакалавров по направление 02.03.03 «Математическое обеспечение и администрирование информационных систем»

Студент группы Б8204 \_\_\_\_\_\_/ Бомко В.Ф.

(подпись)

Руководитель

к.т.н. \_\_\_\_\_\_\_ Остроухова С.Н

(подпись)

\_\_\_\_\_\_\_ «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2019 г.

Защищен с оценкой \_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) Фамилия И.О.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2019 г.

г. Владивосток

2019 г.

Оглавление

[Введение 3](#_Toc23319678)

[1 Описание алгоритмов 4](#_Toc23319679)

[1.1 Быстрая сортировка 4](#_Toc23319680)

[1.2 Пирамидальная сортировка. 7](#_Toc23319681)

[1.2 Интроспективная сортировка. 9](#_Toc23319682)

[2 Разработка КП 12](#_Toc23319683)

[2.1. Архитектура КП 12](#_Toc23319684)

[2.2 Требования к КП 13](#_Toc23319685)

[3 Сценарий диалога КП с пользователем 16](#_Toc23319686)

[4 Экспериментальные исследования 25](#_Toc23319687)

[4.1. Эксперимент над первой группой: 25](#_Toc23319688)

[4.2. Эксперимент над второй группой: 27](#_Toc23319689)

[4.3. Эксперимент над третьей группой: 29](#_Toc23319690)

[4.4. Эксперимент над четвёртой группой: 31](#_Toc23319691)

[4.5. Эксперимент над пятой группой: 33](#_Toc23319692)

[Заключение 36](#_Toc23319693)

# Введение

В связи с широким внедрением электронно-вычислительных машин в различные сферы человеческой деятельности, процесс создания программного обеспечения для этих машин стали изучать с научной точки зрения. Выяснилось, что для того, чтобы хранить и обрабатывать огромные объемы информации нужно применять особые алгоритмы. Поиск необходимой информации может быть затруднён, ввиду неупорядоченности и большого количества обрабатываемых данных. Для упорядочивания данных прибегают к использованию сортировок.

В современных реалиях существует множество сортировок, отличающихся между собой скоростью и принципом работы. Многие из этих алгоритмов неэффективны при работе с большими объемами данных. Время выполнения таких алгоритмов пропорционально квадрату числа элементов, что не приемлемо для того, чтобы назвать сортировку эффективной.

В данной курсовой работе мы будем исследовать один из самых быстрых известных универсальных алгоритмов сортировки – быструю сортировку, а также ее модификацию – интроспективную сортировку.

Целью курсовой работы является разбор, изучение и сравнение алгоритмов быстрой и интроспективной сортировок.

Задачи курсовой работы:

* Изучить теоретическую основу обеих сортировок;
  + Алгоритм быстрой сортировки;
  + Алгоритм интроспективной сортировки;
* Разработать программное обеспечение для проведения экспериментов;
* Провести экспериментальное исследование обеих сортировок;
* Провести сравнительный анализ полученных результатов;

1. **Описание алгоритмов**
   1. **Быстрая сортировка**

Одной из самых универсальных сортировок является быстрая сортировка. Она показывает хорошие результаты при работе с различными видами данных и менее требовательна к ресурсам компьютера, чем другие методы сортировок.

**Идея быстрой сортировки.**

Основная идея быстрой сортировки состоит из трёх шагов[1]:

1. Выбор опорного элемента – означает выбрать элемент из массива, назовём его опорным. Существует несколько методов выбора опорного элемента, но в данном случае, в качестве опорного элемента будем выбирать первый, центральный или последний элемент сортируемой последовательности. Такой метод выбора опорного элемента называют “Медиана из трёх”. Выбранный опорный элемент помещается на место последнего элемента, остальные – в порядке возрастания на первую и серединную позицию.

2. Разбиение – процесс перераспределения элементов в массиве таким образом, что элементы меньше опорного помещаются перед ним, а больше или равные после. Рассмотрим следующую вариацию алгоритма разбиения. Для нашего случая, опорный элемент, выбранный на предыдущем шаге, должен быть средним из возможных. Изначально, алгоритм запоминает номер первого элемента в последовательности, назовём его индексом i. Затем, проходит от начала до предпоследнего элемента последовательности и сравнивает элементы с опорным. Если элемент меньше или равен опорному, то меняет значение текущего обозреваемого элемента и элемента с индексом i местами, и увеличивает индекс i на единицу. После того, как последовательность будет пройдена полностью, алгоритм меняет местами значение элемента с индексом i и опорного элемента. Данный алгоритм получил название “Разбиение Ломуто”.

3. Рекурсивно применить первые два шага к двум подмассивам слева и справа от опорного элемента. Рекурсия не применяется к массиву, в котором только один элемент или отсутствуют элементы.

**Пример быстрой сортировки.**

Пусть исходная последовательность выглядит следующим образом - {7, 5, 8, 3, 2, 4, 1, 0, 6, 9, 10}. Назовём её массив А.

1. Производим поиск опорного элемента. Сравниваем первый, последний и серединный элементы. Выбираем максимальный по значению элемент, ставим его на место последнего элемента, остальные элементы располагаем в порядке возрастания на позиции первого и серединного элементов.

Теперь массив выглядит следующим образом: A: {7, 5, 8, 3, 2, 10, 1, 0, 6, 9, 4}

2. Производим перестановки в массиве A согласно разбиению Ломуто. Опорный элемент выделим серым цветом. Элемент с индексом i подчеркнём, а текущий просматриваемый элемент обозначим жирным.

А: {**7**, 5, 8, 3, 2, 10, 1, 0, 6, 9, *4*}

А: {7, **5**, 8, 3, 2, 10, 1, 0, 6, 9, *4*}

А: {7, 5, **8**, 3, 2, 10, 1, 0, 6, 9, *4*}

А: {7, 5, 8, **3**, 2, 10, 1, 0, 6, 9, *4*}

Текущий просматриваемый элемент меньше опорного элемента, это означает, что его нужно поменять местами с элементом с индексом ***i*** и к индексу ***i*** прибавить единицу. Далее действуем аналогично.

А: {3, 5, 8, **7**, 2, 10, 1, 0, 6, 9, *4*}

А: {3, 5, 8, 7, **2**, 10, 1, 0, 6, 9, *4*}

А: {3, 2, 8, 7, **5**, 10, 1, 0, 6, 9, *4*}

А: {3, 2, 8, 7, 5, **10**, 1, 0, 6, 9, *4*}

А: {3, 2, 8, 7, 5, 10, **1**, 0, 6, 9, *4*}

А: {3, 2, 1, 7, 5, 10, **8**, 0, 6, 9, *4*}

А: {3, 2, 1, 7, 5, 10, 8, **0**, 6, 9, *4*}

А: {3, 2, 1, 0, 5, 10, 8, **7**, 6, 9, *4*}

А: {3, 2, 1, 0, 5, 10, 8, 7, **6**, 9, *4*}

А: {3, 2, 1, 0, 5, 10, 8, 7, 6, **9**, *4*}

Теперь, когда счётчик позиция текущего просматриваемого элемента совпадает с позицией перед опорным элементом, производим обмен значений элемента с индексом ***i*** и опорным элементом.

А: {3, 2, 1, 0, 4, 10, 8, 7, 6, **9**, *5*}

3. Повторяем предыдущие два шага, но уже для двух новых массивов: A1:{3, 2, 1, 0} и A2:{10, 8, 7, 6, 9, 5}.

**Оценка сложности алгоритма быстрой сортировки.**

Быстрая сортировка принадлежит к категории обменных сортировок (т.е. требует всего лишь небольшого вспомогательного стека), на выполнение сортировки из N элементов в среднем затрачивается время, пропорциональное N log N и для него характерны исключительно короткие внутренние циклы. Его недостатком является то, что он неустойчив, для его выполнения в наихудшем случае требуется N2 операций, он хрупок в том смысле, что даже простая ошибка в реализации может пройти незамеченной и вызвать ошибки в работе алгоритма на некоторых файлах. Худший случай получается тогда, когда на каждом шаге рекурсии, массив делится на подмассивы длиной n-1 и 1.

Общая сложность алгоритма определяется глубиной рекурсии, так как размер исходного массива постоянен и суммарно на каждом уровне потребуется *O(n)* операций[2].

Таким образом:

* Лучший случай: *O(N log2 N)*
* Средний случай: *O(N log2 N)*
* Худший случай: *O(N2)*

Мы уже рассмотрели ситуацию, когда быстрая сортировка становится неэффективной, это происходит тогда, когда на каждом шаге рекурсии массив делится на подмассивы длиной n и n-1. Для устранения неэффективности сортировку модифицируют, добавляя в алгоритм части алгоритмов других сортировок. Одной из модификаций быстрой сортировки является интроспективная сортировка.

**1.2 Пирамидальная сортировка.**

**Идея пирамидальной сортировки.**

Пирамидальная сортировка, или сортировка кучей использует сортирующее дерево. Сортирующее дерево или куча, это такое дерево, у которого выполнены следующие условия[3]:

1. Значение в любой вершине не меньше, чем значения её потомков.
2. Расстояние всех листьев до корня отличается не более чем на 1 слой. Корнем называют элемент, находящийся на первом уровне дерева.
3. Последний слой заполняется слева-направо без «дырок».

Пирамидальная сортировка сортирует последовательность без привлечения дополнительной памяти за время *O(N log N).*

Для понимания работы пирамидальной сортировки можно представить, что мы обменяли корневой элемент с самым последним элементов в последнем слое. Тогда последний элемент станет самым большим. Если после этого исключить этот элемент из кучи, то первые ***n-1*** элементов будут удовлетворять условиям сортирующего дерева, за исключением, может быть, корня. Если произвести восстановление свойств кучи, то ***n-1*** станут кучей, а последний, ***n***-ый элемент будет больше всех. Повторяя эти действия ***n-1*** раз, мы отсортируем массив.

Восстановление свойств кучи означает то, что если какой-то элемент меньше, чем его сыновья, то мы меняем его местами с сыном и повторяем восстановление свойств уже для сына. Сыном называют элемент сортировки, у которого есть элемент уровнем выше.

**Пример пирамидальной сортировки.**

Дано множество А: { 440, 551, 122, 423, 944, 185, 66, 677}, необходимо отсортировать его по возрастанию.

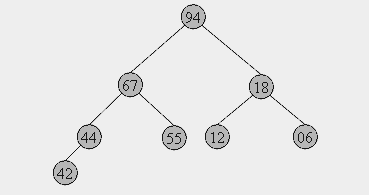


Рисунок 1 - Массив в построенном виде

Построили пирамиду по всем правилам, массив в построенном виде изображён на рисунке 1. Переходим к этапу сортировки:

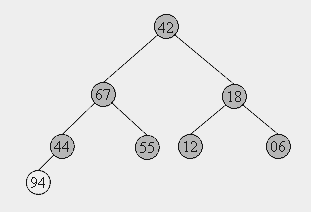


Рисунок 2 - Первый шаг сортировки

Берем верхний элемент пирамиды и меняем с последним местами. Теперь "забываем" об этом элементе и далее рассматриваем массив размерности (n-1) , данные операции представлены на рисунке 2. Для превращения его в пирамиду достаточно просеять лишь новый первый элемент. Просеивание происходит по следующим правилам: элемент-корень опускается ниже и ниже, пока есть элементы-сыны больше него.

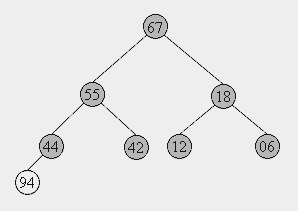


Рисунок 3 - Пирамида на первом шаге после просеивания

Повторяем шаг 1, пока обрабатываемая часть массива не уменьшится до одного элемента. Процесс повторения шага изображён на рисунках 3, 4 и 5

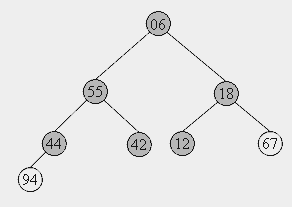


Рисунок 4 - Сортировка элемента "6"

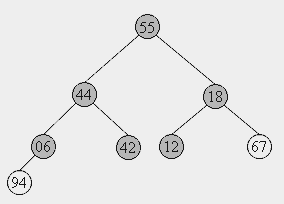


Рисунок 5 - Просеивание элемента "6"

* 1. **Интроспективная сортировка.**

**Идея интроспективной сортировки.**

Основная идея алгоритма заключается в использовании сильных сторон двух алгоритмов сортировки: быстрой и пирамидальной. Таким образом, алгоритм использует быструю сортировку как основную, но переключается на пирамидальную сортировку, когда глубина рекурсии превысит некоторый, заранее установленный, уровень[4].

**Пример интроспективной сортировки.**

Пусть исходная последовательность выглядит следующим образом - {7, 5, 8, 3, 2, 4, 1, 0, 6, 9, 10}. Назовём её массив А.

Первым делом, необходимо посчитать глубину допустимой рекурсии для последовательности такого размера. Высчитывается она по формуле logN\*2, где N – длина последовательности. Глубину рекурсии обозначим как maxdepth, в нашем примере она равна 2. Теперь применяем разбиение Ломуто к нашему массиву A. Получили два подмассива A1: {3, 2, 1, 0} и A2: {10, 8, 7, 6, 9, 5}. Теперь применяем разбиение Ломуто к получившимся подмассивам, глубина рекурсии maxdepth снижается на единицу и теперь равна 1. Продолжаем действия до тех пор, пока maxdepth не будет равен 0. Как только maxdepth станет равным 0, подмассив, который будет участвовать в алгоритме будет отсортирован алгоритмом пирамидальной сортировки.

**Оценка сложности алгоритма интроспективной сортировки.**

Опытным путем было выяснено, что на худшем наборе данных для алгоритма быстрой сортировки «медиана из трёх» (рассматривался массив из 100 тысяч элементов) интроспективная сортировка работает примерно в 200 раз быстрее. Это происходит потому, что алгоритм переключается на пирамидальную сортировку, когда глубина рекурсии превысит некоторый уровень.

Пирамидальная сортировка в свою очередь характерна тем, что у неё нет ни вырожденных, ни лучших наборов данных, любые массивы она сортирует всегда с одинаковой временной сложностью - O(N log N).

Таким образом:

* Лучший случай: *O(N).*
* Средний случай: *O(N log2 N).*
* Худший случай: *O(N log2 N).*

1. **Разработка КП**
2. **Архитектура КП**

Для проведения экспериментального исследования необходимо разработать КП, состоящую из модулей, указанных на рисунке 1.

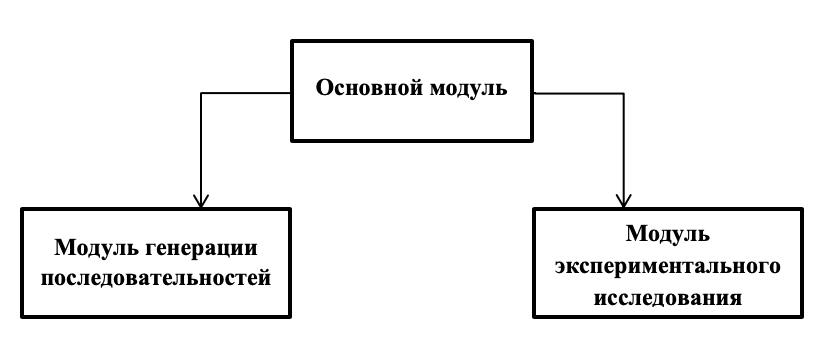
****

Рисунок 6 – Модули программы.

**Основной модуль** предоставляет возможность перейти к генерации последовательностей, либо к окну выбора файлов с последовательностями для анализа.

**Модуль генерации последовательностей** позволяет создавать несколько текстовых файлов с последовательностями типа «Вид последовательности» с количеством элементов, равным «Число элементов» для каждого файла.

**Модуль экспериментального исследования** позволяет выбрать файлы для проведения эксперимента, сортирует методами быстрой и интроспективной сортировок все последовательности в файлах и подсчитывает данные для анализа:

* Время работы сортировки
* Количество сравнений
* Количество обменов

Модуль так же позволяет сохранить отсортированные в ходе эксперимента последовательности. Модуль строит графики по собранным данным в ходе анализа.

* 1. **Требования к КП**

**Требования к данным в генерации последовательностей:**

Входные данные:

1. Тип последовательности (далее type) должен соответствовать одному из следующих типов: возрастающая, убывающая, частично упорядоченная, с одинаковыми элементами, со случайными элементами.

2. Число элементов (далее num) в последовательности не должно превышать 10000000 и не быть меньше 1.

Выходные данные: файл, содержащий последовательность заданного типа и размерности, состоящую из целых чисел, разделённых пробелом. Количество элементов в файле задано пользователем. Элементом файла является целое число. Файл имеет своё уникальное имя вида: «Имя».«Расширение»; «Имя»::= «type»\_ «num»\_ «номер файла с последовательностью»; «Расширение»::= «txt»; «type» ∈ {‘up’, ‘down’, ‘part’, ‘same’, ‘def’}, где ‘up’ - возрастающая, ‘down’ – убывающая, ‘part’ – частично упорядоченная, ‘same’ – с одинаковыми элементами, ‘def’ – со случайными элементами; «num» ∈ {1..10000000}; «Номер файла с последовательностью» ∈ {0, 1, 2…};

**Функциональные требования для генерации последовательностей:**

1. КП должна позволять вводить: число элементов последовательности и тип генерируемой последовательности.

2. КП должна уметь генерировать несколько последовательностей выбранных видов за 1 генерацию.

3. КП должна сообщать пользователю об ошибке, если число элементов, указанное пользователем, не удовлетворяет условиям входных данных.

4. КП должна сохранять сгенерированную последовательность в файл, с автоматически генерируемым именем.

5. КП должна проверять наличие файлов с одинаковыми названиями и, если они существуют, генерируемый файл называть в соответствии с порядковым номером в директории, куда сохраняются файлы.

**Требования к данным в экспериментальном исследовании:**

Входные данные:

Список файлов, каждый из которых:

1. Является текстовым файлом, в котором должна быть записана равно одна последовательность. Все элементы данной последовательности – целые числа, в качестве разделяющего элемента выступает пробел. Количество элементов в файле не может быть меньше 1 и превышать 10000000.

2. Имеет своё уникальное имя вида, которое определяется аналогично имени файла при генерации последовательности.

Выходные данные

1. Гистограммы:

а) На первой гистограмме будет отображена зависимость времени выполнения алгоритма сортировки от выбранного типа последовательности и числа элементов в этой последовательности.

б) На второй гистограмме будет отображена зависимость числа сравнений, которые произвела КП в ходе выполнения алгоритмов быстрой и интроспективной сортировок, от выбранного типа последовательности и числа элементов в этой последовательности.

в) На третьей гистограмме будет отображена зависимость числа обменов, которые произвела КП в ходе выполнения алгоритмов быстрой и интроспективной сортировок, от выбранного типа последовательности и числа элементов в этой последовательности.

2. Список файлов, каждый из которых:

а) Является текстовым файлом, в котором должна быть записана ровно одна отсортированная последовательность. Все элементы данной последовательности взяты из входного файла, который был выбран для проведения эксперимента.

б) Имеет своё уникальное имя вида: «Имя входного файла»\_sorted.«Расширение»

**Функциональные требования для экспериментального исследования**

1. КП должна позволять выбирать файлы для эксперимента в директории компьютера.

2. КП должна уметь выполнять алгоритмы быстрой и интроспективной сортировок.

3. КП должна уметь засекать время работы быстрой и интроспективной сортировок для сортируемых последовательностей и по полученным данным строить график.

4. КП должна производить подсчёт числа обменов, исполненных во время выполнения алгоритмов быстрой и интроспективной сортировок на заданных последовательностях, и по полученным данным строить график.

5. КП должна производить подсчёт числа сравнений, исполненных во время выполнения алгоритмов быстрой и интроспективной сортировок на заданных последовательностях, и по полученным данным строить график.

**Требования к удобству КП**

1. КП должна быть предназначена для любого человека, который имеет навык работы на компьютере и понимает все пять типов последовательностей.

2. КП должна выдавать все сообщения на русском языке, за исключением некоторых имён файлов и папок.

3. Интерфейс КП должен быть понятен и дружелюбен

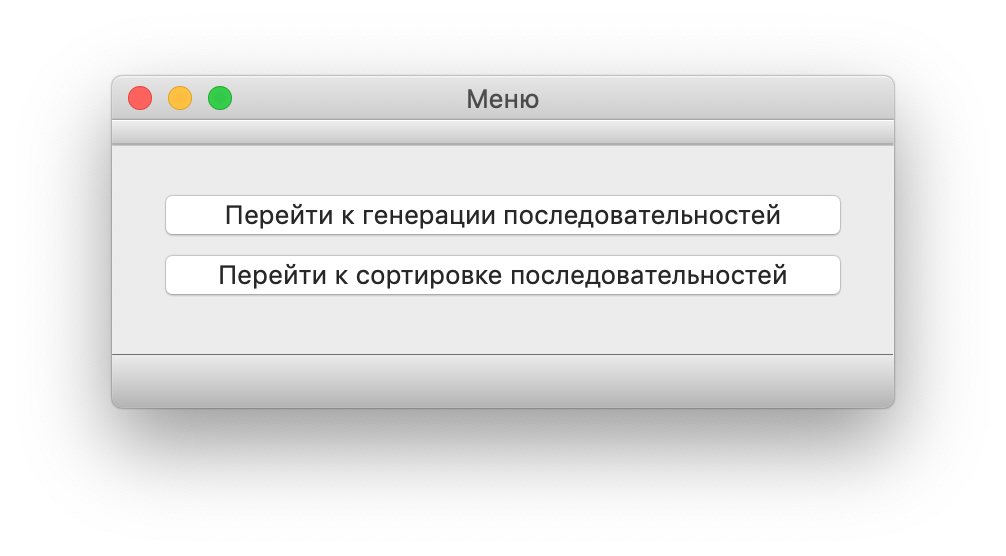
4. Процедура запуска КП должна быть понятна и проста для пользователя

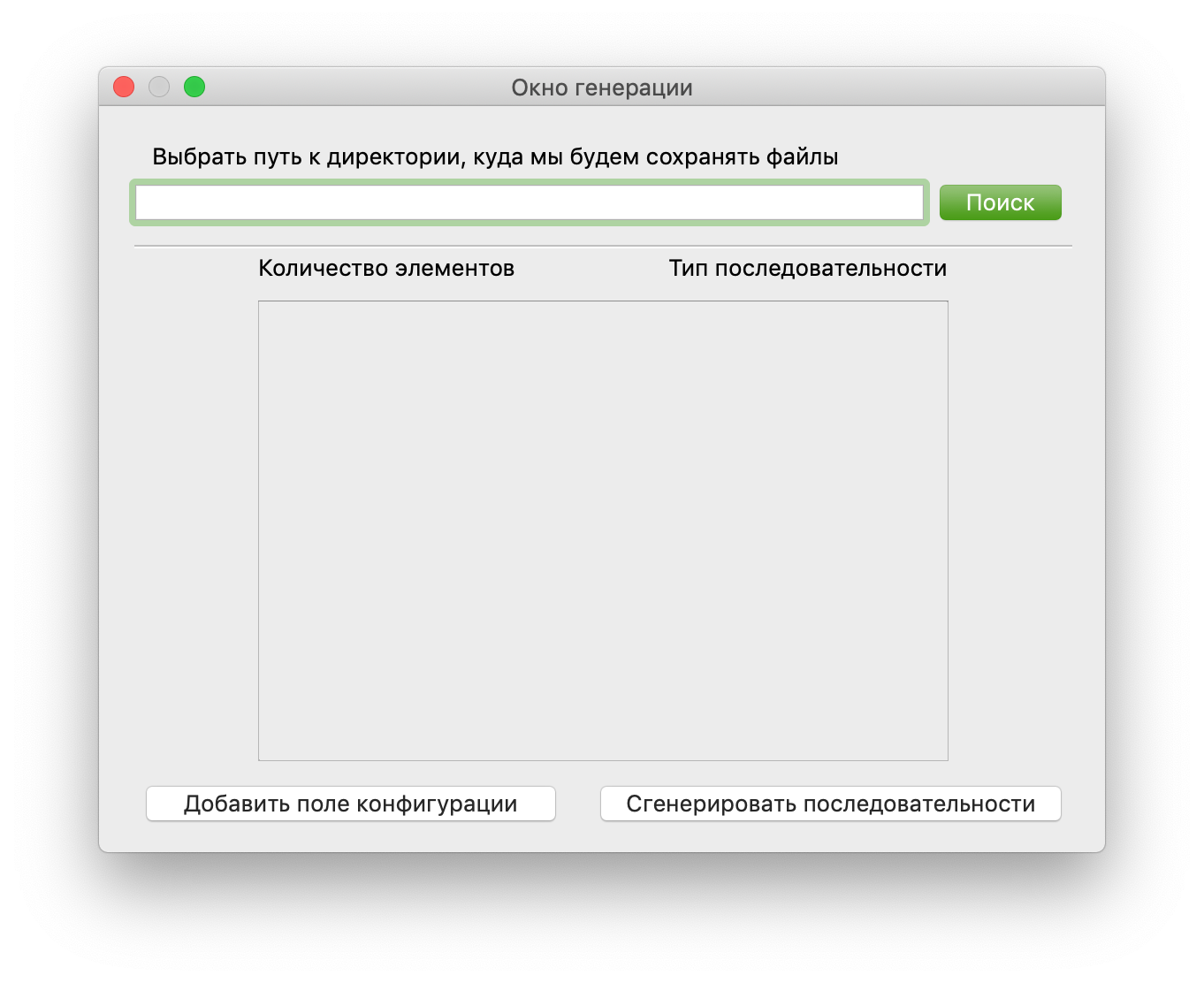
**Требования к мобильности КП**

КП должна быть переносимой без изменений из одной среды в другую в рамках Vista/7/8/10 и их 32-х и 64-х разрядные версии.

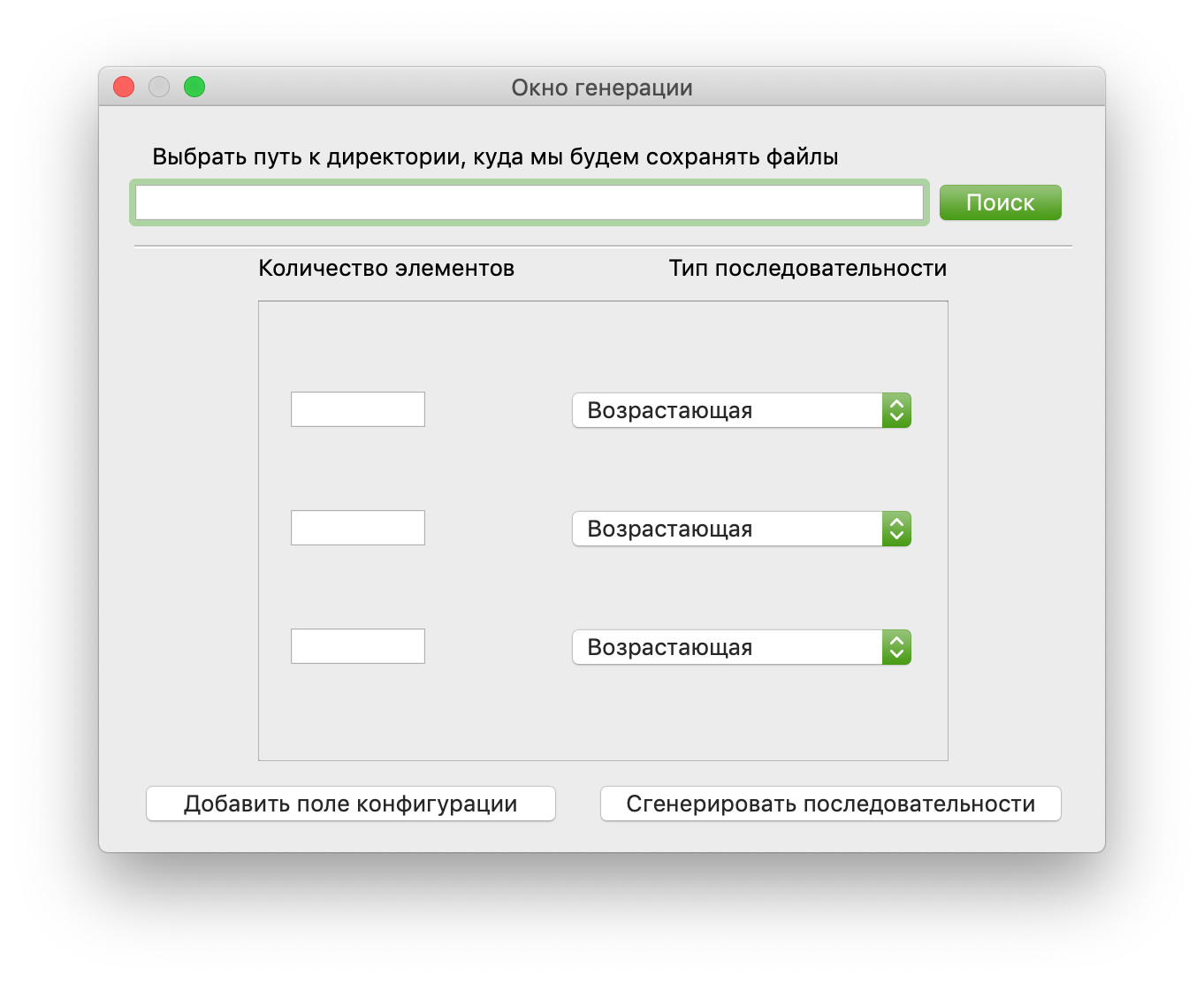
**3 Сценарий диалога КП с пользователем**

При запуске программы открывается главное окно. Главное окно изображено на рисунке 7. На нём изображены кнопки перехода к различным модулям программы. При нажатии кнопки «Перейти к генерации последовательностей» произойдёт переход к модулю генерации последовательностей. Окно генерации последовательности изображено на рисунке 8. При нажатии кнопки «Перейти к сортировке последовательностей» произойдёт переход к модулю экспериментального исследования, окно экспериментального исследования изображено на рисунке 15 стр.21.

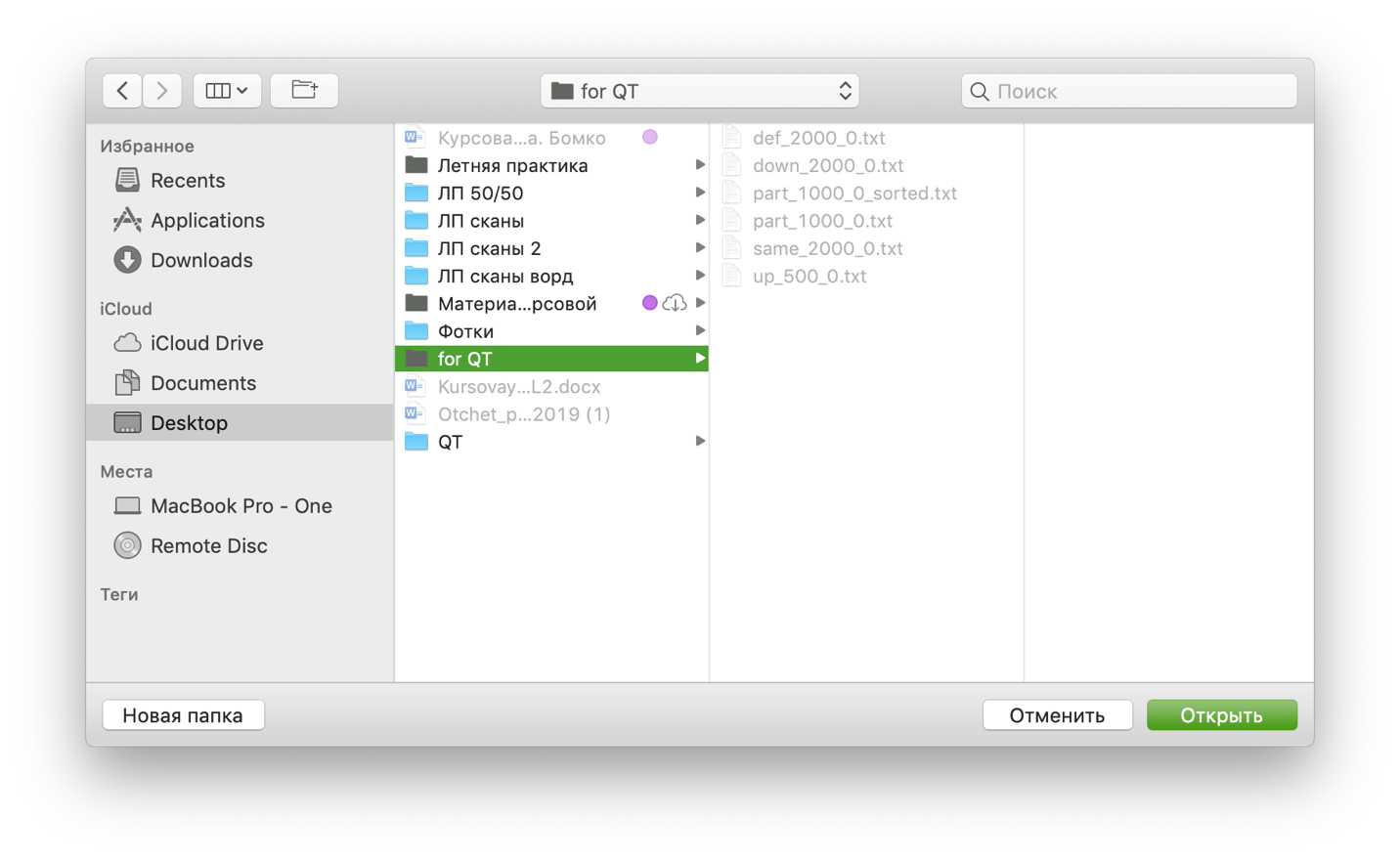
Рисунок 7 – Главное окно.

Рисунок 8 – Модуль генерации последовательностей.

Изначально, модуль генерации последовательности не содержит полей для конфигурации генерируемых файлов. Чтобы добавить поля конфигурации, необходимо нажать на кнопку «Добавить поле конфигурации». За одно нажатие кнопки создаётся ровно 1 поле. Далее рассмотрим окно, в котором добавлено 3 поля. Окно с тремя полями изображено на рисунке 9.

Рисунок 9 – Модуль генерации последовательностей с полями.

При нажатии кнопки «Поиск», открывается окно выбора директории, куда будут сохраняться сгенерированные файлы. Окно выбора директории изображено на рисунке 10. После того как будет выбрана директория сохранения файлов, следует приступить к конфигурации генерируемых последовательностей. В полях, находящихся под надписью «Количество элементов», будет записываться количество элементов в генерируемой последовательности. Справа от этого поля, в выпадающем списке, будем выбирать тип генерируемой последовательности. Выпадающий список выбора типа генерируемой последовательности изображено на рисунке 11. После того как все поля будут сконфигурированы, нажаитие кнопки «Сгенерировать последовательности» приведёт к проверке программой всех полей и выводу ошибок, если поле с количеством элементов будет неверно заполнено, сообщение об этой ошибке можно увидеть на рисунке 12. Если при заполнении полей были встречены ошибки, то программа их проигнорирует и продолжит выполнять генерацию, начиная со следующего поля. Если при генерации все поля оказались неверно заполненными, то программа выведет сообщение о том, что ни одна последовательность не сгенерирована, окно вывода этой ошибки можно увидеть на рисунке 13, в остальных случаях, выведет сообщение о том, что сгенерированы все возможные последовательности, это сообщение изображено на рисунке 14.

Рисунок 10 – Выбор директории сохранения файлов.

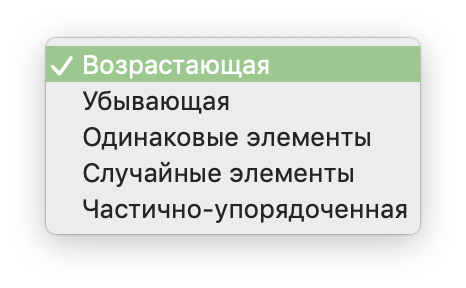
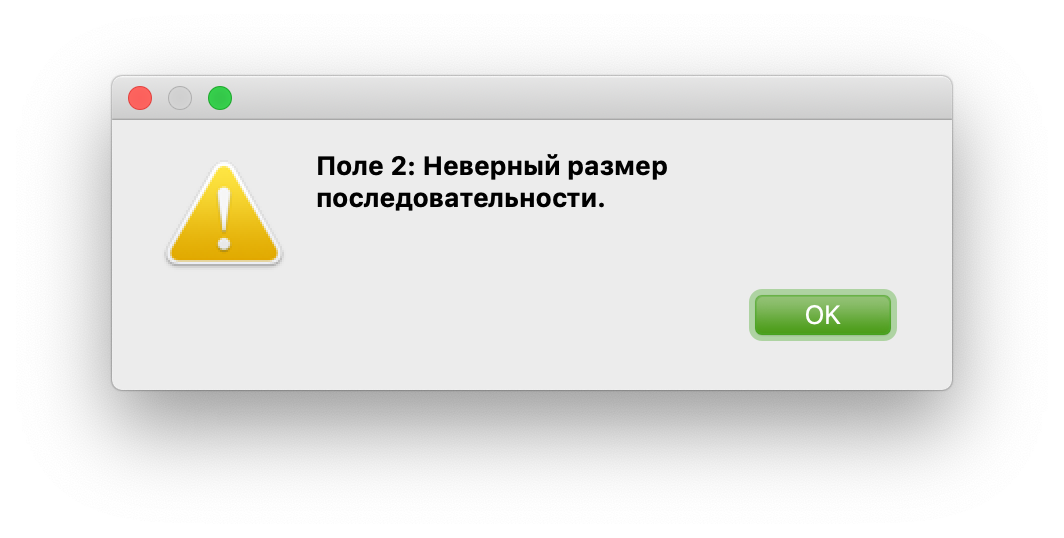
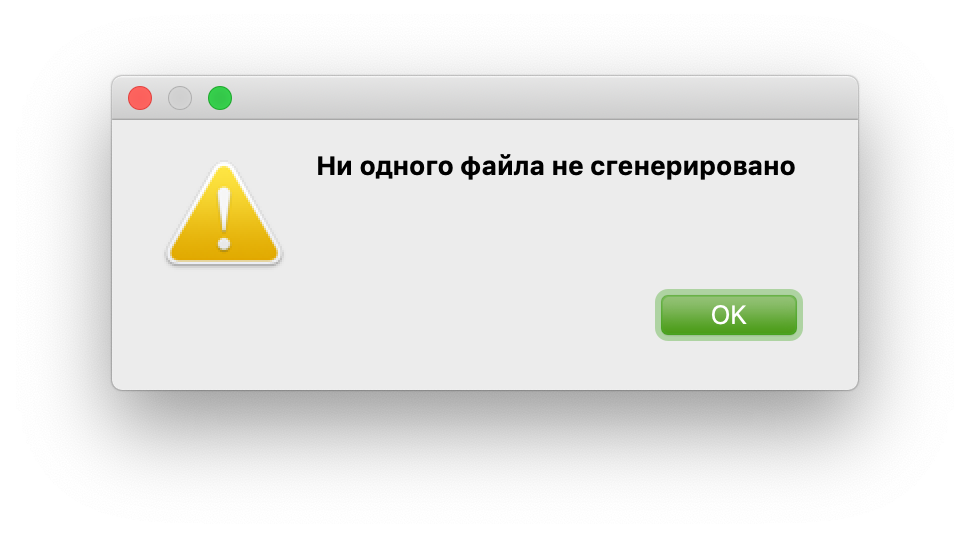
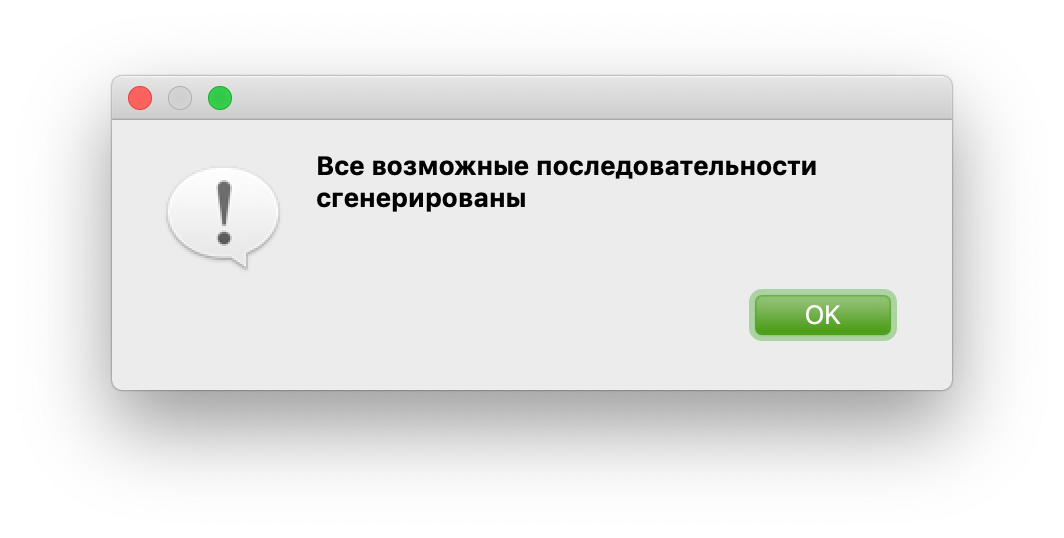
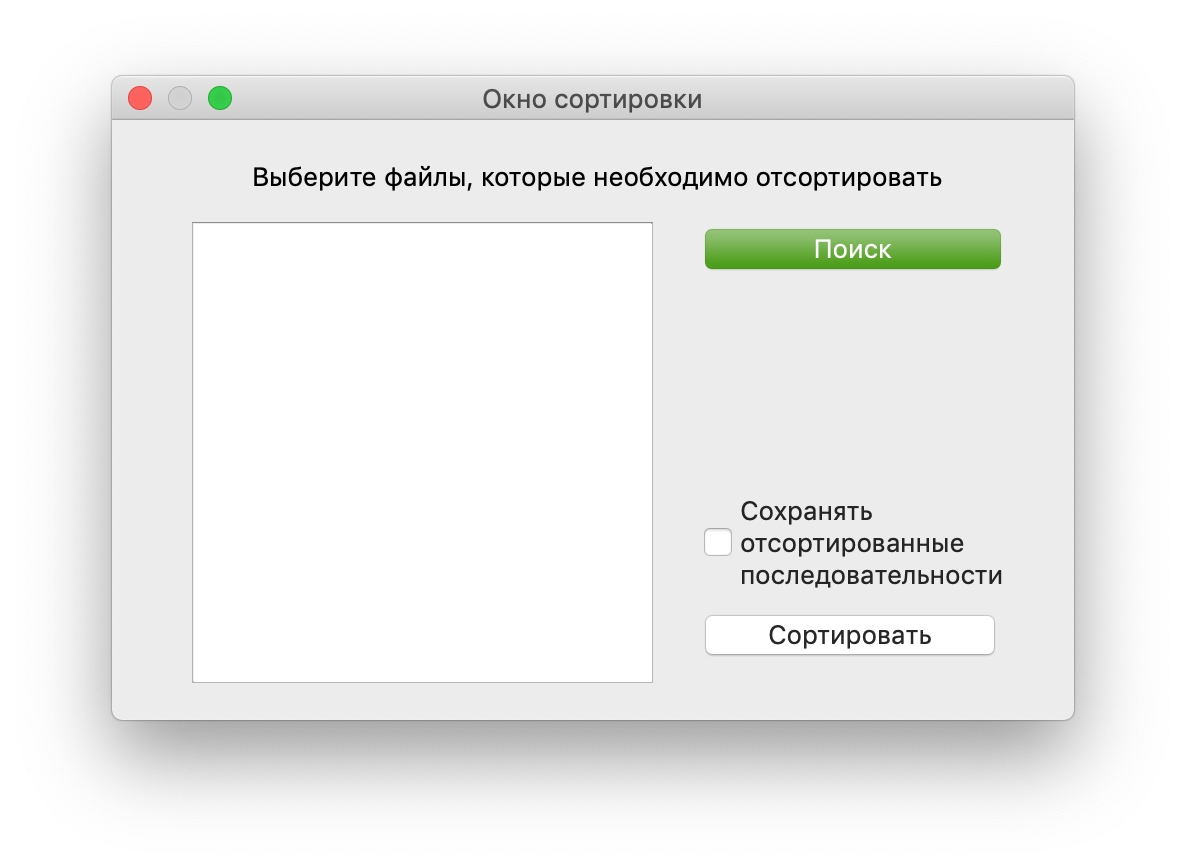


Рисунок 11 – Выбор типа последовательности.

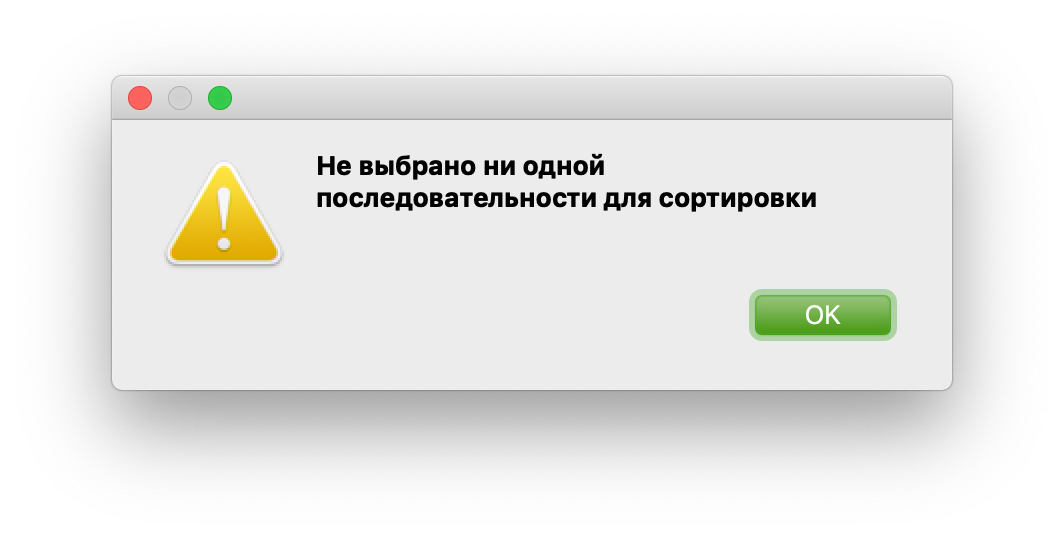
Рисунок 12 – Ошибка: Неверны назмер последовательности.

Рисунок 13 – Ошибка: Ни одного файла не сгенерировано.

Рисунок 14 – Все возможные последовательности сгенерированы.

Рисунок 15 – Модуль экспериментального исследования.

При нажатии кнопки «Поиск», откроется окно выбора файлов, в этом окне следует выбрать файлы, с помощью которые мы хотим использовать для экспертимента. Эти файлы будут добавлены в список выбранных файлов. Если пользователь поставит галочку слева от надписи «Сохранять отсортированные последовательности», то все отсортированные последовательности будут сохранены в той же папке, откуда были взяты последовательности для эксперимента. При нажатии кнопки «Сортировать», программа проверит список выбранных файлов, если в списке не будет ни одного файла, то программа выдаст ошибку представленную на рисунке 16. Если в списке выбранных файлов будет хотя бы одна запись, то программа произведёт сортировки, посчитает количество перестановок, сравнений и затраченное время и выведет эти данные в виде 3х гистограмм. Все типы гистограмм можно увидеть на рисунках 17, 18 и 19.

Рисунок 16 – Ошибка: Не выбрано ни одной последовательности.

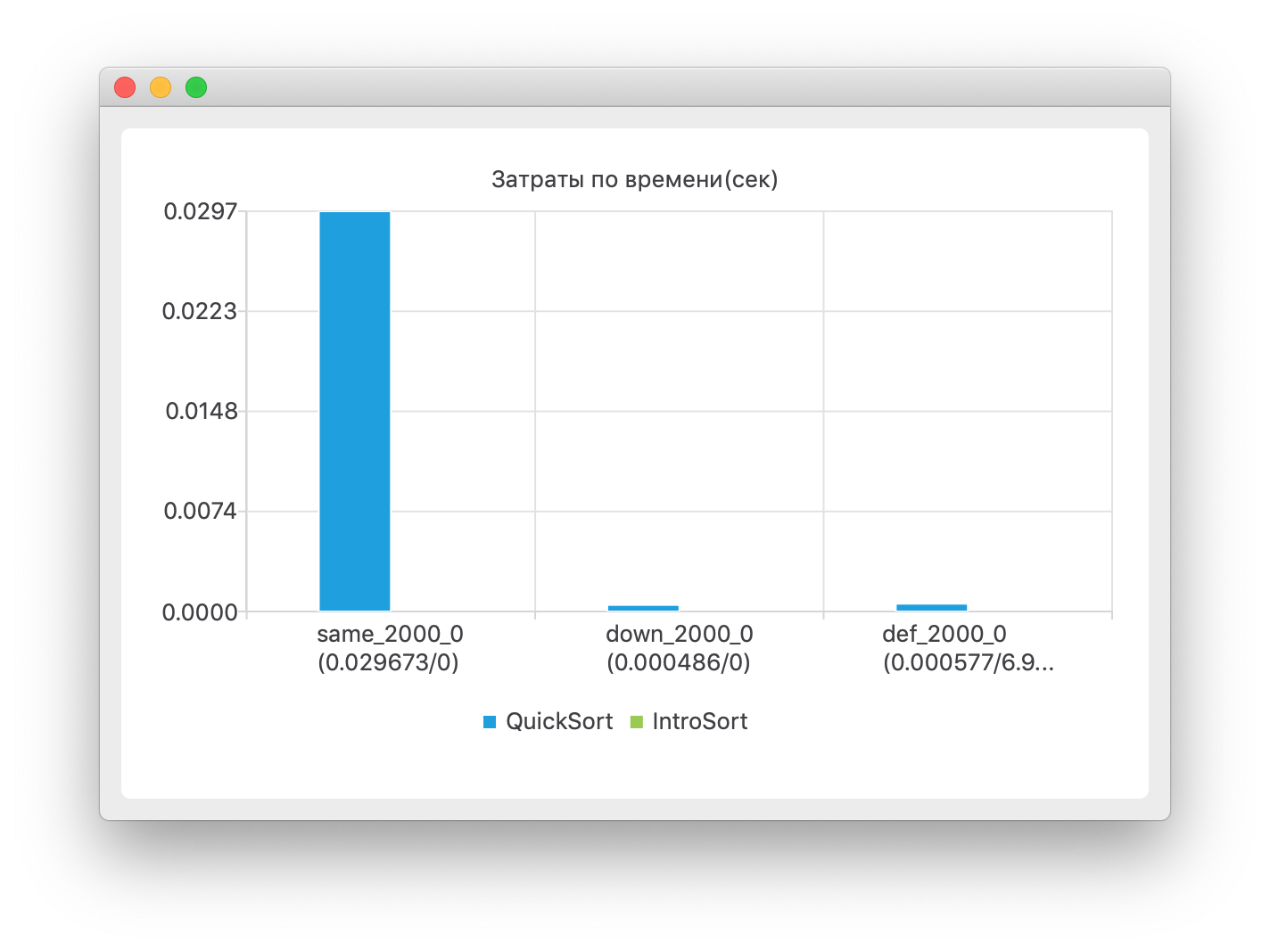


Рисунок 17 – Гистограмма времени.

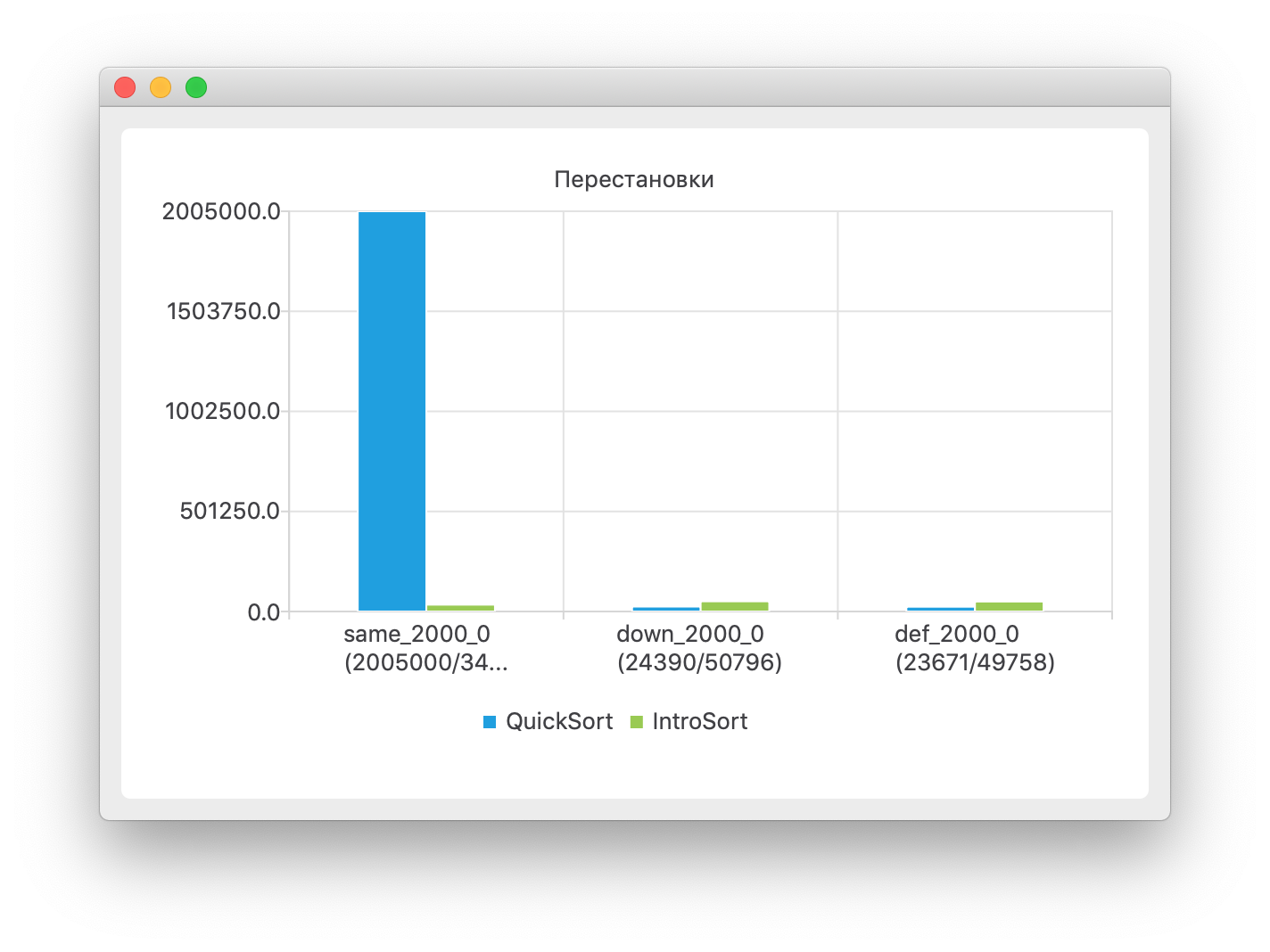


Рисунок 18 – Гистограмма перестановок.

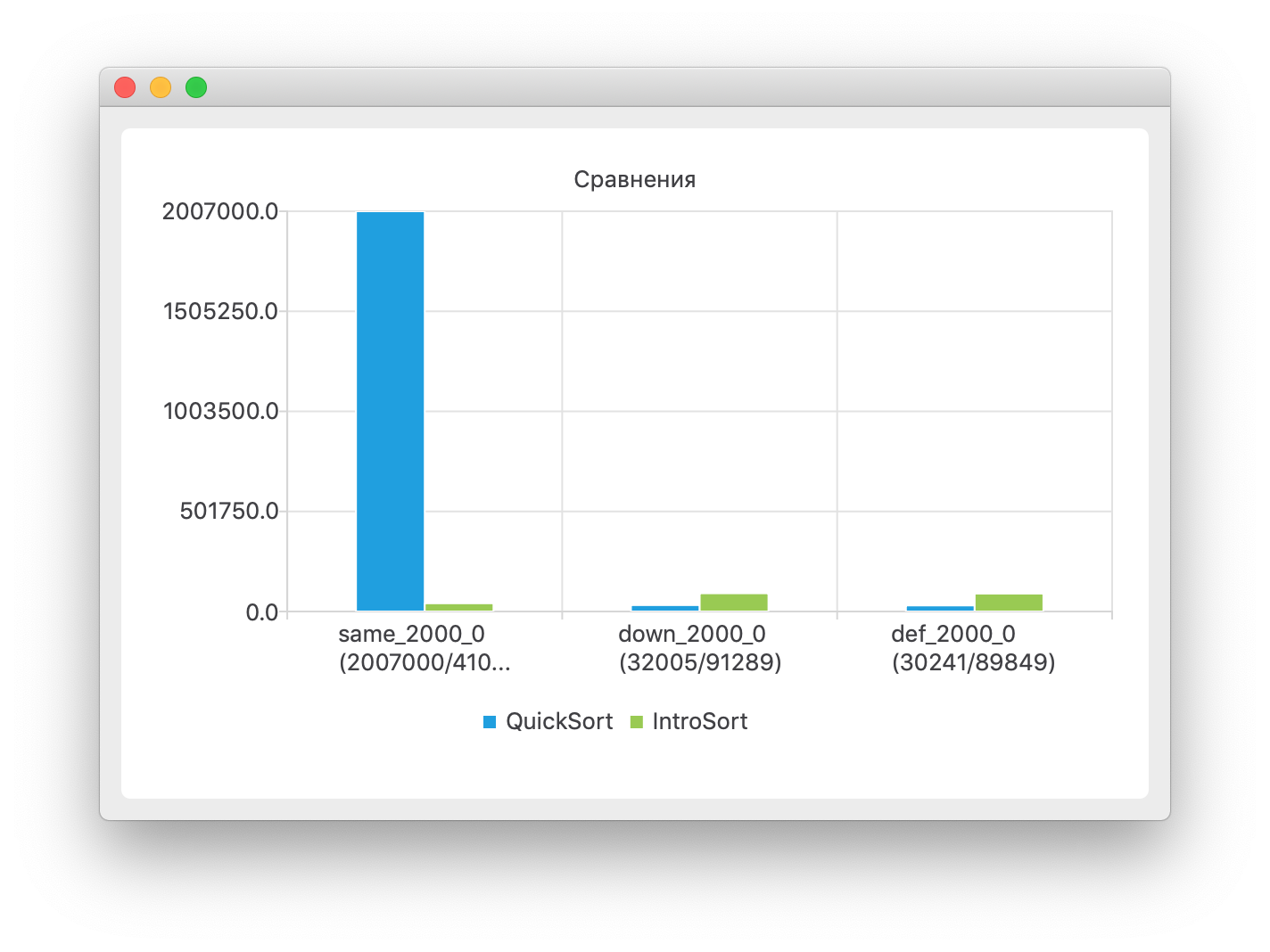


Рисунок 19 - Гистограмма сравнений.

На гистограмме, на оси абцисс отображены названия файлов, с помощью которых мы производим эксперимент, а под ними отображены численные значения сравниваемой характеристики.

1. **Экспериментальные исследования**

Для проведения экспериментальных исследований были выбраны следующие наборы последовательностей.

1. Пять случайных последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

2. Пять частично-упорядоченных последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

3. Пять убывающих последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

4. Пять возрастающих последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

5. Пять последовательностей с одинаковыми элементами размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

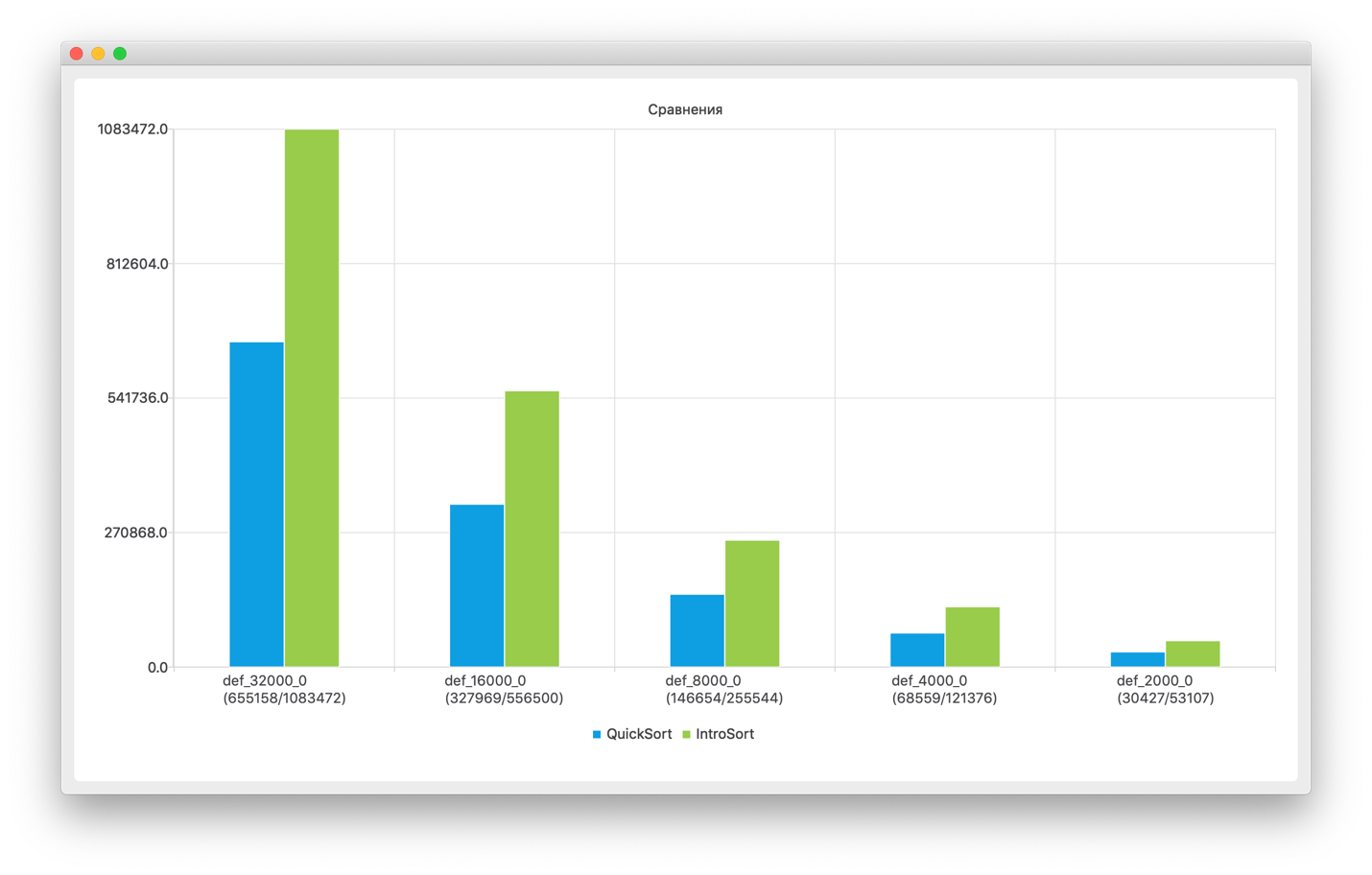
* 1. **Эксперимент над первой группой:**

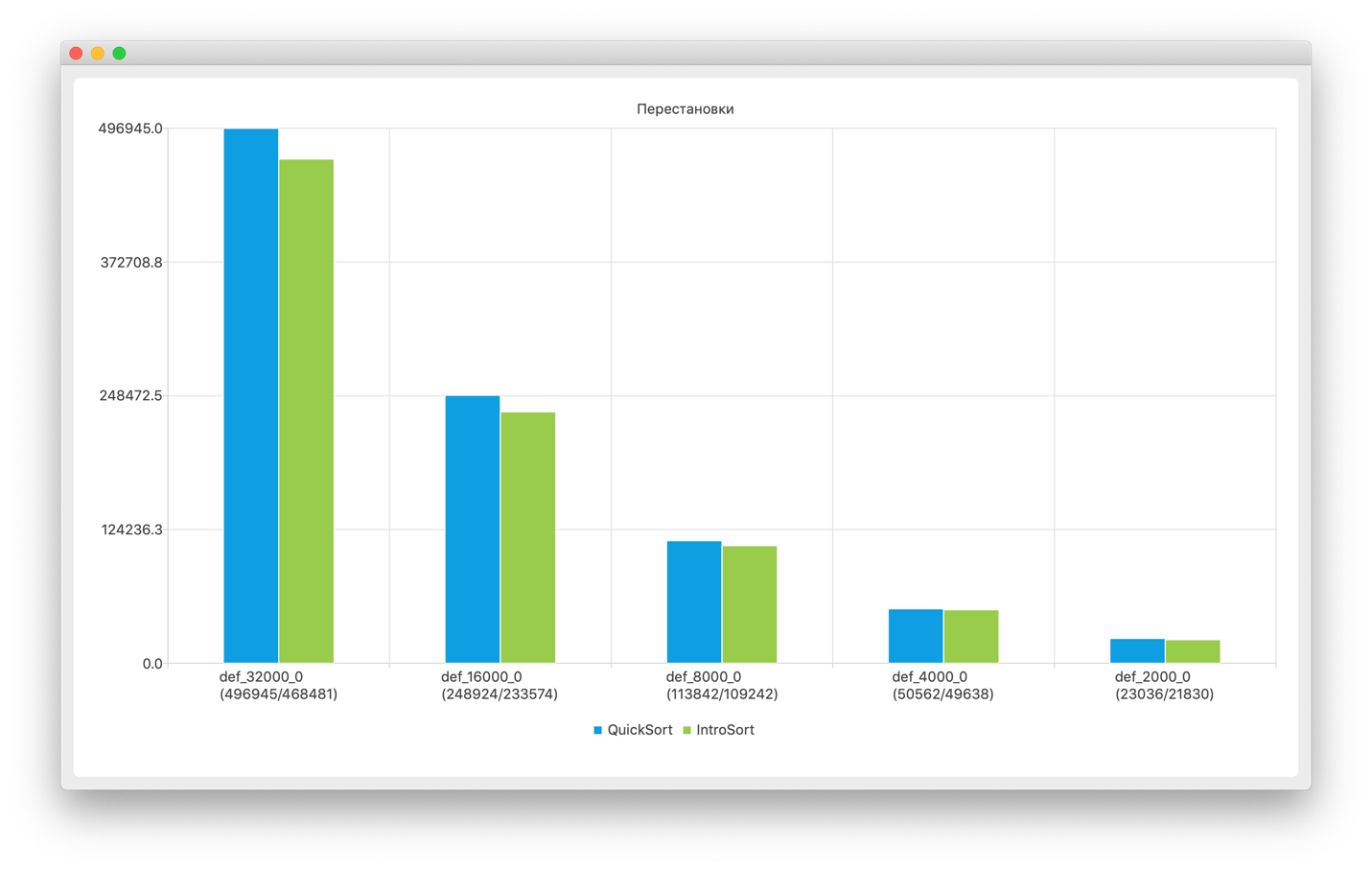
**Цель эксперимента:** Доказать, что на случайных последовательностях, быстрая сортировка справляется лучше интроспективной.

**Входные данные:** Пять случайных последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

**Выходные данные:** Результаты эксперимента над первой группой последовательностей представлены на рисунках 20, 21 и 22.

Рисунок 20 – Гистограмма времени для случайных последовательностей.

Рисунок 21 – Гистограмма количества сравнений для случайных последовательностей.

Рисунок 22 – Гистограмма количества перестановок для случайных последовательностей.

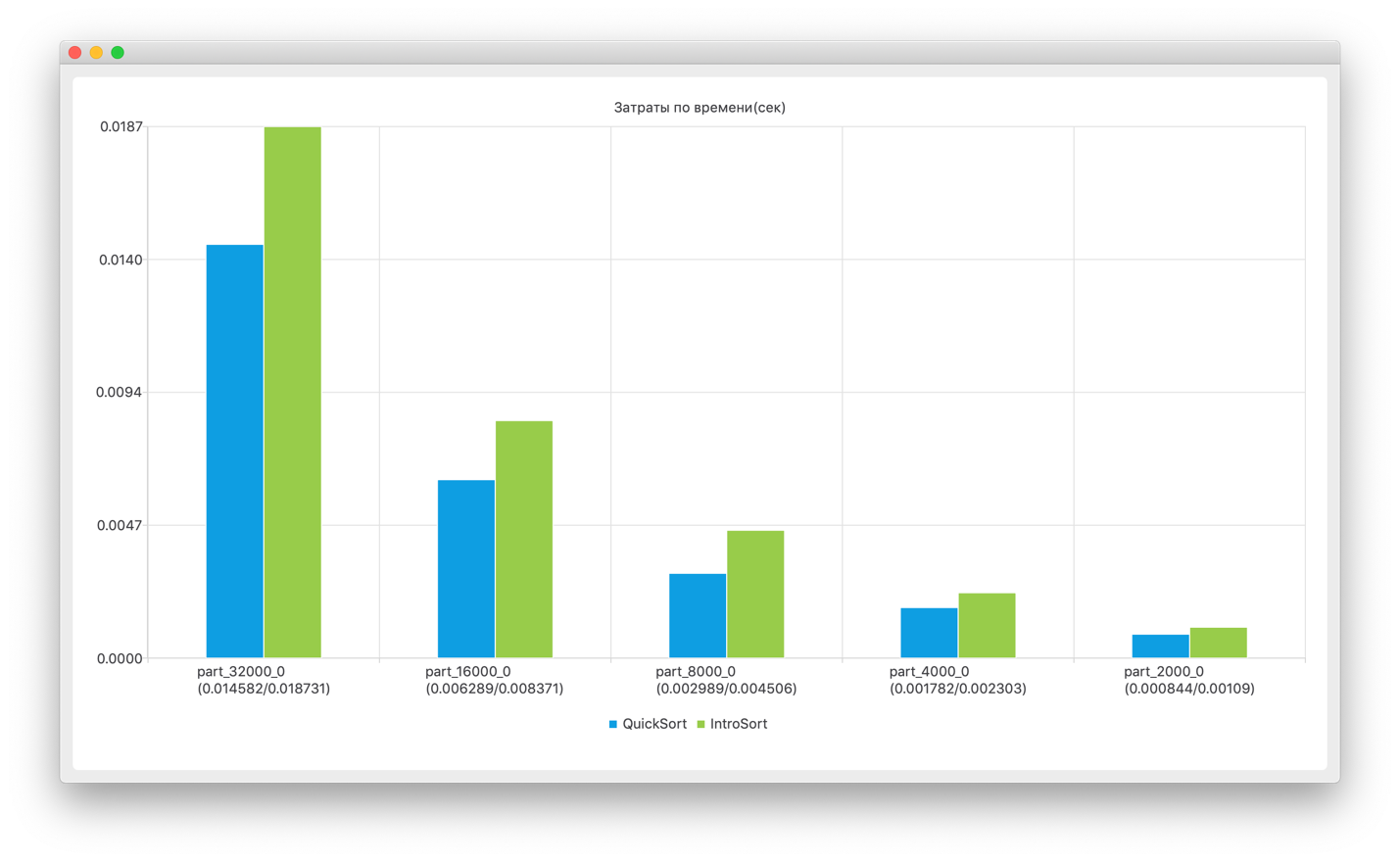
**Обоснование результата:** Как видно из эксперимента, интроспективная и быстрая сортировки, для случайной последовательности требуют примерно одинаковое количество обменов, что подтверждает сложность той и другой сортировки. Сложность для обеих сортировок при среднем случае составляет *O(NlogN)*

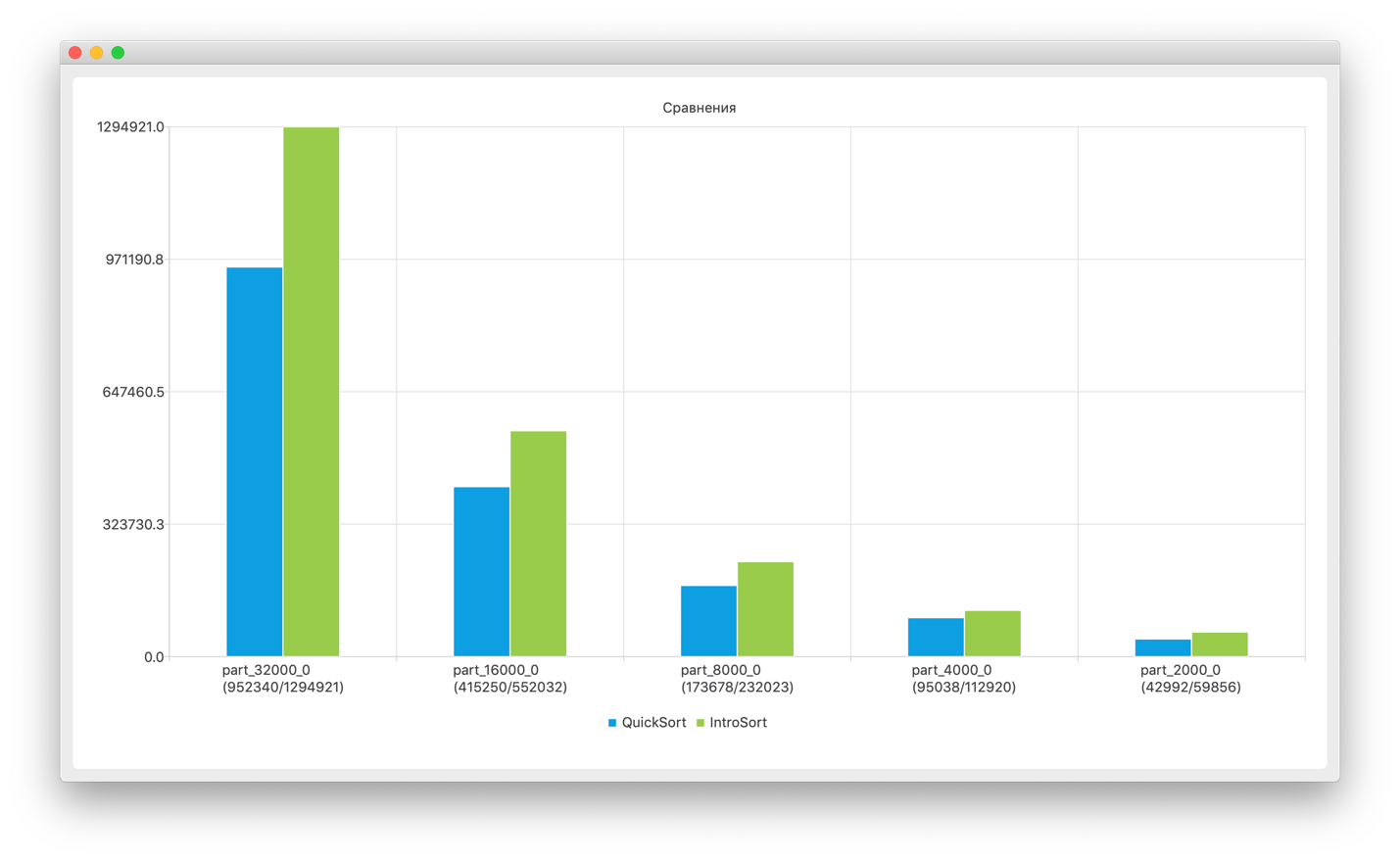
* 1. **Эксперимент над второй группой:**

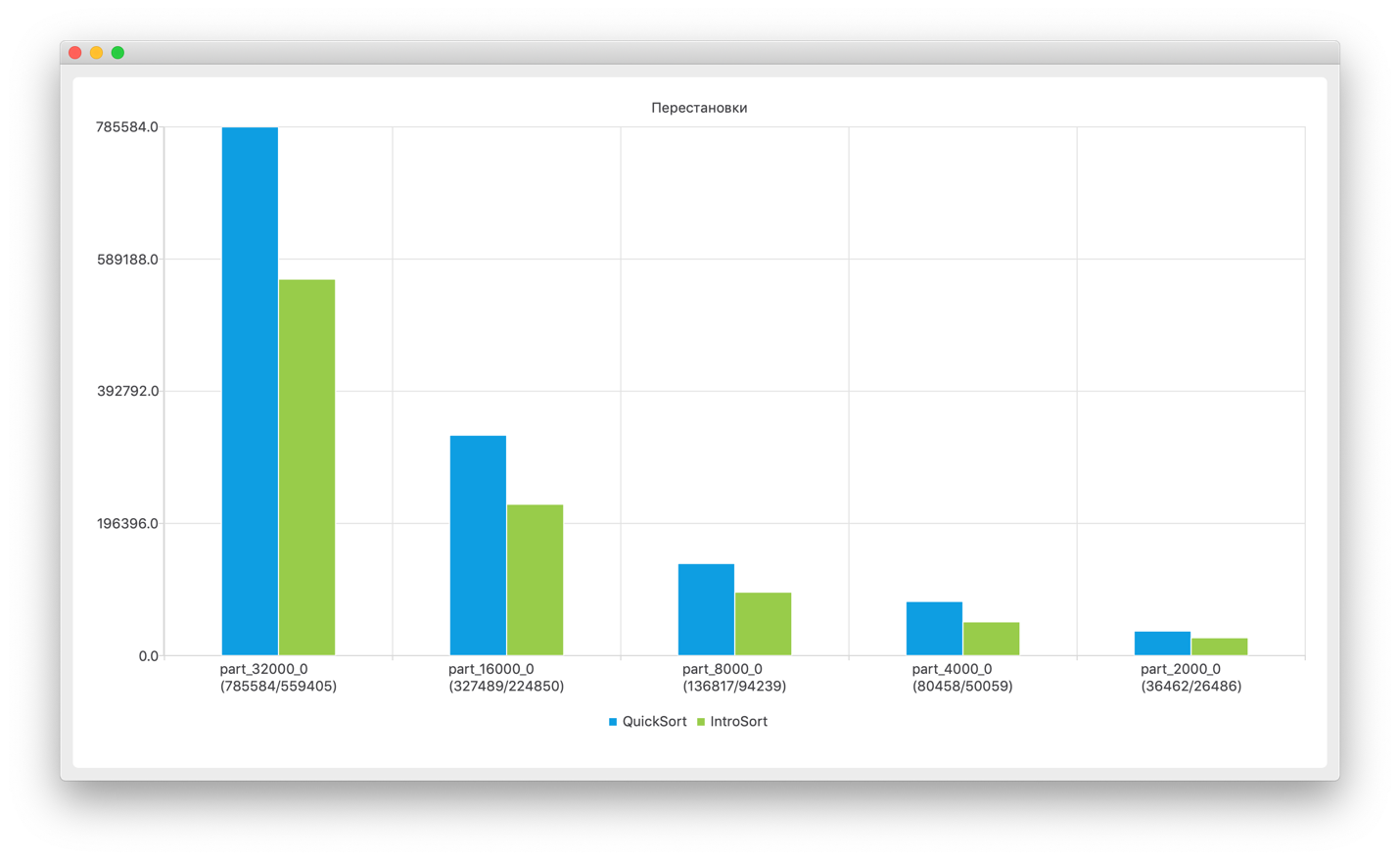
**Цель эксперимента:** Доказать, что на частично-упорядоченных последовательностях, быстрая сортировка справляется лучше интроспективной.

**Входные данные:** Пять частично-упорядоченных последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

**Выходные данные:** Результаты эксперимента над первой группой последовательностей представлены на рисунках 23, 24 и 25.

Рисунок 23 – Гистограмма времени для частично-упорядоченных последовательностей.

Рисунок 24 – Гистограмма количества сравнений для частично-упорядоченных последовательностей.

Рисунок 25 – Гистограмма количества перестановок для частично-упорядоченных последовательностей.

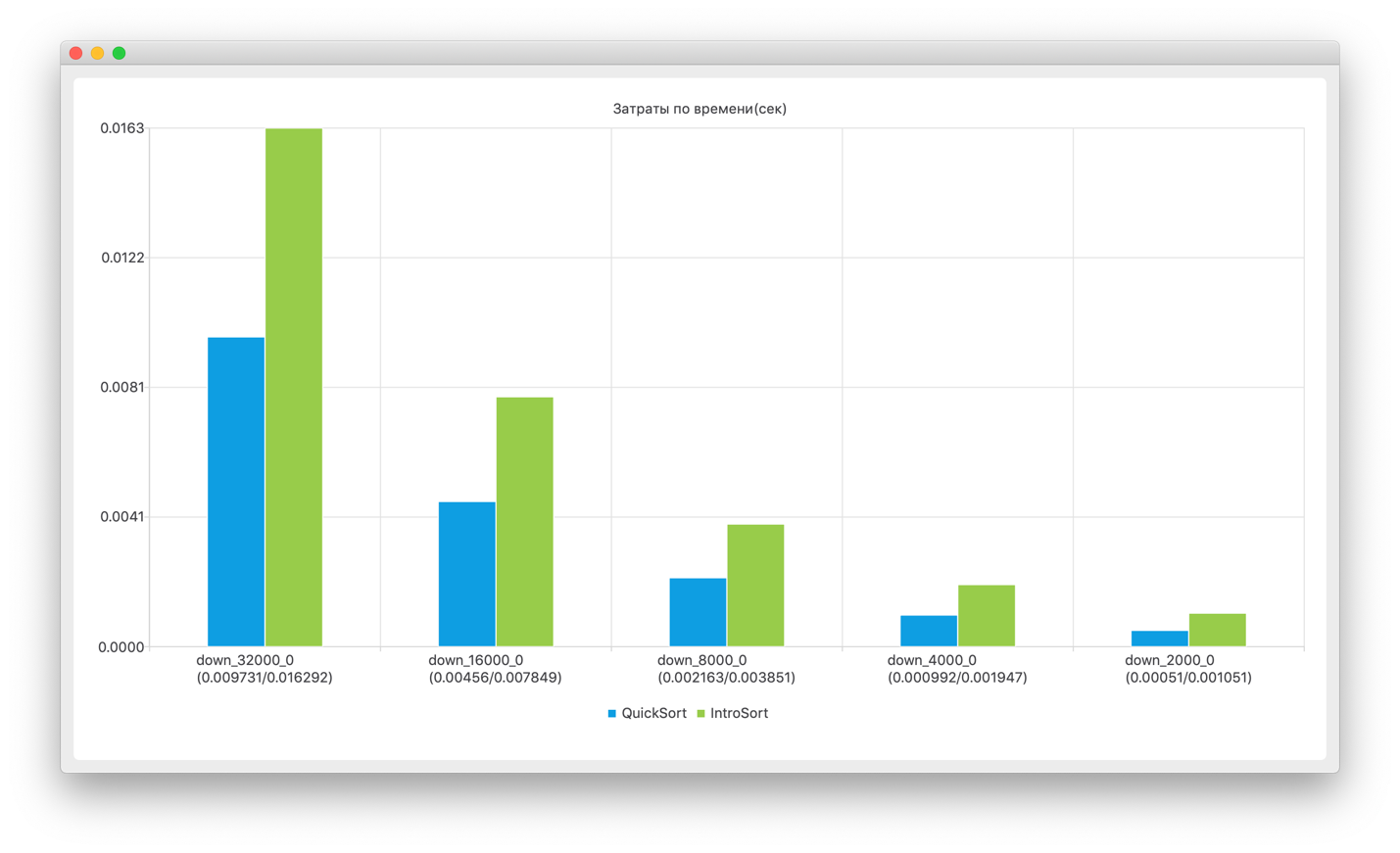
**Обоснование результата:** Как видно из эксперимента, интроспективная сортировка справляется с сортировкой массива, в среднем, за меньшее количество обменов. Этот факт объясняется тем, что отсортированная последоваетльность для быстрой сортировки – худший случай, отсюда ясно, что внутри последовательности существуют отсортированные куски, которые тормозят выполнение сортировки. В среднем, интроспективная и быстрая сортировка подтверждают свои оценки сложности, на данном типе последовательности.

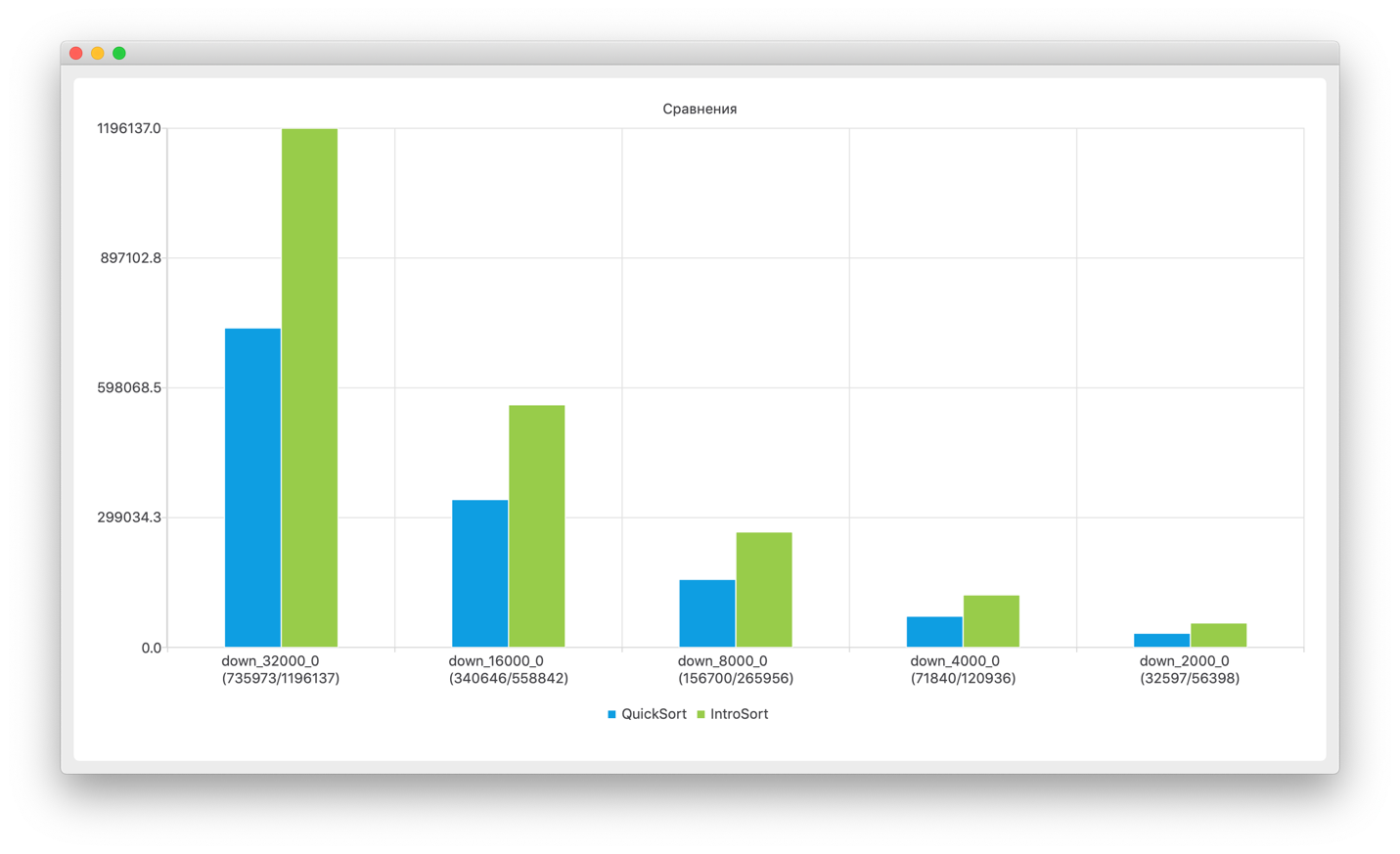
* 1. **Эксперимент над третьей группой:**

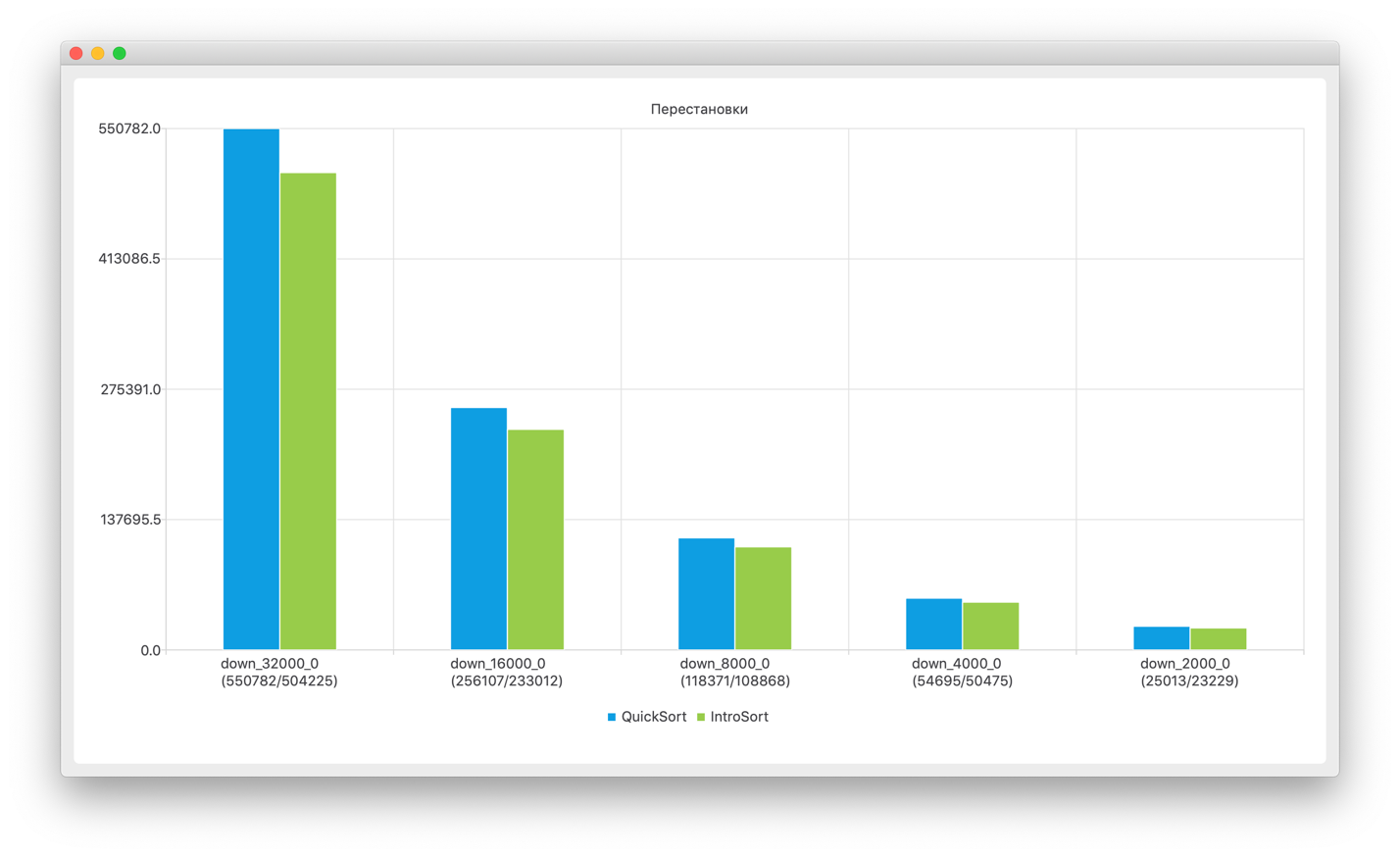
**Цель эксперимента:** Доказать, что на убывающих последовательностях, быстрая сортировка справляется лучше интроспективной.

**Входные данные:** Пять убывающих последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

**Выходные данные:** Результаты эксперимента над первой группой последовательностей представлены на рисунках 26, 27 и 28.

Рисунок 26 – Гистограмма времени для убывающих последовательностей.

Рисунок 27 – Гистограмма количества сравнений для убывающих последовательностей.

Рисунок 28 – Гистограмма количества перестановок для убывающих последовательностей.

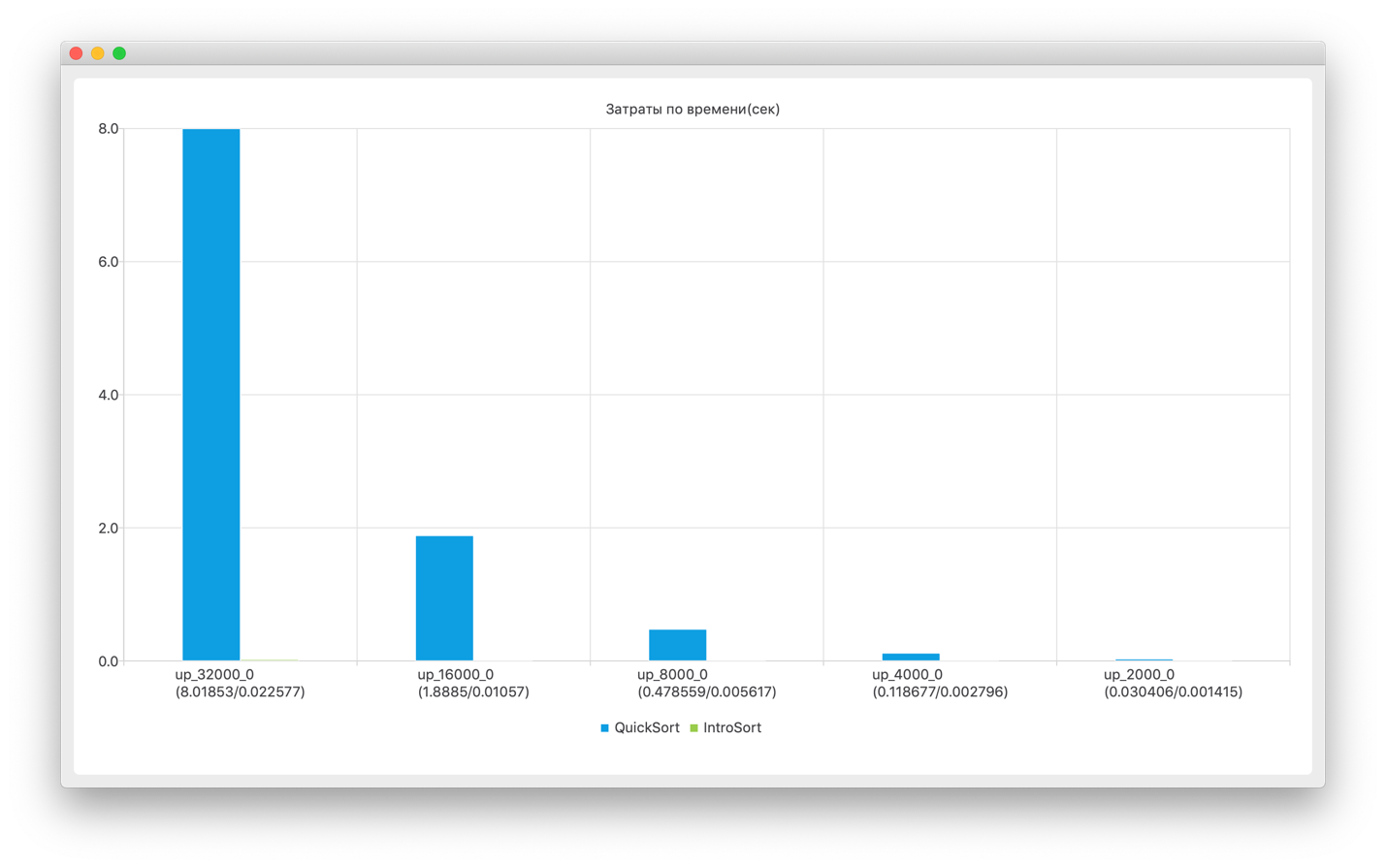
**Обоснование результата:** Как видно из эксперимента, интроспективная и быстрая сортировки, в целом, справляются за одно и то же количество обменов, что подтверждает оценку сложности *O(NlogN)*.

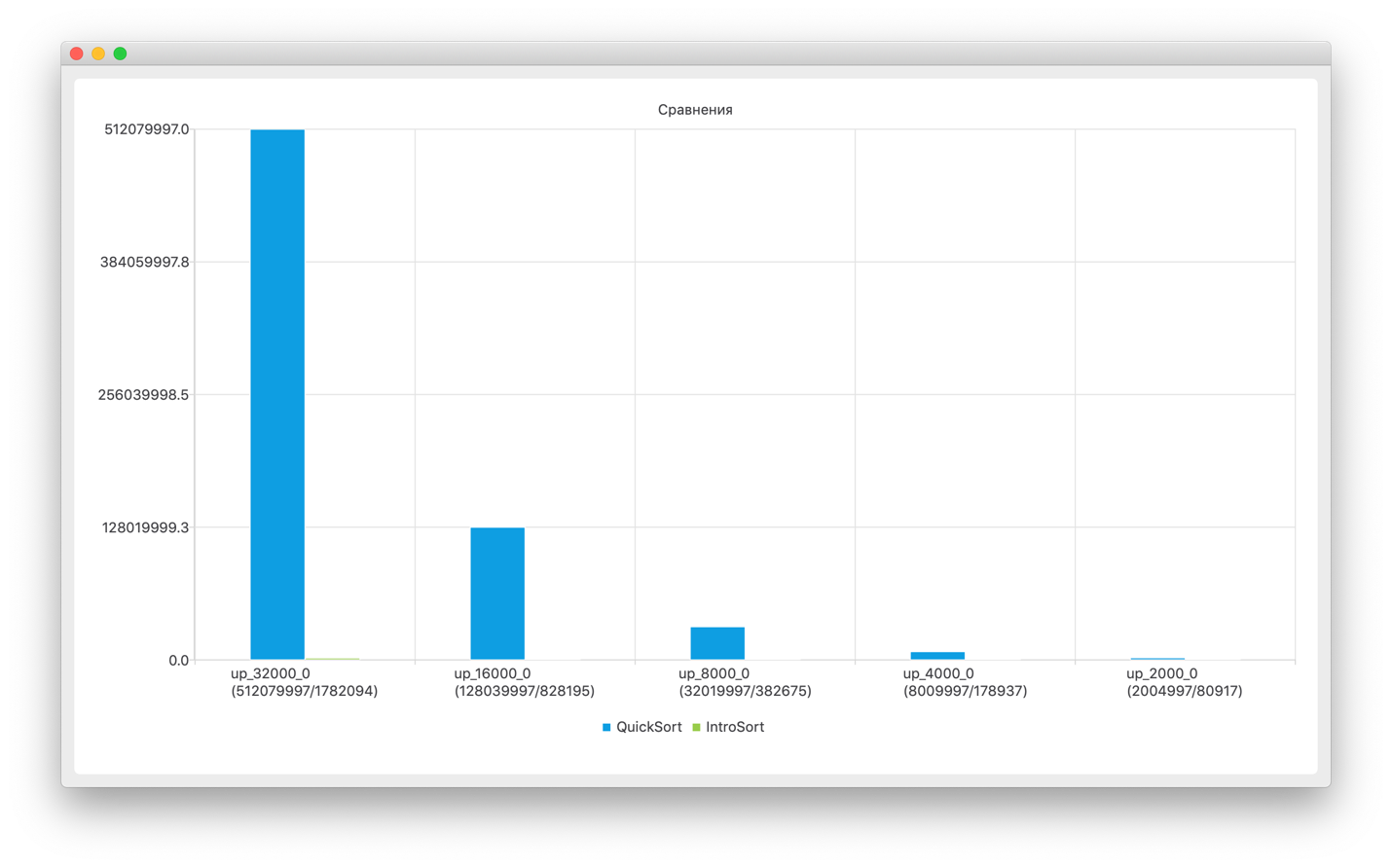
* 1. **Эксперимент над четвёртой группой:**

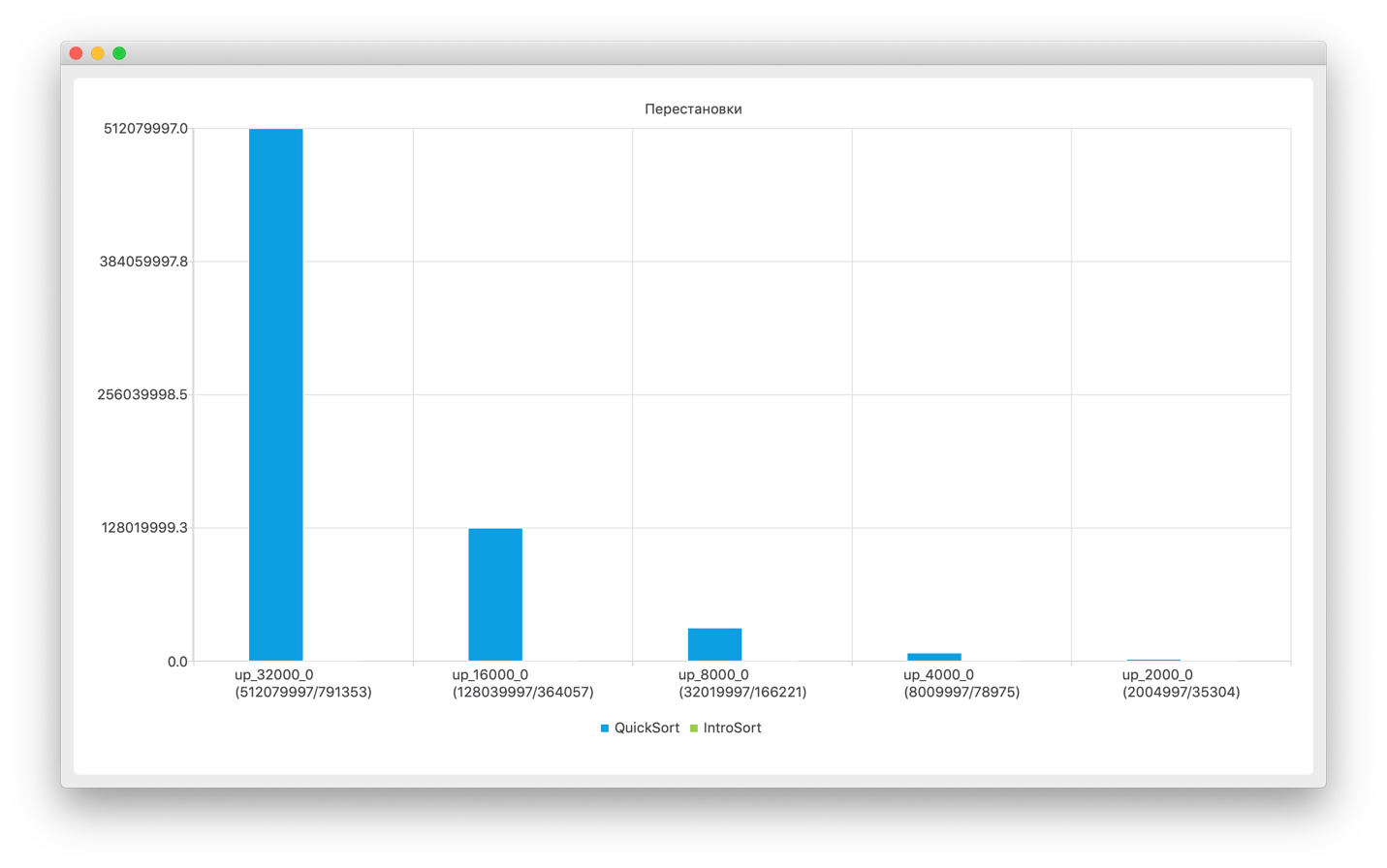
**Цель эксперимента:** Доказать, что на возрастающих последовательностях, быстрая сортировка справляется хуже интроспективной.

**Входные данные:** Пять возрастающих последовательностей размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

**Выходные данные:** Результаты эксперимента над первой группой последовательностей представлены на рисунках 29, 30, 31.

Рисунок 29 – Гистограмма времени для возрастающих последовательностей.

Рисунок 30 – Гистограмма количества сравнений для возрастающих последовательностей.

Рисунок 31 – Гистограмма количества перестановок для возрастающих последовательностей.

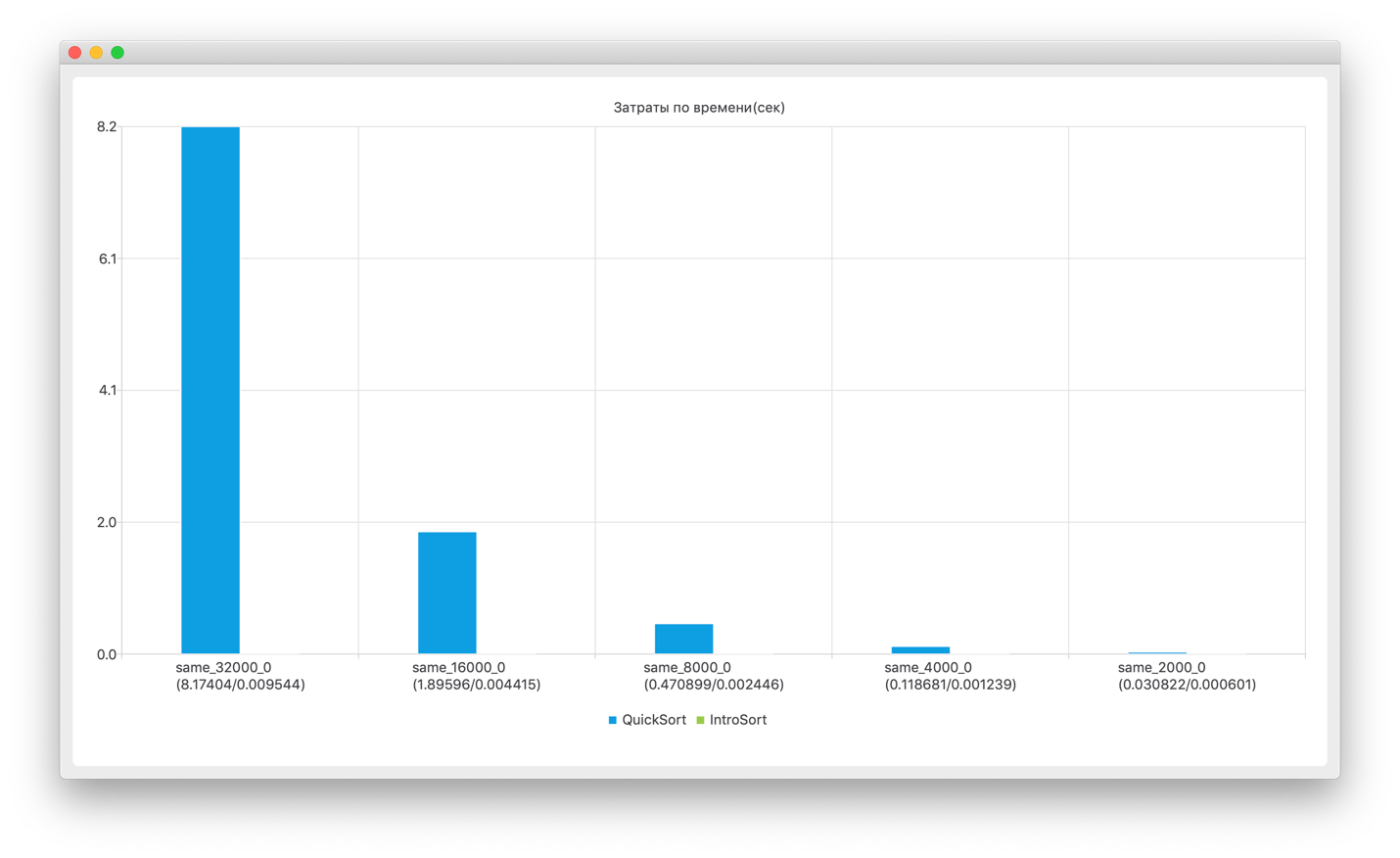
**Обоснование результата:** Как видно из эксперимента, интроспективная сортировка для возрастающей последовательности требует в разы меньше сравнений и перестановок чем быстрая сортировка. В данном эксперименте последовательности выбранные для оценки являются худшим случаем для быстрой сортировки, они наглядно демострируют деградацию сложности быстрой сортировки до *O(N2)*.

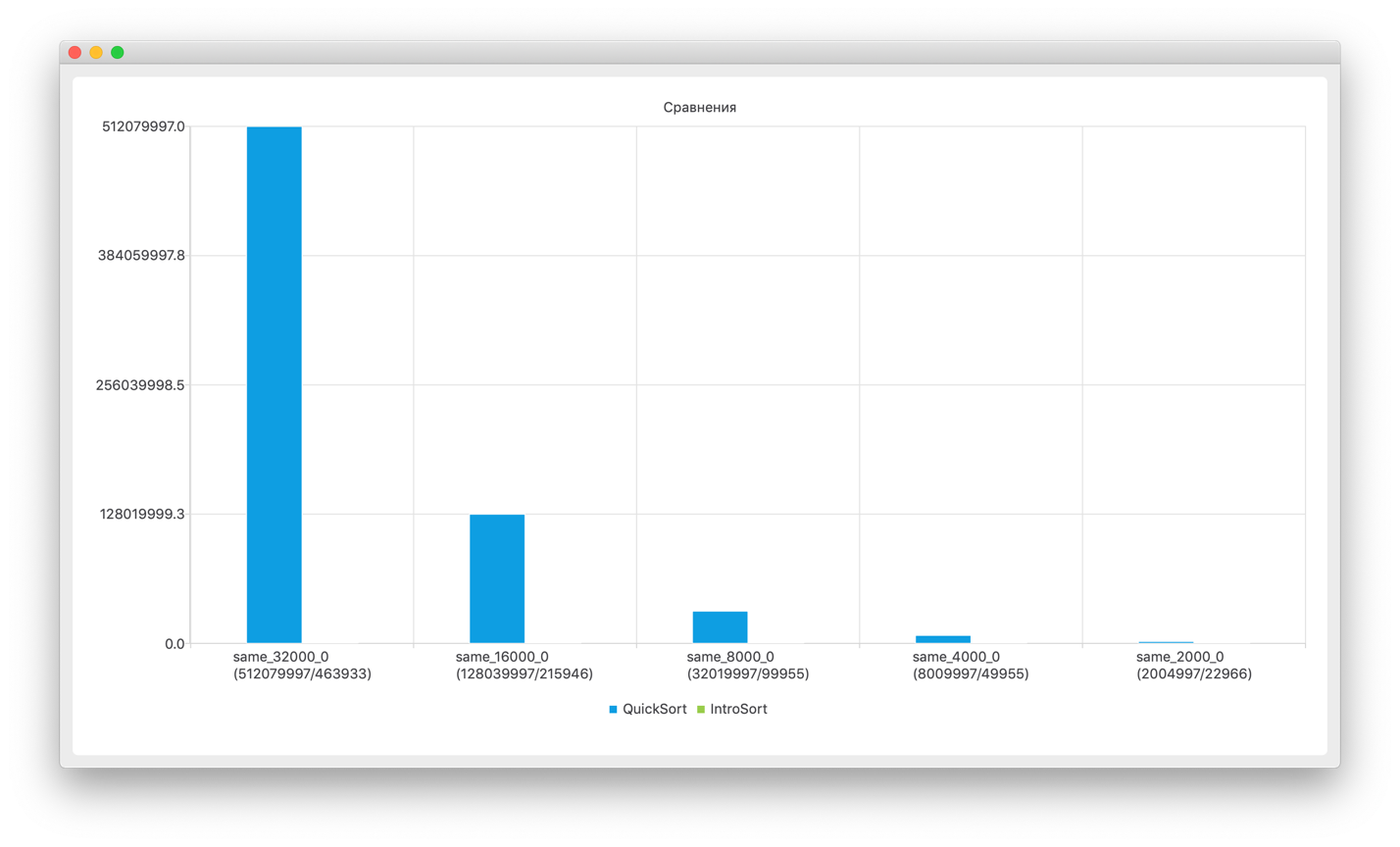
* 1. **Эксперимент над пятой группой:**

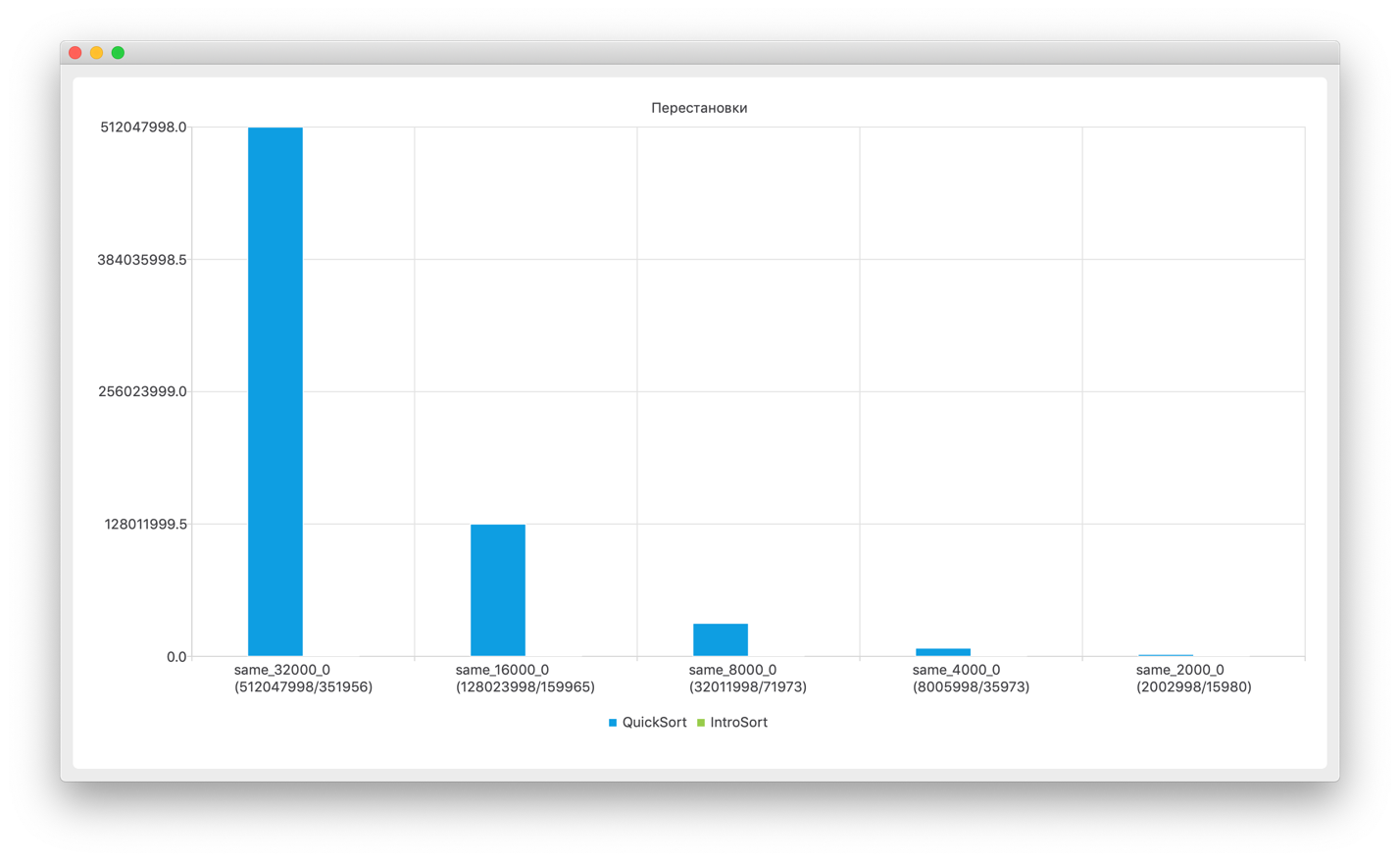
**Цель эксперимента:**Доказать, что на последовательностях с одинаковыми элементами, быстрая сортировка справляется хуже интроспективной.

**Входные данные:** Пять последовательностей с одинаковыми элементами размерности 2000, 4000, 8000, 16000 и 32000 соответственно.

**Выходные данные:** Результаты эксперимента над первой группой последовательностей представлены на рисунках 32, 33 и 34.

Рисунок 32 – Гистограмма времени для последовательностей с одинаковыми элементами.

Рисунок 33 – Гистограмма количества сравнений для последовательностей с одинаковыми элементами.

Рисунок 34 – Гистограмма количества перестановок для последовательностей с одинаковыми элементами.

**Обоснование результата:** Как видно из эксперимента, интроспективная сортировка для последовательности с одинаковыми элементами требует в разы меньше сравнений и перестановок чем быстрая сортировка. В данном эксперименте последовательности выбранные для оценки являются худшим случаем для быстрой сортировки, они наглядно демострируют деградацию сложности быстрой сортировки до *O(N2)*

# Заключение

В рамках курсового проекта:

* был изучен алгоритм быстрой сортировки;
* был изучен алгоритм интроспективной сортировки;
* была проанализирована сложность алгоритма;
* была изучена работа платформы Qt Creator;
* была разработана компьютерная программа для проведения экспериментов;
* был проведен ряд экспериментальных исследований.

Из проведенных экспериментов были сделаны следующие выводы:

* Быстрая сортировка, в среднем, работает так же, как и интроспективная. Сложности обеих сортировок на убывающих, частично-упорядоченных и случайных последовательностях являются средними случаями сортировок и равны O(NlogN) если судить о количестве произведённых обменов.
* Интроспективная сортировка работает более эффективно на возрастающих последовательностях и последовательностях с одинаковыми элементами, чем быстрая сортировка; В данном случае, так как рассматривается худший результат, сложность быстрой сортировки должна быть O(N2). Эти два эксперимента не только подтверждают сложность быстрой сортировки, но и демонстрируют то, что интроспективная сортировка, в среднем, справляется в десятки раз лучше.

**Список литературы**

1. Сэджвик, Р. Алгоритмы на C++, Пер. – Моргунов А.А. Издательство «Вильямс» 2019.

2. Кормен, Т., Лейзерсон, Ч. Ривест, Р. Штайн, К. Алгоритмы: построение и анализ, Пер. – Красиков И. В. Издательство «Вильямс» 2013.

3. Левитин, А. В. Алгоритмы. Введение в разработку и анализ. Издательство «Вильямс» 2006.

**Электронные ресурсы**

4. Мюссер, Д. Introspective Sorting and Selection Algorithms, 1997. [Электронный ресурс] – Режим доступа: http://www.cs.rpi.edu/~musser/gp/introsort.ps