Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ»

КАФЕДРА «ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ

Студент **Ильченко Ева Андреевна**

Группа **ИУ7-24Б**

Тип практики **Проектно-технологическая практика**

Название предприятия **НУК ИУ МГТУ им. Н. Э. Баумана**

Студент _____ _____

Руководитель практики _____ **Ломовской И. В.**

Руководитель практики _____ **Кострицкий А. С.**

Оценка _____

2024 г.

Оглавление

Задание №0. Настройка окружения

Задание

Требуется:

1. Настроить домашнее окружение, установив определённый далее набор пакетов
2. Проверить работоспособность окружения, ответив на представленные далее вопросы

Выводы

Задание выполнено.

Задание №1.1. Автоматизация функционального тестирования

Задание

Требуется:

1. Реализовать скрипты отладочной и релизной сборок.
2. Реализовать скрипты отладочной сборки с санитайзерами.
3. Реализовать скрипт очистки побочных файлов.
4. Реализовать компаратор для сравнения последовательностей действительных чисел, располагающихся в двух текстовых файлах, с игнорированием остального содержи- мого.
5. Реализовать компаратор для сравнения содержимого двух текстовых файлов, рас- полагающегося после первого вхождения подстроки «Result: ␣ ».
6. Реализовать скрипт pos_case.sh для проверки позитивного тестового случая по определённым далее правилам.
7. Реализовать скрипт neg_case.sh для проверки негативного тестового случая по определённым далее правилам.
8. Обеспечить автоматизацию функционального тестирования.

Выводы

Задание выполнено

Задание №1.2. Автоматизация функционального тестирования. Аргументы командной строки приложения

Задание

Требуется:

1. Реализовать скрипты отладочной и релизной сборок.
2. Реализовать скрипты отладочной сборки с санитайзерами.
3. Реализовать скрипт очистки побочных файлов.
4. Реализовать компаратор для сравнения содержимого двух текстовых файлов.
5. Реализовать скрипт pos_case.sh для проверки позитивного тестового случая по определённым далее правилам.
6. Реализовать скрипт neg_case.sh для проверки негативного тестового случая по определённым далее правилам.
7. Обеспечить автоматизацию функционального тестирования.

Выводы

Задание выполнено

Задание №2. Этапы получения исполняемого файла

Задание

Для выполнения этого задания напишите простую программу на языке Си. В программе должны использоваться директива include, директива define, комментарии. В отчёте приведите текст программы.
Найдите описание ключей v и save-temps. С помощью этих ключей изучите, какие этапы, с помощью каких утилит и с какими параметрами выполняются для получения исполняемого файла компиляторами gcc и clang.

Вывод компилятора на консоль при использовании ключей v и save-temps очень большой. Для удобства его анализа вывод лучше перенаправить в файл. Проанализировав вывод, разделите его на этапы. Вам придется отделит команды от их выдачи. Для проверки правильности выделения команды рекомендуется скопировать ее из вывода и повторить самостоятельно. Если команда выполняется с ошибкой, либо Вы ее неправильно выделили, либо передали не все параметры. Обратите внимание, что способы передачи параметров для разных утилит могут быть разными (параметры командной строки, переменные среды окружения и т. п.).

Выводы

Задание выполнено

Задание №3.1. Отладка

Задание

Требуется:

1. Найдите с помощью gdb ошибки в приложенных программах. Опишите, как Вы это сделали и ответьте на вопросы
2. Составьте две таблица размеров типов.
3. Покажите, как в памяти представлены переменные типов char, int, unsigned, long long. Рассмотрите как положительные значения этих переменных, так и отрицательные. Результаты поясните.
4. Покажите, как в памяти представлен массив целых чисел. Продемонстрируйте особенности выполнения операции сложения указателя с целым числом на примере массива.
5. Изучите работу с отладчиком в Qt Creator или Visual Code (в той IDE, которую используете для выполнения лабораторных работ). Будьте готовы ответить на вопросы о соответствии команд gdb и аналогичных действий в Qt Creator или Visual Code.

Выводы

Задание выполнено

Задание №3.2. Представление в памяти многомерного статического массива

Задание

Изучите, как в памяти представлен многомерный статический массив, для чего:

1. Опишите трёхмерный массив целых чисел, размеры которого равны 2, 3 и 4 соответственно.
2. Покажите дамп памяти, который содержит этот массив полностью. Дамп должен быть получен с помощью команды «x /Nxb адрес».
3. Покажите, из каких компонент состоит этот массив, постепенно фиксируя размерность массива. Соответствующие компоненты необходимо показать в дампе.
4. Опишите указатели для работы с этими компонентами. Чему равен размер элемента соответствующего компонента? Проведите теоретический расчёт. Результата проверьте с помощью gdb.
5. Напишите заголовок функции для обработки компонент соответствующего уровня.

Выводы

Задание выполнено

Задание №3.3. Представление в памяти строк и массивов строк

Задание

1. Изучите, как в памяти представлена строка языка Си. Покажите дамп памяти (дамп должен быть получен с помощью команды «x /Nxb адрес»), который содержит строку полностью. Объясните, какие байты за что отвечают.
2. Изучите разные способы хранения массива слов в языке Си. Для каждого способа:
   a. Покажите дамп памяти, который содержит этот массив полностью. Для получения дампа используйте команду, указанную выше.
   b. Прокомментируйте, что есть что в этих дампах памяти. Почему байты имеют именно такое значение?
   c. Рассчитайте суммарный размер памяти, который занимает каждая структура данных. Каков размер «полезных» и «вспомогательных» данных? Поясните свои расчеты.

Выводы

Задание выполнено

Задание №3.4. Представление в памяти структур и объединений

Задание

1. Изучите как располагаются в памяти локальные переменные
a. Опишите несколько локальных переменных разных типов.
b. На дампе (дамп должен быть получен с помощью команды «x /Nxb адрес») памяти покажите, как они располагаются в памяти.
c. Составьте таблицу, в которой укажите имя переменной, её размер, значение адреса, по которому располагается переменная.
d. Проанализируйте зависимость значения адреса переменной от её размера.
2. Изучите как в памяти представлены структуры.
a. Опишите структуру, содержащую несколько полей разного типа.
b. Покажите дамп памяти, который содержит эту структуру. На дампе покажите расположение каждого поля структуры.
c. Составьте таблицу, в которой укажите имя поля, его размер, значение адреса, по которому располагается поле. Проанализируйте зависимость значения адреса поля от его размера.
d. По какому адресу располагается переменная структурного типа? Какое поле структуры повлияло на значение этого адреса?
e. Упакуйте структуру и выполните предыдущие пункты для упакованной структуры.
f. Переставляя поля Вашей структуры, добейтесь, чтобы занимаемое структурой место стало минимальным. Упаковка для выполнения этого задания не используется.
g. Есть ли у получившейся структуры «завершающее» выравнивание? Чему оно равно? Если нет, то почему его нет?

Выводы

Задание выполнено

Задание №4. Исследование характеристик программного обеспечения

Задание

На основе задачи №4 ЛР№2 (сортировка) по курсу «Программирование на Си» проведите сравнение производительности работы программы для разных способов работы с элементами одномерного массива:

- использование операции индексации a[i];
- формальная замена операции индексации на выражение *(a + i);
- использование указателей для работы с массивом.

Измерение производительности необходимо выполнить двумя способами:

1. Реализовать инфраструктуру измерения времени выполнения функции в самой программе.
2. Реализовать инфраструктуру измерения времени выполнения функции вне программы. В этом случае внутри программы выполняется замер одного выполнения функции (время выполнения функции выводится на экран или в файл), а повторные замеры выполняются путем многократного запуска самой программы.

Независимо от способа организации замеров повторные замеры продолжаются до тех пор, пока величина RSE превышает 1%

Для замера времени выполнения функции использовать

- функцию clock_gettime, для clockid равного CLOCK_MONOTONIC_RAW;
- TSC (единица измерений — ticks, не нужно переводить в секунды, это не так-то просто).

При выполнении замеров вторым способом использовать только функцию clock_gettime.

Для сравнения производительности следует реализовать несколько скриптов:

1. build_apps.sh, вызвав который, можно получить весь набор необходимых исполняемых файлов.
2. update_data.sh, вызвав который, можно добавить некоторые данные в датасет данных исследования.
3. make_preproc.sh|py, вызвав который, можно подготовить данные из набора, провести первичный анализ: посчитать среднее арифметическое,

медианное, найти максимум и минимум, вычислить нижний и верхний квартили, etc.
4. make_postproc.sh|py, вызвав который, можно получить указанные ниже графики.
5. go.sh, вызвав который, можно получить данные исследования (скрипт вызывает по очереди предыдущие четыре).

В отчёте привести следующие графики:
1. Обычный кусочно-линейный график зависимости времени выполнения в любых единицах измерения времени от числа элементов массива.
2. Кусочно-линейный график с ошибкой (среднее, максимум, минимум).
3. График с усами (среднее, максимум, минимум; нижний, средний и верхний квартили) для варианта обработки «через квадратные скобки» при уровне оптимизации.

Графики

Рис. 1.1 - Обычный кусочно-линейный график для замера тиками (см. Приложение А)
Рис. 1.2 - Обычный кусочно-линейный график для замера секундами (см. Приложение Б)
Рис. 2.1 - Кусочно-линейный график с ошибкой для замера тиками (см. Приложение В)
Рис. 2.2 - Кусочно-линейный график с ошибкой для замера секундами (см. Приложение Г)
Рис. 3.1 - График с усами для замера тиками (см. Приложение Д)
Рис. 3.2 - График с усами для замера секундами (см. Приложение Е)

Таблицы

1) Использование операции индексации a[i]
Замер в программе clock_gettime()

Размер	t, мс	Кол-во повторо	RSE, %
15500	347.52	180	0.93
16000	370.11	180	0.93
16500	393.34	180	0.93
17000	418.01	180	0.93

17500	442.98	180	0.93
18000	468.41	180	0.93
18500	496.18	180	0.93
19000	521.70	180	0.93
19500	549.39	180	0.93
20000	577.84	180	0.93

Замер в программе TSC

Размер	t, мс	Кол-во повторо	RSE, %
15500	482400.21	180	0.93
16000	515507.50	180	0.93
16500	547593.34	180	0.93
17000	612877.47	180	0.93
17500	636265.32	180	0.93
18000	652796.21	180	0.93
18500	692038.89	180	0.93
19000	726591.65	180	0.93
19500	766780.61	180	0.93
20000	807019.17	180	0.93

Замер вне программы

Размер	t, мс	Кол-во повторо	RSE, %
15500	347.42	40	0.72
16000	370.26	40	0.72
16500	393.60	40	0.72
17000	418.09	40	0.72

17500	442.97	40	0.72
18000	469.45	40	0.72
18500	497.15	40	0.72
19000	524.12	40	0.72
19500	552.88	40	0.72
20000	584.62	40	0.72

2) Формальная замена операции индексации на выражение *(a + i)
Замер в программе clock_gettime()

Размер	t, мс	Кол-во повторо	RSE, %
15500	349.95	200	0.94
16000	372.37	200	0.94
16500	396.33	200	0.94
17000	420.07	200	0.94
17500	445.25	200	0.94
18000	471.33	200	0.94
18500	496.94	200	0.94
19000	523.41	200	0.94
19500	549.72	200	0.94
20000	577.42	200	0.94

Замер в программе TSC

Размер	t, мс	Кол-во повторо	RSE, %
15500	482501.58	20	0.76
16000	513490.54	20	0.76
16500	546530.43	20	0.76

17000	593532.88	20	0.76
17500	624453.38	20	0.76
18000	654117.88	20	0.76
18500	684910.09	20	0.76
19000	726290.26	20	0.76
19500	763251.73	20	0.76
20000	802582.58	20	0.76

Замер вне программы

Размер	t, мс	Кол-во повторо	RSE, %
15500	348.73	20	0.82
16000	370.75	20	0.82
16500	393.91	20	0.82
17000	417.90	20	0.82
17500	443.10	20	0.82
18000	468.58	20	0.82
18500	494.77	20	0.82
19000	523.75	20	0.82
19500	549.26	20	0.82
20000	577.72	20	0.82

3) Использование указателей для работы с массивом

Замер в программе clock_gettime()

Размер	t, мс	Кол-во повторо	RSE, %
15500	347.91	20	0.54
16000	370.48	20	0.54

16500	393.72	20	0.54
17000	418.50	20	0.54
17500	442.23	20	0.54
18000	468.60	20	0.54
18500	495.62	20	0.54
19000	521.73	20	0.54
19500	549.55	20	0.54
20000	577.34	20	0.54

Замер в программе TSC

Размер	t, мс	Кол-во повторо	RSE, %
15500	488616.2	340	0.97
16000	517175.74	340	0.97
16500	576082.92	340	0.97
17000	580145.94	340	0.97
17500	614106.45	340	0.97
18000	660882.10	340	0.97
18500	698904.11	340	0.97
19000	724000.15	340	0.97
19500	789720.48	340	0.97
20000	813836.67	340	0.97

Замер вне программы

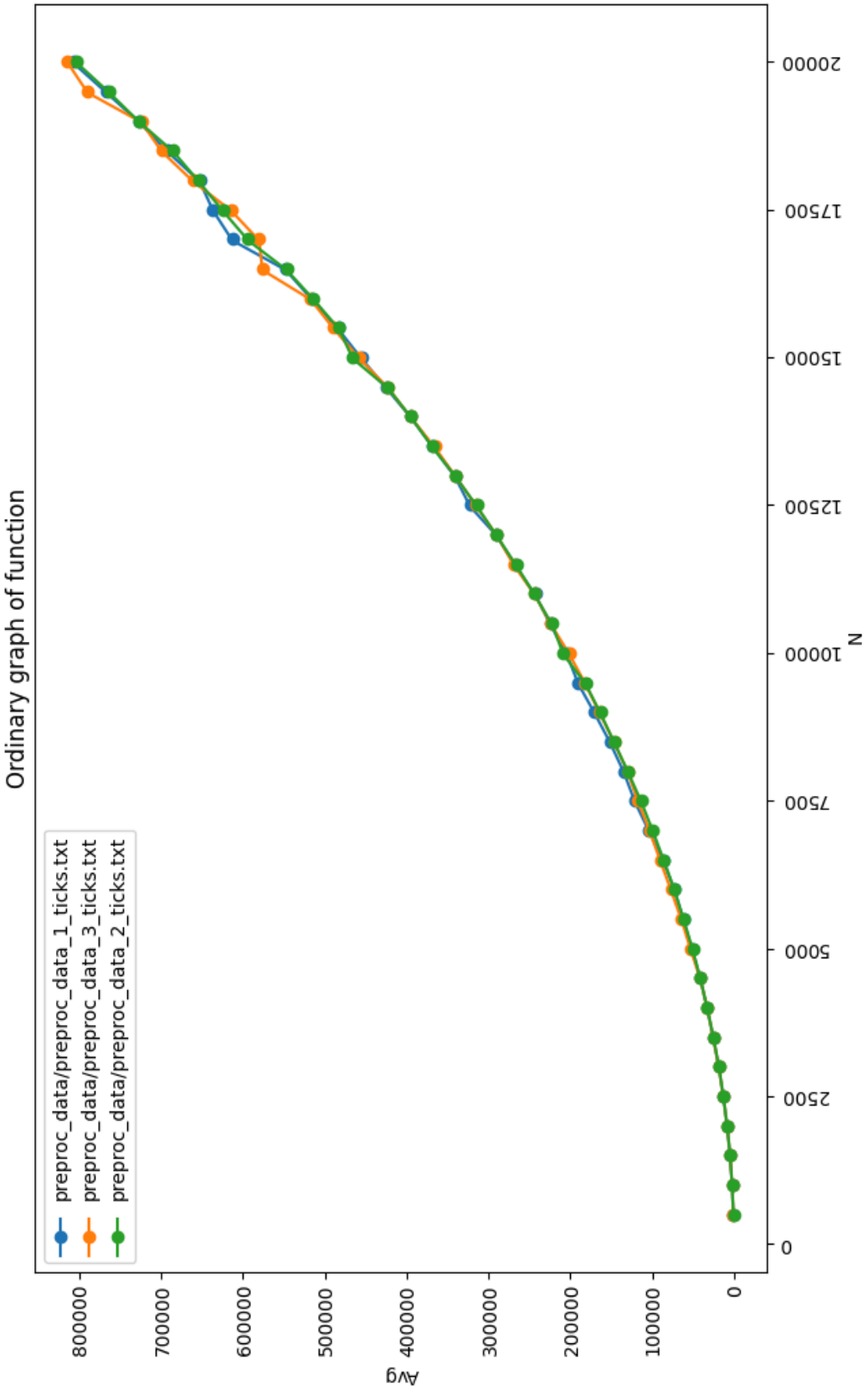
Размер	t, мс	Кол-во повторо	RSE, %
15500	354.32	80	0.92
16000	376.90	80	0.92

16500	406.79	80	0.92
17000	437.84	80	0.92
17500	464.02	80	0.92
18000	481.78	80	0.92
18500	508.81	80	0.92
19000	535.43	80	0.92
19500	555.50	80	0.92
20000	599.15	80	0.92

Выводы

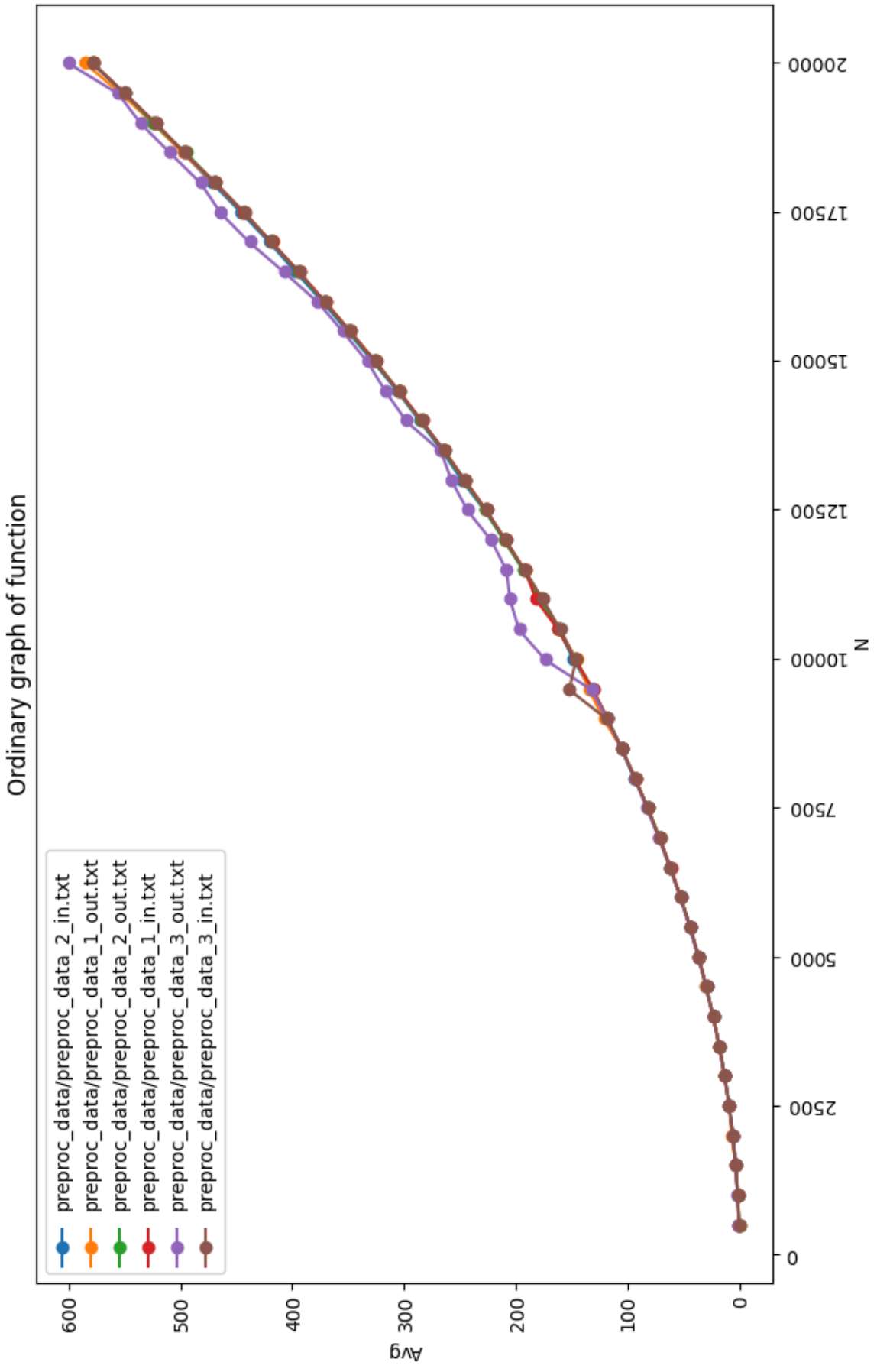
Задание выполнено

Приложение



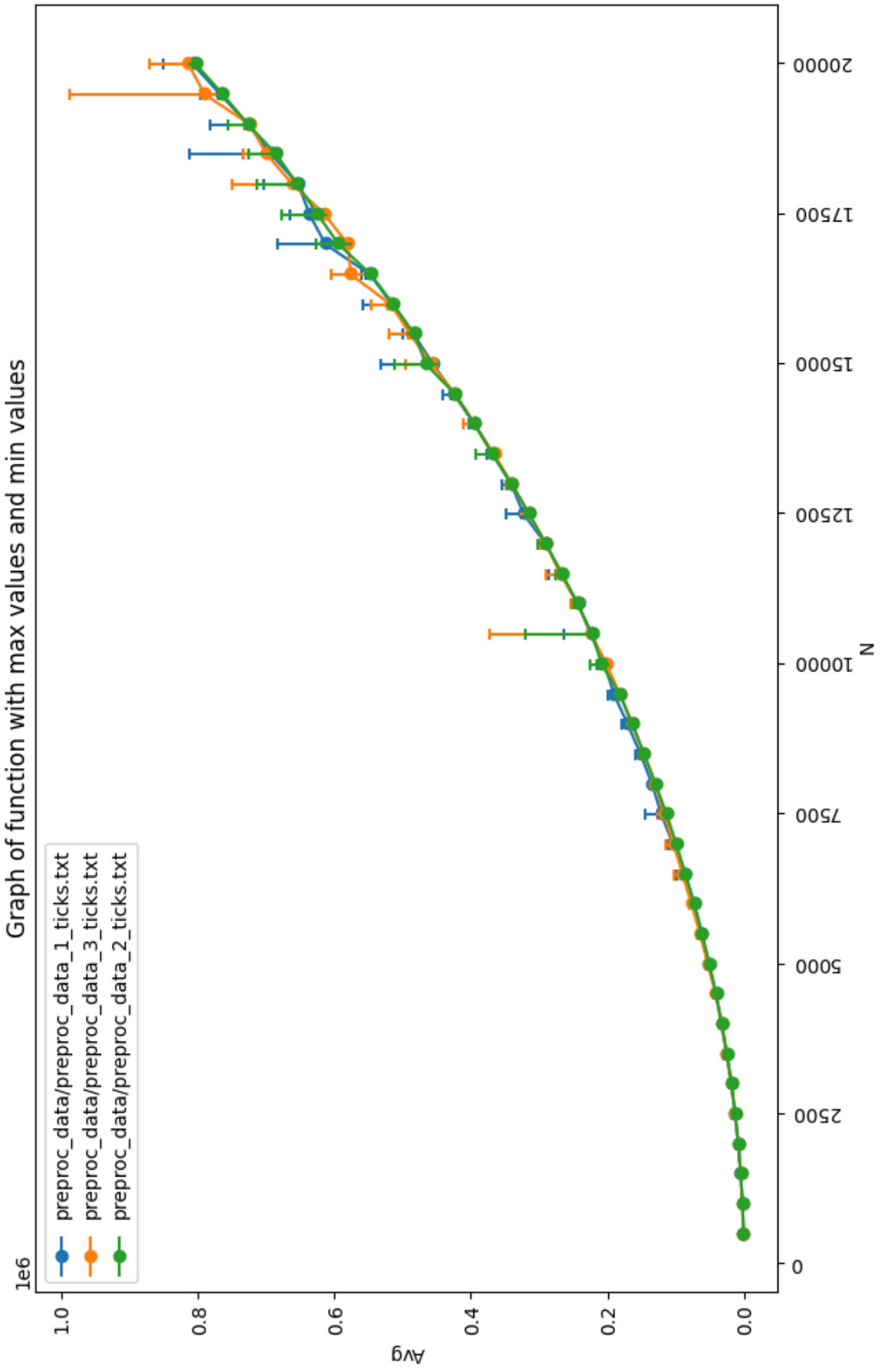
Приложение А

Рис. 1.1 - Обычный кусочно-линейный график для замера тиками



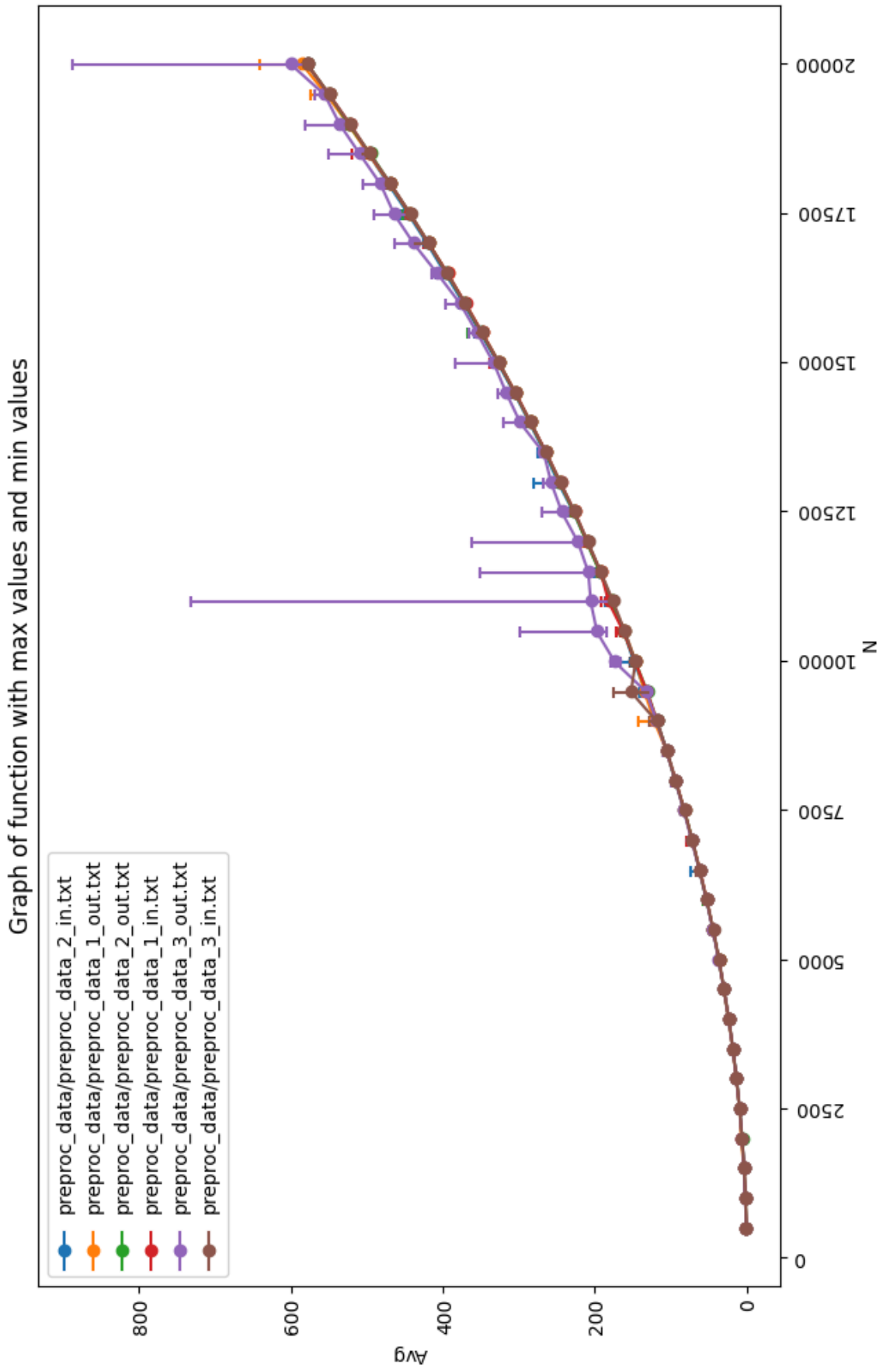
Приложение Б

Рис. 1.2 - Обычный кусочно-линейный график для замера секундами



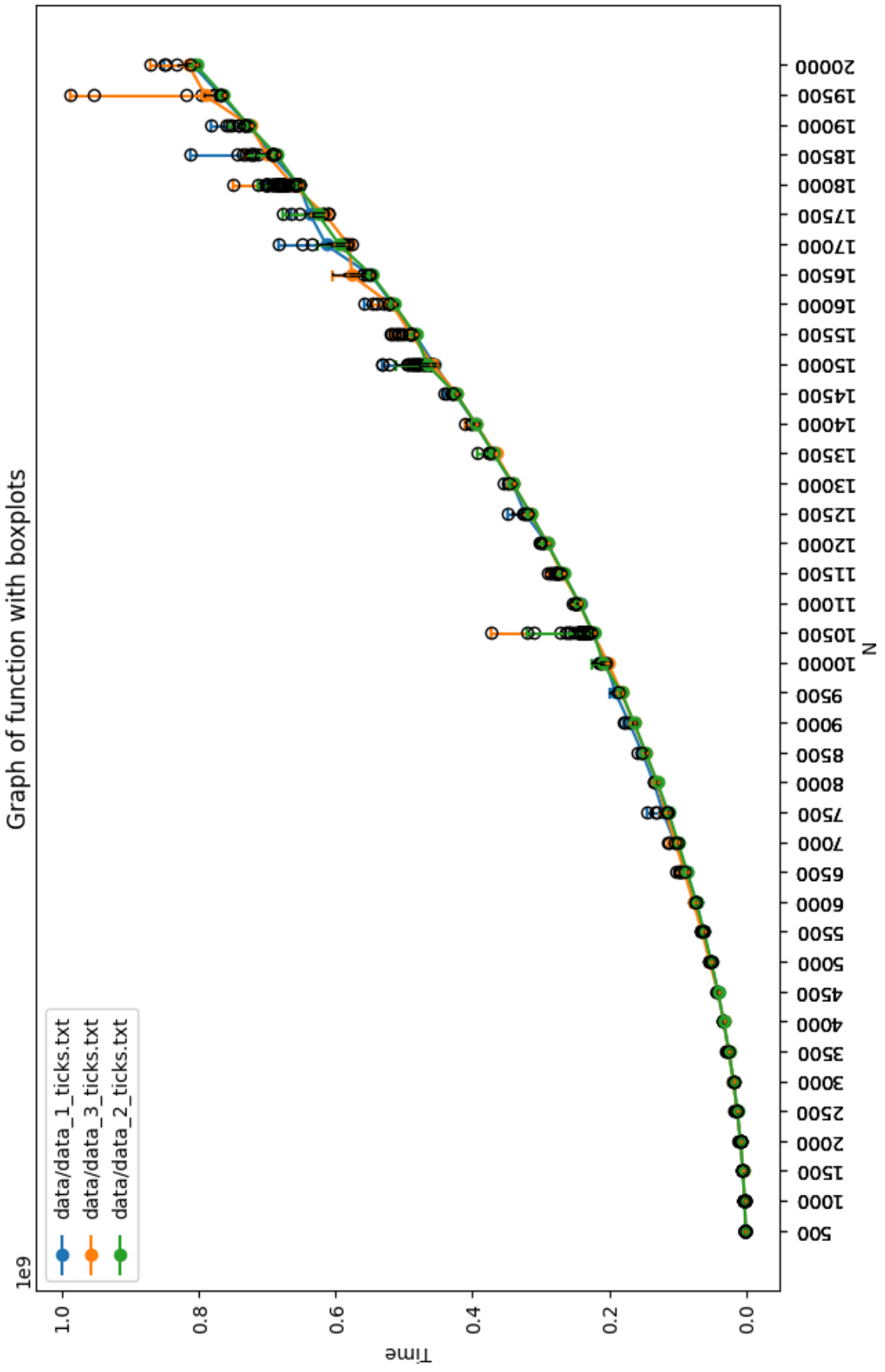
Приложение В

Рис. 2.1 - Кусочно-линейный график с ошибкой для замера тиками



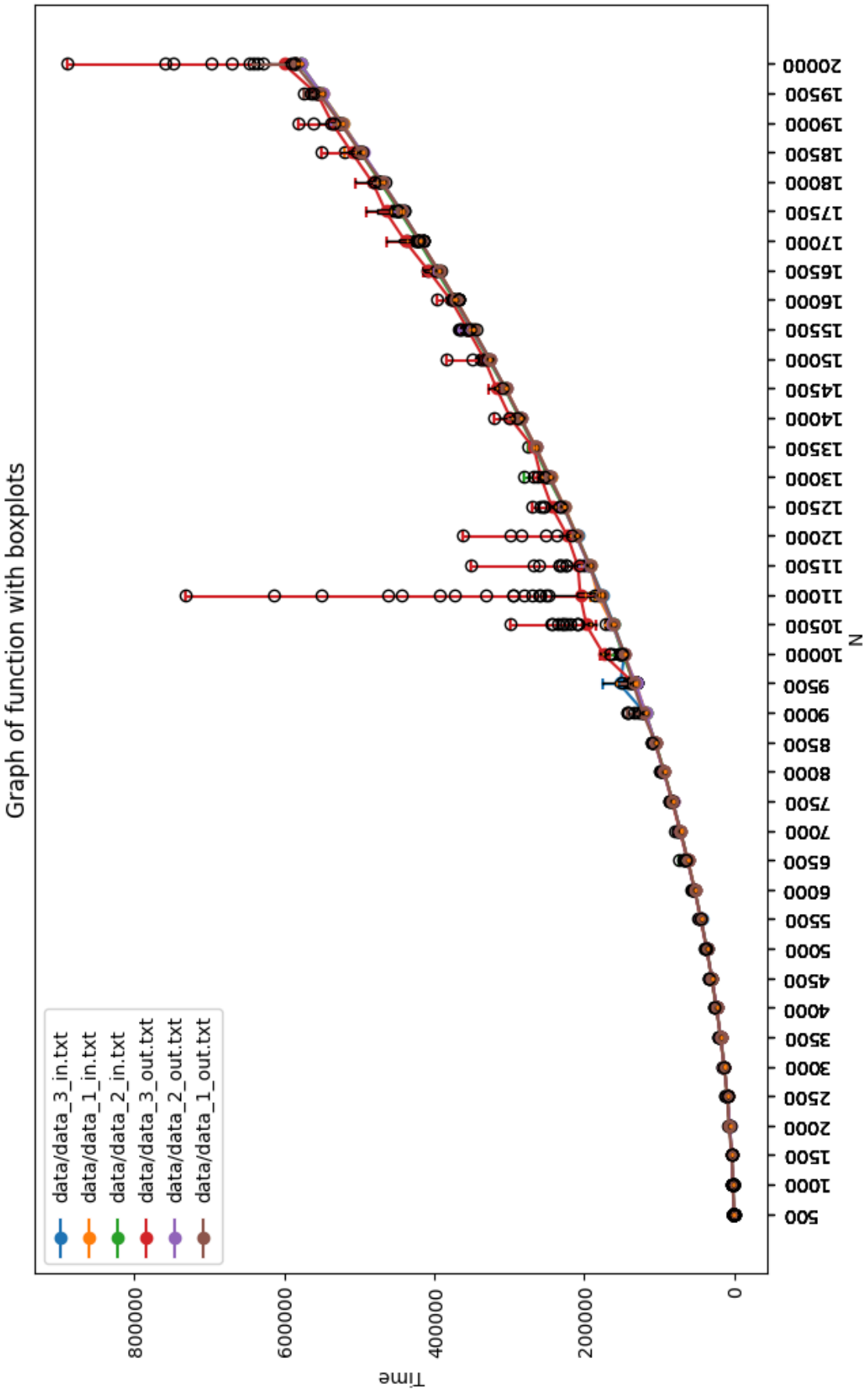
Приложение Г

Рис. 2.2 - Кусочно-линейный график с ошибкой для замера секундами



Приложение Д

Рис. 3.1 - График с усами для замера тиками



Приложение Е

Рис. 3.2 - График с усами для замера секундами