Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Санкт-Петербургский государственный университет» Факультет прикладной математики – процессов управления

**ОТЧЁТ**

**На тему**

**«Эмпирический анализ алгоритма Прима построения минимального остовного дерева взвешенного связного неориентированного графа»**

Дисциплина: «Алгоритмы и анализ сложности»

Направление: «Фундаментальная информатика и информационные технологии»

Выполнил:

Студент 3 курса 21.Б11-ПУ

Авершин Александр Сергеевич

Санкт-Петербург 2023

Оглавление

[Описание алгоритма 2](#_Toc151577742)

[История 2](#_Toc151577743)

[Область применения 2](#_Toc151577744)

[Вспомогательная информация 2](#_Toc151577745)

[Описание алгоритма 3](#_Toc151577746)

[Априорный анализ 3](#_Toc151577747)

[Анализ времени 3](#_Toc151577748)

[Анализ памяти 4](#_Toc151577749)

[Программная реализация 5](#_Toc151577750)

[Входные данные 5](#_Toc151577751)

[Генерация данных и проведение измерений 5](#_Toc151577752)

[Код программы на Python 5](#_Toc151577753)

[Ссылка на репозиторий 5](#_Toc151577754)

[Эмпирический анализ 5](#_Toc151577755)

[Оценка трудоёмкости 5](#_Toc151577756)

[Графическое представление 6](#_Toc151577757)

[Анализ результатов 11](#_Toc151577758)

[Характеристики используемой среды 11](#_Toc151577759)

[Список литературы 12](#_Toc151577760)

# Описание алгоритма

## История

Открыт Ярником [1930], через много лет переоткрыт Примом [1957] и Дейкстрой [1959]. В литературе обычно называется просто «алгоритмом Прима» [3].

История алгоритма связана с задачей поиска минимального остовного дерева в взвешенном связном графе. В начале 20 века подобные задачи стали активно исследоваться в контексте транспортных сетей, электроэнергетики и других областей, где требовалось эффективное и оптимальное соединение различных узлов. Войцех Ярник представил свой алгоритм в работе "O jistém problému minimálním" ("О некоторой задаче минимальной") в 1930 году, которая, к сожалению, осталась практически незамеченной в западной литературе. Этот алгоритм был первым полным описанием эффективного метода поиска минимального остовного дерева в графе.

Алгоритм Прима стал более широко известным после публикации Робертом Примом в 1957 году. Прим предложил аналогичный метод, независимо открытый Ярником. Алгоритм Прима также был включен в статью Эдсгера Дейкстры, опубликованную в 1959 году, где он был представлен в контексте решения задачи кратчайшего пути в графе.

## Область применения

Алгоритм Прима применяется для решения задач в области поиска минимального остовного дерева в графах.

* **Энергетические сети:** в энергетических системах алгоритм Прима помогает минимизировать расходы на провода
* **Технологии маршрутизации:** применяется в компьютерных и телефонных сетях для оптимизации сетевой инфраструктуры.
* **Графовые базы данных:** применяется для анализа и оптимизации структуры данных в графовых базах данных.
* **Биоинформатика:** алгоритм Прима может быть использован для построения филогенетических деревьев и анализа генетических данных
* **Логистика и транспорт:** применяется для решения задач оптимизации маршрутов в логистике и транспортной отрасли.

## Вспомогательная информация

* **Остовное дерево** (spanning tree) связного графа представляет собой связный ациклический подграф (т.е. дерево), которое содержит все вершины графа. Минимальное остовное дерево (minimum spanning tree) взвешенного связного графа представляет собой остовное дерево с наименьшим весом, где вес дерева определяется как сумма весов всех его ребер. Задача о минимальном остов- ном дереве представляет собой задачу поиска минимального остовного дерева для данного взвешенного связного графа. [1]
* **Взвешенный граф**: Граф, в котором каждому ребру присвоено численное значение, называемое весом. В контексте алгоритма Прима, вес ребра представляет стоимость или длину этого ребра.
* **Связный граф:** Граф, в котором существует путь между любыми двумя вершинами. В контексте минимального остовного дерева, граф должен быть связным, чтобы охватывать все его вершины.
* **Неориентированный граф:** Граф, в котором рёбра не имеют направления. Отсутствие направления на рёбрах означает, что ребро {A, B} эквивалентно ребру {B, A}

## Описание алгоритма

Вот краткое описание алгоритма Прима:

1. **Инициализация:** создать пустую коллекцию рёбер Edges и пустую коллекцию вершин Vertex, в которые будем фиксировать информацию об остовном дереве
2. **Выбор начальной вершины:** произвольно выбрать начальную вершину, заносим её в Vertex.
3. **Поиск минимального ребра:** найти такое минимальное ребро (A, B): A ∈ Vertex и B ∉ Vertex. Добавить (A, B) в Edges
4. **Добавление вершины:** добавить вершину “B”, которая принадлежит найденному ребру.
5. **Повторение:** повторять шаги 3-4 до тех пор, пока все вершины не будут включены в остовное дерево.

Алгоритм закончит работу, когда все вершины графа включены в остовное дерево. Результатом работы алгоритма является минимальное остовное дерево, покрывающее все вершины графа с минимальной суммой весов рёбер.

# Априорный анализ

## Анализ времени

**Инициализация пустого списка приоритетов (priority queue):** O(1) - константное время, не зависящее от размера графа.

**Инициализация меток вершин:** O(V) - линейное время относительно количества вершин в графе. Каждой вершине присваивается временная метка.

**Итерации по вершинам:** O(V log V) - алгоритм Прима использует минимальную кучу (priority queue), что приводит к времени O(V log V) для обработки каждой вершины.

**Обновление меток смежных вершин:** O(E log V) - для каждого ребра (E) требуется выполнение операций в приоритетной очереди, что занимает логарифмическое время относительно количества вершин.

Итоговая сложность алгоритма Прима:

O(1)+O(V)+O(VlogV)+O(ElogV)

Объединяя слагаемые, получаем общую сложность:

O(VlogV+ElogV)=O(Elg(V))[2]

**Лучший случай:** Лучший случай возникает, когда граф разреженный и имеет минимальное количество рёбер. В таком случае, трудоемкость ближе к *O*(*V*log*V*), так как количество рёбер минимально, и внутренние операции в алгоритме выполняются в пределах вершин, не зависящих от количества рёбер.

**Средний случай:** В среднем случае, для большинства практических графов, трудоемкость остается *O*(*V*log*V*+*E*log*V*). Средний случай характеризуется умеренным количеством рёбер и вершин.

**Худший случай:** Худший случай возникает, когда граф плотный и содержит максимальное количество рёбер. В таком случае трудоемкость остается *O*(*V*log*V*+*E*log*V*), и больше уделяется внимание рёбрам, что может привести к более высокой трудоемкости по сравнению с лучшим или средним случаем.

## Анализ памяти

**Хранение графа:** для ненаправленного графа в NetworkX, по умолчанию используется структура данных, представляющая граф, как список смежности. Так что будет расходоваться O(E+V) памяти, где E – количество рёбер, а V – количество вершин.

**Хранение вершин:** память требуется для хранения информации о каждой пройденной вершине O(V).

**Хранение рёбер:** также требуется памят для хранения пройденных рёбер O(E).

**Итоговая оценка:** O(V+E)

**Худший случай:** худший случай достигается в случае, если на вход подаётся очень плотный граф

# Программная реализация

## Входные данные

Входные данные представляют из себя структуру данных Graph из библиотеки NetworkX. Эта структура представляет из себя список смежности, то есть для каждой вершины хранятся её соседние вершины и вес.

## Генерация данных и проведение измерений

* **Генерация графа:** перед каждым запуском алгоритма происходит генерация нового графа. Сначала соединяются все вершины, чтобы гарантировать связность, а после псевдо случайно граф заполняется ребрами до указанного значения.
* **Измерение времени выполнения:** в начале алгоритма фиксируется начальное время выполнения, а в конце измеряется время выполнения после завершения алгоритма. Разница между начальным и конечным временем предоставляет информацию о времени выполнения алгоритма.
* **Повторение тестов:** Этот процесс повторяется 100 раз для каждого размера графа, чтобы исключить случайные факторы, например, прерывание. По итогу вычисляется средний результат.

## Код программы на Python

В коде алгоритм Прима выполняется методом из библиотеки networkx: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.tree.mst.minimum\_spanning\_tree.html

## Ссылка на репозиторий

# <https://github.com/asavershin/analysis>

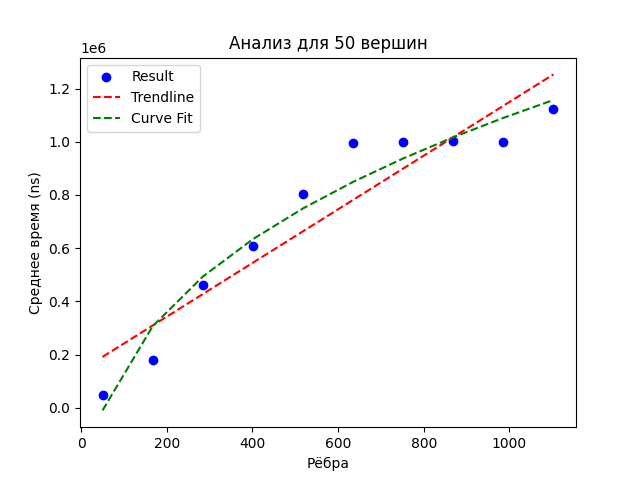
# Эмпирический анализ

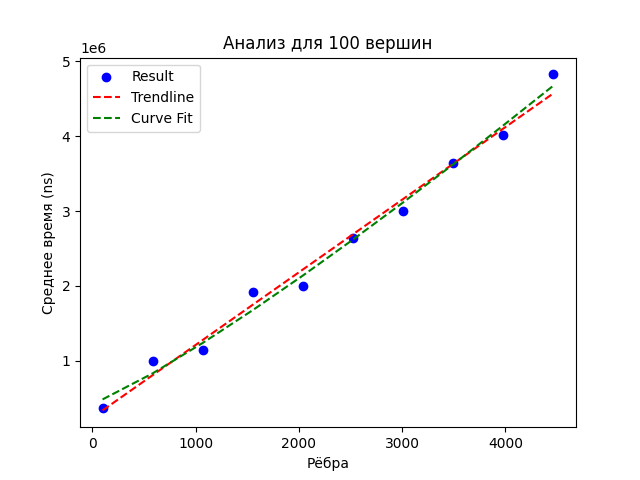
## Оценка трудоёмкости

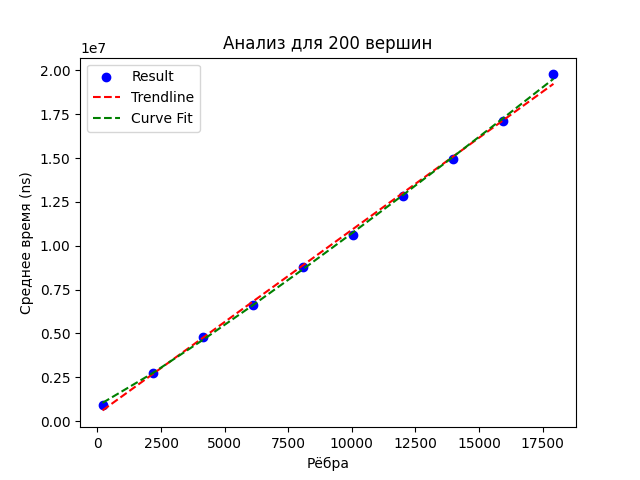
Оценка трудоёмкости производилась путём замера времени до начала работы алгоритма и после. Для этого использовался метод time\_ns из библиотеки time. Проводилось по 100 замеров для каждого типа вершин (50, 100, 200, 400, 800), потом отбирались значения, которые лежат между 20% и 80% квантилем. Перед каждым запуском замера производится принудительный вызов сборщика мусора, чтобы точно очистить память от предыдущего теста, это позволяет минимизировать выбросы при тестировании. Аналогично были проведены оценки трудоёмкости при фиксированном количестве рёбер.

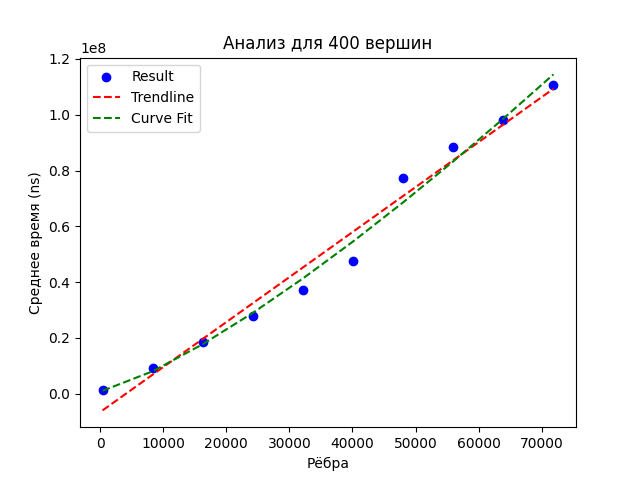
## Графическое представление

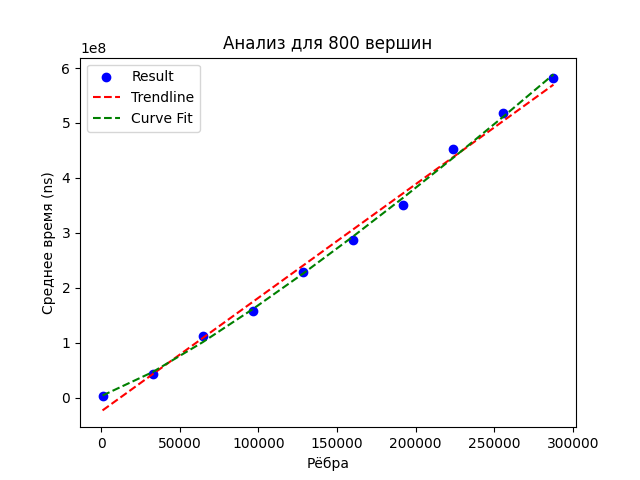
Зеленая пунктирная линия соответствует кривой наименьших квадратов, красная пунктирная линия соответствует линии тренда.

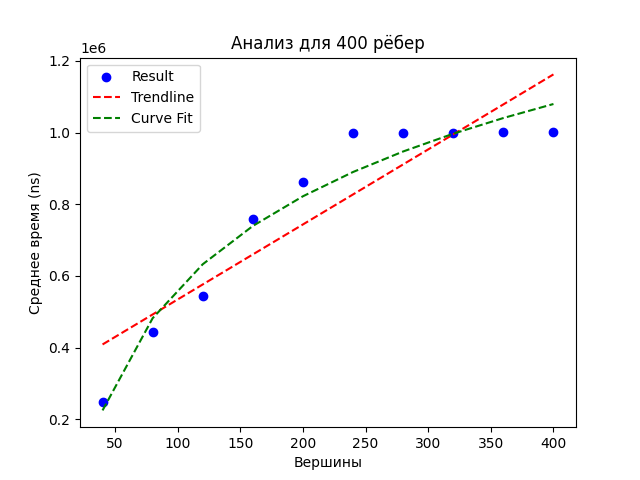


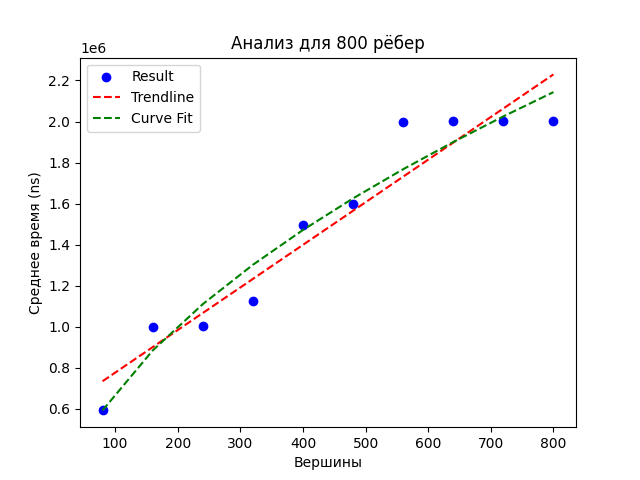


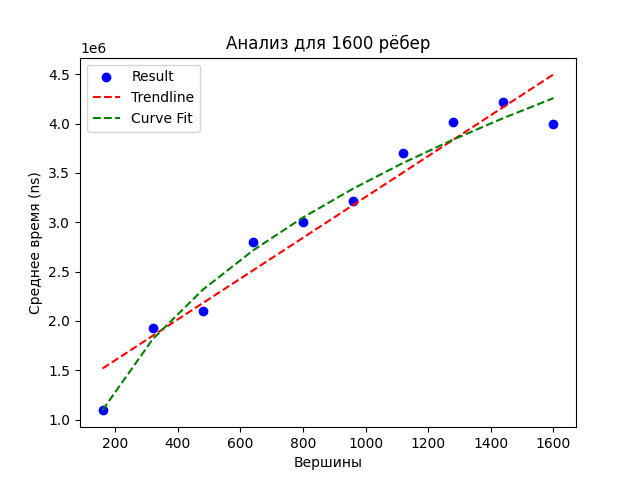


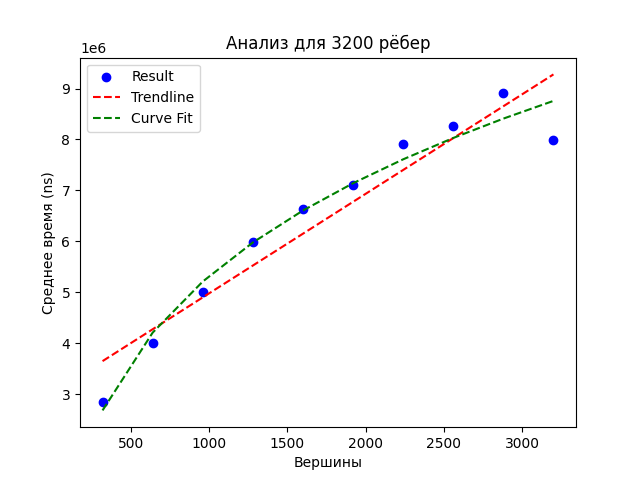


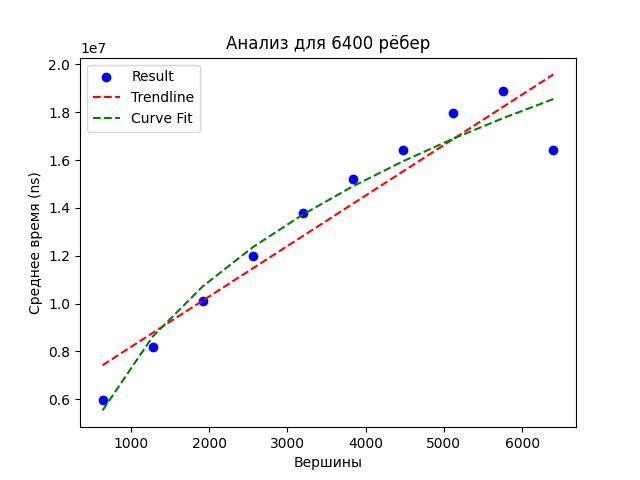










****

## Анализ результатов

Пусть T(V, E) – трудоёмкость алгоритма Прима.

Если мы зафиксируем вершины, и будем увеличивать рёбра то асимптотика трудоёмкости будет O(E), что соответствует полученным нами результатам, а .

Если мы зафиксируем рёбра, и будем увеличивать вершины то асимптотика трудоёмкости будет O(lgV), что соответствует полученным нами результатам, а .

# Характеристики используемой среды

Процессор: 12th Gen Intel(R) Core(TM) i7-12650H 2.30 GHz

Оперативная память: 16,0 ГБ (доступно: 15,7 ГБ)

Операционная система: Windows 11 Pro

Язык разработки: Python 3.11.2

# Список литературы

1. Алгоритм Прима [Раздел книги] // Алгоритмы. Введение в разработку и анализ / авт. книги Левитин Ананий. - 2006.
2. Алгоритм Прима [Раздел книги] // Алгоритмы. Построение и анализ второе издание / авт. книги Кормен Томас [и др.].
3. Охотин А. С. Лекция 2. Наименьшее остовное дерево во взвешенном во взвешенном графе [В Интернете]. - 16 Июнь 2023 г.. - https://users.math-cs.spbu.ru/~okhotin/teaching/algorithms3\_2023/okhotin\_algorithms3\_2023\_l2.pdf.