**Министерство науки и высшего образования Российской Федерации**

**Государственное образовательное учреждение**

**высшего профессионального образования**

**ГОУ ВПО “Пермский государственный научно-исследовательский университет”**

*Кафедра информационной безопасности и систем связи*

Отчёт

О выполнении

Практической работы на тему

**“Сравнение различных алгоритмов решения NP-полных задач”**

По дисциплине “ Алгоритмы и анализ сложности”

**Выполнили:**

Студенты 3-го курса

Бузмаков Иван,

Кондратенков Валерий,

группы ММ/О КМБ-19-СП

Пермь 2021

# Задача о правильной раскраске графа в минимальное количество цветов

Раскраска вершин графа в разные цвета называется правильной, если вершины, соединенные ребром (смежные вершины), раскрашены в разные цвета, а хроматическим числом графа называется минимальное число цветов, которыми можно правильно раскрасить вершины графа. То есть необходимо получить хроматическое число графа, с помощью трех разных алгоритмов («жадный», оптимизированный «жадный» и полный перебор всех возможных n! перестановок вершин).

Дано: число вершин в графе (n) и сам граф в виде матрицы смежности.

Получить: правильная раскраска графа с использованием минимального количества цветов (вывести количество использованных цветов, а также список пар «номер вершины – номер цвета»).

# Описание входных данных

Число вершин в графе(n) – целочисленное значение;

Граф в виде матрицы смежности – двумерный логический массив (если имеется ребро, соединяющее две вершины, то в ячейку массива записывается значение «истина»).

# Краткое описание алгоритмов

Полный перебор: перебираются все возможные перестановки вершин, и к каждой перестановке применяется жадный алгоритм, после чего количество цветов, необходимое для раскраски графа с очередным обходом (последовательности вершин), сравнивается с минимальным числом цветов, полученных при применении жадного алгоритма к предыдущим обходам (последовательностям вершин);

Тривиальный «жадный» алгоритм: может выбираться случайная последовательность вершин для окраски. В нашем случае, мы берем вершины по порядку, т.е. от 1-ой до n-ной: 1) выбирается первая вершина последовательности, раскрашивается в один из цветов; 2) для следующей вершины в последовательности просматриваются смежные с ней окрашенные вершины, и выбирается доступный цвет для её окраски; 3) выполняем те же действия (пункт 2) со всеми вершинами в последовательности пока не раскрасим все вершины графа;

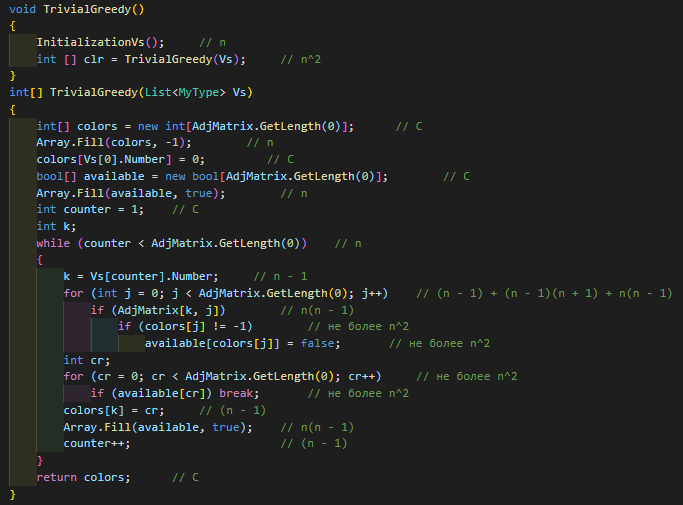
«Жадный» с оптимизацией: для каждой вершины вычисляем её степень и формируем последовательность по убыванию степени вершины, т.е. на первом месте окажется вершина графа с максимальной степенью. Далее к этой последовательности применяем жадный алгоритм.

# Исходный код программы

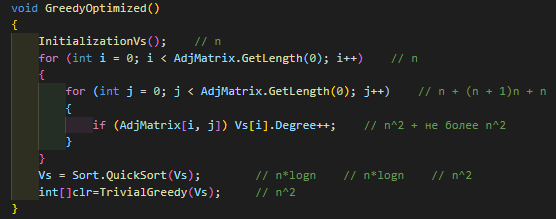
По [ссылке](https://github.com/INDRA909/ColoringGraph-) на репозиторий GitHub можно найти файлы исходного кода, Excel-таблицу с нашими расчетами и архивированную версию репозитория «Проект.rar».

# Теоретическая оценка сложности

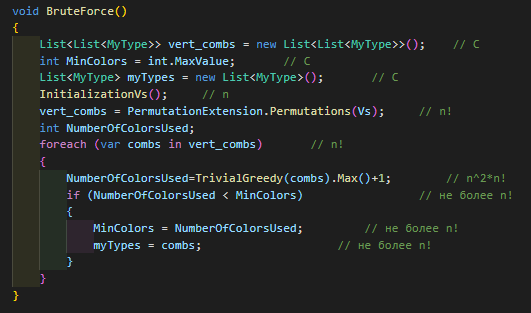
Оценка для тривиального «жадного» алгоритма: порядок роста получился . Построив график и линию тренда по полученным экспериментальным путем времени выполнения от объема входных данных, получаем значение константы . Также данное значение константы можно получить из полученного времени экспериментальным и теоретическим способами, как отношение среднего времени при экспериментальной оценке к значению, полученному из порядка сложности функции, при одинаковых входных данных.



Оценка для тривиального оптимизированного алгоритма: порядок роста получился . Также как и для тривиального алгоритма строим график и линию тренда и получаем значение константы . Значение константы ещё можно получить из полученного времени экспериментальным и теоретическим способами, как отношение среднего времени при экспериментальной оценке к значению, полученному из порядка сложности функции, при одинаковых входных данных.



Оценка для алгоритма полного перебора: порядок роста получился . Значение константы вычисляем, используя время полученное экспериментальным и теоретическим способами, как отношение среднего времени при экспериментальной оценке к значению, полученному из порядка сложности функции, при одинаковых входных данных, получим константу .



# Используемые языки программирования и среда разработки

Язык программирования – C#

Среда разработки – Visual Studio Community 2022 (.NET 6) Версия 17.0.2

# Характеристики оборудования

Модель процессор: Intel Xeon E5 2678 v3 (12 ядер/24 потока)

Тактовая частота процессора: 2.5 – 2.9 ГГц

Объем оперативной памяти: 48 ГБ

# Максимально допустимое количество вершин

Для алгоритма полного перебора максимально допустимым являются 11 вершин. Если попытаться увеличить количество вершин графа, то объема оперативной памяти тестового оборудования становится недостаточно для решения задачи за приемлемое время.

Для тривиального и оптимизированного тривиального алгоритмов максимально допустимого ограничения по количеству вершин, при решении задачи за приемлемое время, не было получено на тестовом оборудовании по причине нехватки максимально возможного объема данных для хранения двумерного массива.

# Графики зависимости времени работы от объема входных данных

1. Тривиальный алгоритм

График 1

График 2

1. Тривиальный оптимизированный алгоритм

График 3

График 4

1. Полный перебор

График 5

График 6

# Вывод о соответствии теоретических и экспериментальных данных

На графиках *(1,3)* теоретического и экспериментального времени выполнения алгоритмов можно заметить, что разница по времени выполнения для 1 и 2 алгоритмов довольно небольшая, что свидетельствует о правильно определенном порядке роста функции сложности тривиального и тривиального оптимизированного алгоритмов. Также на графике *(5)* для 3 алгоритма можно увидеть, что до того момента, пока функция факториала от *n* вершин не начинает резко увеличиваться (при значении 11), разница находится в пределах погрешности.

Определение худшего случая для «жадного» алгоритма является очень сложной задачей. Поэтому, предположим, что худшим случаем следует считать полный граф, а лучшим – граф с числом связности *n.*

Говоря о разнице между временем выполнения в худшем и лучшем случаях, стоит сказать, что на всех графиках *(2,4,6)* прослеживаются зависимости – в худшем случае время выполнения выше, чем в лучшем случае, а также, чем больше количество входных данных, тем больше разница во времени между худшим и лучшим случаями.

# Таблицы

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм | Кол-во вершин | Кол-во тестов в серии | Среднее время работы, мс |
| Тривиальный | 13000 | 10 | 2815,5 |
| 10000 | 10 | 1641,5 |
| 7000 | 10 | 811,1 |
| 4000 | 10 | 261,3 |
| 1000 | 10 | 16,7 |
| Тривиальный оптимизированный | 13000 | 10 | 5985,9 |
| 10000 | 10 | 3526,4 |
| 7000 | 10 | 1749,3 |
| 4000 | 10 | 575,9 |
| 1000 | 10 | 36,8 |
| Полный перебор | 11 | 10 | 85496,8 |
| 10 | 10 | 6848,2 |
| 9 | 10 | 611,2 |
| 8 | 10 | 55,6 |
| 7 | 10 | 5,4 |

*Таблица 1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Алгоритм | Кол-во вершин | Кол-во тестов в серии | Процент тестов, совпадающих с точным | Среднее относительное отклонение |
| Тривиальный | 11 | 100 | 56% | 0,1225 |
| 10 | 100 | 55% | 0,1238 |
| 9 | 100 | 69% | 0,09 |
| 8 | 100 | 75% | 0,0775 |
| 7 | 100 | 78% | 0,0783 |
| Тривиальный оптимизированный | 11 | 100 | 88% | 0,0308 |
| 10 | 100 | 82% | 0,0482 |
| 9 | 100 | 94% | 0,0183 |
| 8 | 100 | 96% | 0,0125 |
| 7 | 100 | 98% | 0,0067 |
| Полный перебор | 11 | 100 | 100% | 0 |
| 10 | 100 | 100% | 0 |
| 9 | 100 | 100% | 0 |
| 8 | 100 | 100% | 0 |
| 7 | 100 | 100% | 0 |

*Таблица 2*

# Заключение

Таким образом, основываясь на проделанной работе, можно сделать вывод, самым быстрым является тривиальный алгоритм, однако, для решения данной задачи, он дает довольно большой процент отклонения от точного решения – алгоритма полного перебора, по сравнению с оптимизированным тривиальным алгоритмом *(см. Таблица 2)*, который в свою очередь намного чаще совпадает с точным решением, но немного уступает по времени вычисления хроматического числа заданного графа.

Самым медленным оказался алгоритм полного перебора, что совсем неудивительно, так как для его реализации используются все перестановки вершин для обхода графа, что составляет *n!*, и для каждой из них применяется тривиальный алгоритм.

В результате можно сказать, что оптимальным алгоритмом для определения правильной раскраски графа является оптимизированный тривиальный алгоритм, который выполняется за приемлемое время и дает небольшой процент отклонения от точного решения.