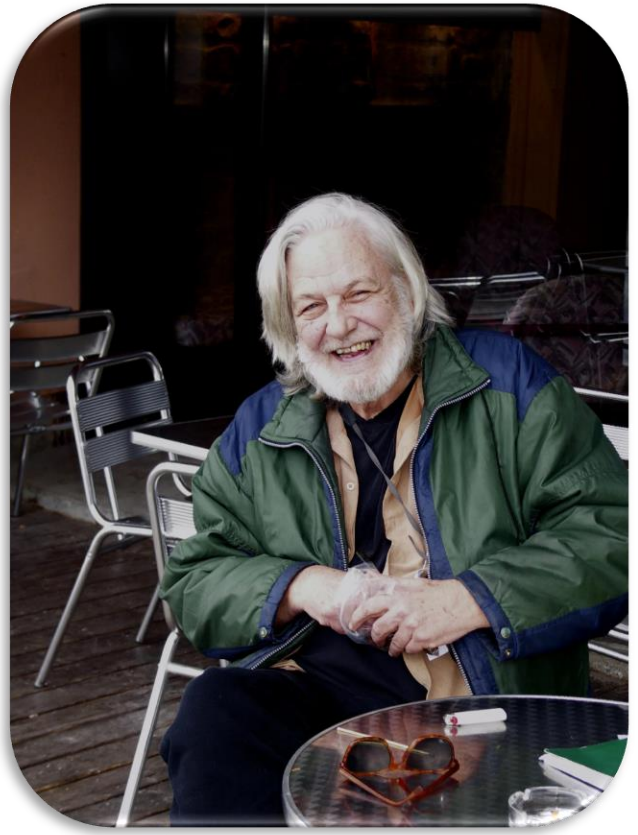
Алгоритм сжатия цветков

Студент Яцевич Ксения
Б9121-09.03.03 пикд

Историческая справка

Алгоритм разработал Джек Эдмондс в 1961 году и опубликовал в 1965 году.

Основная причина важности алгоритма – первое доказательство возможности нахождения наибольшего паросочетания за полиномиальное время.



Описание алгоритма

Идея алгоритма

Алгоритм сжатия цветков (*англ. Blossom algorithm*) — это алгоритм для построения наибольших паросочетаний на графах.

Описание алгоритма

Оценка сложности

Всего n итерации, на каждой выполняется обход в ширину O(m).
Операции сжатия цветков может быть O(n_1).
Сжатие соцветий работает за O(n_2)c
Стоит отметить: n_1 = n_2

Общая асимптотика алгоритма: $O(n(m+n^2)) = O(n^3)$.

Описание алгоритма

Дополняющий(увеличивающий) путь

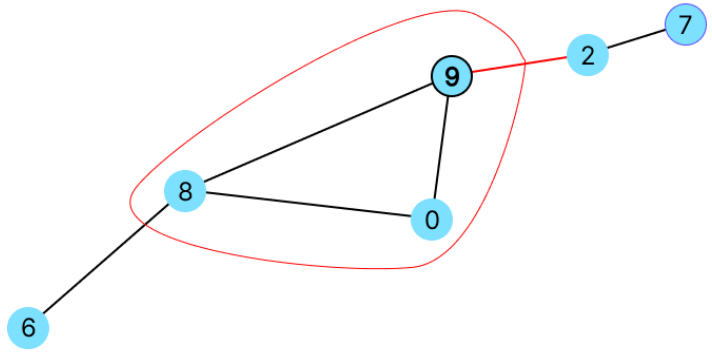
Мы сможем найти максимальное паросочитание путем инверсии дополняющего пути.

Дополняющий путь - чередующаяся цепь, которая начинается и кончается свободными вершинами.

Описание алгоритма

Сжатие цветка

Сжатие всего нечётного цикла в одну псевдо-вершину (соответственно, все рёбра, инцидентные вершинам этого цикла, становятся инцидентными псевдо-вершине).

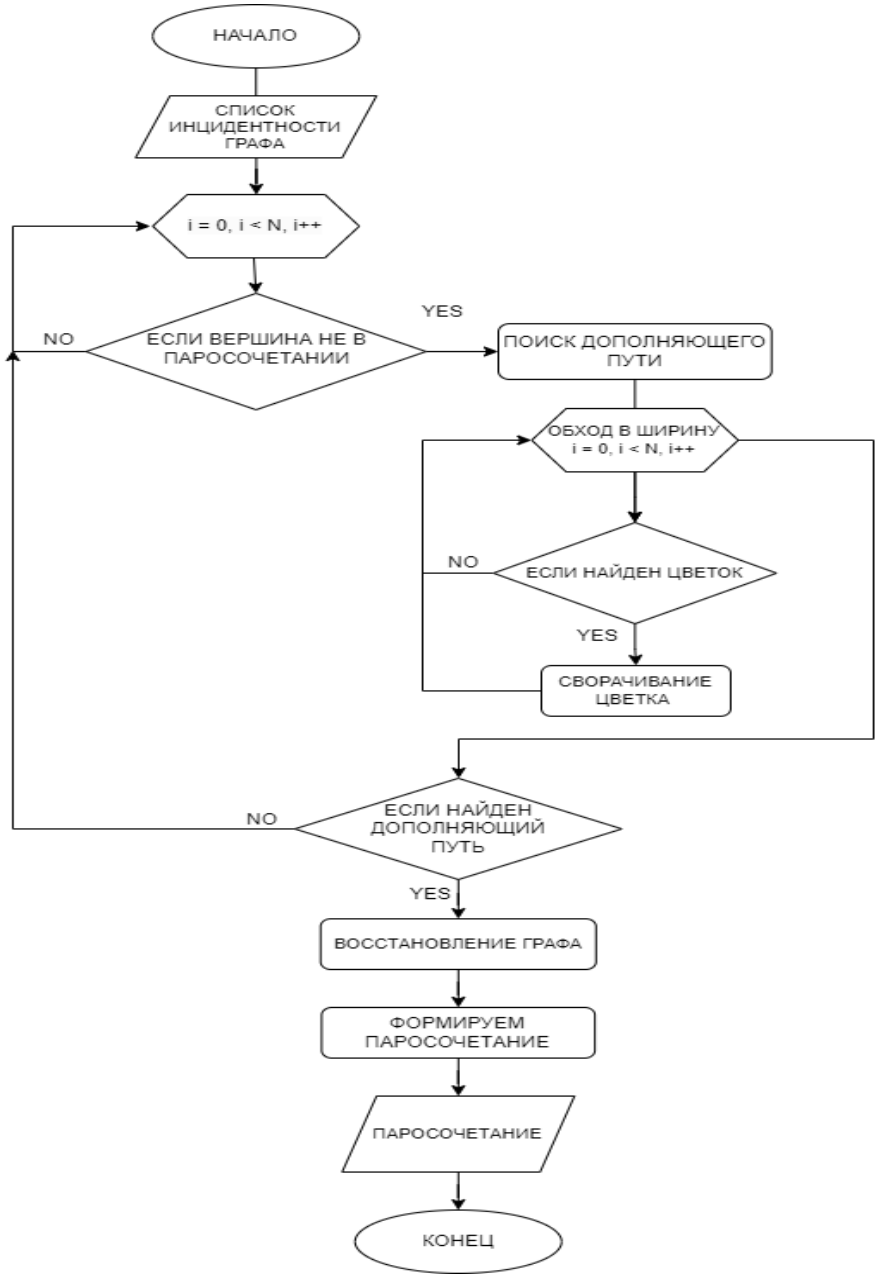


Описание алгоритма

Теорема Эдмондса

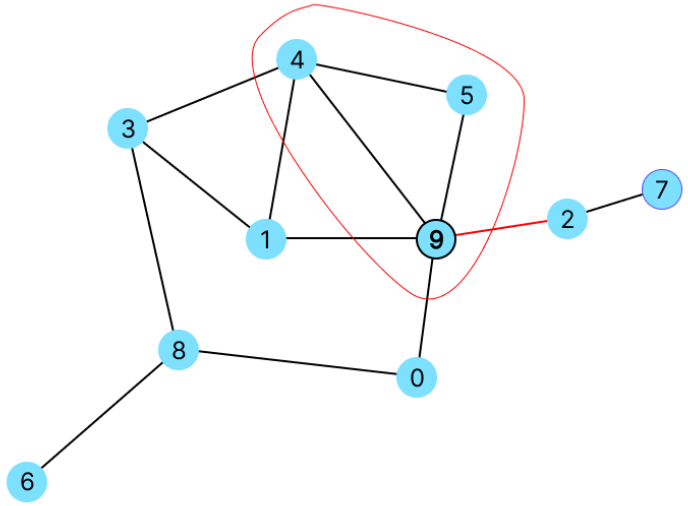
Пусть граф $\overline{G}$ был получен из графа G сжатием одного цветка. Тогда в графе $\overline{G}$ существует увеличивающая цепь тогда и только тогда, когда существует увеличивающая цепь в G.

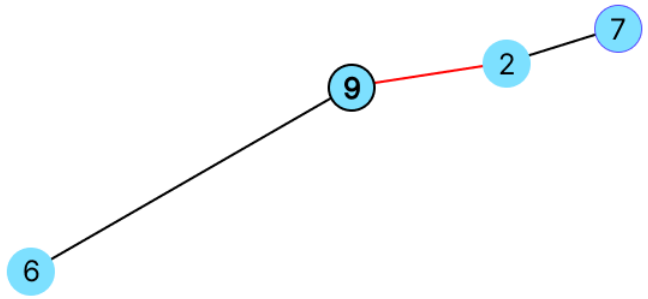
Общая схема алгоритма

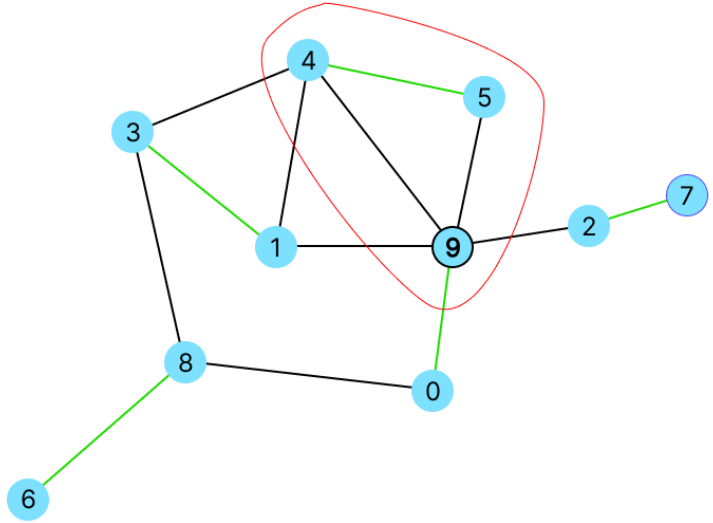


Пример работы алгоритма

Произвольный граф → Поиск цветков → Сжатие цветков → Нахождение дополняющего пути → Восстановление графа → Максимальное паросочетание







Описание реализации

Написана библиотека *blossom.h* , в которой реализованы следующие функции:

print_match() **- принимает паросочетание и выводит его в консоль**

get_match() **- принимает список инцидентности и вектор, куда будет записано паросочетание. Помещает паросочетание в переменную *a*. Включает в себя функции:**

lca() **- находит общего ближайшего предка для вершин цветка**

mark_path() **- помечает чередующийся путь**

find_path() **- ищет дополняющий путь из каждой вершины. Результат работы - последняя вершина дополняющего пути**

Тестирование

Критерии проверки решений:

1) Уникальность вершин
2) Количество ребер
3) Подлинность (существование) ребер

График зависимости памяти и времени от количества вершин

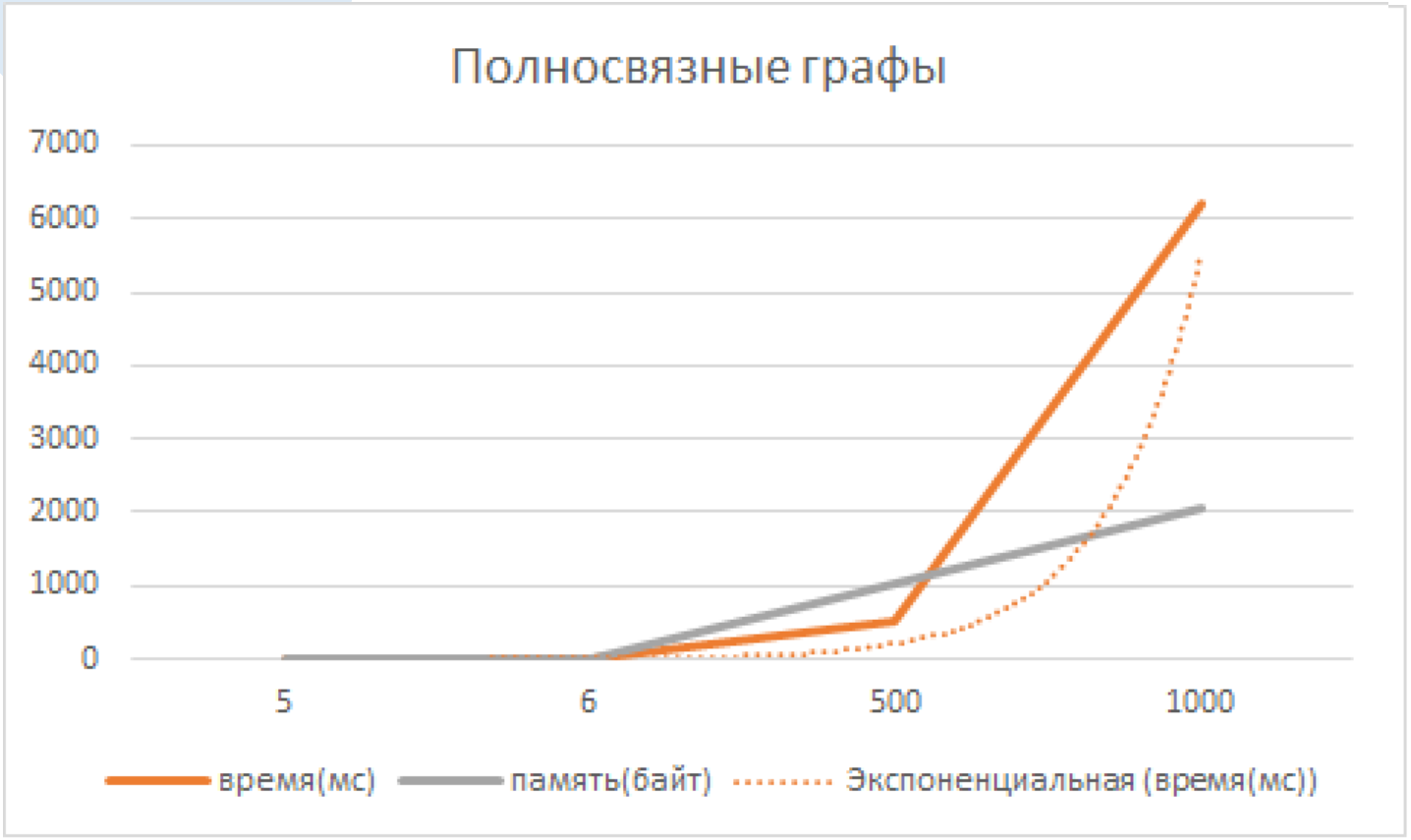


График зависимости памяти и времени от количества вершин

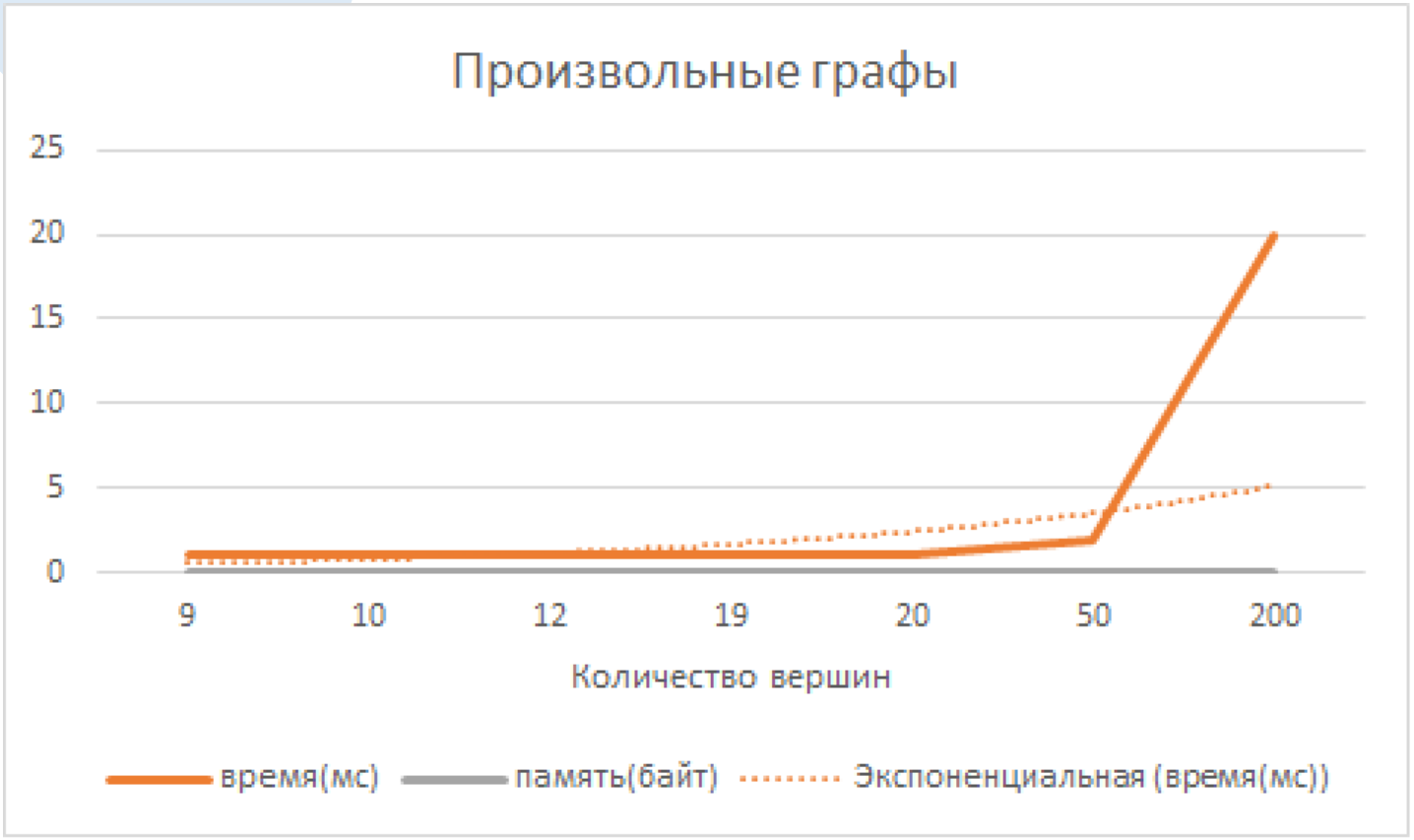
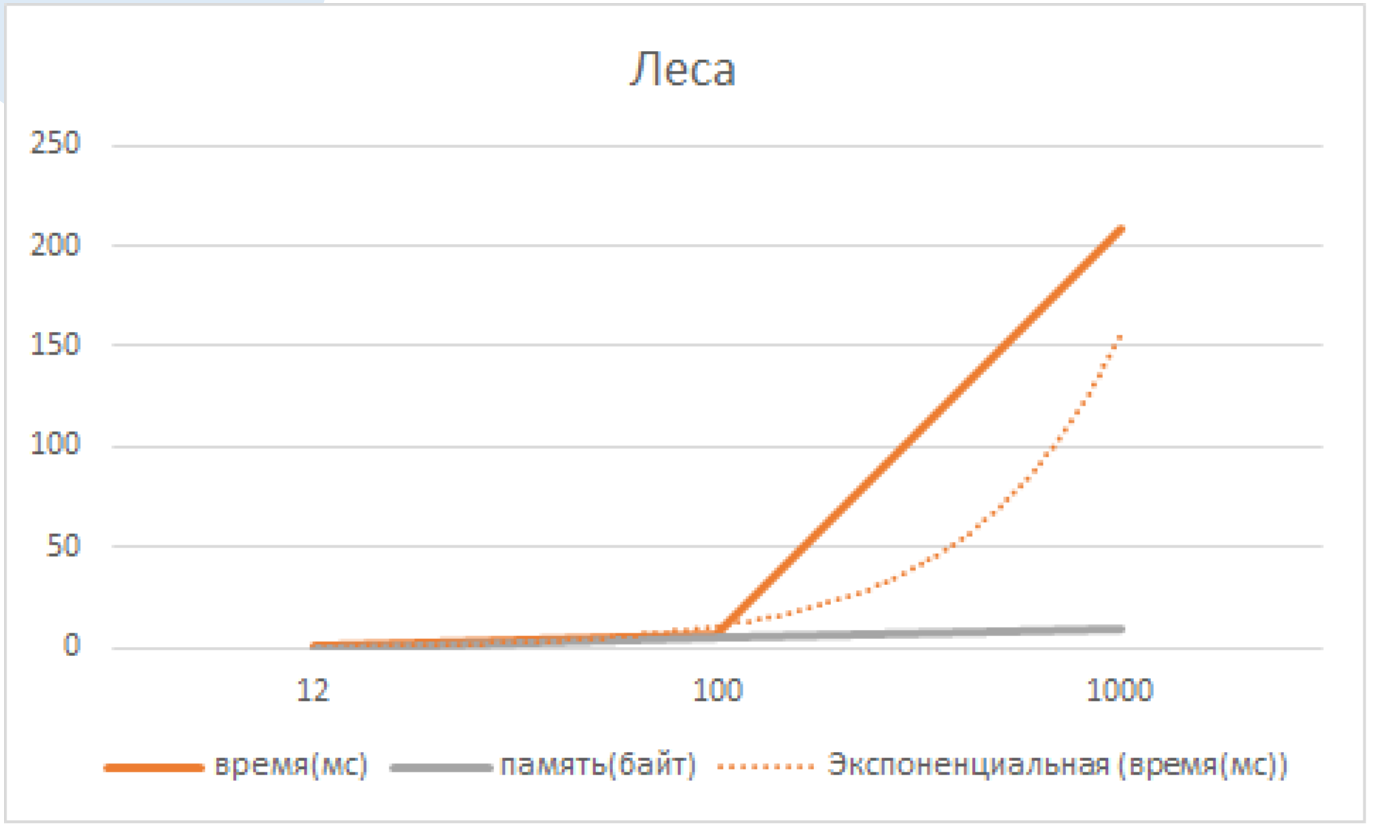


График зависимости памяти и времени от количества вершин

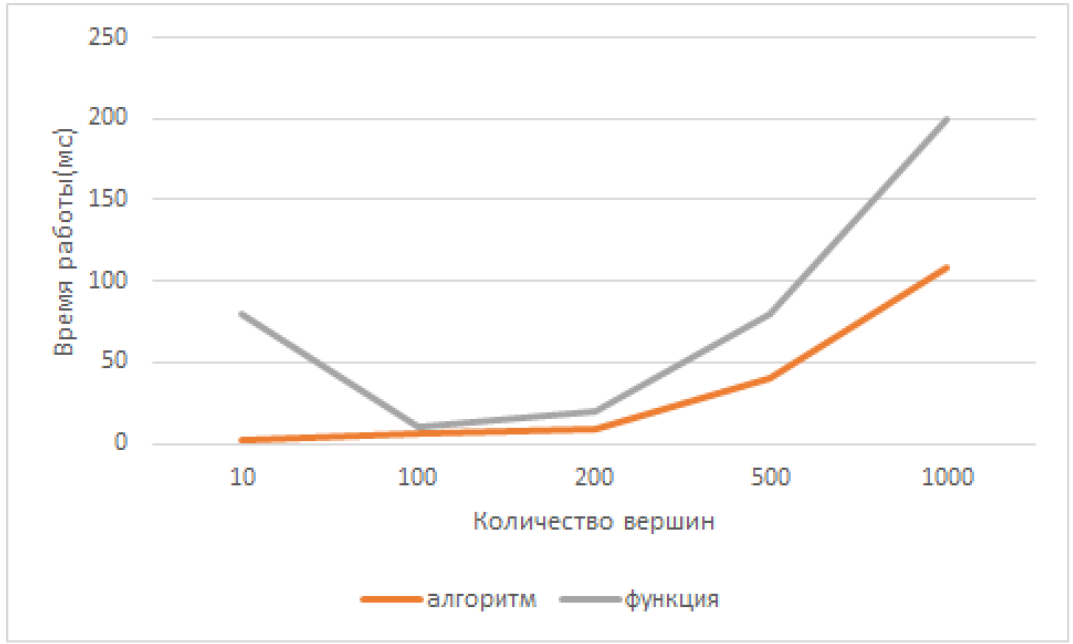


Сравнительный анализ

Критерии проверки решений:

Время работы алгоритма и функции примерно одинаково на графах до 100 алгоритм выигрывает по времени выполнения

При большем количестве вершин на полносвязных графах функция работает значительно быстрее, чем реализованый алгоритм.



Заключение

1. Появление алгоритма «Сжатие цветков» позволило решать новые задачи на графах с нечетными циклами

2. Реализация библиотеки позволяет использовать алгоритм в других проектах

3. Алгоритм работает корректно и достаточно эффективно.

4. Алгоритм работает эффективнее с малым количеством данных, но не сохраняет преимущество при больших объемах данных.