Теория генераторов: отчет по практике

Выполнил студент КНиИТ 431 группы

Серебряков Алексей Владимирович

ГЛАВА II

N 37

Для доказательства утверждения, используем определение неразличимости слов в авт. Мили. Два слова a и b неразличимы автоматом A, если для ∀ вх. слова x, a и b приводят автомат в одно и то же состояние и имеют одинаковую реакцию на вх. слово x.

Дано:

1. V_2 и W_2 неразличимы авт. A.
2. V_1V_2 и W_1W_2 неразличимы авт. A.

Необх. показать, что:

1. V_1W_2 и W_1W_2 неразличимы авт. A.
2. W_1 и V_1 не обязательно неразличимы авт. A.

Док-во:

① Для док-ва неразл. V_1W_2 и W_1V_2 используем св-во конкатенации (И) слов. Если V_1V_2 и W_1W_2 неразличимы $\Rightarrow \forall x$, авт. A приведет V_1V_2 и W_1W_2 в одно и то же состояние с одинаковой реакцией. Поскольку конкатенация слов V_1 и W_2 так же будет вх словом, можно сказать, что авт. A приведёт V_1W_2 и W_1V_2 в одно и то же состояние с одинак. реакцией на ∀ x. Т.О V_1W_2 и W_1V_2 неразличимы авт. A.

② Для показа того, что W_1 и V_1 не обязательно неразл. рассмотрим пример:

Пр: Пусть A - авт. у которого вх слово 01 он переводит из сост. S_1 в сост. S_2 с реакцией "a". Рассмотрим два вх слова: $V_1 = 0$, $W_1 = 1$. В данном случае авт. A на слово "0" перейдет в сост. S_2 из сост. S_1 с реакц. A.

, а на слово "1" перейдет из сост. S_2 в себя же с реакцией "а". Т.О. V_1 и W_2 различимы, поскольку приводят авт. A в разные состояния.

Т.О. показано, что если V_2 и W_2, W_1W_2 и V_1V_2 неразличимы авт. A, то W_1V_2 и V_1W_2 также неразличимы но W_1 и V_1 не обязат. неразличимы.

Глава III

№ 1

Матричный способ технически не будет отличаться от стандартного решения диагностической задачи. Можно исп-ть стандартный алгоритм, проводя операции замены значений меток, смены порядка строк.

	1	2	3	4	5
1	α_0			β_1	
2	α_0				β_1
3	β_1				α_0
4			α_1	β_1	
5		α_1			β_1

Zv	α	β
1	0	1
2	0	1
3	0	1
4	1	1
5	1	1

Обр. двумерные массивы
Т.е. $z[1][\alpha]=0$
$S[3][\beta]=1$

Sv	α	β
1	1	4
2	1	5
3	5	1
4	3	4
5	2	5

Обращение по индексам строки P:

i	gr	α	α_g	β	β_g
1	a	1	·	4	·
2	a	1	·	5	·
3	a	5	·	1	·
4	b	3	·	4	·
5	b	2	·	5	·

Т.е. $\alpha_{group} = P[\alpha][group]$
$\beta_{group} = P[\beta][group]$

После построения P_i таблицы аналогично стандартному решению. Необх. искать различия в колонках $(\alpha_g ; \beta_g)$ и переходить в P_{i-1}, используя полученные α и β.

№ 12

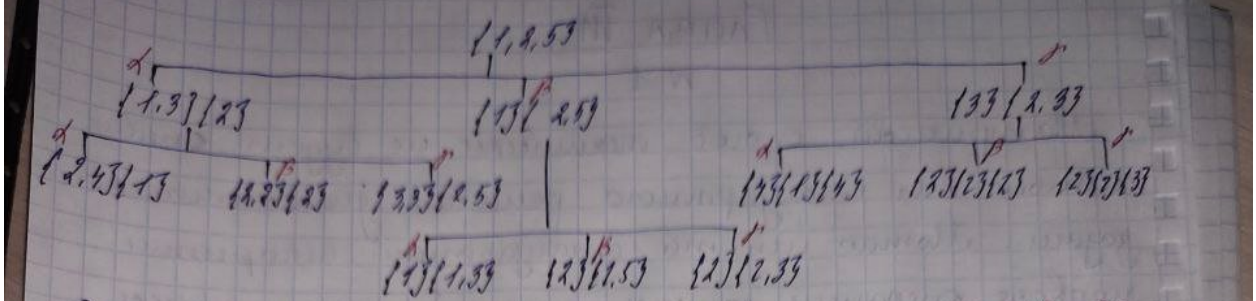
{1, 2, 3, 4, 5}

α: {2, 4}, {1, 1, 3} β: {2, 2} {2, 5, 5} γ: {3, 3} {2, 4, 3}

α: {1, 1} {2, 2, 4} β: {2, 5} {2, 2} γ: {2, 4} {3, 3, 3} α: {4, 4} {1, 1} β: {4} {2, 2} {2, 5} γ: {2} {3, 4}.

α: {1, 1} {1, 3, 3} β: {2, 2} {2, 5, 5} γ: {2, 2} {2, 3, 3}

Ответ: Нач. состояние {1}. Диагност. посл-ть: γα, γβ, γγ



Ответ: Нач. состояние {1,2,4,5}. Диагност. посл: r_4, d_β, d_5

№ 52.

Условие отсутствие синхр. поведения в термах графа достижимости можно опр. ещё образом:

— В орграфе достижимости состояний G' отсутствует синхр. поведение, если для $\forall$ сост. S_1 и S_2 графа G', $\nexists$ путь от S_1 до S_2, такого, что все дуги на этом пути имеют одинаковые метки (реакции).

— Иными словами, если для $\forall S_1, S_2$ $\nexists$ поведения состояний, такой, что для какой пары последовательных состояний в этой последовательности реакции совпадают, то отсутствует синхр. поведение в автомате.

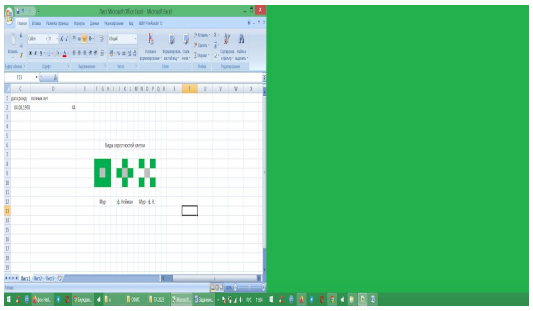
— Это условие означает, что невозможно найти поведение состояний, кот. приведёт все состояния автомата к 1 и тому же начальному состоянию с одинаковыми реакциями, при условии, что в графе достижимости состояний $\nexists$ путь, где все дуги имеют одинаковые метки (реакции).

ГЛАВА 7

Условие задания:

Номер в списке = $12_{10} = 0110_2$

1. $12 \ mod \ 3 \ = \ 0$ => Объединение клетки с окрестностью Мура (8 соседей)



2. Правило перехода: $x_2 = 1$ => клетка приобретает цвет, который имело четное количество клеток из ее окресности.
3. $x_3 x_4 = 10$ =>Требуется найти начальное состояние (или доказать его невозможность), при котором черные клетки в какой-то момент исчезнут, но потом появятся

Доказательство
Для доказательства, что в данном автомате невозможна ситуация, когда все черные клетки исчезнут и затем появятся, можно использовать принцип инвариантности.

Пусть S будет множеством всех черных клеток в начальный момент времени t=0. Докажем, что S остается инвариантом при применении правила перехода автомата.

По определению правила перехода, клетка становится черной в момент времени t+1, если количество черных соседей в момент времени t является четным числом. И наоборот, клетка становится белой в момент времени t+1, если количество черных соседей в момент времени t является нечетным числом.

Предположим, что все черные клетки исчезли в момент времени t=k, но затем появились в момент времени t=k+1. Это означает, что в момент времени t=k все клетки в множестве S были белыми, и все соседи клеток из S также были белыми. Однако, чтобы клетка из S стала черной в момент времени t=k+1, необходимо, чтобы количество черных соседей в момент времени t=k было нечетным числом. Это противоречит предположению, что все клетки в множестве S и их соседи были белыми в момент времени t=k.

Таким образом, мы пришли к противоречию, и можно заключить, что в данном автомате невозможна ситуация, когда все черные клетки исчезнут и затем появятся.

В качестве примеров приведу записи с работой программы:

1. 9x9_sleep(1).mp4 – поле размером 9 на 9 клеток, начальная клетка (черная) в центре, изображение обновляется каждую 1 секунду
2. 40x40_no_sleep.mp4 – поле размером 40 на 40 клеток, начальная клетка (черная) в центре, изображение обновляется по мере вычисления поля
3. 40x40_no_sleep.mp4 – поле размером 40 на 40 клеток, начальная клетка (черная) в центре, изображение обновляется каждую 0.05 долю секунды

4. 180x180_no_sleep.mp4 – поле размером 180 на 180 клеток, начальная клетка (черная) в центре, изображение обновляется по мере вычисления поля

Дополнительное задание

Re: Задачи на автомат

А.С. Богомолов alexbogomolov@yandex.ru 23 мая в 13:56
Я ›

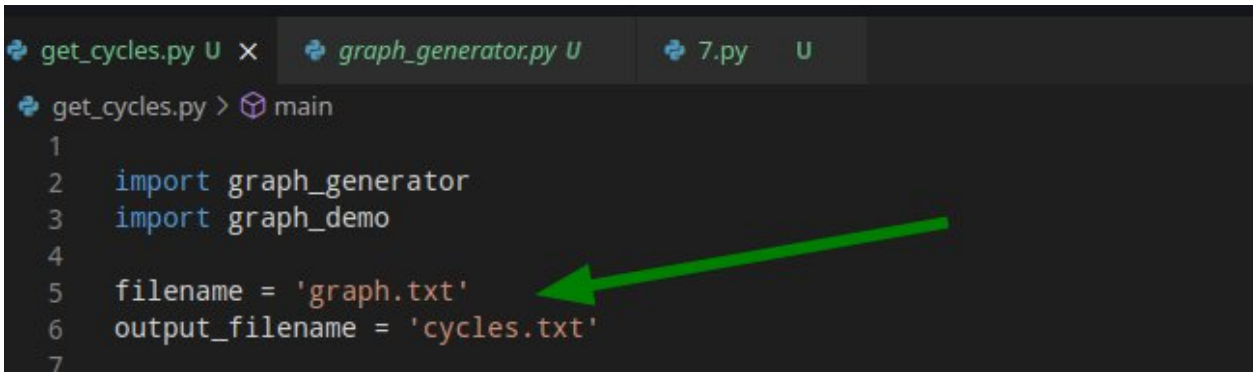
Добрый день!
По практике надо решать задачи из разделов 3 и 7 (они у вас вообще не представлены, либо я что-то пропустил из писем). По теории: нужна небольшая утилитка для поиска контуров в орграфе, дуги которого помечены "+" и "-".

Граф с вершинами 1, 2, ... N задается таблицей из N строк. В строке i на j-том месте стоит пометка входящей дуги из вершины j в вершину i, плюс или минус, или 0, если нет входящей дуги из j в i.
Формат входа - txt (значения разделены пробелами).

Программы выдает все контуры, где дуги только с плюсом, все контуры с минусом, и все знакопеременные контуры с количеством вершин от 2 и дальше. Контур представляется как послед-ть вершин.
Формат выхода - txt с перечислением контуров по одному на строчку: сначала положительные, потом отрицательные, потом знакопеременные (значения разделены пробелами).

--
А.С. Богомолов, д.т.н.,
зав. лаб. системных проблем управления, г.н.с. ФИЦ СНЦ РАН,
проф. каф. МК и КН КНИТ СГУ

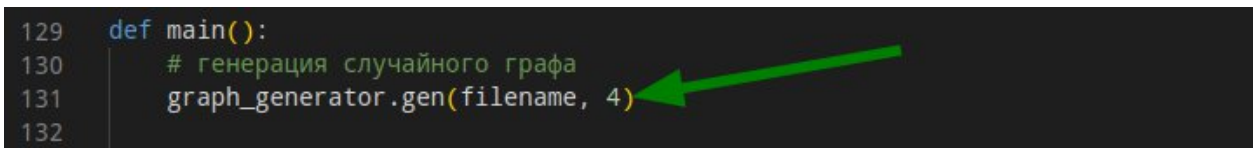
Основная программа get_cycles.py представляет собой код на python, которую можно запусить с помощью команды **python3 get_cycles.py** . Перед запуском программы необходимо проверить, имеются ли все необходимые зависимости. Кроме того в начале программы необходимо указать имя файла с матрицей графа и имя файла для записи контуров.



Зависимости программы:

1. Библиотека networkx (можно установить с помощью pip install networkx)
2. Библиотека matplotlib (можно установить с помощью pip install matplotlib)
3. Модуль random (встроенный)
4. Программа graph_generator.py

Данная программа необходима для тестирования. Она генерирует случайный орграф с количеством вершин, равным n. Параметр n задается вручную (не рекомендуется ставить n > 10 из-за глубины вычислений).



При необходимости проверить конкретный граф, нужно закомментировать строку 131 и ввести матрицу графа вручную в файл с матрицей графа (в стандарте graph.txt).

```
129  def main():
130      # генерация случайного графа
131      # graph_generator.gen(filename, 4)
132
```

5. Программа graph_demo.py

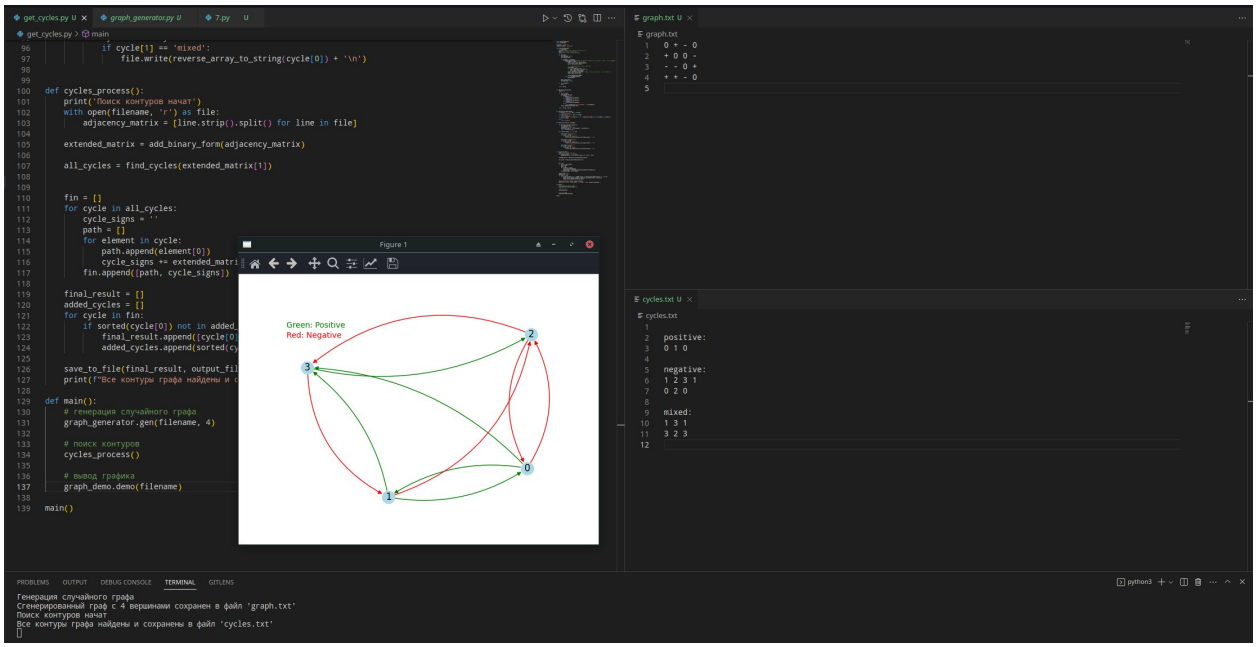
Данная программа необходима для тестирования. Она генерирует изображение проверяемого графа. При необходимости можно отключить отображение, закомментировав строку 137:

```
135
136      # вывод графика
137      # graph_demo.demo(filename)
138
139  main()
```

Тестирование программы:

Для тестирования будут использованы все модули и программы, указанные выше. Программа генерирует случайный орграф, заносит его матрицу в файл graph.txt, вычисляет его контуры (+,- и смешанные) и строит изображение графа для наглядности.

N = 4



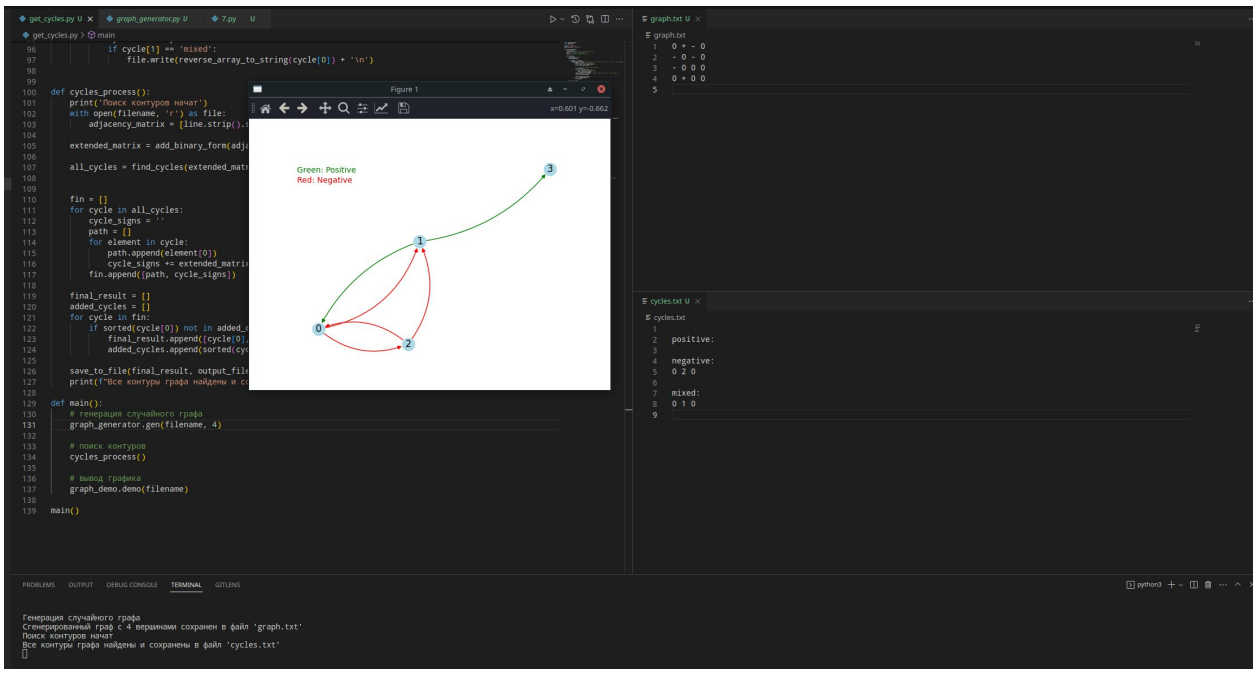


Figure 1
Green: Positive
Red: Negative
graph.txt
0 + - 0
- 0 - 0
- 0 0 0
0 + 0 0
cycles.txt
positive:
0 2 0
negative:
mixed:
0 1 0
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
Генерация случайного графа
Сгенерированный граф с 4 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

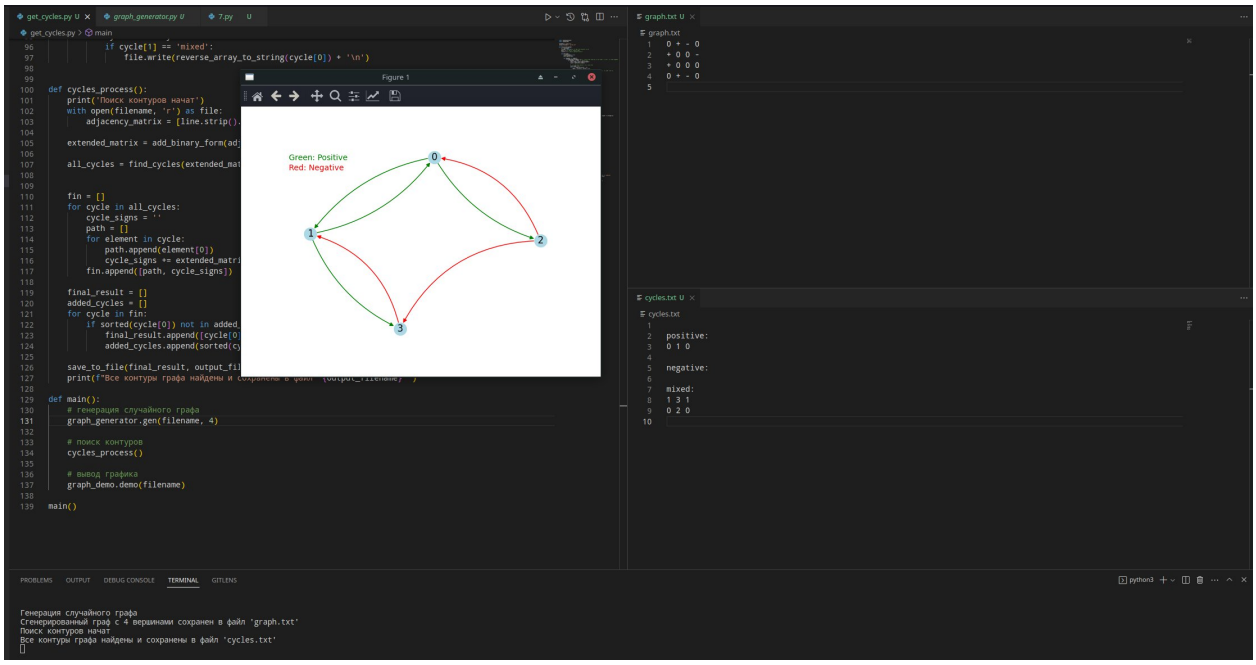


Figure 1
Green: Positive
Red: Negative
graph.txt
0 + - 0
+ 0 0 -
+ 0 0 0
0 + - 0
cycles.txt
positive:
0 1 0
negative:
mixed:
1 3 1
0 2 0
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS
Генерация случайного графа
Сгенерированный граф с 4 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

N = 3

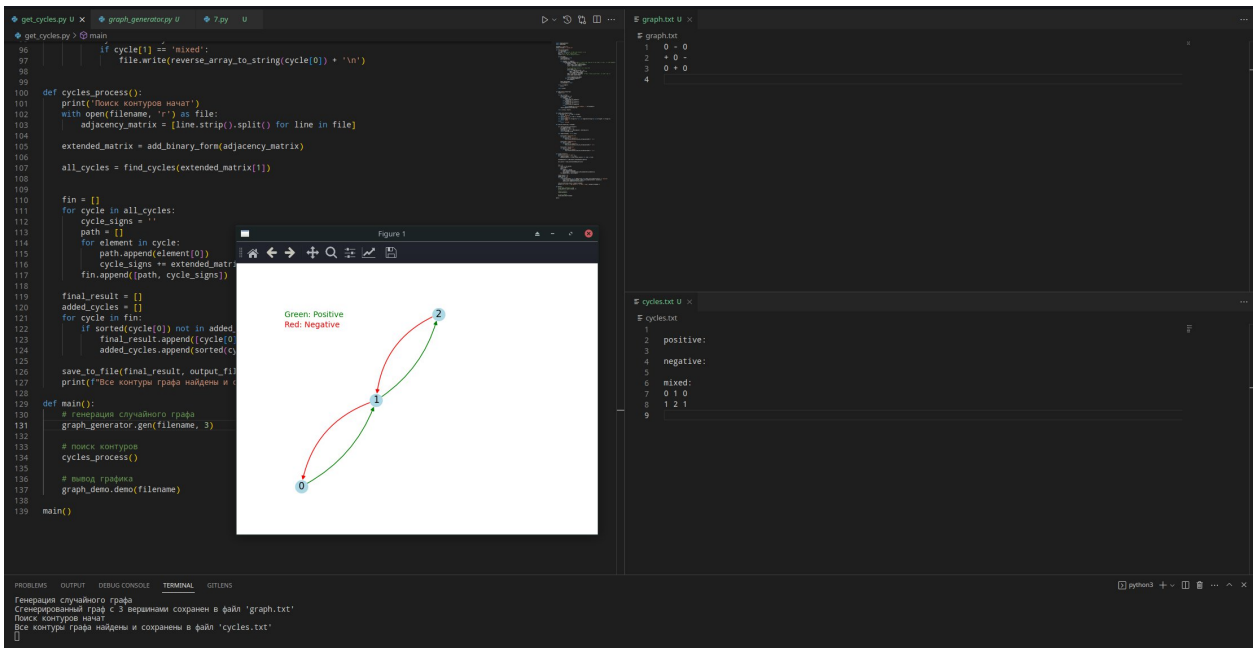


Figure 1
Green: Positive
Red: Negative
graph.txt
0 - 0
+ 0 -
0 + 0
cycles.txt
positive:
negative:
mixed:
0 1 0
1 2 1
Генерация случайного графа
Сгенерированный граф с 3 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

N = 5

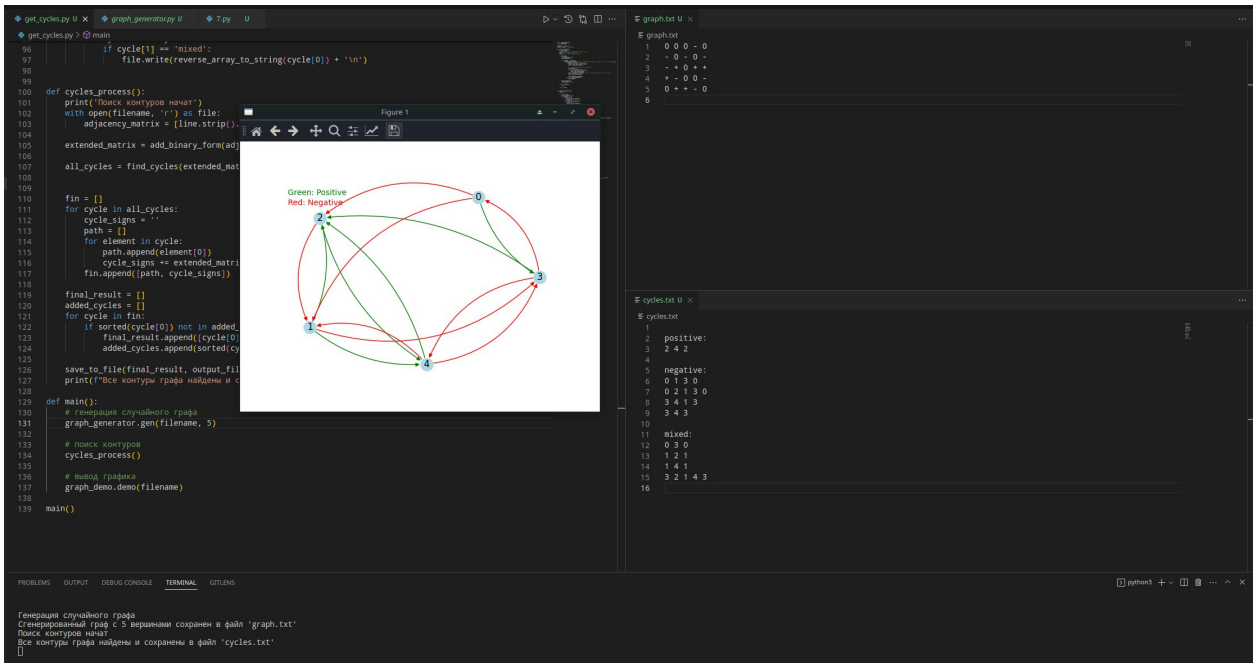


Figure 1
Green: Positive
Red: Negative
graph.txt
0 0 0 - 0
- 0 - 0 -
- + 0 + +
+ - 0 0 -
0 + + - 0
cycles.txt
positive:
2 4 2
negative:
0 1 3 0
0 2 1 3 0
3 4 1 3
3 4 3
mixed:
0 3 0
1 2 1
1 4 1
3 2 1 4 3
Генерация случайного графа
Сгенерированный граф с 5 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

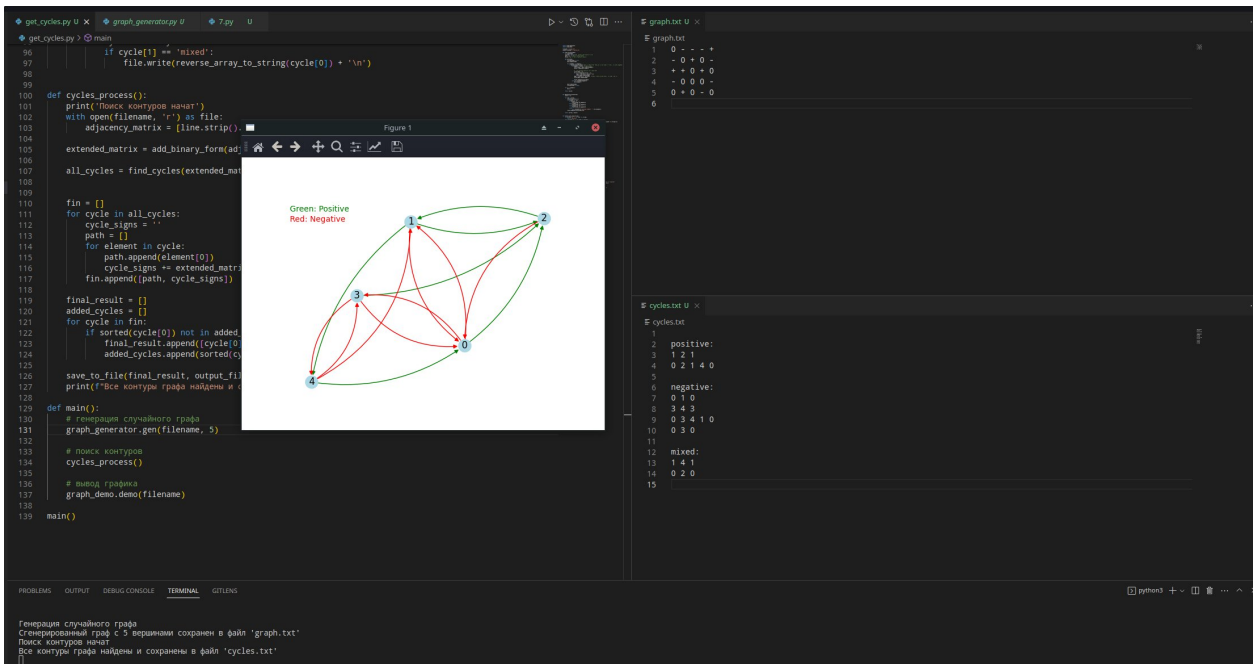


Figure 1
Green: Positive
Red: Negative
graph.txt
0 - - - +
- 0 + 0 -
+ + 0 + 0
- 0 0 0 -
0 + 0 - 0
cycles.txt
positive:
1 2 1
0 2 1 4 0
negative:
0 1 0
3 4 3
0 3 4 1 0
0 3 0
mixed:
1 4 1
0 2 0
Генерация случайного графа
Сгенерированный граф с 5 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

N = 6

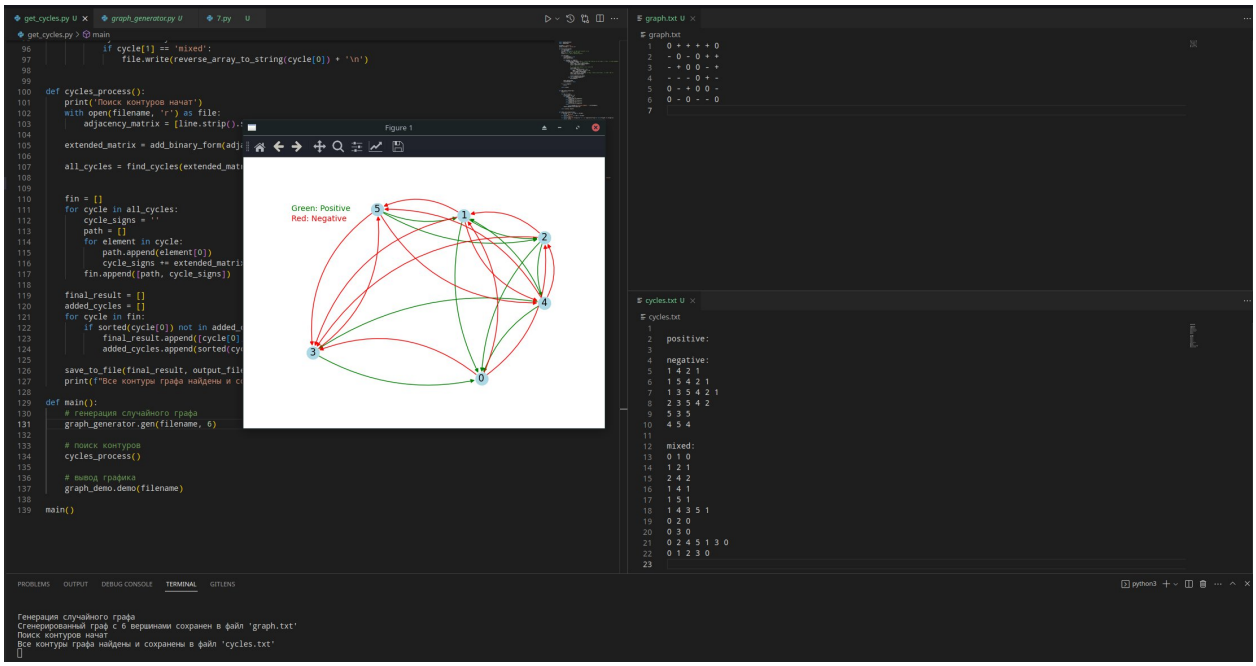


Figure 1
Green: Positive
Red: Negative
graph.txt
0 + + + + 0
- 0 - 0 + +
- + 0 0 - +
- - - 0 + -
0 - + 0 0 -
0 - 0 - - 0
cycles.txt
positive:
negative:
1 4 2 1
1 5 4 2 1
1 3 5 4 2 1
2 3 5 4 2
5 3 5
4 5 4
mixed:
0 1 0
1 2 1
2 4 2
1 4 1
1 5 1
1 4 3 5 1
0 2 0
0 3 0
0 2 4 5 1 3 0
0 1 2 3 0
Генерация случайного графа
Сгенерированный граф с 6 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

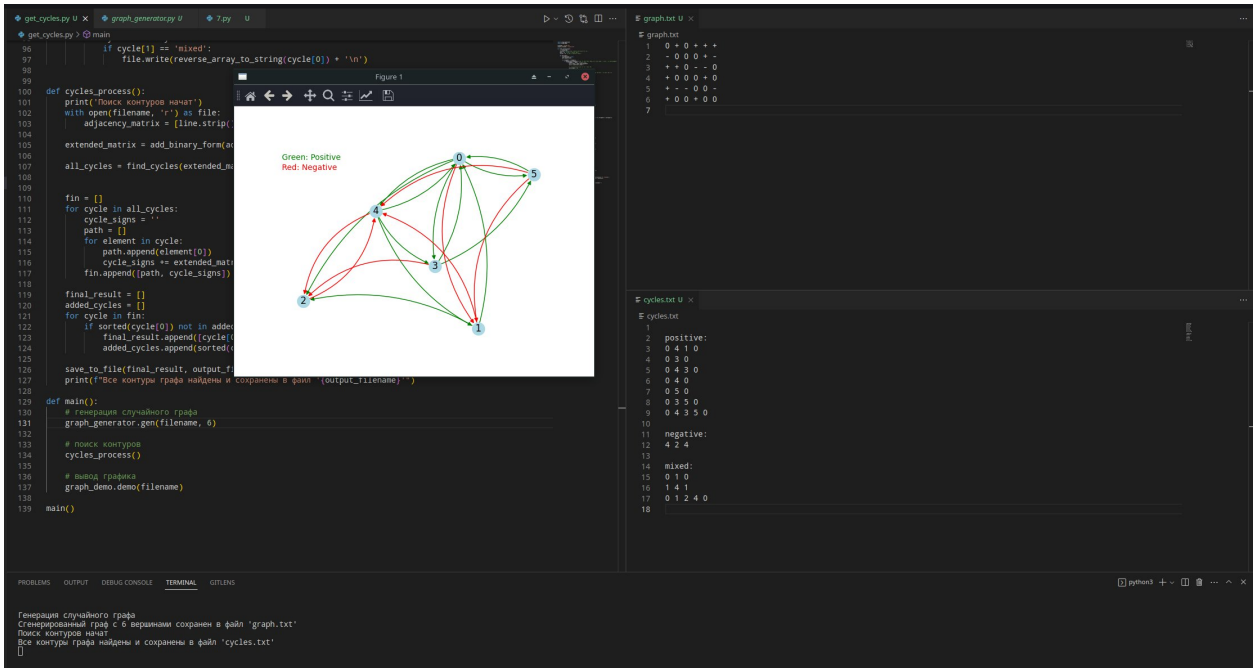


Figure 1
Green: Positive
Red: Negative
graph.txt
0 + 0 + + +
- 0 0 0 + -
+ + 0 - - 0
+ 0 0 0 + 0
+ - - 0 0 -
+ 0 0 + 0 0
cycles.txt
positive:
0 4 1 0
0 3 0
0 4 3 0
0 4 0
0 5 0
0 3 5 0
0 4 3 5 0
negative:
4 2 4
mixed:
0 1 0
1 4 1
0 1 2 4 0
Генерация случайного графа
Сгенерированный граф с 6 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'

N = 10

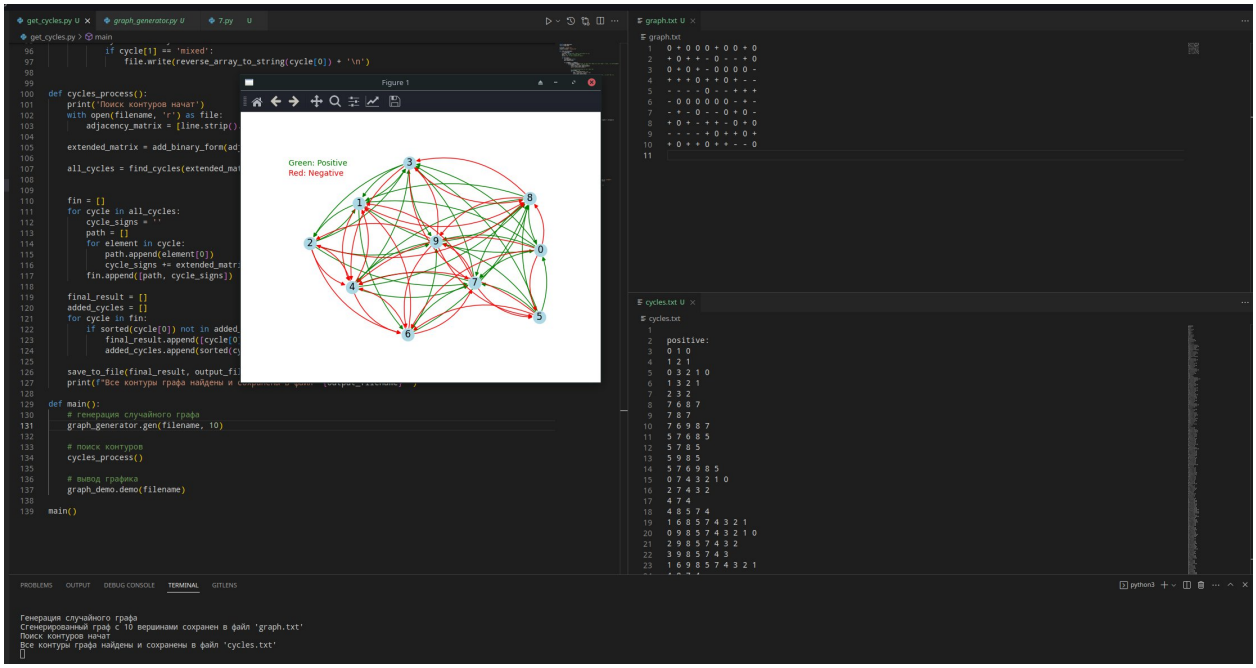


Figure 1
Green: Positive
Red: Negative
graph.txt
0 + 0 0 0 + 0 0 + 0
+ 0 + + - 0 - - + 0
0 + 0 + - 0 0 0 0 -
+ + + 0 + + 0 + - -
- - - - 0 - - + + +
- 0 0 0 0 0 0 - + -
- + - 0 - - 0 + 0 -
+ 0 + - + + - 0 + 0
- - - - + 0 + + 0 +
+ 0 + + 0 + + - - 0
cycles.txt
positive:
0 1 0
1 2 1
0 3 2 1 0
1 3 2 1
2 3 2
7 6 8 7
7 8 7
7 6 9 8 7
5 7 6 8 5
5 7 8 5
5 9 8 5
5 7 6 9 8 5
0 7 4 3 2 1 0
2 7 4 3 2
4 7 4
4 8 5 7 4
1 6 8 5 7 4 3 2 1
0 9 8 5 7 4 3 2 1 0
2 9 8 5 7 4 3 2
3 9 8 5 7 4 3
1 6 9 8 5 7 4 3 2 1
Генерация случайного графа
Сгенерированный граф с 10 вершинами сохранен в файл 'graph.txt'
Поиск контуров начат
Все контуры графа найдены и сохранены в файл 'cycles.txt'