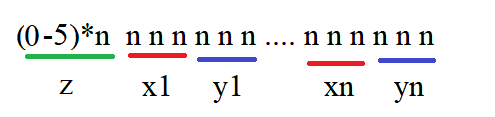
Алгоритм

Найкращій алгоритм, який я зміг придумати працює наступним чином:

На вхід подається число, яке складається з великої кількості цифр. Перші 1-5 цифр будуть використовуватися для встановлення центра розбитого скла, кожні наступні 6 цифр будуть використовуватися для знаходження координати точки, в яку слід провести промінь.



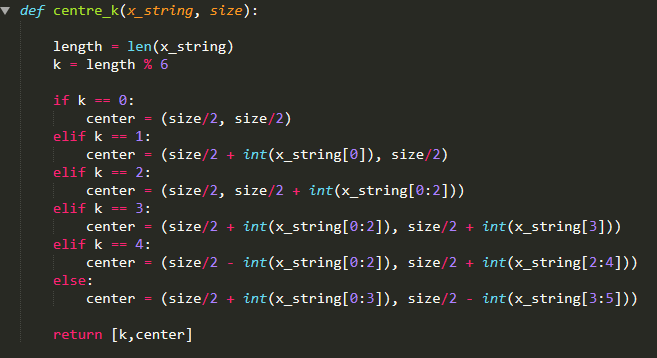
Я дізнаюсь довжину числа z шляхом знаходження залишку від ділення довжини повного числа на 6 (Оскільки кожні 6 елементів будуть використовуватися для знаходження 2 координати тріщини, враховуючи, що 1 – центр). Координати центру визначаються по z.

Існує 6 варіантів:

Залишок від ділення довжини числа на 6 (позначається z)= 0,1,2,3,4,5

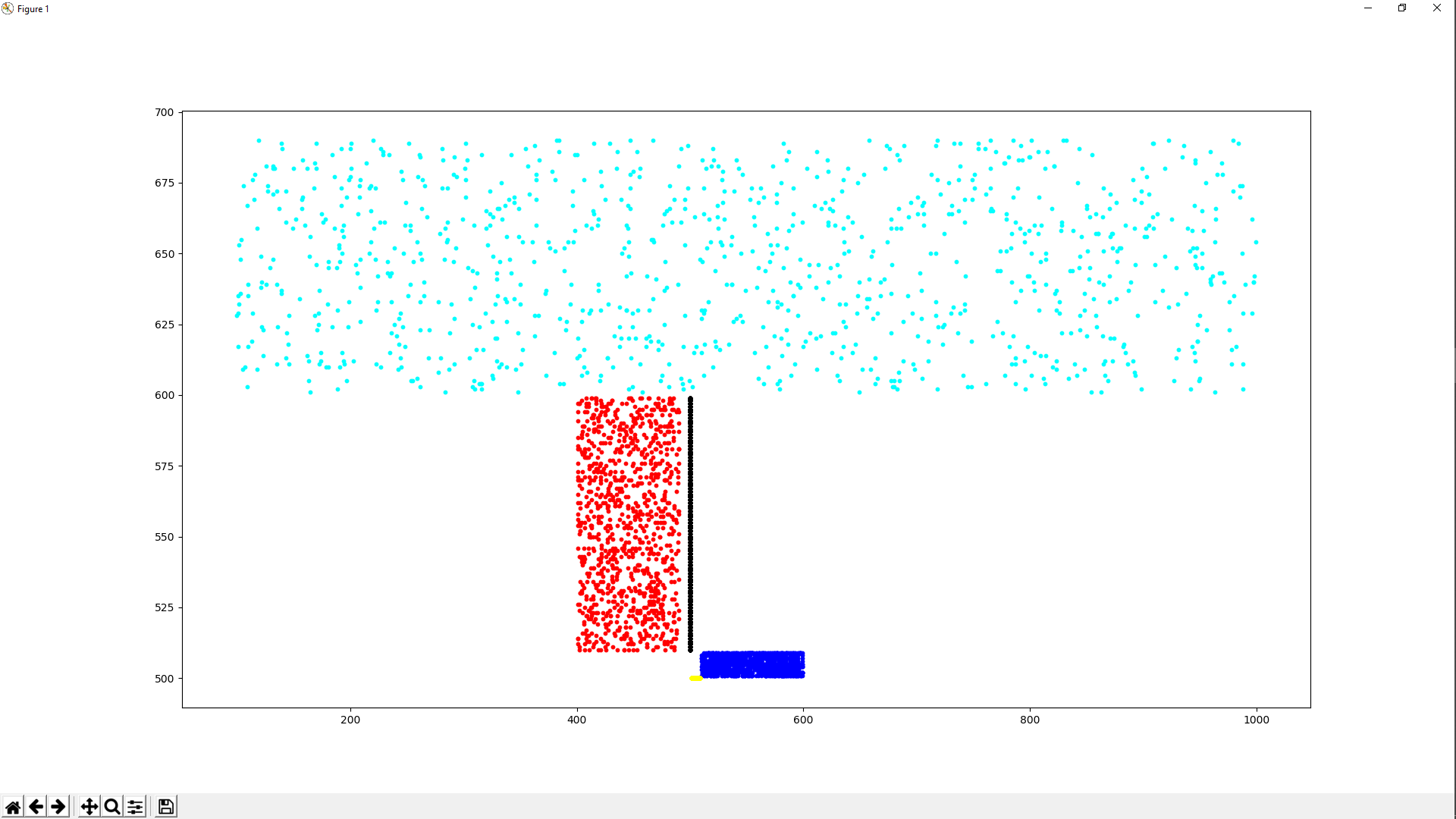
Для кожного варіанту необхідно побудувати таку множину значень, щоб вони не перетиналися.

Я зробив це наступним чином (Впевнений, що існує набагато кращий варіант розбиття, який більш нормований відносно центру та розміру вікна, але обмежень(де повинен знаходитись центр) не було, тому цей варіант задовольняє потреби):

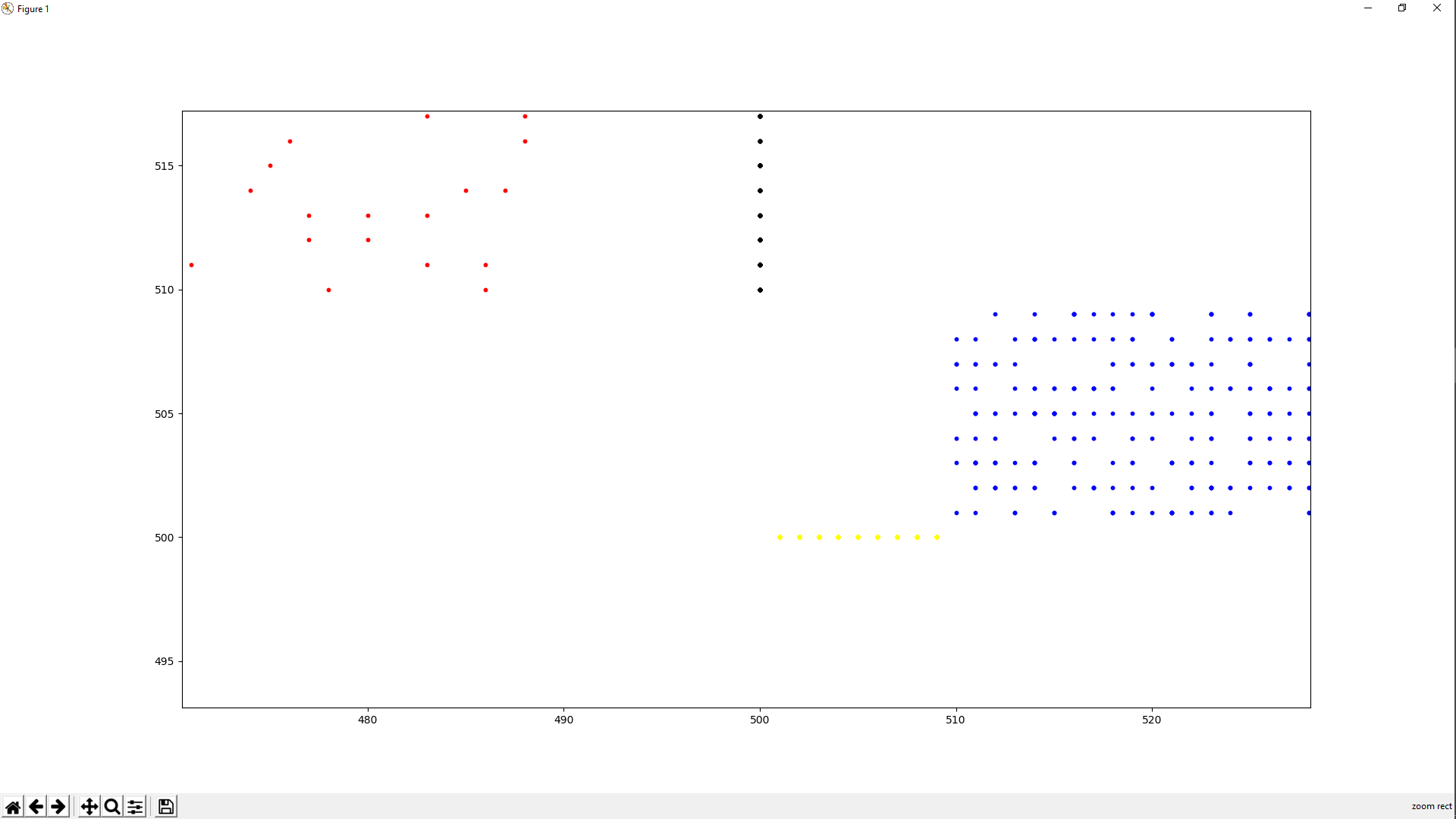


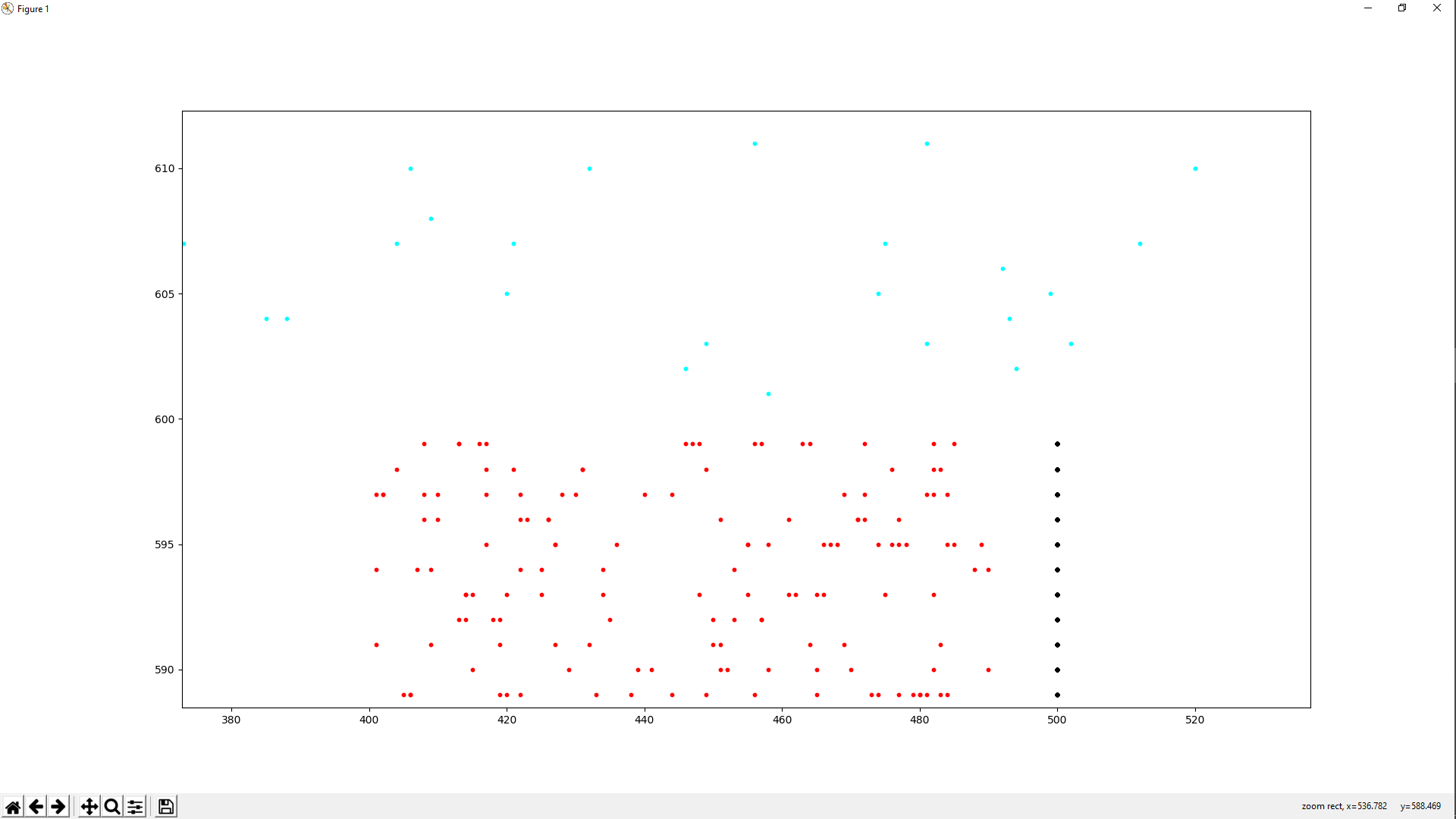
Де x\_string – введене число в форматі стрінг, size – розмір вікна, що = 1000 в моїй задачі.

Щоб перевірити, що розбиття не перетинається, я створюю 1000 випадкових точок центру і дивлюсь на їх місцезнаходження.



Збільшимо, щоб переконатись:



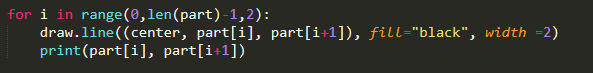


Чудово, отже наші 0-5 елементів визначаюсь координати центра, на даному етапі ніякого накладання результатів не відбувається.

Це я зробив для того, щоб отримати n-кількість координат по 6 значень в кожній. Малюємо тріщину.

Для графічного відображення я використовую PIL .

Код виглядає наступним чином:



де part – це масив, який складається з n-кількості координат точок тріщини. Тобто це вхідне число, без перших 0-5 цифр і розбите по 3 розряди.

Приклад: Для числа х масив part виглядає наступним чином



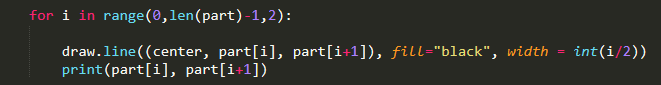
Зауваження!

В цей момент можливе накладання, для чисел що мають масив part таким

1. [111, 222, 333, 444 …..]
2. [333, 444, 111, 222, …..]

Результат зображений на екрані буде однаковий, тому необхідно виправити цю ситуацію.

Я починаю використовувати порядок в масиві part для визначення товщини лінії.

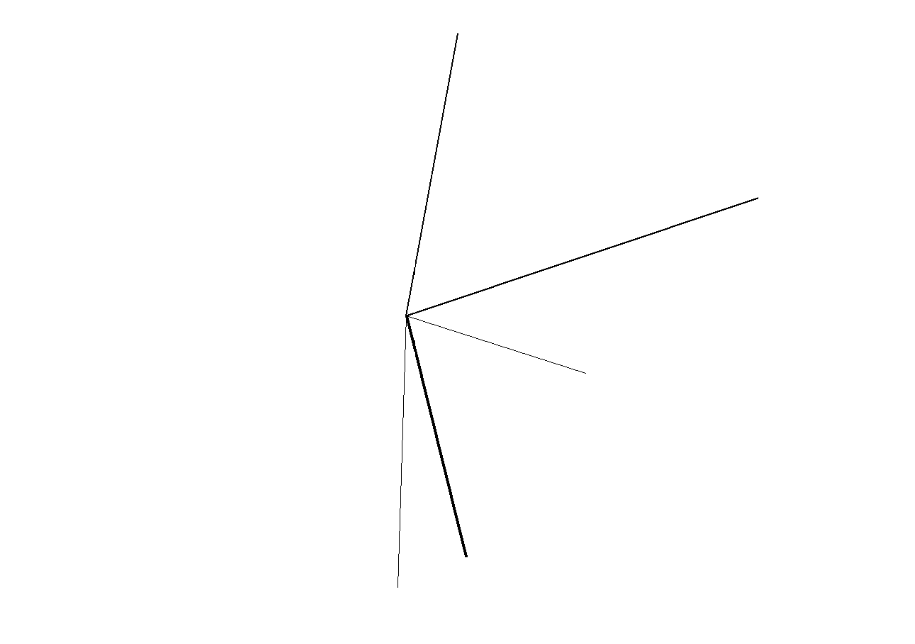
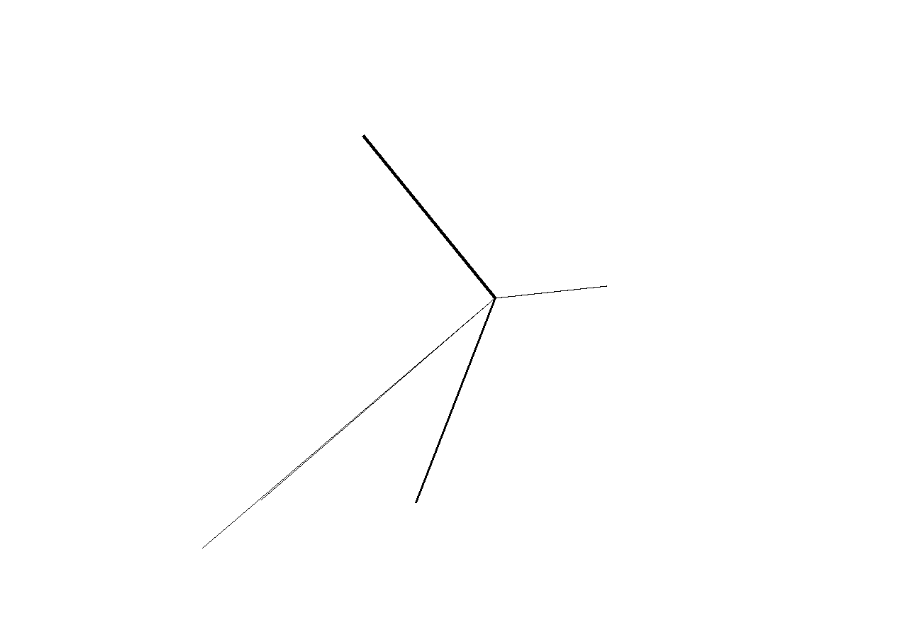


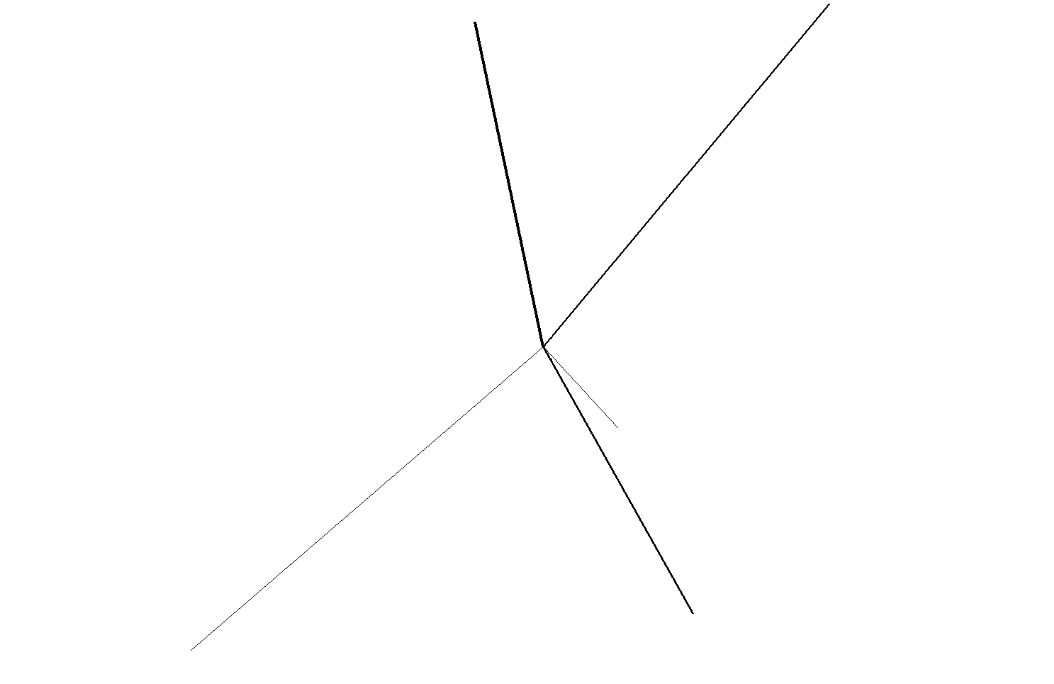
Тепер накладання не буде і при перестановці доданків в масиві товщина ліній буде змінюватись.

Далі зберігаю в пдф / ets , оскільки svg не підтримується в бібліотеці PIL.

Ознайомитись з кодом більш детально можна безпосередньо в файлі коду.

Результати роботи для випадкових чисел 37, 38, 39 порядків:





Плюси алгоритми:

* Не залежить від порядку числа.
* Для кожного введеного числа будує різну тріщину, що я розумів у якості головної задачі.
* Для 2 однакових чисел видає однаковий результат (не використовує функції бібліотек random, тощо).

Мінуси:

* Центр системи не збалансований відносно центра вікна.
* Якщо введене число менше 6 розрядів, то програма не намалює нічого. Помилка не станеться, але і результат буде відсутній (в контексті великих чисел це проблемою не є). Виведеться порожній екран.
* Для числа 100 порядку товщина лінії стає надто великою, відображення незручне.

Інші ідеї:

* За основу взяти цей самий алгоритм, проте використовувати полярні координати. Розбити область на n- секторів, в кожний з яких буде попадати відрізок. Це забезпечить кращий візуальний вигляд, та прибере проблему з товщиною лінії, оскільки в цьому випадку її використовувати не потрібно зовсім.
* Використовувати замість товщини лінії інший параметр – можливо, колір прямої.
* Використовувати рандомні функції, якщо для нас не так важливе накладання.
* Використовувати декілька точок, як центр системи.

В пошукових мережах знайти рішення не вдалось.