Algorytmy geometryczne

Sprawozdanie z ćwiczenia 4.

Paweł Lamża

Dane techniczne urządzenia na którym wykonano ćwiczenie:

Laptop z systemem Windows 10 x64

Procesor: AMD Ryzen™ 5 4600H

Pamięć RAM: 16GB

Środowisko: Jupyter notebook

Ćwiczenie zrealizowano w języku Python 3, z wykorzystaniem bibliotek numpy oraz matplotlib

Opis ćwiczenia

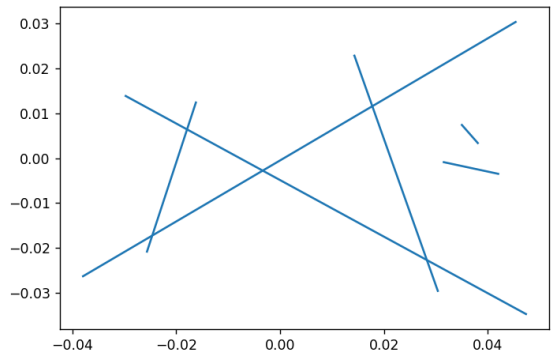
Ćwiczenie polegało na zaimplementowaniu algorytmów:

-Zamiatania, sprawdzający czy choć jedna para odcinków w zadanym zbiorze się przecina

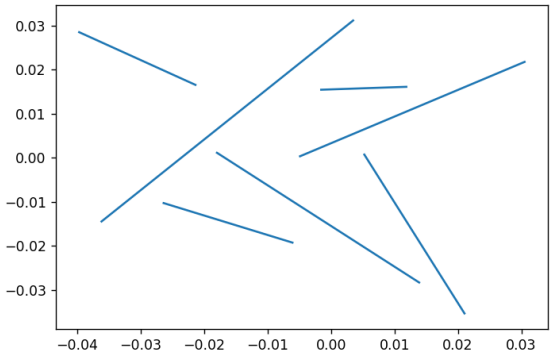
-Zamiatania, wyznaczający wszystkie przecięcia odcinków w zadanym zbiorze

1. Generacja figur.

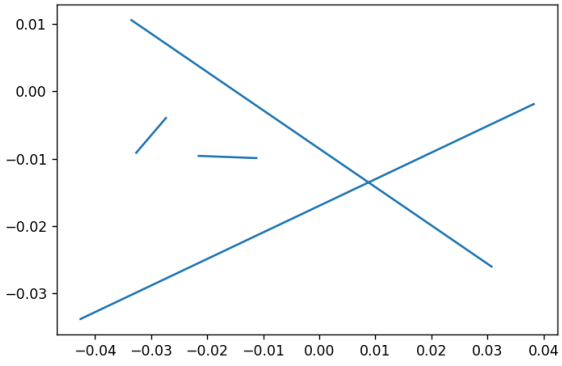
W celu wykonania ćwiczenia wygenerowano 4 zestawów odcinków. Zestaw 1. i 3. Sprawdzają ewentualne przecięcia które mogłyby być naliczane kilka razy. Zestaw 2. Sprawdza poprawność działania algorytmów dla zestawu w którym nie występują przecięcia, natomiast zestaw 4. to zestaw losowo wygenerowanych 10 odcinków. Poniżej kolejne zestawy:



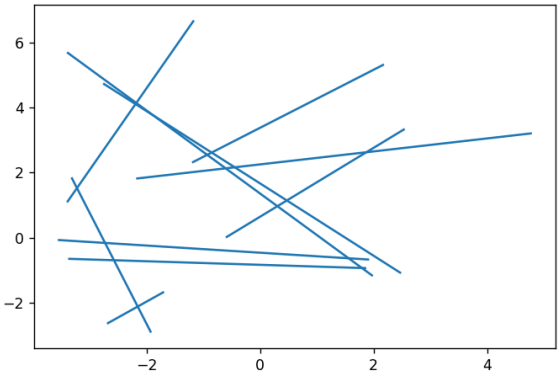
Wykres 1.1, Zestaw 1



Wykres 1.2, Zestaw 2



Wykres 1.3, Zestaw 3



Wykres 1.4, Zestaw 4

2. Metoda obliczania wyznacznika i tolerancja dla zera.

Do obliczania wyznacznika użyłem wyznacznika macierzy 3x3 i wyznacznika macierzy 2x2 własnej implementacji, a jako tolerancje dla zera przyjąłem 10^-12.

3. Struktury Danych:

Odcinki przechowuje jako obiekty klasy Line, a punkty jako obiekty klasy Point. Klasy te umożliwiają sortowanie tych obiektów w strukturach danych tak jak chcemy. W klasie Line przechowywana jest też informacja o tym gdzie znajduje się w tej chwili zamiatacz (tzn. na jakim X). Klasa Point przechowuje informacje o indeksach pierwszego (s) i drugiego (t) odcinka, tworzących dany punkt. Jeśli punkt nie jest punktem przecięcia to indeksy są sobie równe tj. s = t.

4. Algorytm zamiatania, sprawdzający czy choć jedna para odcinków w zadanym zbiorze się przecina:

Opis algorytmu:

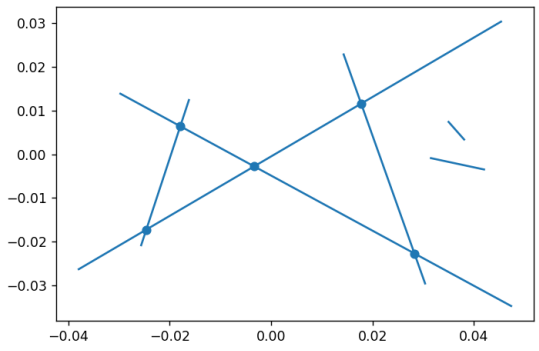
1. Zamieniamy wejściowe linie na punkty i linie jako obiekty klas (odpowiednio Point, Line).
2. Obliczamy w jakich granicach chcemy rysować naszą miotłę.
3. Deklarujemy struktury danych:
    - Struktura zdarzeń Q jako posortowana rosnąco po X lista końców odcinków.
    - Struktura stanu T jako SortedSet.
4. Ustawiamy pozycję zmiatacza na pozycje pierwszego punktu.
5. Przechodzimy do głównej części algorytmu który wykonuje się tak długo jak struktura zdarzeń Q jest niepusta:
    - Pobieramy punkt ze struktury zdarzeń.
    - Jeśli punkt jest początkiem odcinka:
        - Aktualizujemy położenie zamiatacza i dodajemy odcinek do struktury stanu.
        - Jeśli w strukturze T istnieje odcinek powyżej lub poniżej dodanego odcinka to sprawdzamy czy odcinek dodany przecina się z którymś z nich, do tego celu używam funkcji orient i wyznacznika 3x3.
    - Jeśli punkt jest końcem odcinka:
        - Jeśli w strukturze T istnieje odcinek powyżej i poniżej naszego odcinka to sprawdzamy czy te odcinki się przecinają.
6. Jako wynik zwracamy wartość True/False mówiącą o tym czy istnieje przecięcie odcinków oraz zmienną scenes która zawiera sceny do wyświetlenia animacji.

5. Algorytm zamiatania, wyznaczający wszystkie przecięcia odcinków w zadanym zbiorze:

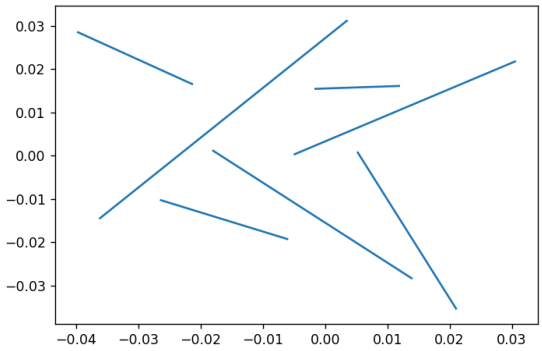
Opis algorytmu:

1. Zamieniamy wejściowe linie i obliczamy granice wyświetlania tak samo jak w poprzednim algorytmie.
2. Deklarujemy struktury danych z tą różnicą do poprzedniego algorytmu, że struktura zdarzeń Q to tym razem SortedSet, ponieważ potrzebujemy możliwości dodawania punktów przecięcia do struktury zdarzeń.
3. Dodajemy wszystkie końce odcinków do struktury zdarzeń.
4. Ustawiamy pozycję zmiatacza na pozycje pierwszego punktu.
5. Przechodzimy do głównego algorytmu:
    - Pobieramy punkt ze struktury zdarzeń
    - Jeśli punkt jest początkiem odcinka:
        - Aktualizujemy położenie zamiatacza i dodajemy odcinek do struktury stanu.
        - Jeśli w strukturze T istnieje odcinek powyżej lub poniżej dodanego odcinka to sprawdzamy czy odcinek dodany przecina się z którymś z nich.
    - Jeśli punkt jest końcem odcinka:
        - Jeśli w strukturze T istnieje odcinek powyżej i poniżej naszego odcinka to sprawdzamy czy te odcinki się przecinają
        - Usuwamy odcinek ze struktury T
    - Jeśli punkt jest punktem przecięcia się dwóch odcinków:
        - Zamiana kolejności:
            1. Usuwamy ze struktury T te dwa odcinki
            2. Zwiększamy pozycję zamiatacza o 0.00001
            3. Dodaje usunięte odcinki do struktury
        - Sprawdzam odcinki z ich nowymi sąsiadami czy się przecinają
6. Jako wynik zwracamy listę punktów przecięcia jako współrzędne (X,Y), listę punktów jako obiekty klasy Point, które posiadają informację o indeksach odcinków przecinających się w danym punkcie, zmienną scenes która zawiera sceny do wyświetlenia animacji.

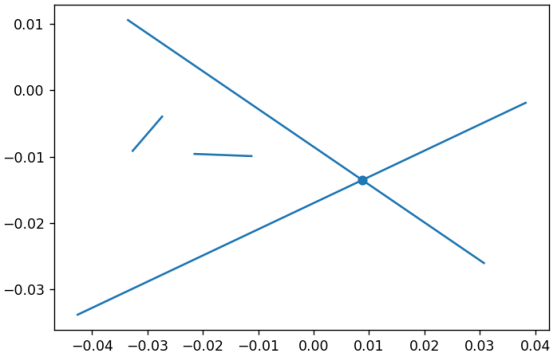
6. Wynik algorytmu:



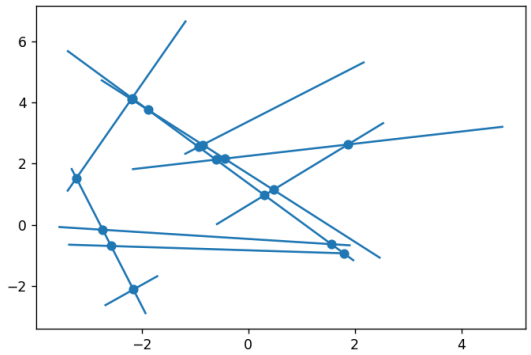
Wykres 7.1 Wynik zestaw 1



Wykres 7.2 Wynik zestaw 2



Wykres 7.3 Wynik zestaw 3



Wykres 7.4 Wynik zestaw 4

7. Wnioski i spostrzeżenia

- Dla wszystkich zadanych zestawów program zachował się tak jak powinien
- Między dwoma algorytmami była konieczna zmiana w strukturze zdarzeń , ponieważ przy wykrywaniu istnienia przecięcia nie musieliśmy dodawać do struktury punktów przecięcia to zwykła lista wystarczyła, natomiast w algorytmie znajdywania wszystkich przecięć dodawanie punktów do struktury zdarzeń było konieczną operacją.
- Program uwzględnia przypadki kiedy dane punkty przecięcia są wykrywane więcej niż raz i jest na to odporny poprzez sprawdzanie indeksów przecinających się odcinków.