Łukasz Stępień gr. IV

Algorytmy geometryczne

sprawozdanie z ćw. 4

1. Cel ćwiczenia:

Implementacja oraz przetestowanie działania algorytmu zamiatania, który wyznacza punkty przecięcia odcinków na płaszczyźnie.

2. Dane techniczne:

Język implementacji: Python

Środowisko programistyczne: Jupyter Notebook

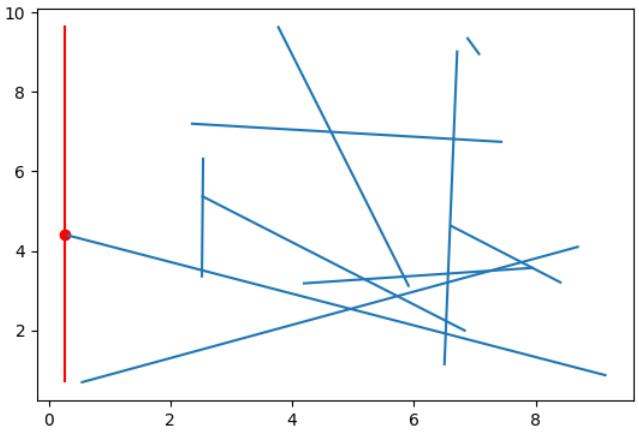
System operacyjny: Microsoft Windows 10 Pro x64

Procesor: Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz, 2904 MHz

3. Zbiory danych

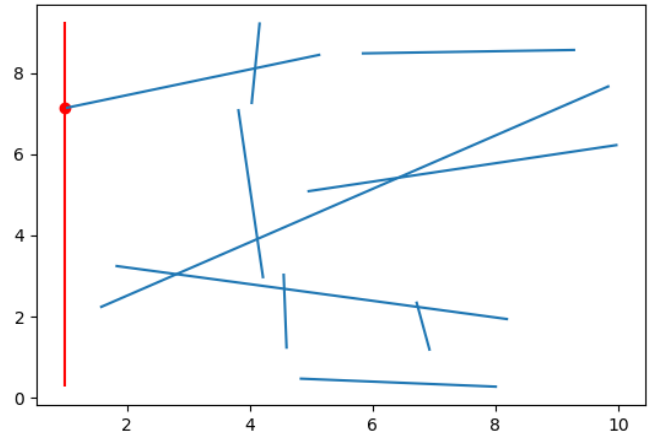
Aplikacja pozwala na tworzenie wprowadzanie odcinków przez użytkownika oraz generowanie losowych odcinków. Wprowadzony / wygenerowany zestaw danych zapisywany jest do pliku w formacie json jako lista linii. Następnie plik ten może być wczytywany, a jego zawartość wykorzystywana w algorytmie. Wprowadzanie ręczne odcinków odbywa się za pomocą dostarczonego narzędzia graficznego. Za pomocą myszki należy wprowadzić kolejne odcinki i wykonać kod służący konwersji i zapisowi do pliku. Generowanie odcinków polega na losowaniu dla każdej linii (których ilość jest zadana parametrem) pary odcinków o współrzędnych mieszczących się w zadanym przedziale.

- Zestaw A:



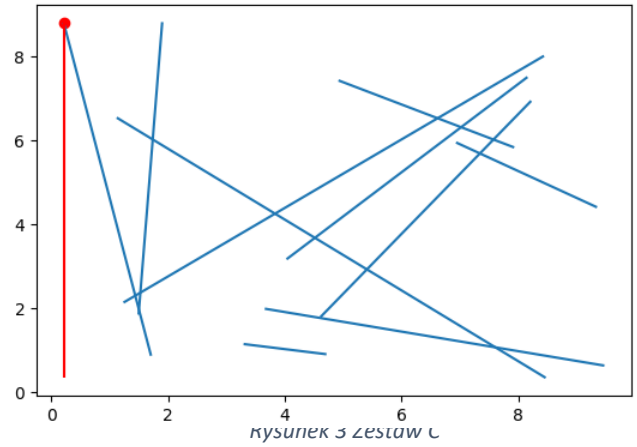
Rysunek 1 Zestaw A

- Zestaw B



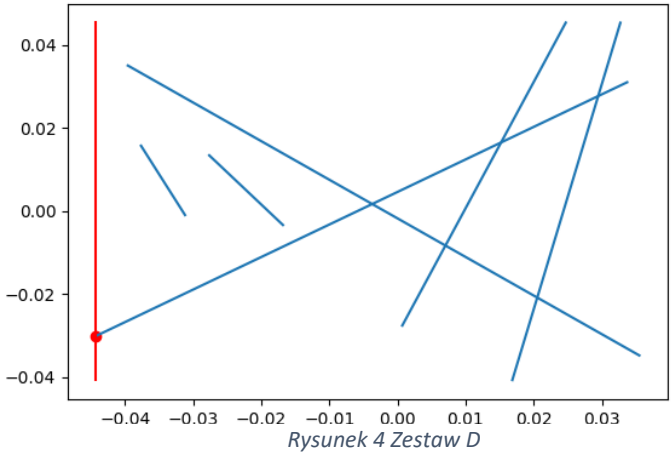
Rysunek 2 Zestaw B

- Zestaw C



Rysunek 3 Zestaw C

- Zestaw D



Rysunek 4 Zestaw D

4. Implementacja algorytmu

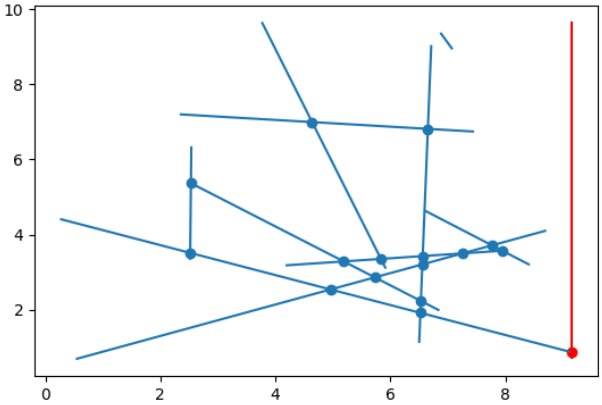
Funkcja *line_intersection* przyjmuje dwa odcinki. Jeśli one się przecinają, to zwraca ich punkt przecięcia w postaci obiektu klasy *Point*, który zawiera informacje, które odcinki go przecinają.

Do implementacji algorytmu zamiatania została użyta klasa *SweepLineStatus.* Jako strukturę stanu oraz zdarzeń użyto *SortedSet*. Problem wykrywania przecięcia więcej niż jeden raz rozwiązano poprzez wykorzystanie struktury *set* jako struktury zdarzeń, co zapobiega dodaniu tego samego zdarzenia dwa razy. Metoda *run* odpowiada za uruchomienie algorytmu zamiatania, *find_intersection* za wykrycie punktu przecięcia się odcinków, *insert_line* oraz *remove_line* za odpowiednio wstawienie lub usunięcie odcinka ze struktury stanu oraz jednocześnie wstawienie punktów przecięcia do struktury zdarzeń.

5. Wynik algorytmu

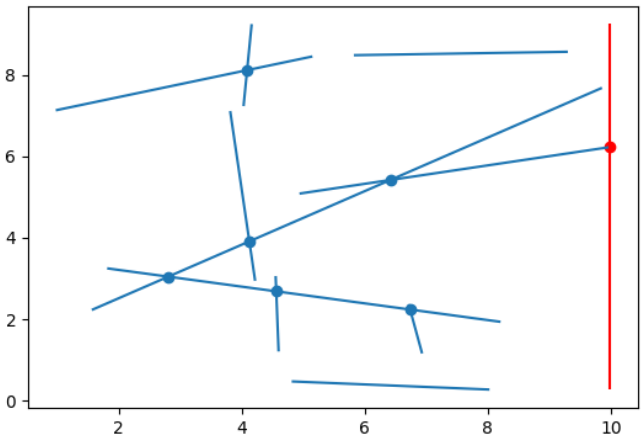
Algorytm na wejściu przyjmuje tablicę linii i zwraca zbiór punktów przecięć w postaci listy krotek, zbiór punktów będących obiektami (a więc zawierający informacje które linie się przecinają) oraz listę scen potrzebnych do wizualizacji.

- Zestaw A



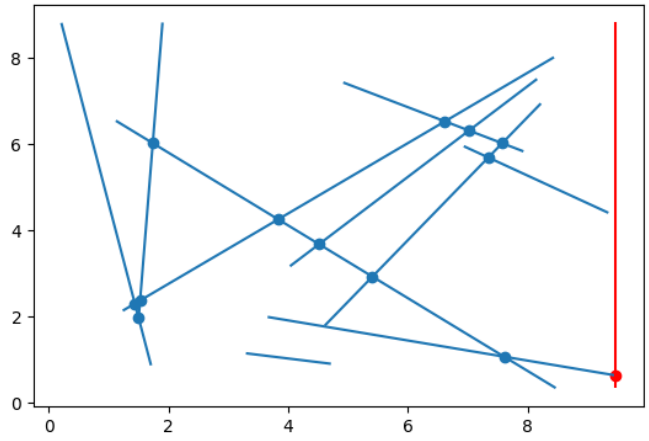
Rysunek 5 Wynik dla zestawu A

- Zestaw B



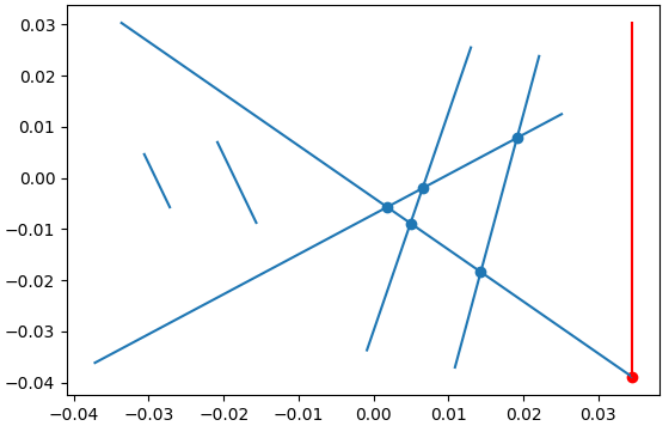
Rysunek 6 Wynik dla zestawu B

- Zestaw C



Rysunek 7 Wynik dla zestawu C

- Zestaw D



Rysunek 8 Wynik dla zestawu D

6. Wnioski

Algorytm zadziałał poprawnie dla testowanych zbiorów. Dzięki zastosowanym strukturom danych złożoność jest optymalna dla tego typu algorytmu. Wykrycie tylko jednego przecięcia jest dosyć proste, natomiast to właśnie obsługa zdarzeń będących punktami przecięcia jest trudnością w algorytmie głównym.