ALGORYTMY GEOMETRYCZNE

Sprawozdanie z ćwiczenia 4.

Bartosz Kucharz

Grupa nr 1

1. Specyfikacja techniczna:

Komputer na którym wykonywano obliczenia:

System operacyjny: Windows 10 x64
Procesor: Intel® Core™ i5-5300U CPU @ 2.30Ghz
Pamięć RAM: 8GB

Obliczenia zostały wykonywane w języku Python 3 z wykorzystaniem bibliotek: numpy random, sortedcontainers i matplotlib.

2. Opis ćwiczenia:

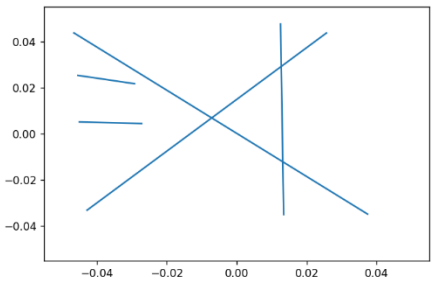
Celem ćwiczenia było zapoznanie się z algorytmem wyznaczania przecięć odcinków na płaszczyźnie oraz jego implementacja.

3. Zestawy danych

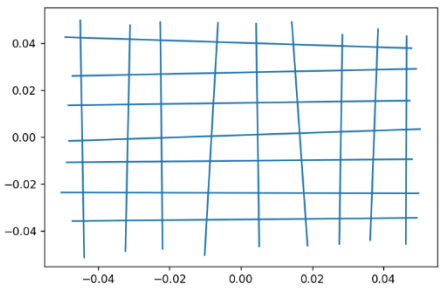
Program korzysta z dostarczonego narzędzia graficznego. Zbiory odcinków można zadawać myszką wprowadzając punkty początkowe i końcowe kolejnych odcinków. Istnieje możliwość zapisu wprowadzonego zbioru odcinków do pliku json i późniejszego odczytu.

Do testów wybrano cztery zbiory punktów:

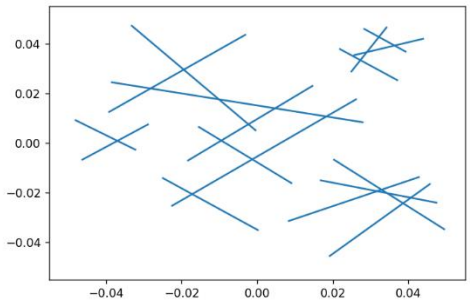
1) Zbiór podany na zajęciach
2) Siatka
3) Iksy
4) Chaotyczny zbiór



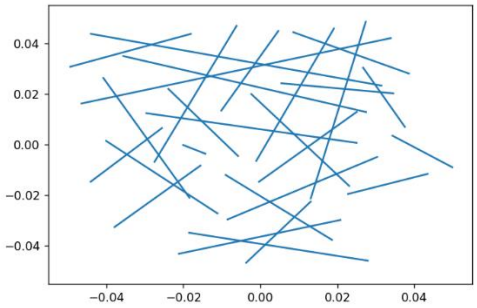
Rys. 1 Zbiór odcinków podany na zajęciach



Rys. 2 Zbiór odcinków "Siatka"



Rys. 3 Zbiór odcinków "Iksy"



Rys. 4 Zbiór odcinków "Chaos"

4. Struktury

Do implementacji struktury stanu oraz struktury zdarzeń wykorzystano zbiór uporządkowany SortedSet z biblioteki sortedcontainers. SortedSet jest oparty na zrównoważonym drzewie binarnym z logarytmicznymi czasami wstawiania i usuwania elementów. Odcinki w strukturze stanu są porównywane względem współrzędnej x punktów przecięcia odcinków należących do struktury z linią zamiatającą.

Zaimplementowano również nowe klasy punktów oraz odcinków.

W klasie punktu poza jego współrzędnymi przechowywana jest referencja do odcinka na którym się znajduję oraz informacja czy jest to punkt początkowy czy końcowy odcinka.

Klasa odcinka przechowuje punkt początkowy oraz końcowy odcinka, współczynnik kierunkowy prostej na której się znajduję oraz przecięcie tej prostej z osią y.

5. Wyznaczanie przecięć

Algorytm jako dane wejściowe otrzymuje zbiór odcinków. Na początku do struktury zdarzeń dodane są wszystkie punkty początkowe i końcowe danych odcinków. Następnie kolejno wyciągane są punkty ze struktury zdarzeń o największej współrzędnej y i w zależności czy jest to początkowy, końcowy czy środkowy punkt odcinka to obsługiwany jest w inny sposób.

Dla zdarzeń będących początkami odcinków, dany odcinek dodawany jest do struktury stanu oraz sprawdzane jest jego przecięcie z lewym i prawym sąsiadem. Jeśli takie przecięcia istnieją to dodawane są do struktury zdarzeń.

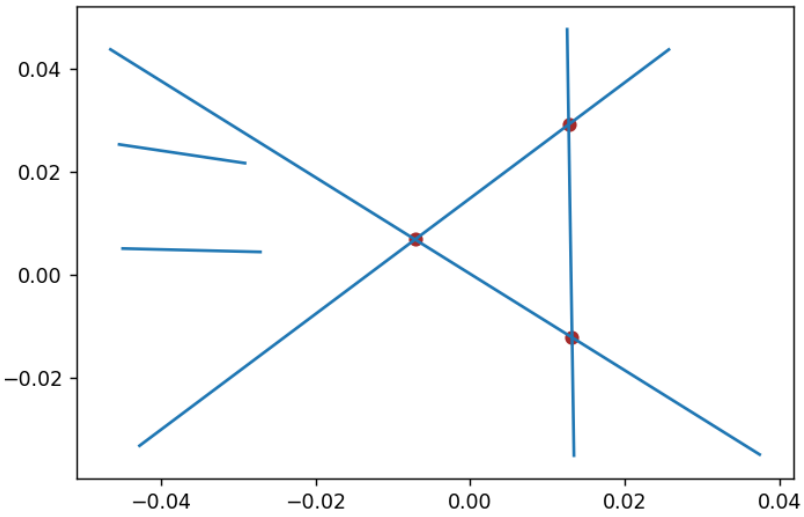
W przypadku punktów końcowych sprawdzane jest przecięcie jego lewego sąsiada z prawym i jeśli istnieje dodawane jest do struktury zdarzeń. Na koniec odcinek do którego należał wyciągnięty punkt zostaje usunięty z struktury stanu.

W przypadku punktu przecięcia, jest on dodawany do wynikowej listy punktów przecięcia. Następnie odcinki przecinające się zamieniane są w strukturze stanu poprzez usunięcie ich dla linii zamiatającej będącej o e przed zdarzeniem i późniejsze dodanie ich z powrotem dla linii zamiatającej o e po zdarzeniu, gdzie e wynosi 1E-16. Na koniec sprawdzane są przecięcia tych odcinków z ich nowymi sąsiadami.

Ostatecznie po wyczerpaniu się punktów w strukturze zdarzeń algorytm zwraca listę punktów własnego stworzonego typu które zawierają informacje o odcinkach które przecinają się w danym punkcie. Zwracana jest również lista scen potrzebna do wizualizacji działania algorytmu.

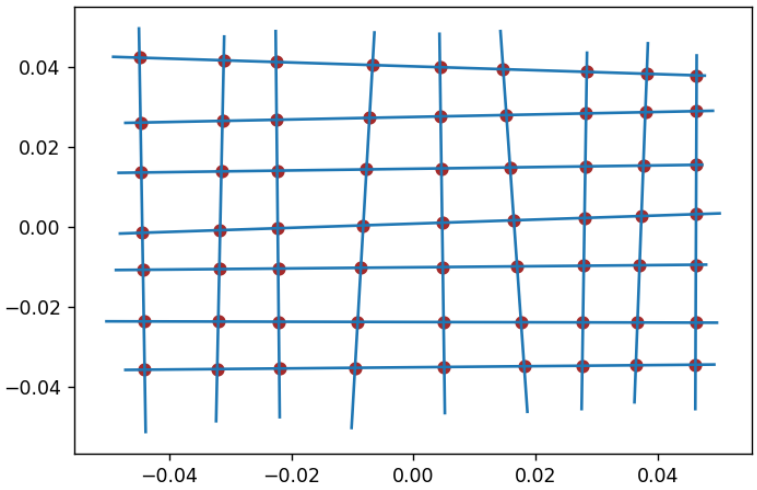
6. Wyniki

Dla zadanych zbiorów odcinków algorytm poprawnie wyznaczył punkty przecięcia.



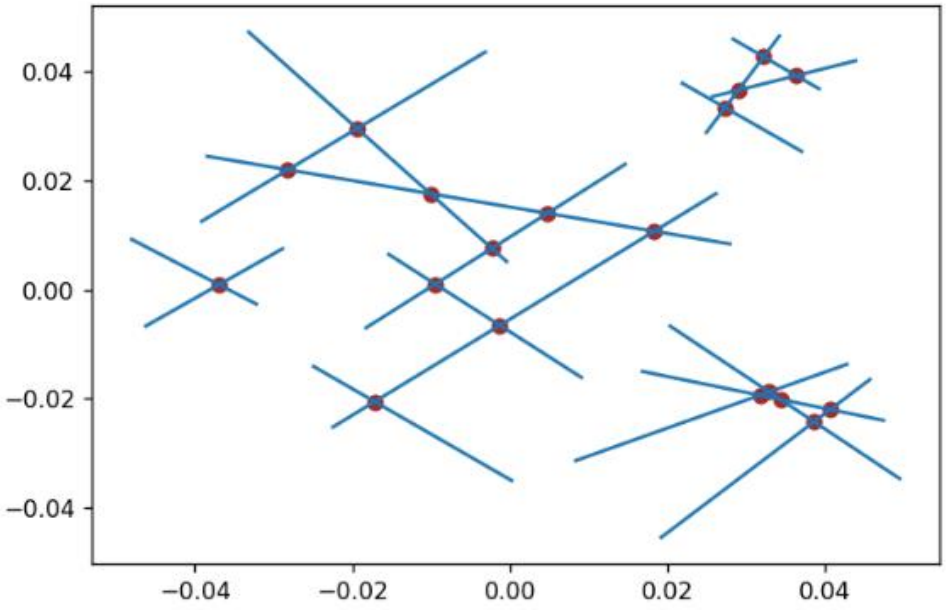
Rys. 5 Zbiór 1. z wyznaczonymi punktami przecięcia

```
Liczba wykrytych przecięć: 3
```



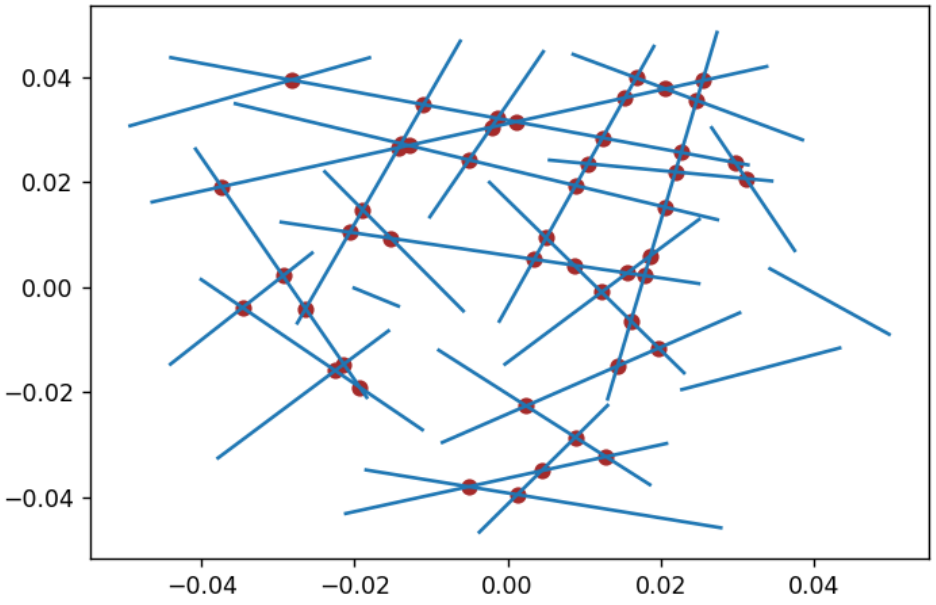
Rys. 6 Zbiór 2. z wyznaczonymi punktami przecięcia

```
Liczba wykrytych przecięć: 63
```



Rys. 7 Zbiór 3. z wyznaczonymi punktami przecięcia

```
Liczba wykrytych przecięć: 19
```



Rys. 8 Zbiór 4. z wyznaczonymi punktami przecięcia

```
Liczba wykrytych przecięć: 48
```

7. Wnioski

Algorytm prawidłowo wyznaczył punkty przecięć odcinków dla zadanych zestawów danych. Zastosowanie implementacji SortedSet z biblioteki sortedcontainers jako struktury stanu i zdarzeń pozwoliło osiągnąć złożoność $O((P+n) \log n)$ gdzie P to liczba przecięć a n to liczba punktów wyznaczających odcinki. Największą trudnością w implementacji algorytmu okazała się obsługa zdarzenia punktu przecięcia, a w szczególności zamiana miejscami odcinków przecinających się. Problem ten można by rozwiązać dodatkowym porównywaniem odcinków w strukturze stanu na podstawie obliczonego wyznacznika.