Sprawozdanie z ćwiczenia czwartego

Algorytmy Geometryczne

Iwo Szczepaniak, Windows 11 – 4,2GHz

Jupyter Notebook, VS Code Python 3.9.12

1) Wstęp

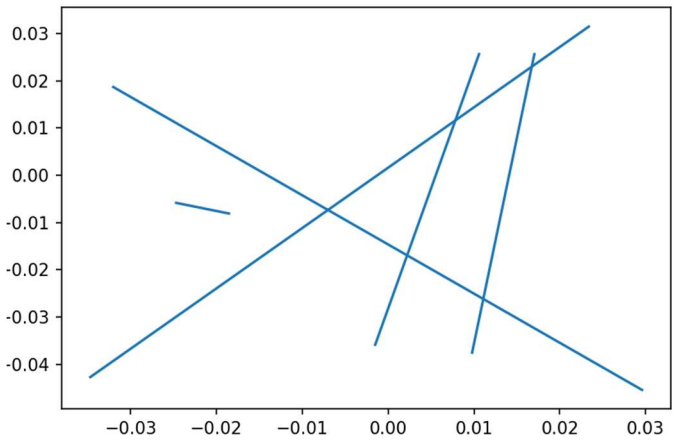
Zadaniem które należy wykonać na laboratorium 4 jest implementacja oraz przetestowanie czterech algorytmów:

- algorytm umożliwiający zadawanie linii przy użyciu myszki, ich zapis oraz odczyt
- algorytm umożliwiający generowanie losowe linii
- algorytm zamiatania sprawdzający czy dowolne dwie pary odcinków przecinają się
- algorytm zamiatania szukający wszystkich przecięć

2) Opis algorytmów

2.1 Rysowanie linii przy użyciu myszki, ich zapis oraz odczyt:

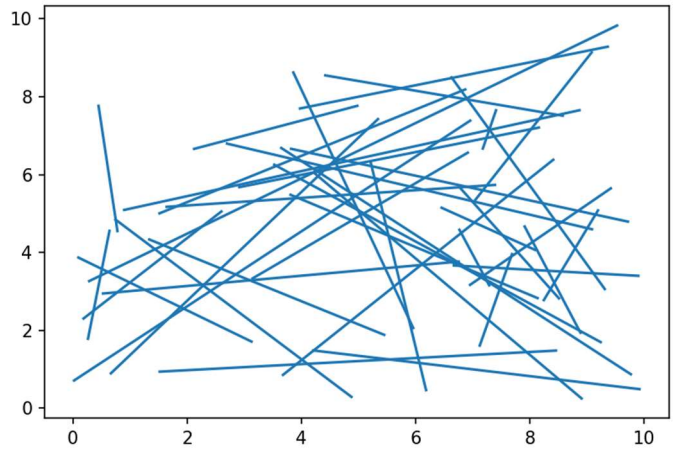
Rysowanie możliwe jest wewnątrz pustego plot'u w nagłówku „Narysuj linie". Następnie, by zapisać figurę należy przejść do nagłówka „Zapisz narysowane linie do pliku". W nim używana jest funkcja plot.get_added_elements(), która wczytuje narysowane linie zapisuje pod nazwą wskazaną przez użytkownika w folderze projektu. Odczytanie wybranego zbioru punktów możliwe jest w nagłówku „Zobacz wybrany zbiór linii". Użytkownik musi podać nazwę pliku, w którym zapisany jest interesujący go zbiór.



Rys 1.

2.2 Generowanie losowe punktów:

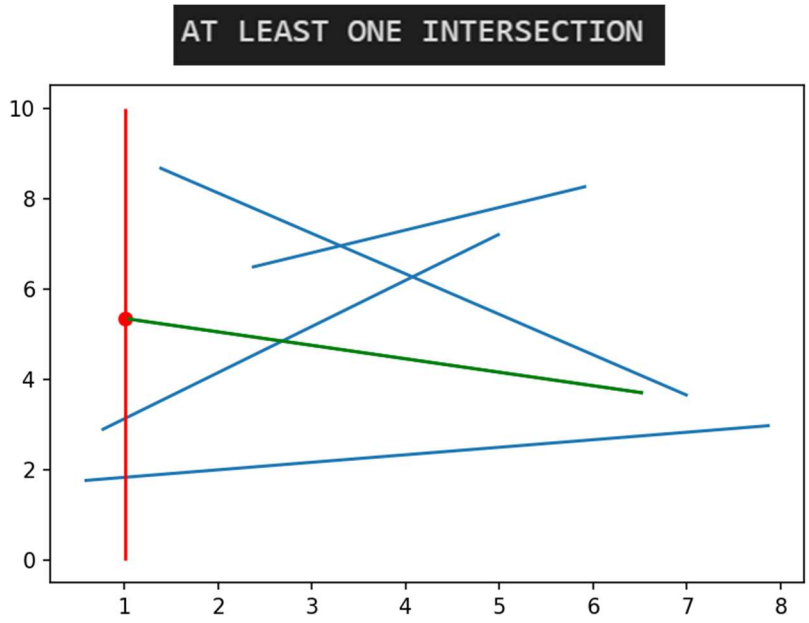
Generowanie punktów odbywa się zgodnie z założeniami zadania(żadne dwa punkty nie spotykają się w jednym miejscu). Funkcja draw_lines umożliwia zobaczenie wygenerowanych punktów.



Rys 2.

2.3 Badanie istnienia choć jednego przecięcia:

Pierwszym istotnym algorytmem do zaimplementowania było szukanie przecięcia w zbiorze punktów. Do przechowania struktury stanu wykorzystano w tym przypadku strukturę SortedSet(sortowanie po współrzędnej y), a do przechowania wydarzeń tablicę(listę), którą kopcujemy, co daje nam na raz możliwość wyciągania elementów tak jak z tablicy oraz tak jak z PriorityQueue. SortedSet reprezentowane jest przez drzewo czerwono-czarne, więc dostęp do wstawiania, usuwania i szukania następnika oraz poprzednika jest O(logn). Algorytm szuka punktu przecięcia w oparciu o algorytm zamiatania, działa on dopóki nie zostanie znalezione przecięcie. Jeśli zdarzenie jest początkiem odcinka, to dodaję do struktury stanu nowy odcinek i sprawdzam czy istnieje przecięcie. Jeżeli zdarzenie jest końcem odcinka, to sprawdzam przecięcie i usuwam linię. W momencie znalezienia przecięcia algorytm zwraca True i przestaje działać, a program wypisuje, że istnieje co najmniej jedno przecięcie.



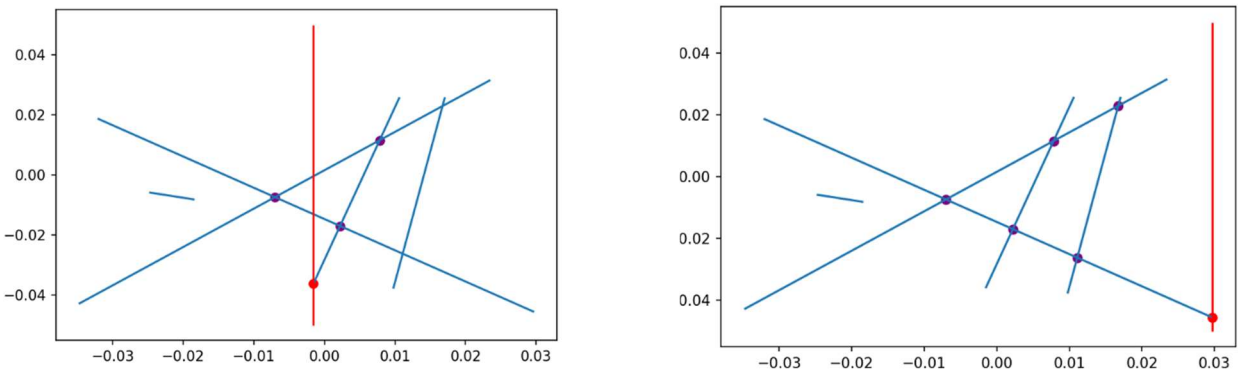
Rys 3.

2.4 Znajdywanie wszystkich przecięć:

Algorytm ten opiera się na podobnej procedurze co poprzedni, jednak nie kończy się na pierwszym przecięciu, ale traktuje każde spotkanie dwóch linii jako nowe zdarzenie. Algorytm ten różni się też reprezentacją danych od poprzednika - tym razem struktura zdarzeń przechowywana jest jako SortedSet tak samo jak struktura stanu. Umożliwia to dodawanie zdarzeń i ich porządkowanie. Dodatkowo osobno przechowywane są punkty przecięcia(crossings). Zarówno przecięcia jak i zwykłe zdarzenia posiadają słowniki które umożliwiają wyciągnięcie linii związanych z danym punktem. Przejście przez przecięcie w zbiorze zdarzeń powinno powodować zamianę kolejności sąsiadów, co jest realizowane za pomocą funkcji order_by(która dla danego x szereguje linie po y) - przy przecięciu współrzędna x zwiększana jest o 10^{-12}. Dzięki temu zabiegowi następuje zamiana sąsiadów w punkcie przecięcia.
Funkcja wypisuje ilość przecięć, ich współrzędne oraz wyświetla wizualizację. W trakcie działania program posługuje się indeksami linii do określania sąsiedztwa.

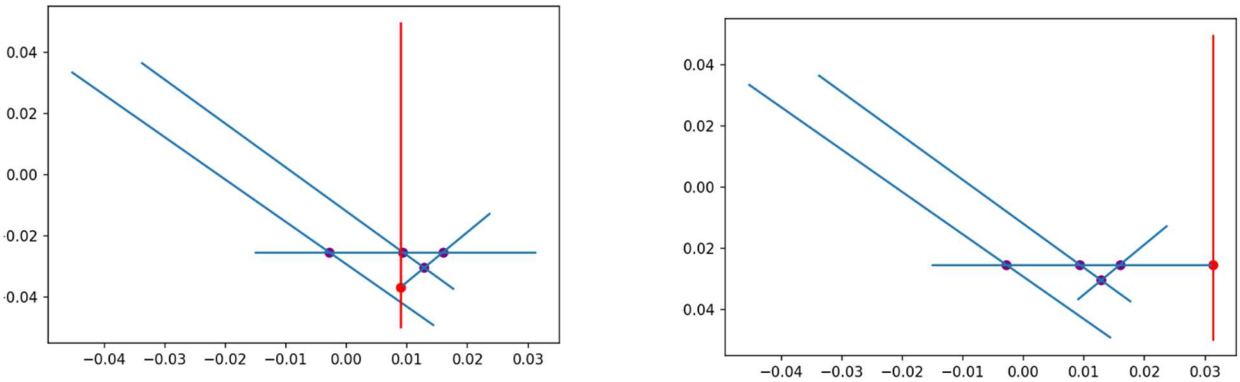
Zbiory punktów zadane na laboratoriach:

```
Ilość przecięć to 5. Przecięcia to:
[(-0.00700899567758, -0.00735671631758), (0.00221863131308, -0.01695626291461), (0.00783551353677, 0.01159872478947),
(0.01109361219668, -0.02618894999941), (0.01674221042352, 0.02297197841861)]
```



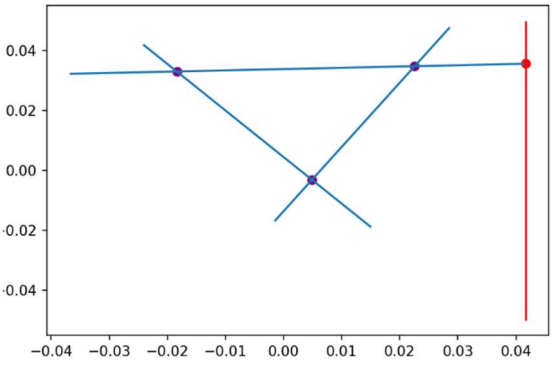
Rys 4. wizualizacja wewnątrz i na końcu działania programu

```
Ilość przecięć to 4. Przecięcia to:
[(-0.00281416847236, -0.02533892566083), (0.00933895898634, -0.02533892566083), (0.01285287332171, -0.03038077026388),
(0.01594032547379, -0.02533892566083)]
```

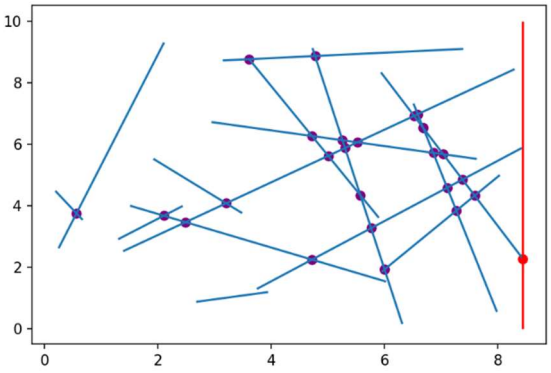


Rys 5. wizualizacja wewnątrz i na końcu działania programu

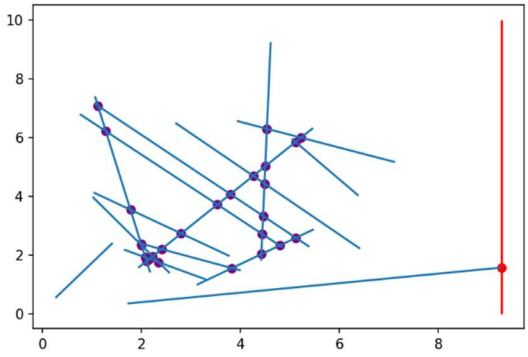
Przykładowe inne zbiory punktów:



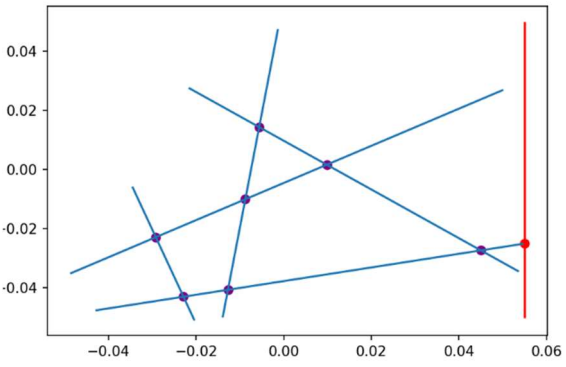
Rys 6. rysowane linie



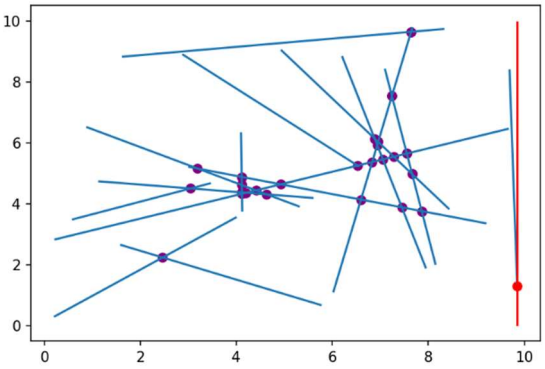
Rys 7. losowo generowane linie



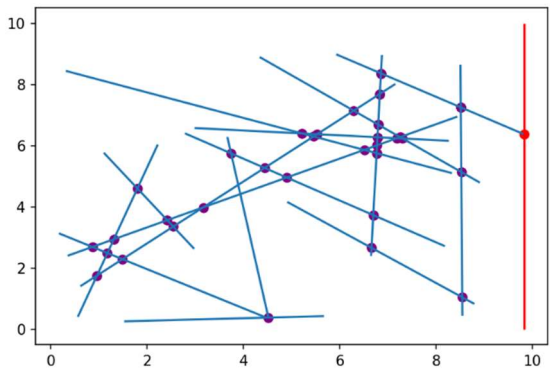
Rys 8. losowo generowane linie



Rys 9. rysowane linie



Rys 10. losowo generowane linie



Rys 11. losowo generowane linie