Laboratorium 4 - Problem przecinania się odcinków

Mateusz Podmokły - II rok Informatyka WI

29 listopad 2023

1 Specyfikacja użytego środowiska

Specyfikacja:

- Środowisko: Jupyter Notebook,
- Język programowania: Python,
- System operacyjny: Microsoft Windows 11,
- Architektura systemu: x64.

2 Przebieg ćwiczenia

Ćwiczenie polegało na zaimplementowaniu algorytmu znajdowania wszystkich przecięć odcinków na płaszczyźnie. Napisany został także algorytm generujący losowe odcinki na płaszczyźnie. Dodatkowo wykorzystane zostało napisane wcześniej narzędzie pozwalające zadawać wielokąty przy użyciu myszki.

2.1 Implementacja struktury zdarzeń

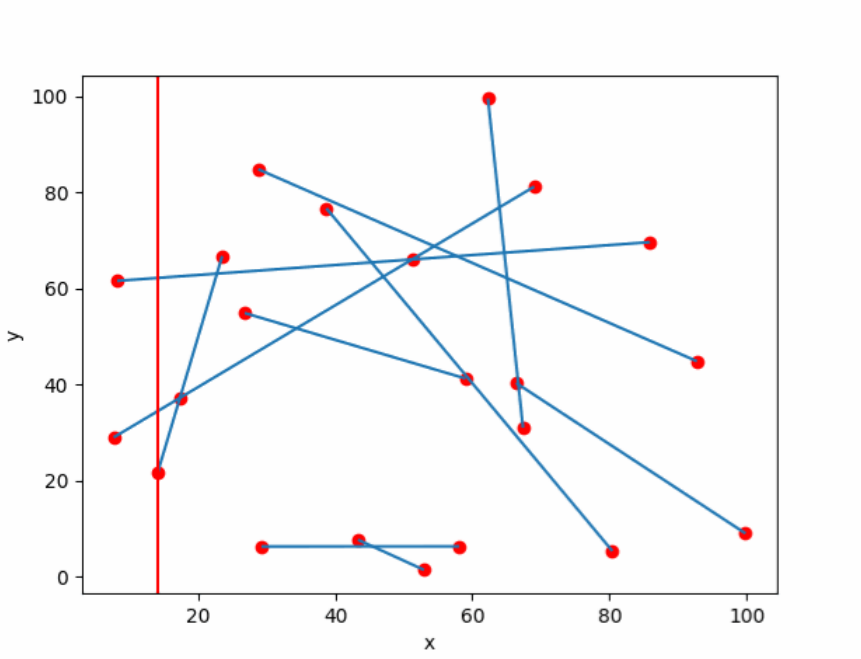
Struktura zdarzeń przechowuje punkty w których zatrzymuje się miotła w celu badania przecięć aktywnych odcinków. Punkty te to początek i koniec każdego odcinka oraz punkty przecięć. Są one posortowane według współrzędnej x. Jako struktura zdarzeń użyta została kolejka priorytetowa, która pozwala szyko wkładać i wyciągać elementy zachowując określony porządek.

2.2 Implementacja struktury stanu

Struktura stanu przechowuje aktywne odcinki, czyli takie, które w danym momencie przecina miotła, posortowane według współrzędnej y. Wykorzystana została struktura SortedList z biblioteki sortedcontainters, która pozwala na aktualizowanie listy w czasie $O(logn)$ oraz na dostęp do sąsiadów danego elementu.

2.3 Implementacja miotły

Miotła wyciąga po kolei punkty ze struktury zdarzeń i przechodzi po nich. Badając przecięcia odcinków, które sąsiadują ze sobą w strukturze stanu. Jeżeli wykrywa nowe przecięcie to dodaje je do struktury zdarzeń.



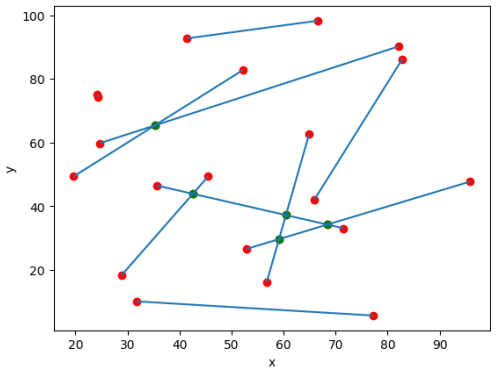
Rysunek 1: Miotła w trakcie przejścia przez odcinki.

2.4 Znajdowanie przecięć odcinków

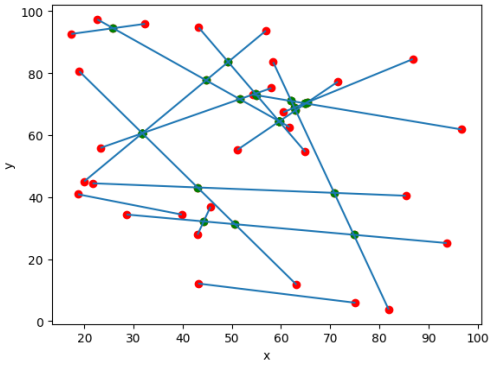
Jeżeli miotła napotyka punkt, który jest przecięciem dwóch odcinków, zapisuje go do listy przecięć, która jest zwracana na końcu programu.

3 Analiza znalezionych przecięć odcinków

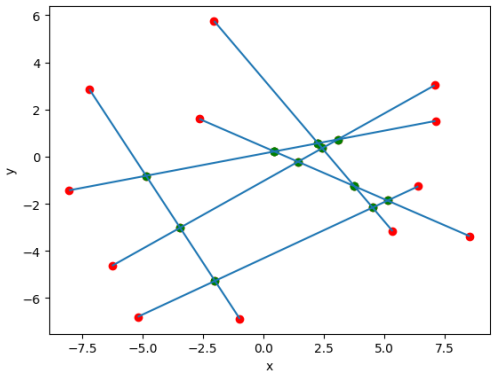
3.1 Odcinki użyte do testów



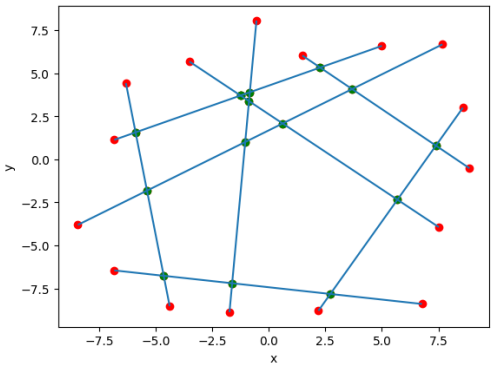
Rysunek 2



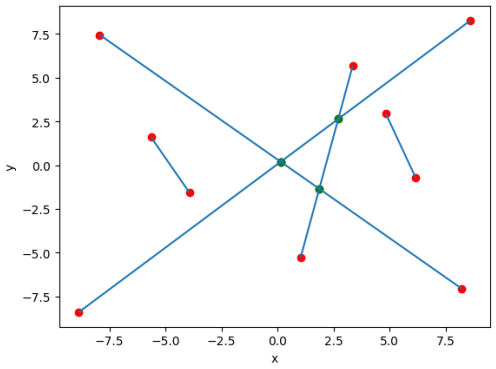
Rysunek 3



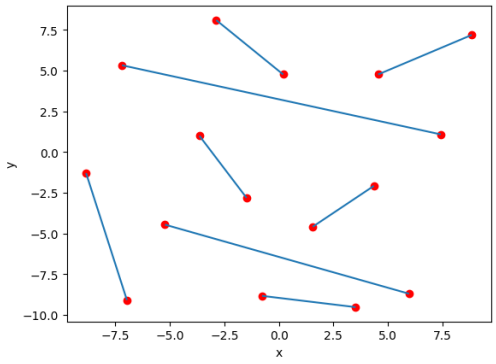
Rysunek 4



Rysunek 5



Rysunek 6



Rysunek 7

3.2 Analiza wyników

Rysunek 3 i rysynek 4 zawierają wygenerowane losowo odcinki, rysunek 4, rysunek 5, rysunek 6 oraz rysunek 7 zostały wygenerowane ręcznie. Rysunek 7 nie zawiera żadnego przecięcia. Rysunek 4 i rysunek 5 zawierają wielokrotne przecięcia tych samych odcinków. Na rysunku 6 charakterystyczne są dwa krótkie odcinki, które nie przecinają się z żadnym innym, ale powodują pownowne sprawdzenie pozostałych odcinków. We wszystkich przypadkach algorytm działa poprawnie.

4 Wnioski

Algorytm dzięki swojej szybkości oraz uniwersalności jest dobrym narzędziem do znajdowania przecięć odcinków na płaszczyźnie. Jego złożoność obliczeniowa wynosi $O(nlogn)$, gdzie n to liczba odcinków. Należy pamiętać, że uzyskanie takiej złożoności obliczeniowej wymaga użycia odpowiednich struktur danych pozwalających na szybkie aktualizacje i odczytywanie struktury zdarzeń i struktury stanu.