



ALGORYTMY GEOMETRYCZNE

Laboratorium 4

Przecinanie się odcinków

Kyrylo Iakymenko

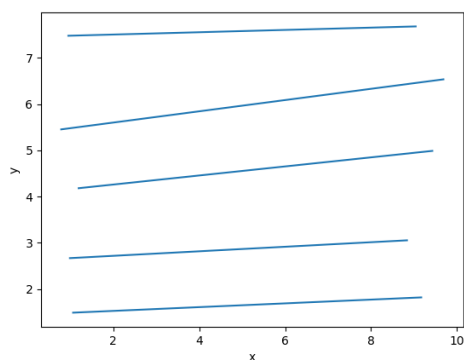
1 Wprowadzenie

Zadaniem analizy przecinania się odcinków jest rozwiązanie fundamentalnego problemu w geometrii komputerowej, mającego szerokie zastosowanie w różnych dziedzinach, takich jak grafika komputerowa, analiza obrazów, planowanie tras czy symulacje fizyczne. Problem ten skupia się na identyfikacji punktów przecięcia się dwóch odcinków na płaszczyźnie, co stanowi kluczowy element w wielu algorytmach i systemach przetwarzania danych geometrycznych.

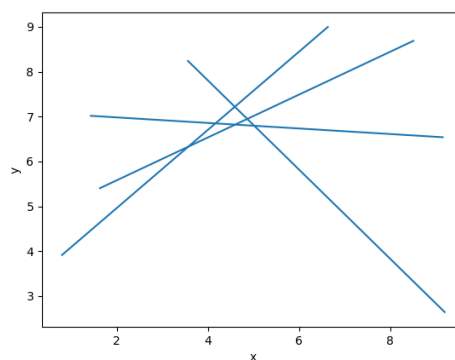
W dalszej części raportu omówimy algorytm zmiatania służący rozwiązaniu problemu przecinania się odcinków oraz przeanalizujemy złożoność czasową i przestrzenną algorytmu.

2 Zbiory testowe

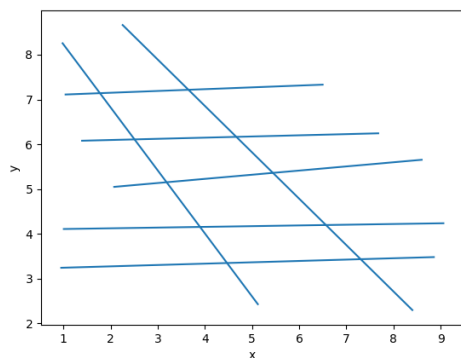
Na potrzeby ćwiczenia stworzyliśmy 6 zbiorów odcinków na płaszczyźnie. Zostały wybrane odpowiednio, żeby przetestować działanie algorytmu w przypadkach zarówno losowych jak i ekstremalnych.



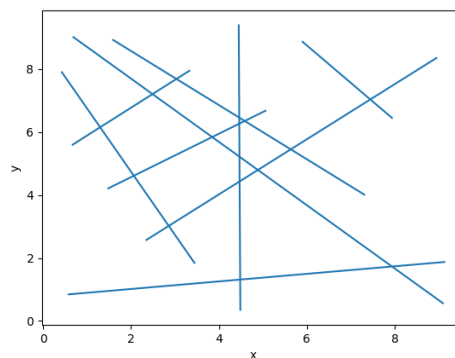
Rys. 1



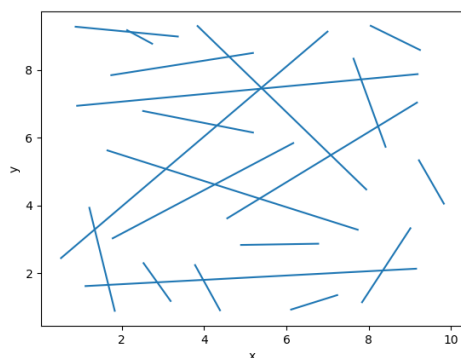
Rys. 2



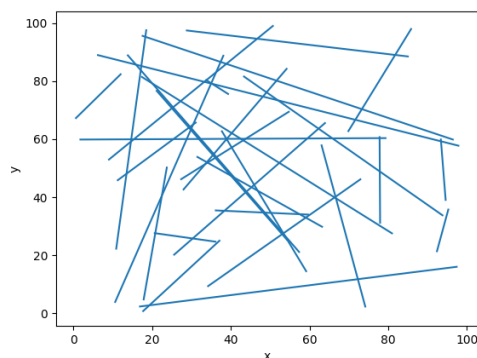
Rys. 3



Rys. 4



Rys. 5



Rys. 6

3 Algorytm zmiatania

Algorytm zmiatania (ang. sweep line algorithm) to efektywna metoda rozwiązania problemu wyznaczania przecięcia się odcinków na płaszczyźnie. Algorytm ten można opisać w kilku krokach:

1. **Tworzenie listy zdarzeń:** Każde zdarzenie jest punktem. Zdarzenia przechowujemy w zbiorze posortowanym (Sorted Set), które są posortowane ze względu na swoją współrzędną x .
2. **Pętla główna:** Po kolei wyciągamy zdarzenia z listy zdarzeń i obsługujemy je za pomocą funkcji `handle_event`.
3. **Pętla wewnętrzna (`handle_event`):** Najpierw sprawdzamy czy zdarzenie jest zdarzeniem początku lub końca odcinka (odpowiednią informację przechowujemy w słowniku, żeby zapewnić szybki dostęp).

Jeżeli nasze zdarzenie jest zdarzeniem początku odcinka - dodajemy odcinek do zbioru posortowanego z odcinkami, które przecina nasza miotła i sprawdzamy czy nie przecina swoich sąsiadów. Jeżeli wykrywamy przecięcie - dodajemy go do zbioru posortowanego wydarzeń.

Jeżeli zdarzenie jest zdarzeniem końca odcinka, usuwamy go z listy odcinków przecinających naszą miotłę.

Jeżeli zdarzenie jest zdarzeniem przecięcia się odcinków - sprawdzamy najpierw czy nie dodaliśmy już przecięcia tej pary odcinków (używamy do tego celu słownika, w którym przechowujemy pary prostych, przecięcia których już zostały zarejestrowane). A pod koniec aktualizujemy (sortujemy) zbiór prostych przecinających miotłę, ze względu na ich współrzędną y po tym przecięciu.

4. **Wynik:** Zwracamy listę przecięć i indeksów odcinków.

3.1 Struktury zdarzeń i stanu miotły

Zarówno struktura zdarzeń, jak i struktura stanu miotły była zaimplementowana za pomocą zbioru posortowanego (Sorted Set) w pythonie. W przypadku miotły przechowujemy w zbiorze odcinki (objekty klasy Line), posortowane ze względu na współrzędną y w punkcie x , który był punktem ostatniego zdarzenia. Zbiór zdarzeń sortujemy po współrzędnej x każdego zdarzenia (zdarzenia w naszym przypadku, to po prostu punkty).

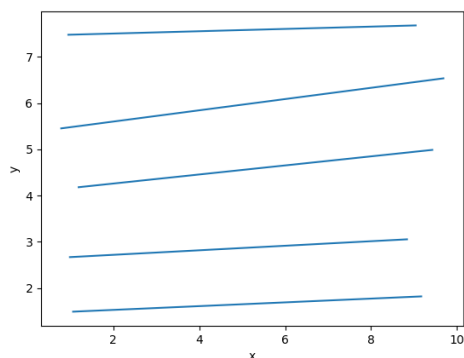
3.2 Wykrywanie przecięć wielokrotnych

Żeby zapobiec sytuacji, w której zapisujemy jedno przecięcie wiele razy, używamy słownika, do którego zapisujemy pary prostych, przecięcia których już zostały zarejestrowane. Przy kolejnych przecięciach sprawdzamy czy dana para prostych nie wystąpiła już wcześniej. Słownik w pythonie pozwala na szybkie sprawdzenie czy dany klucz już był zapisany do tego słownika, co pozwala nie tracić na tej operacji złożoności.

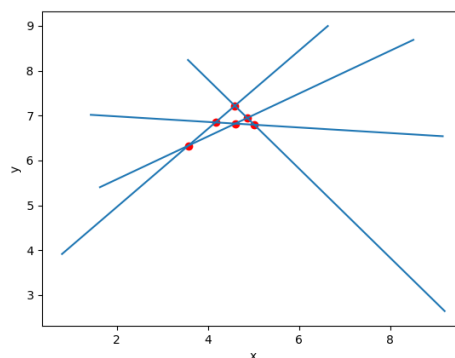
3.3 Wykrywanie przecięć wielokrotnych w algorytmie, który sprawdza czy w zbiorze prostych występują przecięcia

Nie musimy wykonywać podobnych operacji w przypadku tego algorytmu, gdyż wykrycie jednego przecięcia już powoduje zatrzymanie programu.

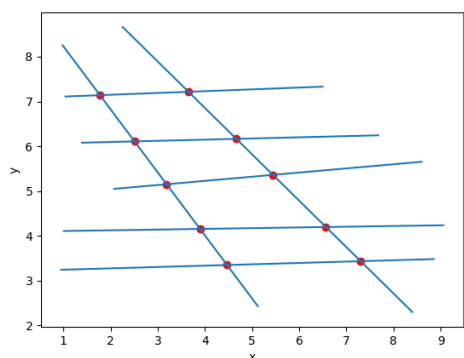
4 Wyniki algorytmu



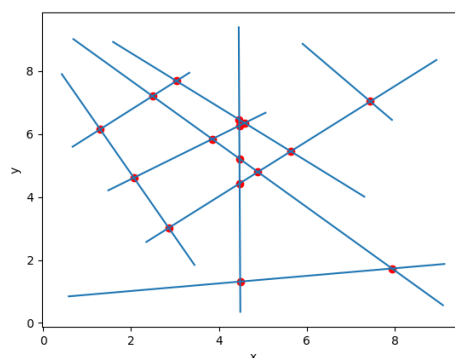
Rys. 7



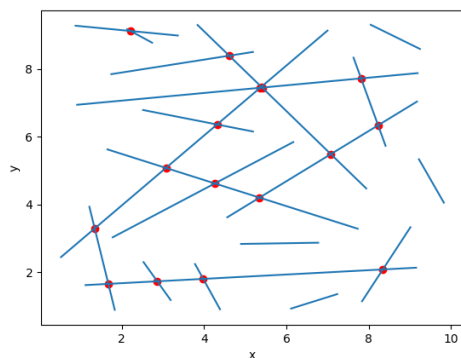
Rys. 8



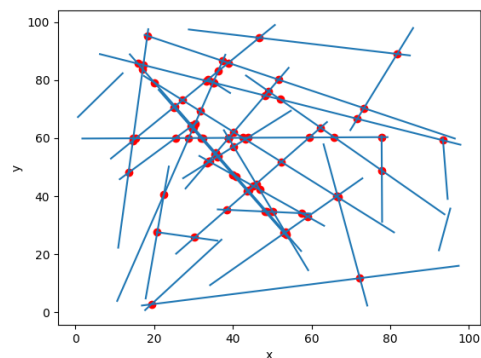
Rys. 9



Rys. 10



Rys. 11



Rys. 12

5 Podsumowanie

Podczas implementacji algorytmu zmiatania przetestowano go na różnych zestawach danych, a uzyskane wyniki i wnioski zostały szczegółowo przedstawione w sprawozdaniu. Wnioski te obejmują skuteczność algorytmu w różnych scenariuszach. Największym wyzwaniem była poprawna obsługa wykrytych przecięć.

W wyniku nasz algorytm poprawnie wykrywa przecięcia odcinków, nie zapisuje duplikatów przecięć oraz ma złożoność $O(n \log n)$.

Warto zaznaczyć, że nasz algorytm może być modyfikowany poprzez użycie innych struktur zdarzeń i miotły i sam algorytm zmiatania jest tylko wzorcem, na podstawie, którego można zaimplementować wiele różnych, ale dających ten sam wynik algorytmów.