Algorytmy geometryczne

Sprawozdanie z laboratorium 4

Przecinanie się odcinków

Iwo Zowada

Gr. 6 Wtorek 15:00-16:30 A

Dane techniczne

Komputer z systemem Windows 10
Procesor Intel Core i7-3770 CPU 3.40GHz
Pamięć RAM 24GB
Program napisany w języku Python w środowisku Jupyter Notebook z wykorzystaniem bibliotek numpy, matplotlib oraz narzędzia przygotowanego przez KN Bit

1. Opis ćwiczenia

Celem ćwiczenia jest poznanie algorytmu zamiatania, który pozwala na wykrywanie punktów przecięcia odcinków na płaszczyźnie. Zadanie obejmuje implementację algorytmu sprawdzającego, czy jakiekolwiek dwa odcinki się przecinają, oraz wyznaczenie wszystkich punktów przecięcia między odcinkami.

W ramach tego laboratorium należało wykonać następujące zadania:

1. Dostosować aplikację graficzną tak, aby można było zadawać odcinki na płaszczyźnie przy użyciu myszki, z dodatkowym zapisem i odczytem podanych prostych. Odcinki poziome powinny być eliminowane, a także żaden odcinek nie powinien mieć końca w punkcie o tej samej współrzędnej x co inny odcinek.
2. Zaimplementować algorytm zamiatania sprawdzający czy jakakolwiek para odcinków się przecina, wizualizacja tego algorytmu oraz opis struktur zdarzeń użytych do realizacji zadania.
3. Zmodyfikować algorytm w taki sposób aby wyznaczał punkty przecięcia odcinków - ile ich jest, jakie mają współrzędne oraz odcinki które przecinając się tworzą to przecięcie.

2. Opis algorytmu zamiatania

Do realizacji tego algorytmu potrzebne nam będą dwie struktury zdarzeń Q i T:

Zbiór zdarzeń (Q):

- Zawiera listę punktów zdarzeń składającą się z punktów początkowych i końcowych wszystkich odcinków.
- Każdy punkt zdarzenia zawiera informacje o:
  - Typie punktu: początek (lewy) lub koniec (prawy) odcinka.
  - Współrzędne punktu oraz identyfikator odcinka, do którego należy.

Zbiór aktywnych odcinków (T):

- Utrzymuje dynamiczną listę odcinków, które aktualnie przecina pionową prostą zamiatającą.

Struktury te zaimplementowałem jako **SortedSet** z pakietu sortedcontainers przez co nie było już potrzeby sortowania punktów po współrzędnych oraz struktura umożliwia dodawanie i usuwanie odcinków w czasie logarytmicznym.

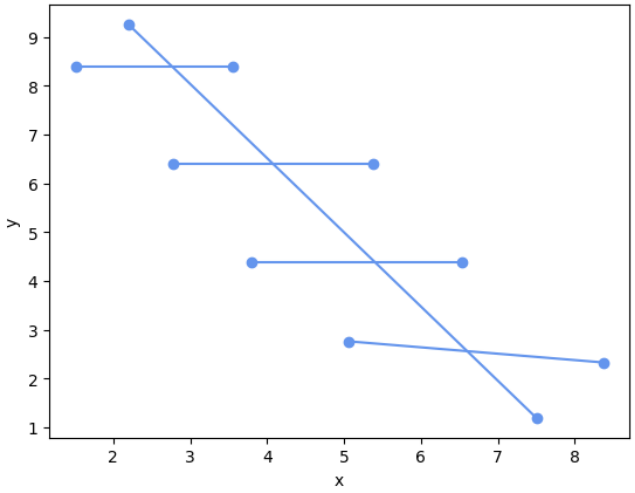
Przegląd punktów

- Przesuwaj pionową prostą zamiatającą od lewej do prawej, odwiedzając punkty zdarzeń w kolejności ze zbioru Q.
- Dla każdego zdarzenia wykonaj odpowiednią operację w T:
  - **Punkt początkowy odcinka:**
    - Dodaj odcinek do zbioru T.
    - Sprawdź potencjalne przecięcia odcinka z jego sąsiadami w T.
  - **Punkt końcowy odcinka:**
    - Usuń odcinek ze zbioru T.
    - Jeśli usunięty odcinek miał dwóch sąsiadów w T, sprawdź, czy te dwa odcinki się przecinają.
  - **Punkt przecięcia wykryty we wcześniejszych krokach algorytmu:**
    - Dla odcinków, które tworzą przecięcie sprawdzamy jeszcze raz obydwóch sąsiadów.

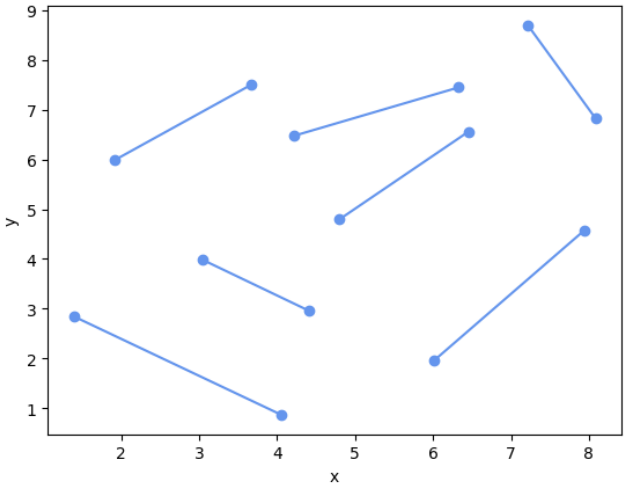
Dzięki zastosowaniu struktur Q i T, algorytm zamiatania działa w czasie $O((n+k)log\ n)$ gdzie n to liczba odcinków, a k to liczba punktów przecięcia.

3. Realizacja ćwiczenia

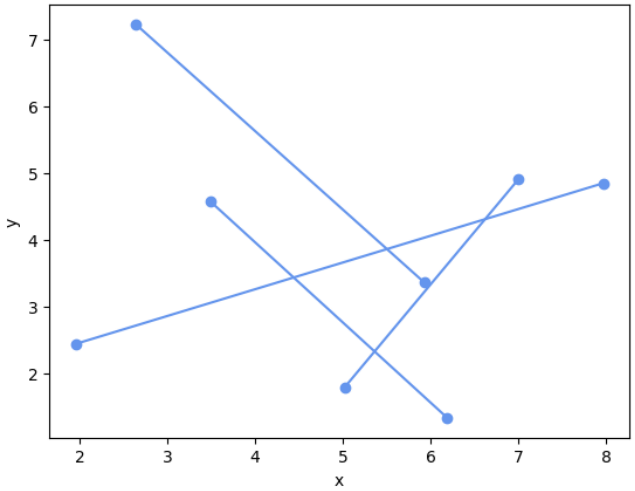
Poniżej prezentuje zbiory odcinków na których testuje mój program:



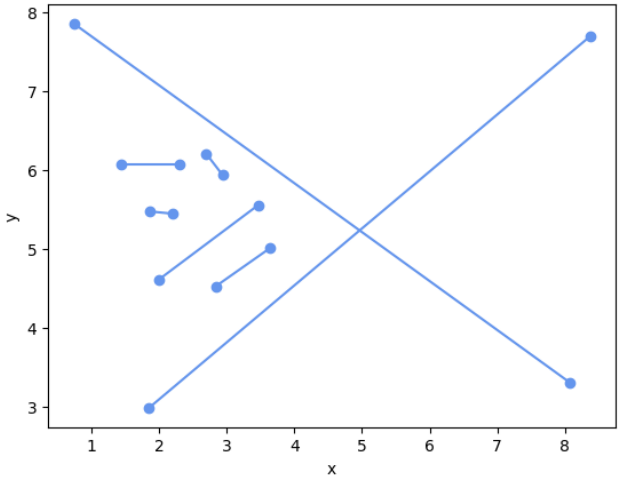
(Rys 3.1): *Zbiór odcinków A*



(Rys 3.2): *Zbiór odcinków B*



(Rys 3.3): *Zbiór odcinków C*



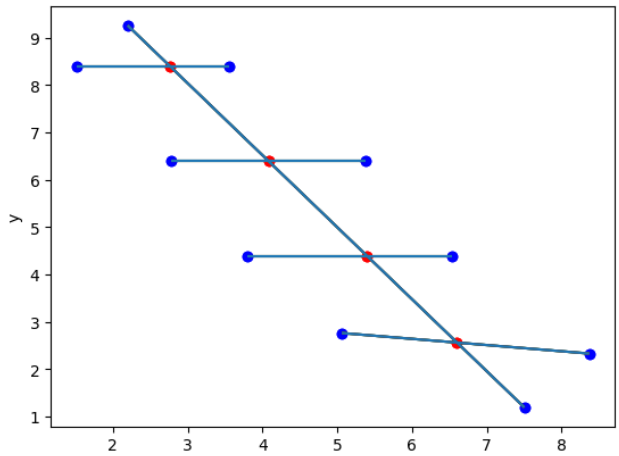
(Rys 3.4): *Zbiór odcinków D*

W poniższej tabeli przedstawiam czy dla danego zbioru odcinków występuje jakiekolwiek przecięcie:

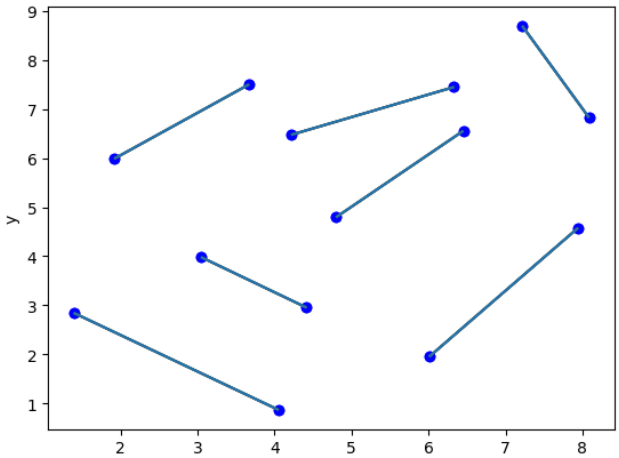
	Odcinki A	Odcinki B	Odcinki C	Odcinki D
Czy występuje przecięcie	Tak	Nie	Tak	Nie

(Tabela 3.1): *Czy występuje przecięcie*

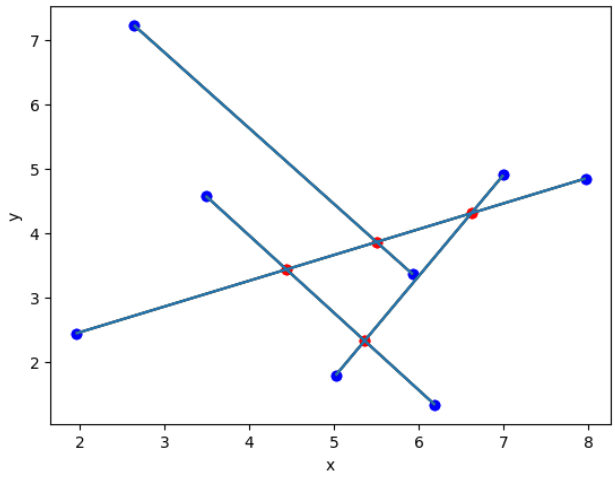
Poniżej prezentuje rysunki z zaznaczonymi punktami, które odpowiadają przecięciom odcinków (kolorem czerwonym są zaznaczone punkty przecięcia):



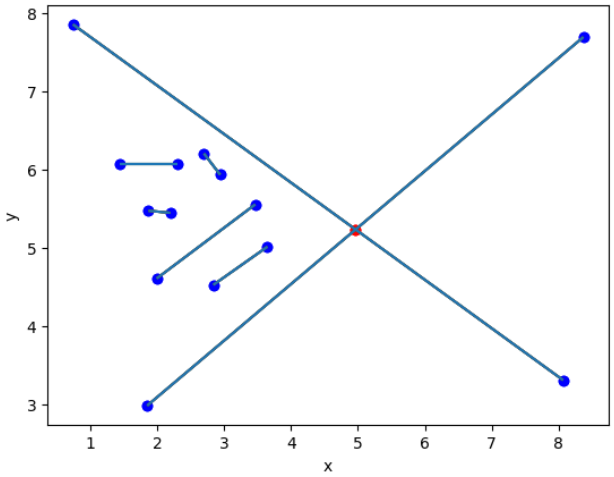
(Rys 3.5): *Punkty przecięcia odcinków A*



(Rys 3.6): *Punkty przecięcia odcinków B*



(Rys 3.7): *Punkty przecięcia odcinków C*



(Rys 3.8): *Punkty przecięcia odcinków D*

Przypadek, gdzie algorytm kilka razy wykrywa to samo przecięcie odcinków jest przedstawiony jako zbiór D. Jak widać na załączonych rysunkach, algorytm obsługuje tego typu przypadki i działa poprawnie.

Podsumowanie i wnioski

Zbiory odcinków, które poddano testom, miały zróżnicowaną specyfikę, co pozwoliło ocenić działanie programu w różnych warunkach. Program radził sobie dobrze zarówno z zestawami testowymi, jak i z zestawami przygotowanymi przez Koło Naukowe AGH Bit, co potwierdzają generowane przez niego rysunki i animacje. Algorytm zastosowany w programie bazował na metodzie zamiatania, omawianej podczas wykładu.