17.12.2023 r.

Mateusz Król

Algorytmy geometryczne

Przecinanie się odcinków

Ćwiczenie nr 4

Spis treści

1. Realizacja ćwiczenia

W ramach czwartego ćwiczenia przygotowałem procedurę umożliwiającą interaktywne wprowadzanie odcinków wykorzystując bibliotekę **open-cv** oraz funkcję generującą zadaną liczbę odcinków z podanego zakresu współrzędnych.

Zaimplementowałem algorytm sprawdzający czy dwa dowolne odcinki na płaszczyźnie się przecinają oraz algorytm obliczający wszystkie zaistniałe przecięcia.

Przy implementacji wykorzystałem klasę **SortedSet** z biblioteki **sortedcontainers** (*https://grantjenks.com/docs/sortedcontainers/introduction.html*).

Umożliwiłem również wizualizację algorytmu znajdywania przecięć odcinków oraz przetestowałem algorytm na różnych zestawach danych. Sporządziłem też porównanie czasowe algorytmu z wykorzystaniem **SortedSet**'a, do algorytmu z wykorzystaniem zwykłem tablicy jako struktury stanu.

2. Implementacja struktur zdarzeń i stanu

- **Struktura zdarzeń**

Strukturą zdarzeń w moim algorytmie jest **SortedSet**, czyli drzewo binarne umożliwiające dodawanie, usuwanie, wyciąganie elementów struktury, będących krotkami trójelementowymi symbolizującymi końce odcinków lub punkty przecięć, w czasie logarytmicznym zachowując przy tym pożądany porządek wyznaczony przez parametr **key** jako wartość współrzędnej **x** tychże punktów.

W strukturze **Q** pierwszy element krotki symbolizuje typ punktu, a drugi i trzeci w przypadku, gdy jest to koniec odcinka, którego odcinka ten punkt dotyczy, a w przypadku punktu przecięcia odcinków, są to indeksy dwóch przecinających się odcinków.

Wszystkie operacje, oprócz porównywania wartości współrzędnej **x** wykonywane są na indeksach, co uniemożliwia dodanie to kolejki dwukrotnie tego samego wierzchołka mimo tego, że wierzchołki wyznaczające przecięcia są wielokrotnie wykrywane (**!**), więc mój program uwzględnia taki przypadek.

Wartości współrzędnych **x**-owych są zapisane w słowniku **intersections** i odczytywane dla każdego porównania w czasie O(1).

- **Struktura stanu**

Strukturą stanu w moim algorytmie jest również **SortedSet**, który pozwala na wykonywanie potrzebnych operacji w czasie logarytmicznym.

Elementami struktury są elementy zaimplementowanej przeze mnie klasy **Segment**, która ma atrybut **index** oznaczający odcinek, do którego dana instancja **Segment** się odnosi oraz posiada nadpisane metody: **__eq__**, **__gt__**, **__hash__**, itd. w celu implementacji modyfikowalnego sposobu porównywania elementów struktury.

Elementy są porządkowane na podstawie punktu wartości współrzędnej **y** przecięcia prostej wyznaczone przez odcinek z prostą $x=x_0$, a w drugiej kolejności po współczynniku kierunkowym tej prostej.

3. Opis sposobu obsługi zdarzeń w zaimplementowanym algorytmie

Zdarzenie początku odcinka:

- wyznaczany i zmieniany jest obecnie rozważany punkt x_0
- do struktury stanu dodawany jest element typu **Segment** odpowiadający rozważanemu odcinkowi
- znajdowany jest element wstawiony i jego potencjalni nowi sąsiedzi
- sprawdzenie czy element wstawiony przecina się z nowymi sąsiadami i uaktualnienie struktury zdarzeń

Zdarzenie końca odcinka:

- wyznaczany i zmieniany jest obecnie rozważany punkt x_0
- znajdowany jest element do usunięcia
- sprawdzenie czy nowi potencjalni sąsiedzi po usunięciu odcinka się przecinają
- uaktualnienie struktury zdarzeń
- usunięcie rozważanego odcinka z struktury stanu

Zdarzenie przecięcia odcinków:

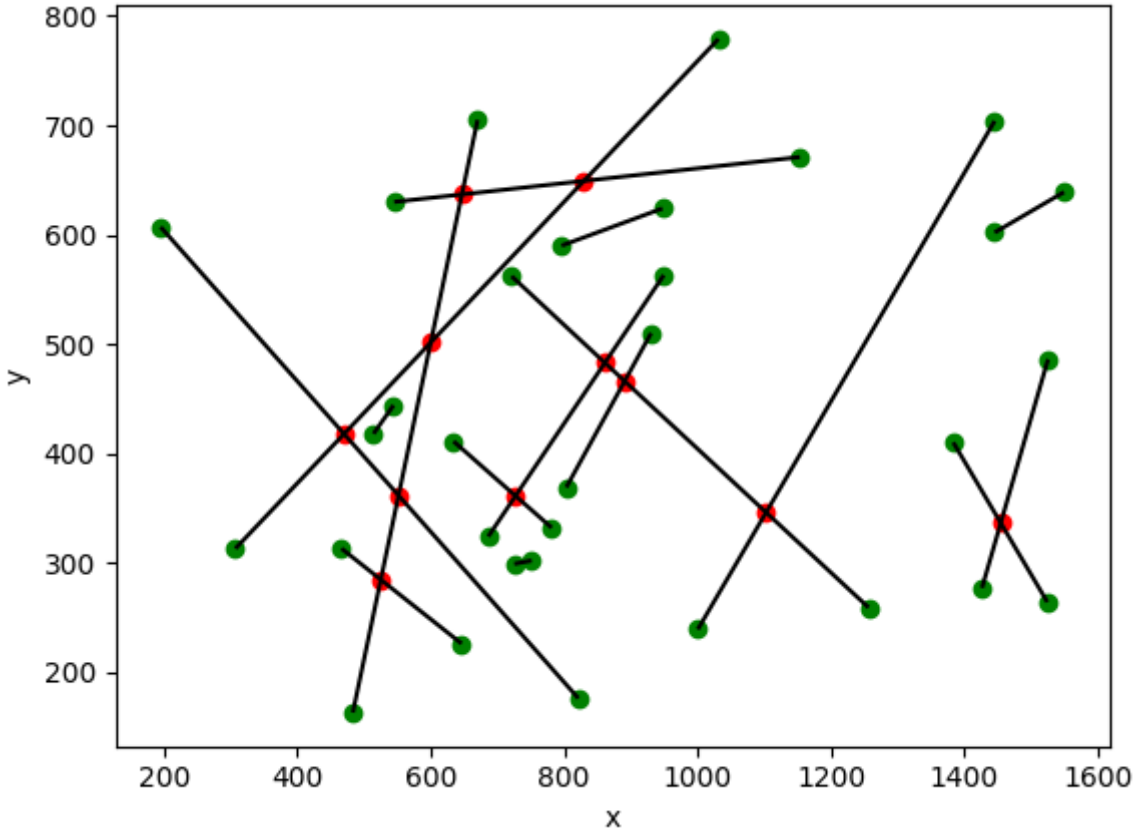
- znajdowane są element do usunięcia
- usunięcie rozważanych odcinków z struktury stanu
- wyznaczany i zmieniany jest obecnie rozważany punkt x_0
- do struktury stanu dodawane są elementy typu **Segment** odpowiadające rozważanym, wcześniej usuniętym odcinkom
- sprawdzenie czy nowi potencjalni sąsiedzi wstawieniu odcinków się przecinają
- uaktualnienie struktury zdarzeń

4. Wizualizacja algorytmu

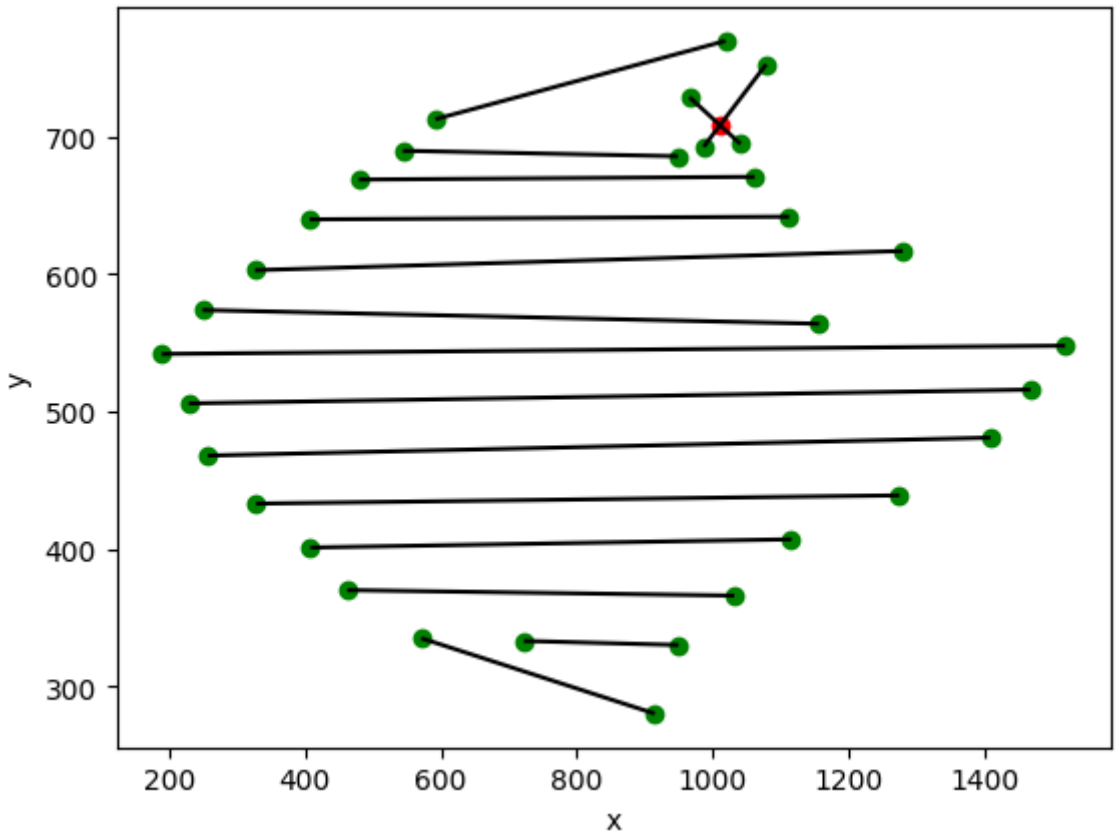
Funkcja **find_intersections_with_visualization** umożliwia wizualizację kolejnych kroków algorytmu lub wyświetlenie stanu końcowego – po zakończeniu działania algorytmu.

Wyniki dla przykładowych zbiorów danych:

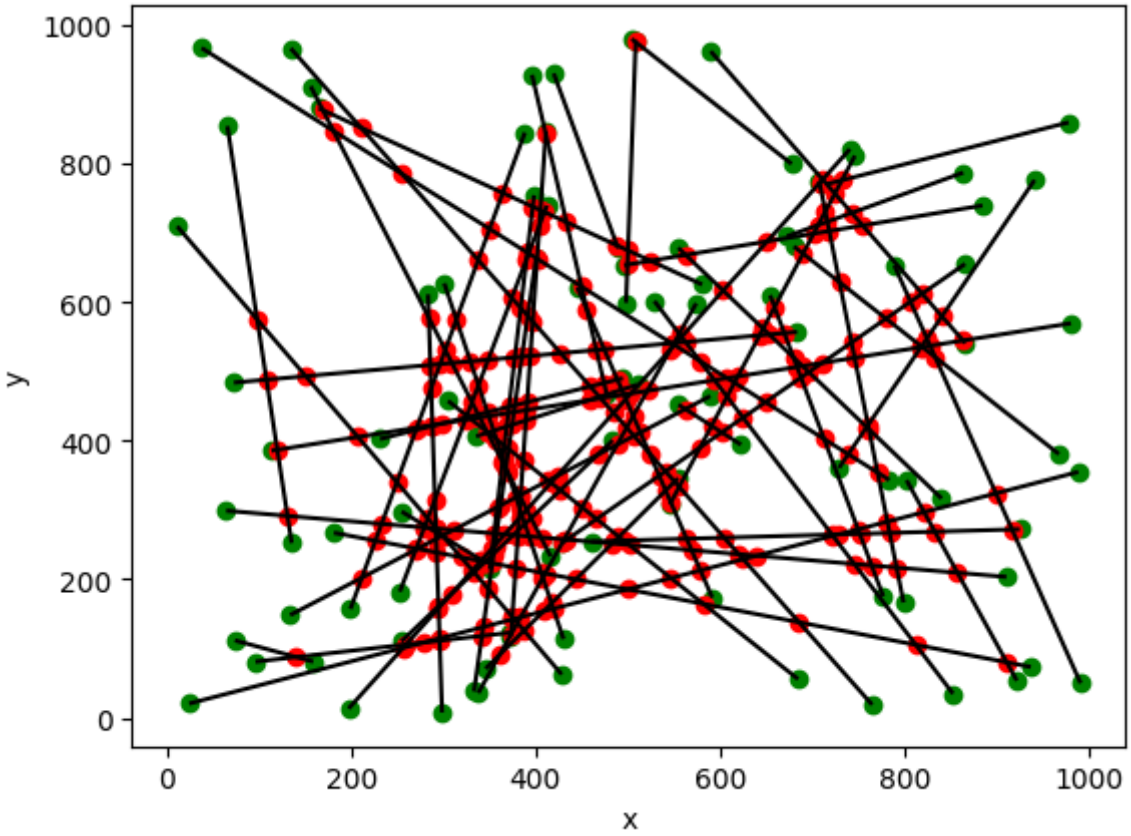
1) zwykły zbiór zadany ręcznie za pomocą funkcji **add_sections()**



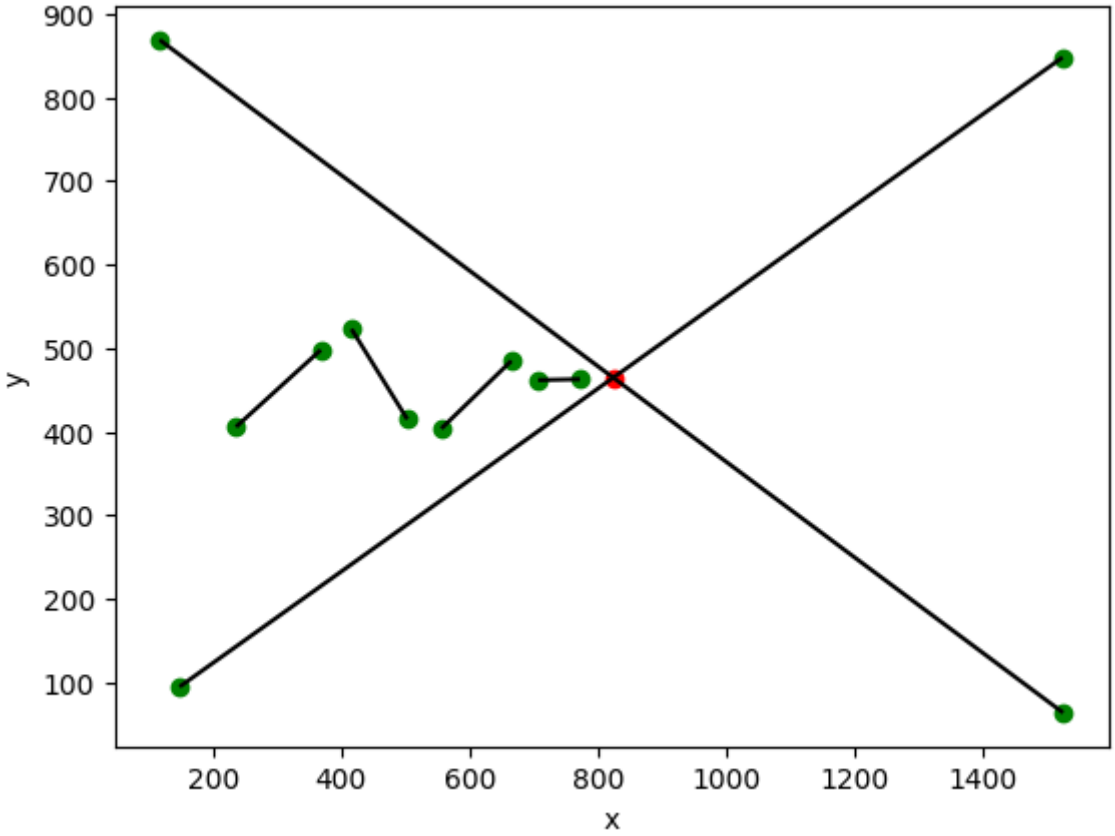
2) zbiór z wieloma odcinkami rozpatrywanymi równocześnie



3) zbiór wygenerowanych losowo 50 odcinków



4) zbiór odcinków, w którym ten sam punkt przecięcia będzie wykrywany wielokrotnie



5. Testy czasowe algorytmów

W poniższej tabeli znajduje się porównanie czasu działania algorytmów dla różnych implementacji struktury stanu: dla implementacji z wykorzystaniem **SortedSet**, oraz tej z wykorzystaniem zwykłej Python-owej listy. Zbiorami testowymi są generowane losowo odcinki na płaszczyźnie z zmienną wartością parametru **n** wyznaczających ich ilość oraz to czy wygenerowane odcinki się przecinają czy nie.

n	ilość przecięć	SortedSet	lista
100	959	0.0375 s	0.0334 s
100	0	0.0042 s	0.0098 s
500	27969	1.565 s	3.153 s
500	0	0.035 s	0.152 s
1000	113670	7.12 s	25.5 s
1000	0	0.0811 s	0.6148 s
1500	262457	17.4 s	91.4 s
1500	0	0.124 s	1.511 s
2000	458412	32.9 s	206.1 s
2000	0	0.172 s	2.524 s

Implementacja z wykorzystaniem klasy **SortedSet** jest szybsza dla każdego z testowanych zbiorów danych oprócz 100 odcinków wygenerowanych losowo. Im więcej rozważanych odcinków, tym jest większa różnica w pomiarach czasowych.

*Z*łożoność czasowa aproksymacyjna algorytmu z wykorzystaniem **SortedSet** wynosi $O((k+n)\cdot \log n)$, gdzie k to ilość przecięć odcinków, a z wykorzystaniem tablic może wynieść nawet $O(n^2 \cdot \log n)$.

6. Podsumowanie

W procedurach do sprawdzania czy istnieje przecięcie na zbiorze odcinków oraz znajdywania wszystkich przecięć na zbiorze odcinków wykorzystałem dokładnie te same struktury zdarzeń i stanu, ale nie było takiej potrzeby.

W procedurze sprawdzania czy istnieje przecięcie na zbiorze odcinków, struktura zdarzeń **Q** mogłaby być zwykła Python-ową listą, która po wrzuceniu odcinków została posortowana po współrzędnej **x** punktów wyznaczających końce odcinków. Wynika to z faktu, że nigdy nie zmieniamy tej struktury, a jedynie szukamy, czy istnieje przecięcie.