Ćwiczenie IV - Przecinanie odcinków

Jakub Frączek - grupa nr 4

1. Wstęp

Opis ćwiczenia

Ćwiczenie polegało na zaimplementowaniu:

- funkcji generującej daną liczbę odcinków z zakresu współrzędnych 2D
- funkcji pozwalającej interaktywnie generować odcinki na płaszczyźnie
- algorytmu zamiatania sprawdzającego, czy w zbiorze istnieje choć jedna para przecinających się odcinków
- algorytmu zamiatania wyznaczającego wszystkie punkty przecięcia odcinków w zbiorze

Dane techniczne - software

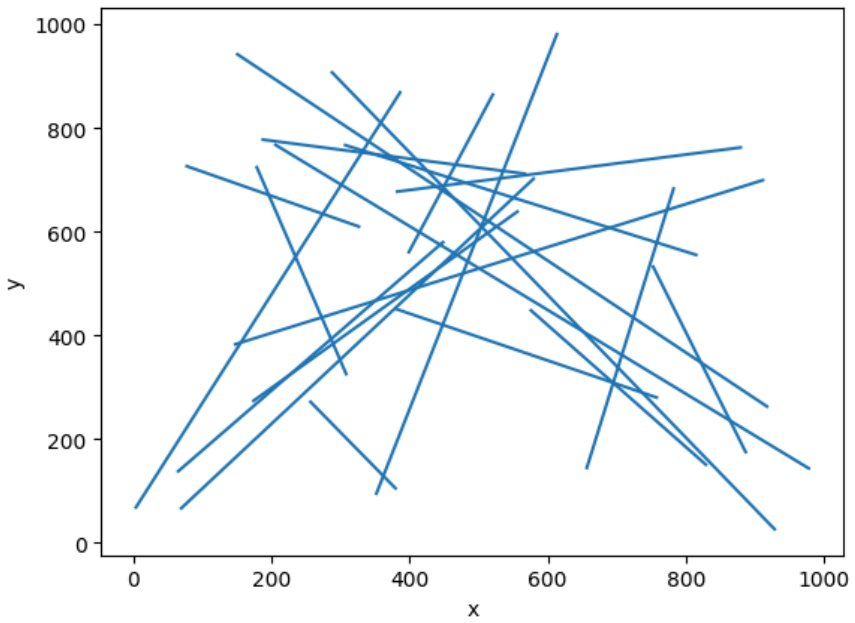
Ćwiczenie zostało zrealizowane w języku Python przy użyciu środowiska Jupyter notebook. Wykorzystane biblioteki to: numpy, random, pandas, matplotlib, queue, bitalg.

Dane techniczne - hardware

Laptop z systemem operacyjnym Linux Mint, procesorem AMD Ryzen 5 5500U 2.1 GHZ oraz 8 GB pamięci RAM.

2. Generowanie odcinków na płaszczyźnie

Funkcja generuje losowo odcinki na płaszczyźnie tak, że żaden odcinek nie jest pionowy. Szansa na taki odcinek jest bardzo mała biorąc pod uwagę, że punkty mają współrzędne rzeczywiste. Aby temu zapobiec, wszystkie współrzędne x - owe są generowane jako unikalne (w zasadzie wystarczyłoby sprawdzić czy współrzędne x - owe odcinka są sobie równe i jeśli tak to generować nowy punkt do skutku). Na wykresie 1. przedstawiono przykładowy zbiór wygenerowanych odcinków.



Wykres 1. Losowo wygenerowany zbiór odcinków

3. Interaktywne generowanie odcinków na płaszczyźnie

Interaktywne generowanie odcinków odbywa się za pomocą funkcjonalności biblioteki matplotlib. Zostaje wyświetlone okno, które pozwala użytkownikowi “wyklikać” interesujące go odcinki.

4. Algorytm sprawdzający, czy w zbiorze istnieje choć jedna para przecinających się odcinków

Struktura zdarzeń Q przechowuje jedynie posortowane rosnąco po współrzędnej x-owej początki i końce odcinków. Natomiast struktura stanu T odcinki posortowane ze względu na współrzędną y-ową. Struktura zdarzeń Q jest zaimplementowana przy użyciu kolejki priorytetowej, a struktura stanu T za pomocą listy. Algorytm postępowania:

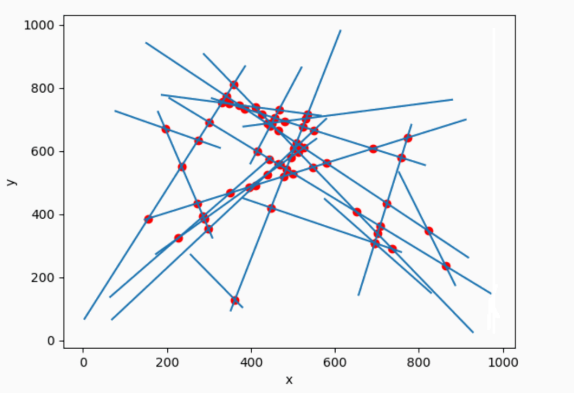
- Należy utworzyć strukturę stanu T oraz strukturę zdarzeń Q
- Umieścić w Q wszystkie punkty początkowe i końcowe
- Następnie w pętli, dopóki Q nie będzie puste:
    - Pobrać z Q punkt o najmniejszej współrzędnej x-owej.
    - Jeśli p jest lewym końcem odcinka to dodać go do T
    - Jeśli p jest prawym końcem odcinka, to usunąć go z T
    - Dla każdej pary nowych sąsiadów w T sprawdzić, czy tworzą one punkt przecięcia:
        - Jeśli tak, to dostać to zwrócić True

5. Algorytm wyznaczający wszystkie przecięcia odcinków w zbiorze.

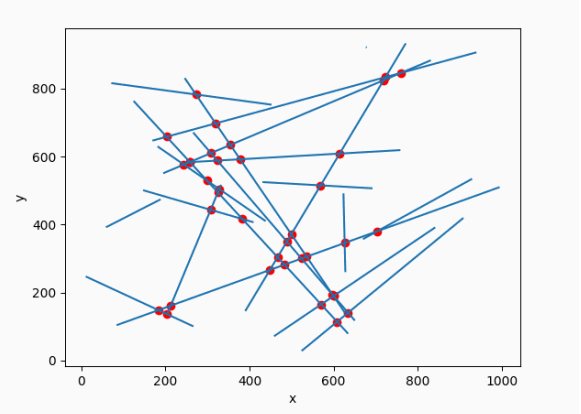
Implementacja tego algorytmu wymagała rozszerzenia funkcjonalności struktury zdarzeń Q o dodawanie do niej punktów przecięć. Implementacja struktury zdarzeń to ponownie kolejka priorytetowa, która pozwala otrzymywać punkty o rosnących współrzędnych x-owych w czasie log(n). Aby zapobiec powstawaniu duplikatów, wszystkie punkty przecięć wraz z identyfikatorami odcinków, które je tworzą są przechowywane w słowniku, a więc sprawdzenie, czy dany punkt jest duplikatem odbywa się w czasie stałym (Sprawdzanie odbywa się po indeksach odcinków tworzących punkt przecięcia). W przypadku tego algorytmu należało także rozszerzyć funkcjonalność struktury stany T, aby umożliwiała ona zamienienie miejscami odcinków po wyjęciu z Q punktu przecięcia. Współrzędne punktu przecięcia są wyliczane poprzez wyprowadzenie równań na proste przechodzącę przez punkty tworzące przecinające się odcinki, a następnie rozwiązanie powstałego układu równań. Pełny algorytm postępowanie jest następujący:

- Należy utworzyć strukturę stanu T oraz strukturę zdarzeń Q
- Umieścić w Q wszystkie punkty początkowe i końcowe
- Następnie w pętli, dopóki Q nie będzie puste:
    - Pobrać z Q punkt o najmniejszej współrzędnej x-owej.
    - Jeśli p jest lewym końcem odcinka to dodać go do T
    - Jeśli p jest prawym końcem odcinka, to usunąć go z T
    - Jeśli p jest punktem przecięcia to zmienić w T kolejność odcinków, które go tworzą
    - Dla każdej pary nowych sąsiadów w T sprawdzić, czy tworzą one punkt przecięcia:
        - Jeśli tak, to dostać ten punkt do Q

Poniżej zaprezentowany jest rezultat działania algorytmu dla 2 różnych zbiorów odcinków. Ponadto w dołączonym notatniku Jupyter znajdują się gify, które pokazują przesuwającą się miotłę krok po kroku.



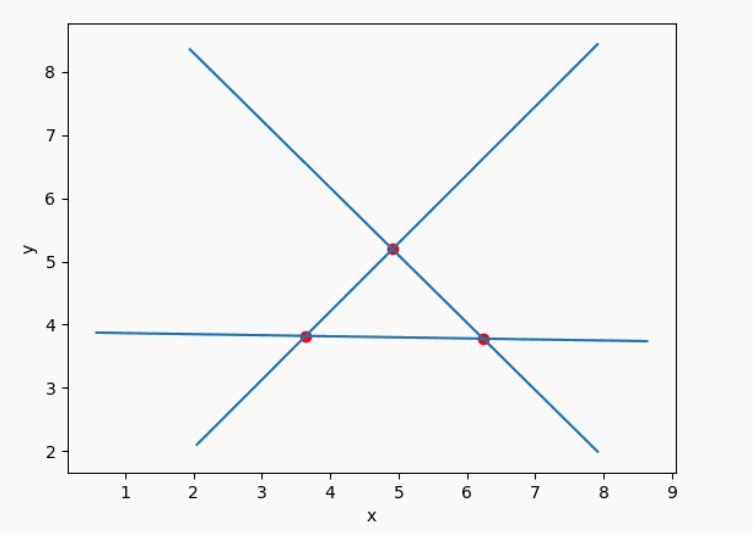
Wykres 2. Efekt działania dla zbioru 1.



Wykres 3. Efekt działania dla zbioru 2.

6. Przypadek, w którym przecięcia są wykrywane więcej niż raz

W tym przypadku po dodaniu punktu do struktury zdarzeń Q i zamienieniu pozycji odcinków tworzących ten punkt w strukturze stanu T, następuje parę zmian w pozycji odcinków w taki sposób, że odcinki tworzące punkt zostają najpierw rozdzielone, a potem znów umieszczone obok siebie i oznaczone jako nowi sąsiedzi. Taka sytuacja jest zaprezentowana na wykresie poniżej.



Wykres 4. Przypadek, w którym algorytm generuje duplikaty punktów przecięć

7. Porównanie czasów wykonania dla różnej ilości odcinków w zbiorze

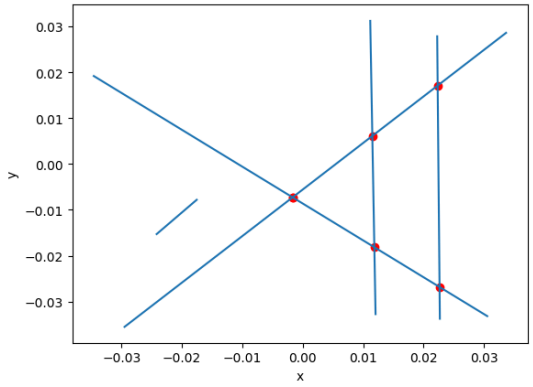
Poniżej w tabeli przedstawione są czasy działania algorytmu wyznaczania punktów przecięcia dla różnych ilości odcinków na zbiorze $(x, y) \in [0, 1000]^2$.

Ilość odcinków ->	10	100	1000	5000
Czas wykonania	1,02 ms	49,23 ms	1850,85 ms	11209,95 ms
Ilość punktów przecięć	9	1182	52316	314009

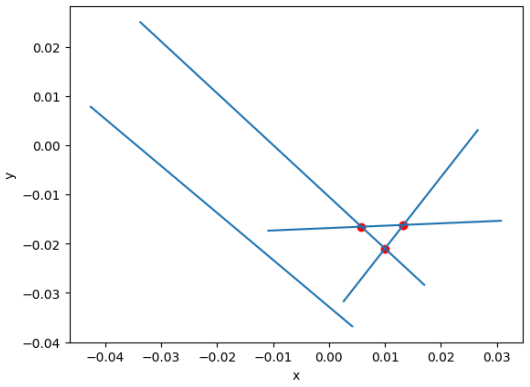
Tabela 1.

8. Wizualizacja rozwiązań dla wybranych zbiorów odcinków

Poniższe zbiory odcinków mogły być problematyczne dla algorytmu m. in. poprzez generowanie duplikatów punktów przecięć oraz nachylenie odcinków względem osi układu.



Wykres 5. Zbiór 3.



Wykres 6. Zbiór 4.

9. Wnioski

Na podstawie przeprowadzonych testów można stwierdzić, że algorytmy działają poprawnie. Złożoność operacji na strukturze T wynosi O(n), można ją poprawić stosując na przykład implementację drzewa AVL, co pozwoliłoby na wykonywanie wszystkich potrzebnych operacji na strukturze stanu w czasie O(log(n)). Poza tym algorytmy są dość wydajne. Jedną z trudności było wyciąganie ze struktury zdarzeń Q punktów z rosnącymi współrzędnymi x-owymi, ale PriorityQueue bardzo dobrze radzi sobie z tym zadaniem. W przypadku duplikatów, łatwym sposobem radzenia sobie z nimi był słownik, na którym potrzebne operacje mogły zostać wykonane w czasie stałym.