

Algorytmy geometryczne – laboratorium 4.

Przecinanie się odcinków.

1. Dane techniczne:

- System operacyjny: Windows 11 Pro (x64)
- Procesor: AMD Ryzen 5 7530U (2.0 – 4.5 GHz)
- Pamięć RAM: 16GB (2133 MHz)
- Środowisko: Jupyter Notebook
- Język programowania: Python 3.13.7

Podczas przeprowadzenia doświadczenia użyto narzędzia do wizualizacji danych autorstwa Koła Naukowego BIT oraz bibliotek *numpy*, *pandas*, *matplotlib*, *tkinter*, *sortedcontainers* oraz *queue*. By zagwarantować powtarzalność generowanych danych użyto polecenia `np.random.seed(42)`.

2. Cel ćwiczenia

- Wdrożenie do zagadnień związanymi z algorytmami zmiatania.
- Wyznaczenie punktów przecięcia w zadanych zbiorach danych, wizualizacja algorytmów.

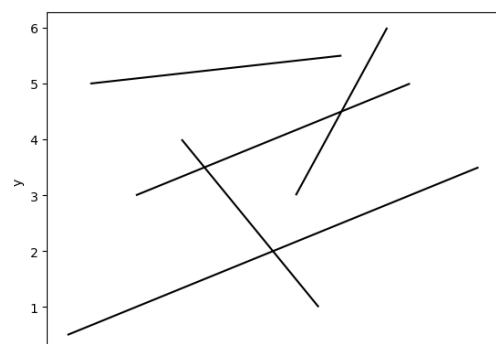
3. Wstęp teoretyczny

3.1. Rozważane zbiory odcinków

Algorytm opracowano dla zbiorów odcinków, na płaszczyźnie dwuwymiarowej, definiowanych za pomocą ich punktów początkowych oraz końcowych.

Odcinki te spełniają następujące założenia:

- Żaden odcinek nie jest pionowy.
- Dwa odcinki przecinają się w co najwyżej jednym punkcie.
- Żadne trzy odcinki nie przecinają się w jednym punkcie.



Rys nr. 1: Przykładowy zbiór odcinków.

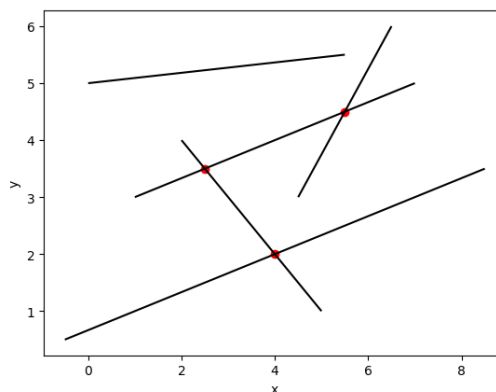
Przykładowy, rozważany zbiór odcinków przedstawiono na rysunku o nr 2.

3.2. Punkty przecięcia odcinków

Współrzędne punktów przecięcia (x_p, y_p) dwóch odcinków s_i oraz s_j dla $i \neq j$ należących do danego zbioru S odcinków wyznaczone zostały za pomocą wzoru wykorzystującego postać kierunkową prostych reprezentujących odcinki s_i oraz s_j .

$$s: y = a \cdot x + b$$

$$x_p = \frac{(b_j - b_i)}{(a_i - a_j)} \quad y_p = a_i \cdot x_p + b_i$$



Rys nr. 2: Przykładowy zbiór odcinków wraz z zaznaczonymi punktami przecięcia.

Wyznaczona współrzędna x_p punktu przecięcia musi spełniać warunek zawierania się między minimum oraz maksimum ze współrzędnych punktu początkowego oraz końcowego. Przykładowy zbiór odcinków S wraz z zaznaczonymi punktami wzajemnego przecinania się pary odcinków został przedstawiony na rysunku o nr 2. Punkty przecięcia oznaczono kolorem **czerwonym**.

3.3. Algorytm weryfikacji występowania co najmniej jednego przecięcia pary odcinków

Program działa analogicznie do algorytmu opisanego w punkcie 3.4, jednak swoje działanie przerywa w momencie znalezienia pierwszego przecięcia.

W jego implementacji nie rozpatruje się punktów zdarzeń o typie punktów przecięcia. Dla wygody implementacyjnej, dla struktury zdarzeń wybrano posortowaną tablicę punktów. W przypadku obu programów nie jest wymagana ta sama struktura zdarzeń, ponieważ ich charakterystyka znacząco różni się od siebie. Tylko w pełnym algorytmie wyznaczania punktów przecięcia odcinków wymagana jest obsługa dołączania nowych zdarzeń do struktury zdarzeń w postaci punktów przecięcia.

3.4. Algorytm wyznaczania punktów przecinania się odcinków

Działanie algorytmu opiera się na symulacji przejścia hiperpłaszczyzny, w ustalonym kierunku, wyznaczonej dla każdego zbioru rozważanych odcinków. Pozycje, w których miotła zatrzymuje się, określone są zdarzeniami. Informacje o nich przechowywane są w strukturze zdarzeń. Dane odcinków potrzebne do obliczeń przechowywane są w strukturze stanu, która jest aktualizowana w każdym napotykanym przez miotłę zdarzeniu.

Algorytm rozpoczyna pracę od dodania każdego z punktów odcinków do struktury zdarzeń

oraz oznaczenia ich jako punkt początkowy odcinka (punkt o mniejszej współrzędnej x) oraz punkt końcowy odcinka (punkt o większej współrzędnej x).

W kolejnych krokach algorytmu ze struktury zdarzeń pobierana jest pozycja oraz typ punktu o aktualnie najmniejszej współrzędnej x , tzn. punktu, w którym symulowana miotła się zatrzyma.

Zachowanie programu w punkcie zależne jest od jego typu, który należy do jednej z poniższych kategorii:

- **Punkt początkowy odcinka** – odcinek, do którego należy dany punkt, oznaczany jest jako aktywny poprzez dodanie do struktury stanu. Kolejno sprawdzana jest obecność aktywnych odcinków, które przecinają hiperpłaszczyznę miotły powyżej oraz poniżej aktualnego punktu. Następnie, stosując metodę opisaną w punkcie 3.2., weryfikowane jest występowanie punktu przecięcia tych odcinków z odcinkiem, którego początek został napotkany. Jeśli którykolwiek z aktualnie rozpatrywanych punktów przecięcia istnieje oraz nie był wcześniej odnaleziony, to zostaje on zapisany oraz dodany do struktury zdarzeń z oznaczeniem typu punktu przecięcia.
- **Punkt końcowy odcinka** – odcinek, do którego należy dany punkt, oznaczany jest jako nieaktywny poprzez usunięcie ze struktury stanu. Kolejno sprawdzana jest obecność odcinków, które przecinają hiperpłaszczyznę miotły powyżej oraz poniżej aktualnego punktu. Następnie, stosując metodę opisaną w punkcie 3.2., weryfikowane jest występowanie punktu przecięcia tych odcinków. Jeśli punkt przecięcia istnieje oraz nie był wcześniej odnaleziony, to zostaje on zapisany oraz dodany do struktury zdarzeń z oznaczeniem typu punktu przecięcia.
- **Punkt przecięcia odcinków** – kolejność odcinków względem miotły, które przecinają się w napotkanym punkcie, zostaje zamieniona w strukturze stanu. Kolejno sprawdzana jest obecność aktywnych odcinków przecinających miotłę powyżej oraz poniżej tych dwóch odcinków. Następnie, stosując metodę opisaną w punkcie 3.2., weryfikowane jest występowanie punktów przecięcia między nowo zamienionym odcinkiem na pozycję wyższą względem miotły i odcinkiem przecinającym miotłę powyżej oraz między nowo zamienionym odcinkiem na pozycję niższą i odcinkiem przecinającym miotłę poniżej. Jeśli którykolwiek z aktualnie rozpatrywanych punktów przecięcia istnieje oraz nie był wcześniej odnaleziony, to zostaje on zapisany oraz dodany do struktury zdarzeń z oznaczeniem typu punktu przecięcia.

Algorytm kończy pracę wraz z obsłużeniem ostatniego punktu w strukturze zdarzeń.

Jako strukturę zdarzeń wybrano kolejkę priorytetową, a jako strukturę stanu – strukturę `SortedSet`. Dzięki dobraniu takich struktur algorytm osiągnął złożoność czasową rzędu $O((n + k) \log n)$, gdzie k to liczba przecięć, a n to liczba odcinków wejściowych.

Na wizualizacjach w postaci rysunków odcinki oraz ich punktu są oznaczone kolorami:

- **czarnym** – odcinek oraz jego punkty początkowe i końcowe.
- **czzerwonym** – wyznaczone punkty przecięcia pary odcinków.

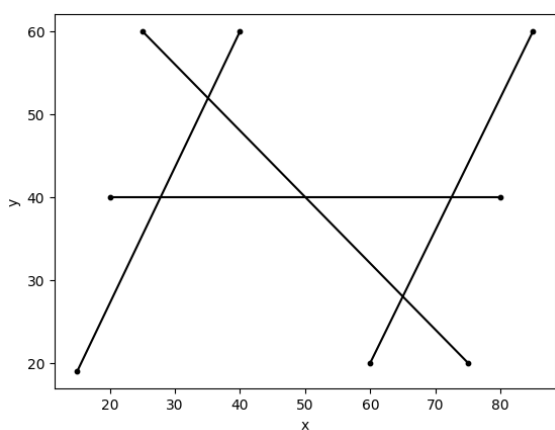
Na wizualizacjach w pliku gif odcinki oraz ich punkty są oznaczone kolorami:

- **czarnym** – jeszcze nienapotkany odcinek,
- **zielonym** – aktywny odcinek w strukturze stanu,
- **szarym** – zdezaktywowany odcinek usunięty ze struktury stanu,
- **niebieskim** – punkty początkowe odcinków,
- **ciemnoniebieskim** – punkty końcowe odcinków,
- **fioletowym** – odcinki, których przecinanie się jest aktualnie sprawdzane,
- **niebieskim** – punkty odcinków dodane do struktury zdarzeń
- **czerwonym** – miotła, aktualnie napotkany przez nią punkt oraz wyznaczone punkty przecięcia pary odcinków.

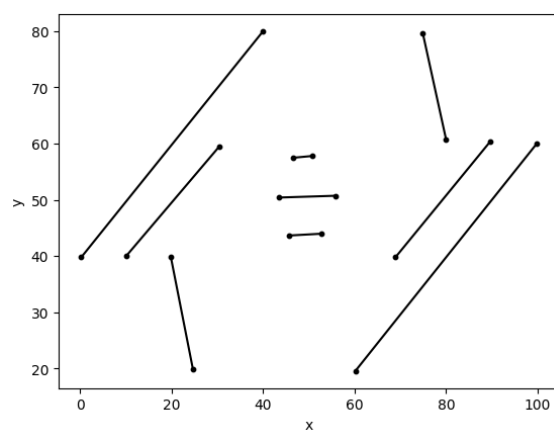
4. Przebieg doświadczenia

Doświadczenie rozpoczęto od wprowadzenia oraz zapisania do pliku 4 testowych zbiorów odcinków z użyciem wcześniej zaimplementowanego, interaktywnego narzędzia.

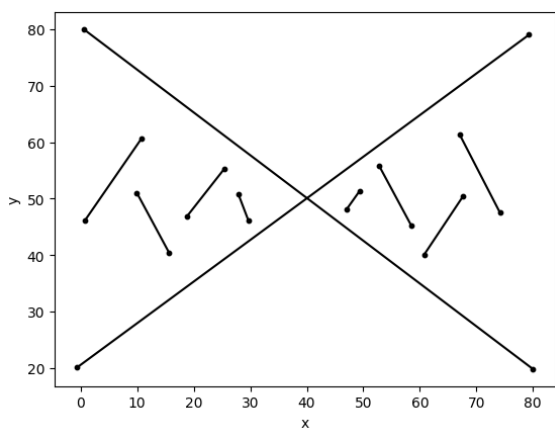
Wprowadzone dane zobrazowano na rysunkach o nr od 3 do 6.



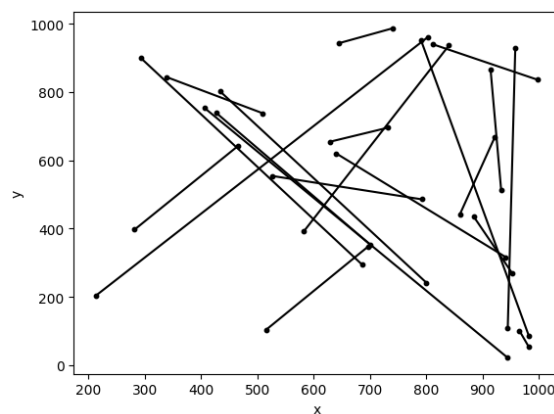
Rys nr. 3: Zbiór odcinków A.



Rys nr. 4: Zbiór odcinków B.



Rys nr. 5: Zbiór odcinków C.



Rys nr. 6: Zbiór odcinków D.

Powyższe wielokąty zostały dobrane tak, by swoimi cechami wpłynąć na jak najdokładniejsze zbadanie algorytmu w zróżnicowanych przypadkach testowych.

Zbiór A został dobrany tak by ukazać pracę algorytmu na prostym zbiorze odcinków.

Zbiór B został dodany do testu jako prosty zbiór, którego odcinki się nie przecinają.

Zbiór C został dobrany tak, by ukazać sposób działania algorytmu w przypadku wielokrotnego wykrywania punktu przecięcia.

Zbiór D został wygenerowany jako zbiór 20 losowych odcinków o punktach początkowych i końcowych ze współrzędnymi z przedziału $[0, 1000]$.

Kolejnym krokiem doświadczenia było zaimplementowanie algorytmów opisanych w punkcie 3 opracowania. Posłużono się nimi w następnych etapach doświadczenia.

Każdy z wielokątów przykładowych odczytano z pliku oraz zweryfikowano pod względem występowania co najmniej jednego punktu przecięcia pary odcinków w zbiorze.

Drugim etapem doświadczenia było dokładne wyznaczenie wszystkich punktów przecięcia wraz z określeniem ich współrzędnych.

5. Analiza wyników

5.1. Występowanie co najmniej jednego punktu przecięcia pary odcinków ze zbioru

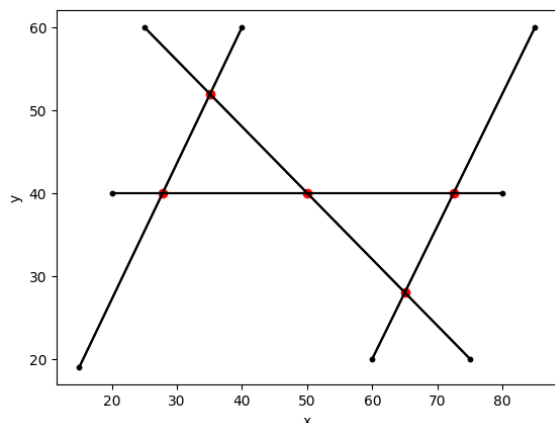
Każdy ze zbiorów wprowadzono i poddano weryfikacji z wykorzystaniem algorytmu opisanego w punkcie 3.3. Otrzymane wyniki wprowadzono do tabeli nr 1.

Czy w zbiorze występuje co najmniej jedno przecięcie pary odcinków?			
A	B	C	D
Tak	Nie	Tak	Tak

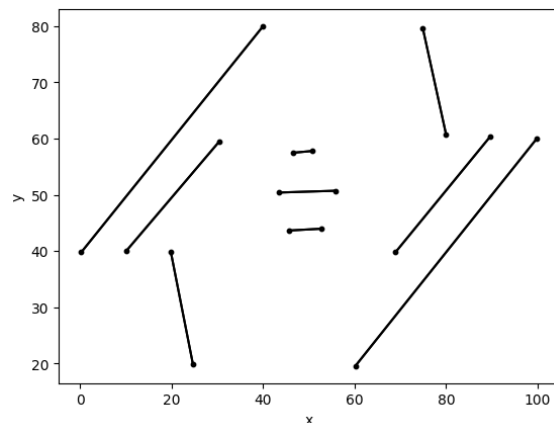
Tabela nr 1: Wyniki algorytmu weryfikującego występowanie co najmniej jednego punktu przecięcia pary odcinków.

5.2. Wyznaczenie punktów przecięcia par odcinków ze zbioru

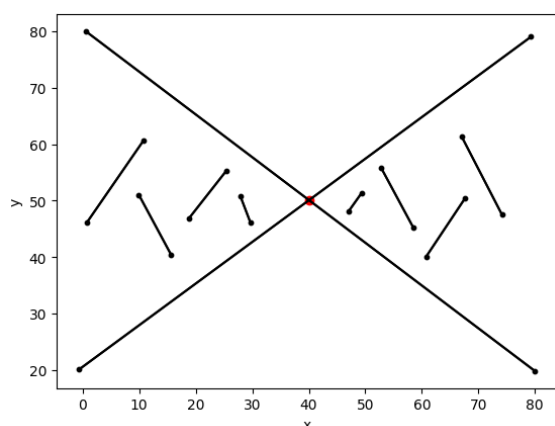
Dla każdego ze zbiorów wyznaczono punkty przecięcia z wykorzystaniem algorytmu opisanego w punkcie 3.4. Otrzymane wyniki zobrazowano na rysunkach o nr od 7 do 10 oraz do tabeli nr 2.



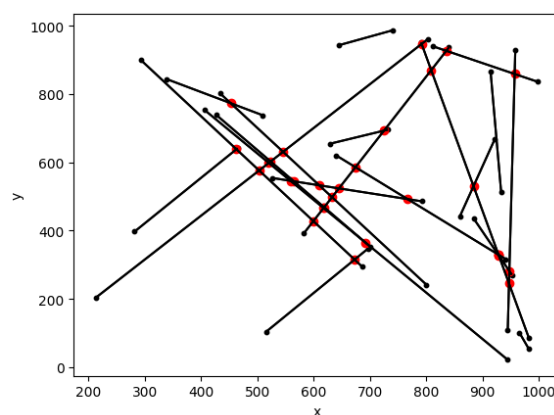
Rys nr. 7: Wyznaczone punkty przecięcia dla zbioru odcinków A.



Rys nr. 8: Wyznaczone punkty przecięcia dla zbioru odcinków B.



Rys nr. 9: Wyznaczone punkty przecięcia dla zbioru odcinków C.



Rys nr. 10: Wyznaczone punkty przecięcia dla zbioru odcinków D.

Liczba punktów przecięć			
A	B	C	D
5	0	1	29

Tabela nr 2: Liczba punktów przecięcia znalezionych w zbiorach

Punkty przecięcia znalezione przez algorytm to:

Dla zbioru A:
 (27.804878048780488, 40.0)
 (50.0, 40.0)
 (35.08196721311475, 51.93442622950819)
 (64.99999999999999, 27.999999999999986)
 (72.5, 40.0)

Dla zbioru B:

Dla zbioru C:

(40.04241349287496, 50.087803492793526)

Dla zbioru D:

(462.7893393904151, 638.20692865964)
(692.0981315490843, 365.09816360935076)
(452.6155549570083, 773.0844690144862)
(503.182413653887, 576.0847033595471)
(520.3785536567021, 598.1447737051155)
(672.8683709512487, 315.11745930352185)
(523.711415727428, 602.4203362278824)
(545.7910074444759, 630.7451432136229)
(558.9216318280794, 545.8366221201818)
(564.9630938161903, 544.2807269337967)
(609.9797568366637, 532.6873064275495)
(599.6152261683083, 427.77658335195713)
(617.7376142044847, 466.01540732041303)
(618.8416477198831, 468.3449536895954)
(632.8078269587793, 497.8140428825851)
(645.0542346429105, 523.6543585597806)
(767.1570244635222, 492.20846958457014)
(674.433720391831, 585.6460222187909)
(725.1889970699768, 692.7412950101882)
(792.0732312966112, 946.6883247736541)
(809.100698368706, 869.7976944455413)
(928.7866959416203, 329.3337630888873)
(835.9504001119386, 926.4514321602112)
(884.3177407316127, 530.1414335223649)
(928.0078345792734, 330.11862402829956)
(929.3138952118145, 326.9530987174858)
(947.4273748279725, 245.15838085224095)
(948.0225636448239, 281.6085243686757)
(957.444981588818, 858.6497533140064)

Uzyskane wyniki są w pełni zgodne z oczekiwanymi rezultatami. Świadczą one o poprawnym wykonaniu zadania. Implementacja algorytmu zwraca poprawne wyniki dla różnych przypadków testowych.

6. Wnioski

Na podstawie analizy wyników doświadczenia nasuwają się następujące wnioski:

- 1) Dowiedziono poprawności implementacji algorytmu wyznaczania punktów przecięcia odcinków dla zadanych zbiorów poprzez ich zbieżność z teoretycznymi przewidywaniami.
- 2) Kolejka priorytetowa oraz struktura `SortedSet` okazały się użytecznymi strukturami danych. Dzięki ich użyciu algorytm osiąga optymalną złożoność obliczeniową, a także dużą łatwość w implementacji.

Wizualizacja wyników algorytmów oraz dokładna selekcja przypadków testowych pozwoliła na weryfikację poprawności zaimplementowanych algorytmów. Posługiwanie się narzędziem graficznym zdecydowanie wspomaga ocenę jakości zaproponowanych rozwiązań, a także wspomaga dogłębne zrozumienie idei stosowanych mechanizmów algorytmicznych.