Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji
Kierunek Informatyka

Algorytmy geometryczne

Laboratorium 4

Przecinanie się odcinków

Kyrylo Iakymenko

Kraków, 5 grudnia 2023

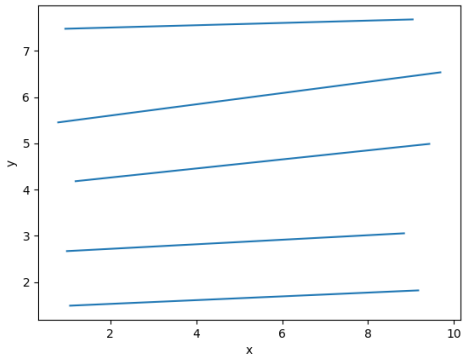
1 Wprowadzenie

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.

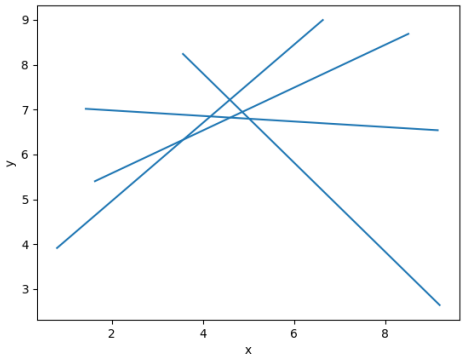
W dalszej części raportu omówimy algorymt zamiatania służący rozwiązania problemu przecinania się odcinków oraz przeanalizujemy złożoność czasową i przestrzenną algorymtu.

2 Zbiory testowe

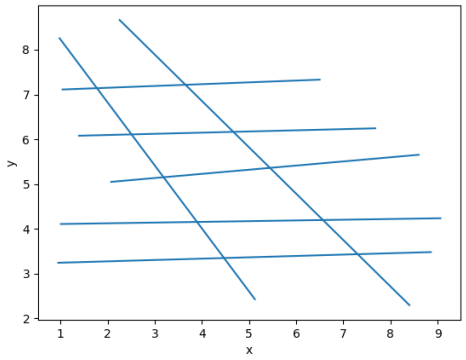
Na potrzeby ćwiczenia stworzyliśmy 6 zbiorów odcinków na płaszczyźnie. Zostały wybrane odpowiednio, żeby przetestować działanie algorytmu w przypadkach zarówno lososwych jak i ekstremalnych.



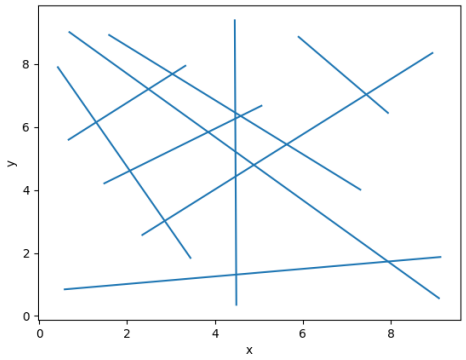
Rys. 1



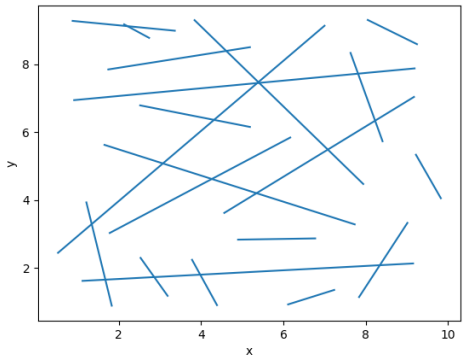
Rys. 2



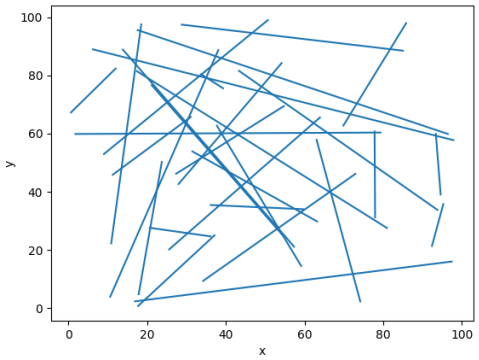
Rys. 3



Rys. 4



Rys. 5



Rys. 6

3 Algorytmy

3.1 Algorytm triangulacji

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

1. **Sortowanie punktów końcowych odcinków:** Algorytm rozpoczyna się od posortowania punktów końcowych odcinków według współrzędnej x. Dzięki temu możemy ustalić kolejność, w jakiej algorytm będzie przetwarzał odcinki.

2. **Sweepline:** Algorytm używa 'miotły', która przesuwa się przez posortowane punkty. Linia ta reprezentuje aktualne położenie algorytmu na płaszczyźnie.

3. **Struktura danych do przechowywania aktywnych odcinków:** Tworzona jest struktura danych, najczęściej używane to drzewo binarne przeszukiwań lub struktura zrównoważona, które przechowuje aktualnie przecinane przez sweepline odcinki.

4. **Przeszukiwanie sweepline:** Sweepline przemieszcza się przez posortowane punkty, a w miarę tego ruchu algorytm aktualizuje strukturę danych, śledząc, które odcinki są obecnie przecinane przez sweepline.

5. **Znajdowanie przecięć:** Przy każdym ruchu sweepline, algorytm sprawdza, czy doszło do przecięcia pomiędzy aktualnym odcinkiem, a innymi odcinkami znajdującymi się nad lub pod sweepline. W przypadku wykrycia przecięcia, jest ono rejestrowane.

6. **Obsługa przypadków szczególnych:** Algorytm uwzględnia różne przypadki przecięć, takie jak przecięcia proste, styczne, czy nakładające się odcinki. Obsługuje również sytuacje szczególne, takie jak wspólne punkty i brzegi odcinków.

7. **Aktualizacja struktury danych:** W miarę poruszania się sweepline, struktura danych jest aktualizowana, a odcinki, które już zostały przetworzone, są usuwane z niej.

Algorytm zamiatania jest stosunkowo prosty do zrozumienia i implementacji, a jego złożoność czasowa wynosi $O(nlogn)$, gdzie n to liczba odcinków. Działa efektywnie, zwłaszcza w przypadku dużych zbiorów danych, co czyni go popularnym narzędziem w problemach związanych z geometrią obliczeniową.

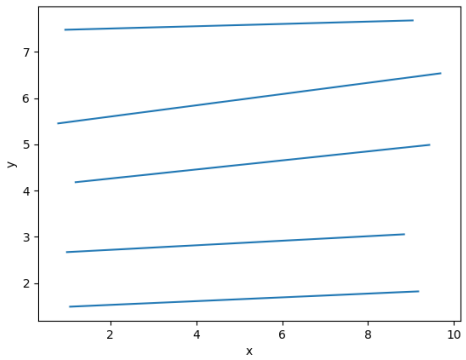
3.2 Algorytm sprawadzenia y-monotoniczności wielokąta

Nasz algorytm najpierw dzieli wierzchołki na 5 rodzajów:

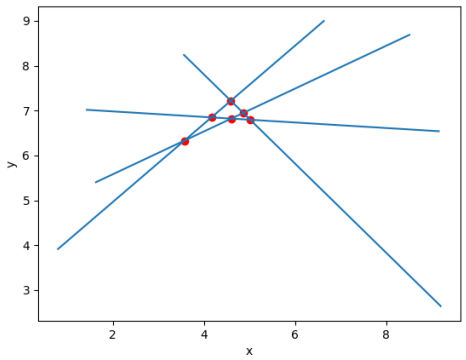
1. Początkowy gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny $< \pi$.
2. Końcowy, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny $< \pi$.
3. Łączący, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny $> \pi$.
4. Dzielący, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny $> \pi$.
5. prawidłowy, gdy ma jednego sąsiada powyżej, drugiego - poniżej.

Robi to licząc wyznacznik i odpowiednio porównując współrzędne swoich sąsiadów. Wtedy sprawdzenie czy wielokąt jest $y-$monotoniczny sprowadza się do sprawdzenia, czy nasz wielokąt posiada wierzchołki łączące lub dzielące. Jeśli tak - nie jest monotoniczny, w przeciwnym wypadku - jest.

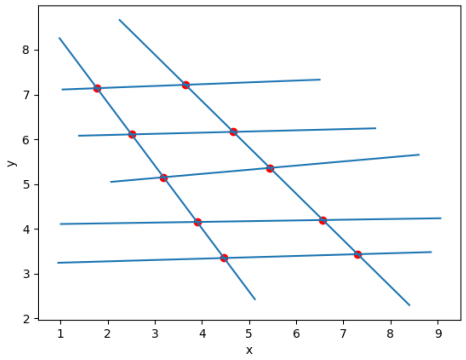
4 Wyniki algorytmu



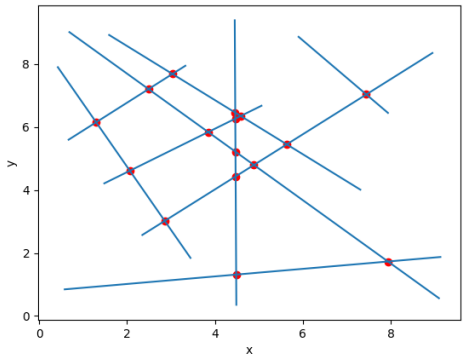
Rys. 7



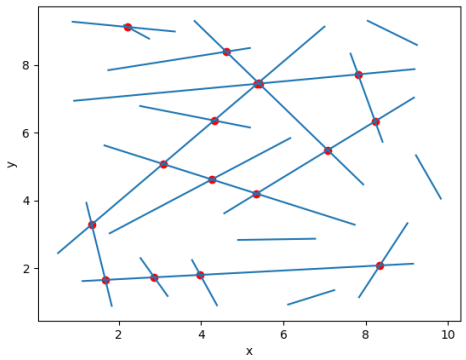
Rys. 8



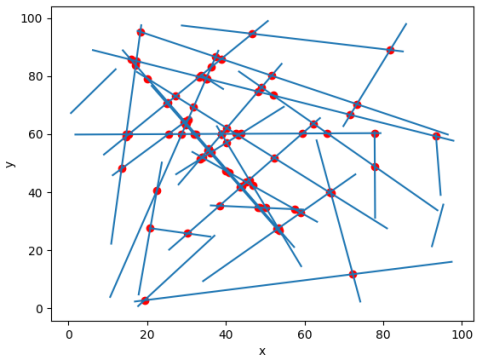
Rys. 9



Rys. 10



Rys. 11



Rys. 12

5 Podsumowanie

Skuteczność Algorytmu: Algorytm zamiatania wykazuje się wysoką skutecznością w identyfikowaniu przecięć odcinków na płaszczyźnie. Jego zdolność do obsługi różnych przypadków przecięć, takich jak przecięcia proste, styczne czy nakładające się odcinki, sprawia, że jest on wszechstronnym narzędziem do rozwiązania problemów geometrii obliczeniowej związanego z odcinkami.

Złożoność Obliczeniowa: Warto zauważyć, że algorytm zamiatania ma relatywnie niską złożoność obliczeniową w porównaniu do niektórych innych metod wyznaczania przecięć

odcinków. Jego efektywność staje się szczególnie istotna w przypadku dużej liczby odcinków, gdzie inne podejścia mogą być bardziej kosztowne obliczeniowo.

Przydatność w Praktyce: Algorytm zamiatania znajduje zastosowanie w wielu praktycznych dziedzinach, takich jak grafika komputerowa, planowanie tras czy analiza obrazów medycznych. Jego prostota implementacyjna i skuteczność czynią go atrakcyjnym narzędziem dla programistów i badaczy zajmujących się problemami geometrii obliczeniowej.

Obsługa Sytuacji Wyjątkowych: Omawiane sprawozdanie uwzględnia obsługę różnych przypadków szczególnych, takich jak wspólne punkty i brzegi odcinków. To potwierdza, że algorytm zamiatania jest elastyczny i potrafi radzić sobie z różnorodnymi sytuacjami, co zwiększa jego użyteczność w praktyce.

Wyzwania Implementacyjne: Pomimo ogólnej efektywności, implementacja algorytmu zamiatania może wymagać uwzględnienia wielu szczegółów. W przypadku bardziej skomplikowanych scenariuszy, programiści powinni być świadomi potencjalnych wyzwań związanych z obsługą sytuacji granicznych i optymalizacją kodu.

Jedną ze zmian, którą trzeba było zrealizować było zwracanie w wyniku triangulacji, zarówno oryginalnego wielokąta jak i triangulacji po indeksach. To pozwala na wygodny odczyt tej struktury i możliwości szybkiej wizualizacji wyników. Także ten sposób przechowywania wyniku algorytmu uniemożliwia błędy związane ze złym dopasowaniem triangulacji do właściwego wielokąta, gdyż zarówno wielokąt, jak i wynik są przechowywane razem. Jeszcze jedną rzeczą na którą trzeba było zwrócić uwagę była reprezentacja triangulacji jako listy par indeksów wierzchołków, zamiast listy par współrzędnych wierzchołków. To pozwala przede wszystkim na zmiejszenie ilości używanej pamięci, ale również jest bardziej eleganckim rowiązaniem, które upraszcza implementacje algorytmu.