Algorytmy Geometryczne
Algorytm Greinera-Hormanna
Obliczanie sumy i iloczynu dwóch wielokątów

Błażej Żuk
07.01.2026

1. Dane techniczne

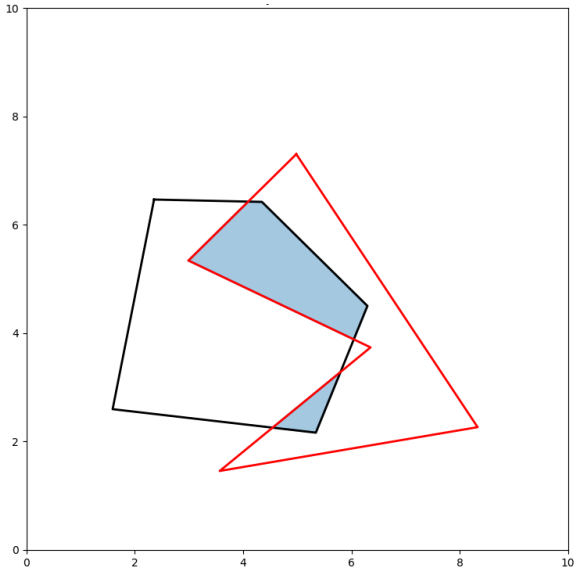
- System operacyjny: Windows 10
- Procesor: AMD Ryzen 5 3600 3,6 GHz
- Język programowania: Python 3.13.0

Do realizacji ćwiczenia wykorzystano biblioteki `Matplotlib` oraz `NumPy`.

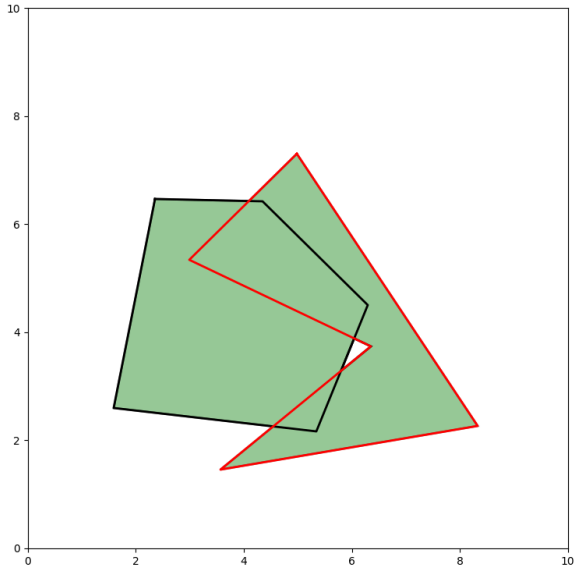
2. Wstęp teoretyczny

2.1. Problem obcinania wielokątów

Zadanie polega na wyznaczeniu wielokąta (lub zbioru wielokątów) będącego iloczynem oraz wielokąta będącego sumą dwóch zadanych wielokątów na płaszczyźnie. Algorytm Greinera-Hormanna jest efektywnym rozwiązaniem tego problemu, działającym w czasie liniowym względem liczby wierzchołków po znalezieniu wszystkich punktów przecięcia.
Na rysunkach 1 i 2 przedstawiono iloczyn i sumę dwóch przykładowych wielokątów. Wielokąty zostały zaznaczone kolorem czarnym i czerwonym, obszar iloczynu kolorem niebieskim, a obszar sumy kolorem zielonym.



Rysunek 1: Iloczyn dwóch wielokątów



Rysunek 2: Suma dwóch wielokątów

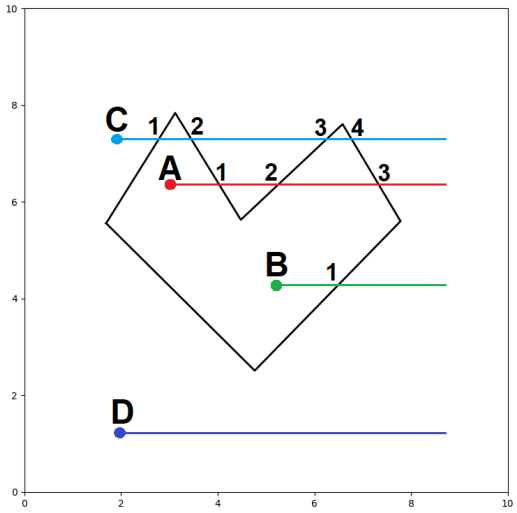
2.2. Założenia

Podczas realizacji zadania przyjęto następujące założenia:

- Każdy wielokąt jest prosty (nie posiada dziur oraz nie przecina samego siebie),
- Krawędzie wielokątów nie nachodzą na siebie,
- Wierzchołek jednego wielokąta nie leży na krawędzi drugiego wielokąta,
- Żadna para wierzchołków nie posiada takiej samej współrzędnej y.

2.3. Położenie punktu względem wielokąta

W trakcie działania algorytmu Greinera-Hormanna wielokrotnie trzeba sprawdzać czy dany punkt leży wewnątrz, czy na zewnątrz wielokąta. W tym celu zastosowano metodę „ray-casting". Na rysunku 3 została przedstawiona wizualizacja tego algorytmu - promienie rzucone przez wewnętrzne punkty A i B przecinają się z wielokątem odpowiednio 3 razy i 1 raz. W przypadku zewnętrznych punktów C i D promienie przecięły się 4 razy i 0 razy.



Rysunek 3: Wizualizacja metody „ray-casting"

2.4. Algorytm Greinera-Hormanna

Algorytm składa się z następujących etapów:

1. **Znajdowanie punktów przecięcia krawędzi** – dla każdej krawędzi jednego wielokąta sprawdzane jest, czy przecina się z krawędziami drugiego wielokąta. Przecięcia są wstawiane do obu wielokątów jako nowe wierzchołki.
2. **Oznaczanie punktów przecięcia jako wejście/wyjście** – dla każdego punktu przecięcia w danym wielokącie określane jest, czy jest to „punkt wejścia" do drugiego wielokąta, czy „punkt wyjścia". W tym celu przechodzimy po wszystkich wierzchołkach wielokąta - jeśli w danym momencie znajdujemy się na zewnątrz drugiego wielokąta, to wiemy, że następne przecięcie będzie wejściem. Analogicznie, jeśli jesteśmy wewnątrz drugiego wielokąta, to następne przecięcie będzie punktem wyjścia.
3. **Konstruowanie wyniku** – zaczynając od dowolnego punktu przecięcia, przechodzimy wzdłuż wielokąta zgodnie z regułą:
   - dla iloczynu: przy wejściu idziemy do przodu, przy wyjściu do tyłu,
   - dla sumy: przy wejściu idziemy do tyłu, przy wyjściu do przodu.

   Po osiągnięciu kolejnego punktu przecięcia przechodzimy na drugi wielokąt.
   Proces powtarzany jest, aż wszystkie przecięcia zostaną odwiedzone.
4. **Analiza przypadków bez przecięć** – jeśli wielokąty się nie przecinają, sprawdzane jest położenie jednego wielokąta wewnątrz drugiego. W tym celu można wykorzystać algorytm ray-casting. W zależności od wyniku zwracany jest odpowiedni wielokąt. Jeśli wielokąty są rozłączne, to w przypadku iloczynu zwracany jest zbiór pusty, a w przypadku sumy oba wielokąty.

3. Realizacja ćwiczenia

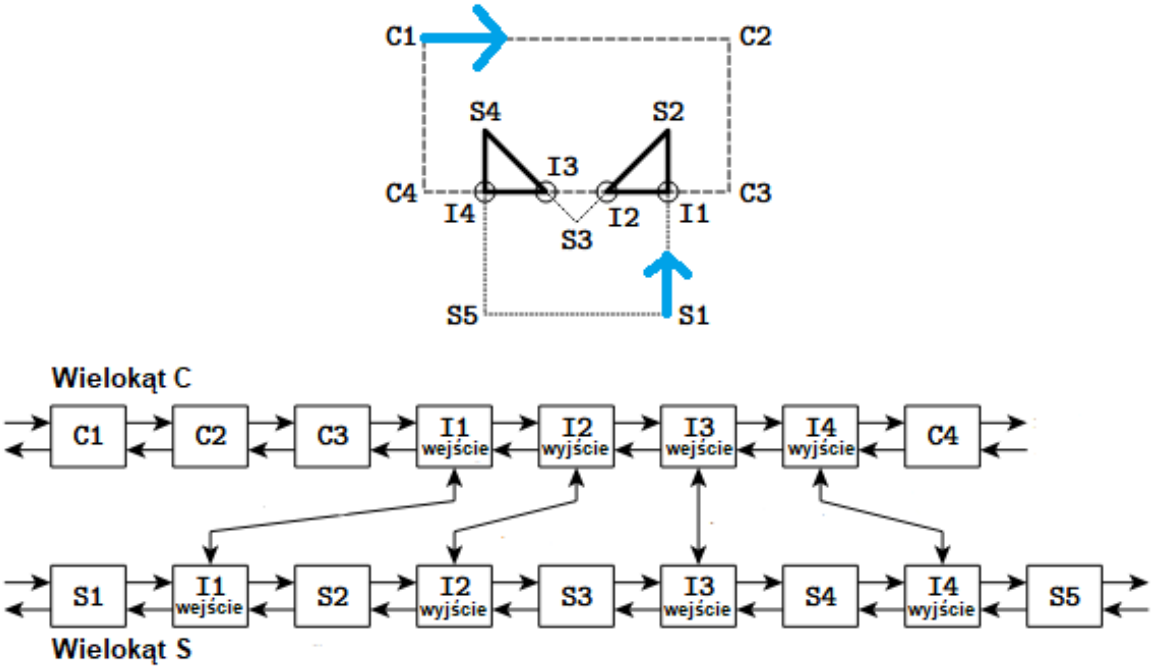
3.1. Struktury danych

Wielokąt przechowywany jest jako dwukierunkowa lista cykliczna obiektów klasy `Vertex`. Obiekty te reprezentują oryginalne wierzchołki wielokątów oraz punkty przecięcia. Każdy obiekt zawiera:

- współrzędne wierzchołka,
- indeks (dla oryginalnych wierzchołków),
- wskaźniki na poprzedni i następny wierzchołek,
- flagi: czy wierzchołek jest punktem przecięcia, czy jest wejściem, czy jest wyjściem,
- wskaźnik na odpowiadający punkt przecięcia w drugim wielokącie (dla punktów przecięcia),
- parametr α (dla punktów przecięcia).

Parametr α określa położenie punktu przecięcia na danym odcinku. Parametr ten jest potrzebny aby wstawić przecięcia do wielokątów w odpowiedniej kolejności.

Na rysunku 4 przestawiona jest wizualizacja struktur przechowujących dwa przykładowe wielokąty, po wstawieniu przecięć. Na górze rysunku znajdują się dwa wielokąty C i S z podpisanymi wierzchołkami oraz punktami przecięć. Na obu figurach zaznaczono niebieską strzałką kierunek zadania wielokąta. Pod wielokątami znajduje się wizualizacja dwóch dwukierunkowych list cyklicznych reprezentujących wielokąty C i S.

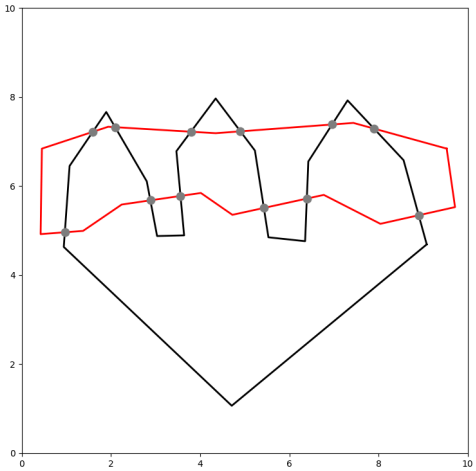


Rysunek 4: Struktury danych dla przykładowych wielokątów

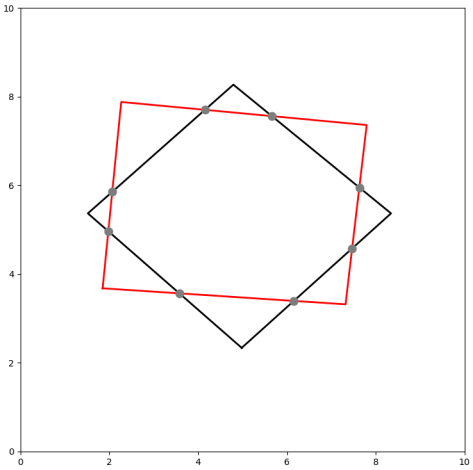
3.2. Zestawy danych

Na rysunkach 5-9 zostały przedstawione pary wielokątów na których przetestowano zaimplementowany algorytm. Wielokąty są zaznaczone kolorem czarnym i czerwonym. Kolorem szarym zaznaczono przecięcia wielokątów.

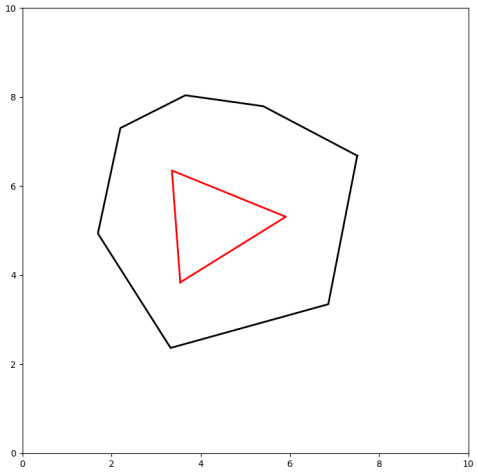
Rysunek 5: Para A



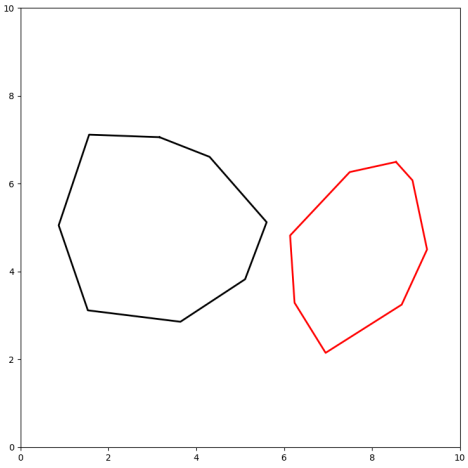
Rysunek 6: Para B



Rysunek 7: Para C



Rysunek 8: Para D



Rysunek 9: Para E

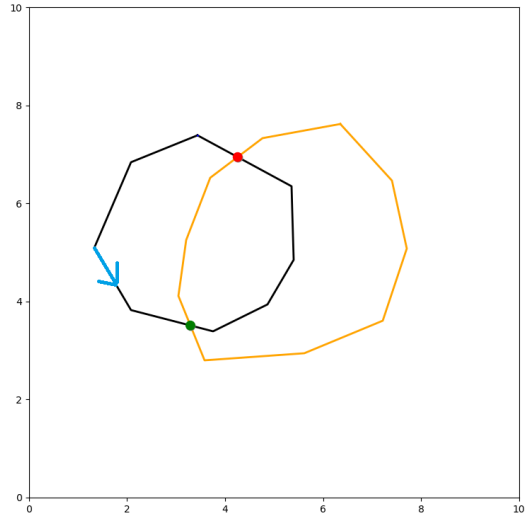
Para A została wybrana, aby sprawdzić działanie algorytmu w prostym przypadku, gdzie dwa wielokąty nachodzą na siebie.
Para B pokaże jak algorytm radzi sobie, kiedy iloczyn składa się z kilku wielokątów.
Para C pozwoli zweryfikować poprawność algorytmu, gdy krawędzie iloczynu należą na przemian do różnych wielokątów.
Pary D i E zostały wybrane aby sprawdzić, co się dzieje w sytuacji gdy wielokąty się nie przecinają.

4. Wyniki

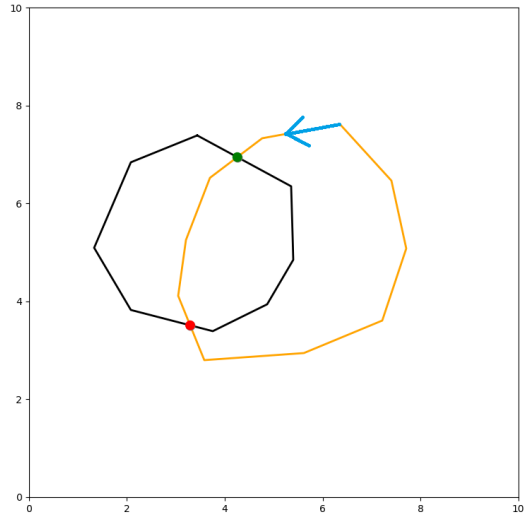
4.1. Oznaczanie przecięć jako punkty wejścia/wyjścia

Na rysunkach 10-15 przedstawiono klasyfikację przecięć par wielokątów A, B i C. Dla par D i E algorytm nie wykrył żadnego przecięcia, co jest zgodne z rzeczywistością.

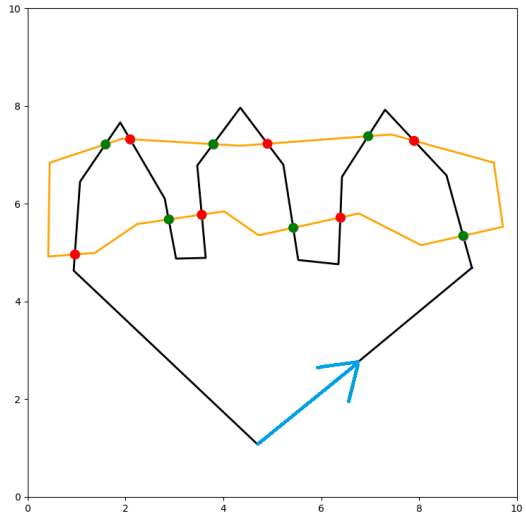
Wielokąty zaznaczono kolorem czarnym i pomarańczowym. Punkty wejścia oznaczono kolorem zielonym, a punkty wyjścia kolorem czerwonym. Niebieską strzałką zaznaczono kierunek wielokąta.



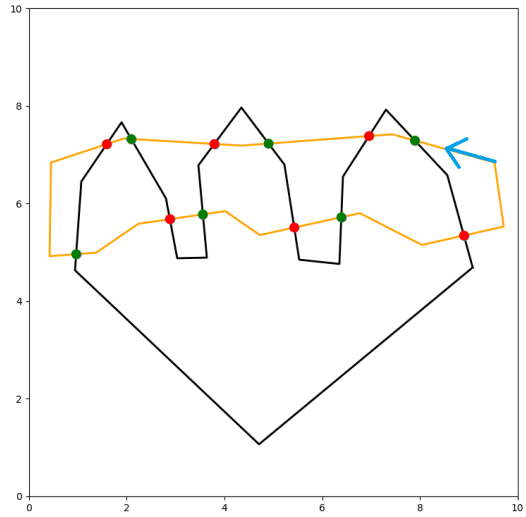
Rysunek 10: Klasyfikacja przecięć pary A dla czarnego wielokąta



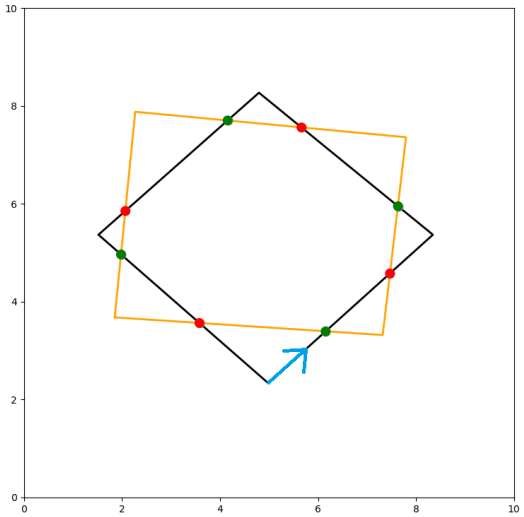
Rysunek 11: Klasyfikacja przecięć pary A dla pomarańczowego wielokąta



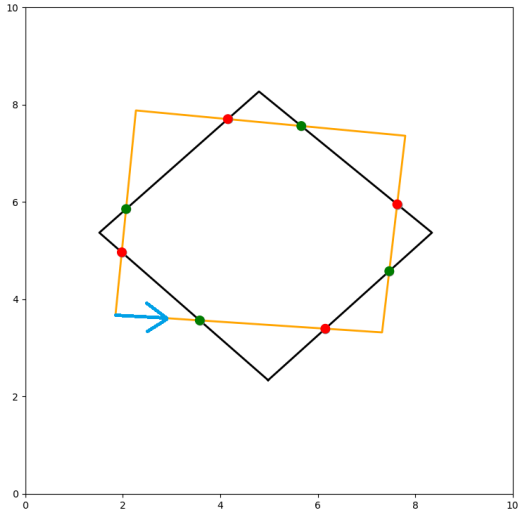
Rysunek 12: Klasyfikacja przecięć pary B dla czarnego wielokąta



Rysunek 13: Klasyfikacja przecięć pary B dla pomarańczowego wielokąta



Rysunek 14: Klasyfikacja przecięć pary C dla czarnego wielokąta



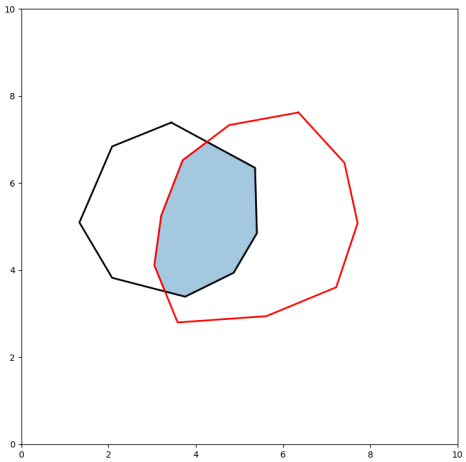
Rysunek 15: Klasyfikacja przecięć pary C dla pomarańczowego wielokąta

Algorytm w każdym przypadku poprawnie wyznaczył punkty wejścia i wyjścia.

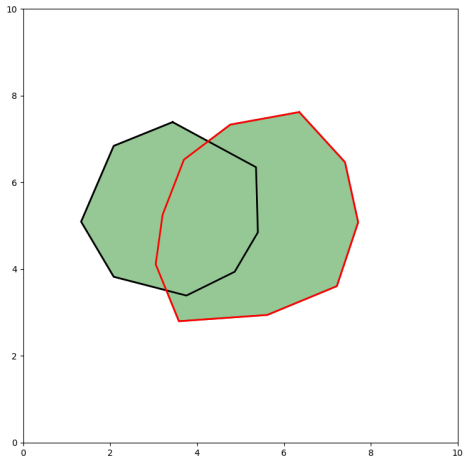
Jak można zauważyć, w każdym przypadku klasyfikacja dla wielokąta pomarańczowego była przeciwna do klasyfikacji dla wielokąta czarnego. Wynika to z faktu, że oba wielokąty były zadane w tym samym kierunku (przeciwnie do ruchu wskazówek zegara). Gdyby wielokąty były zadane w różnych kierunkach, to wówczas klasyfikacja wierzchołków wyglądałaby identycznie dla obu wielokątów - jest to widoczne w przypadku figur przedstawionych w sekcji 3.1.

4.2. Iloczyn i suma

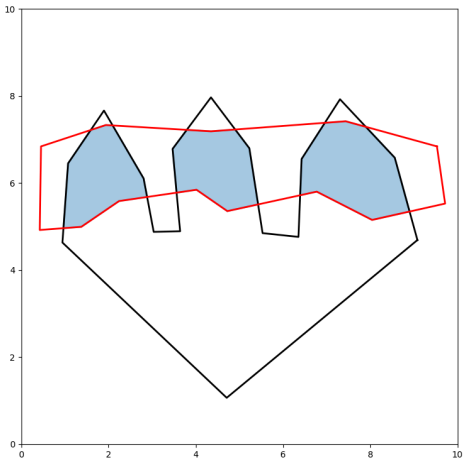
Na rysunkach 16-25 przedstawiono wyznaczony iloczyn i sumę dla każdej analizowanej pary wielokątów. Wielokąty zostały zaznaczone kolorem czarnym i czerwonym, obszar iloczynu kolorem niebieskim, a obszar sumy kolorem zielonym.



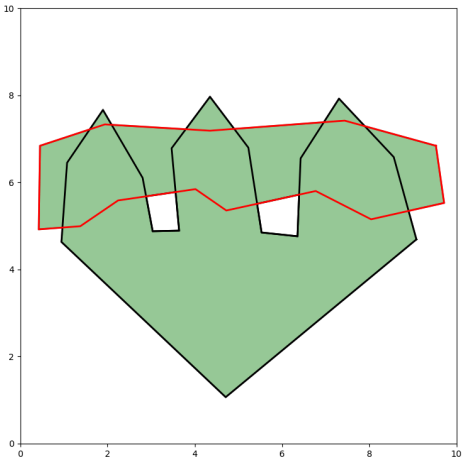
Rysunek 16: Iloczyn wielokątów pary A



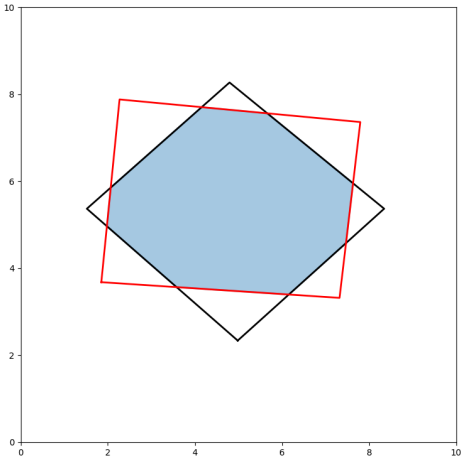
Rysunek 17: Suma wielokątów pary A



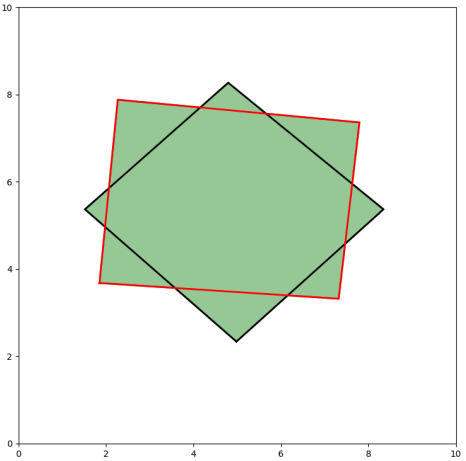
Rysunek 18: Iloczyn wielokątów pary B



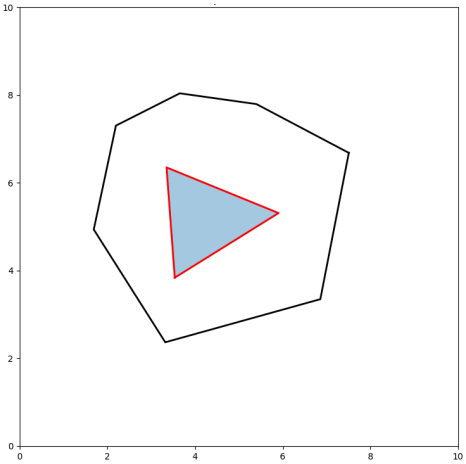
Rysunek 19: Suma wielokątów pary B



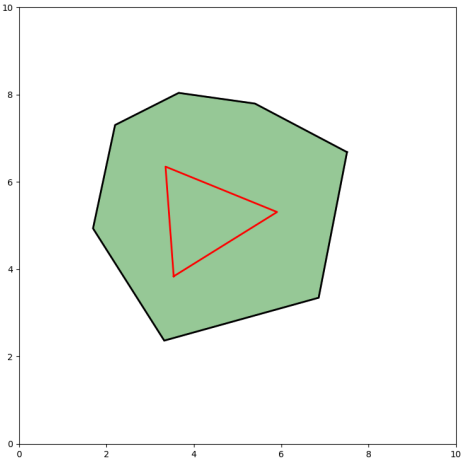
Rysunek 20: Iloczyn wielokątów pary C



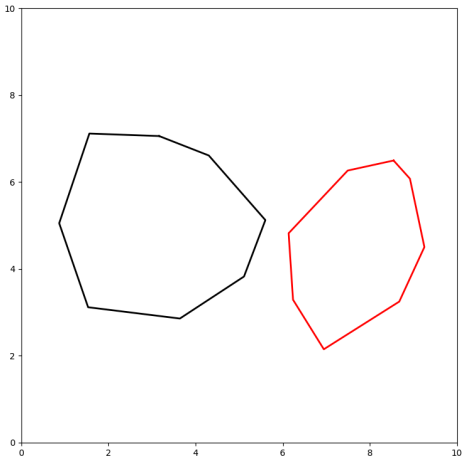
Rysunek 21: Suma wielokątów pary C



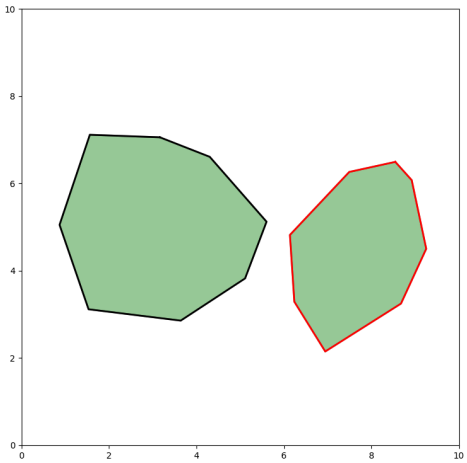
Rysunek 22: Iloczyn wielokątów pary D



Rysunek 23: Suma wielokątów pary D



Rysunek 24: Iloczyn wielokątów pary E

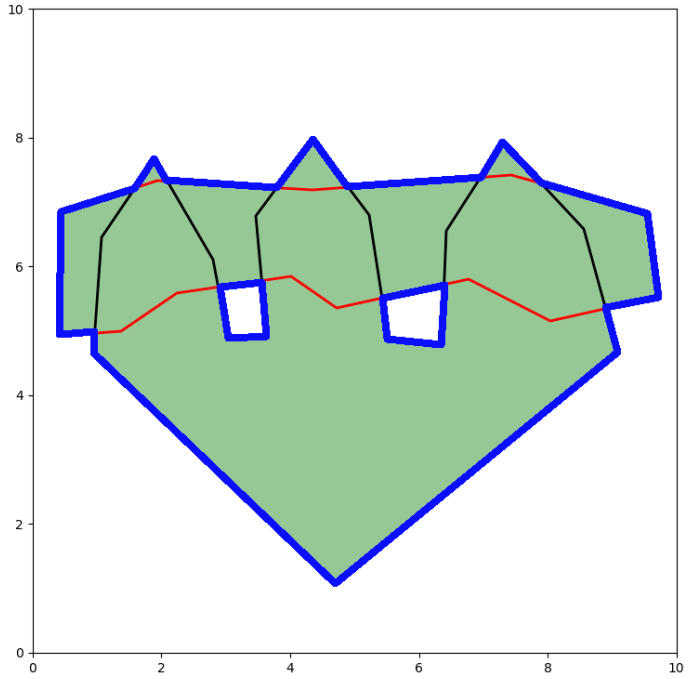


Rysunek 25: Suma wielokątów pary E

Otrzymane wyniki są zgodne z oczekiwaniami:

- dla par A i C algorytm poprawnie wyznaczył wielokąt będący iloczynem,
- iloczyn pary B zawiera trzy różne wielokąty,
- w przypadku pary D, iloczynem jest wielokąt wewnętrzny, a sumą wielokąt zewnętrzny,
- dla pary E iloczynem jest zbiór pusty, a sumą oba wielokąty.

W przypadku pary B, otrzymanym wynikiem sumy były trzy wielokąty - z czego dwa z nich znajdowały się wewnątrz trzeciego, co jest pokazane na rysunku 26. Niebieskim kolorem zaznaczono krawędzie zwróconych wielokątów, a kolorem zielonym wyznaczoną sumę.



Rysunek 26: Zwrócone wielokąty dla sumy pary B

Wielokąty wewnętrzne reprezentują „dziury” - ich wnętrze nie należy do sumy. W celu poprawnego wyznaczenia sumy należy odjąć od obszaru wielokąta zewnętrznego obszary wielokątów wewnętrznych. Aby sprawdzić, które wielokąty są wewnętrzne, a które zewnętrzne, można użyć metody ray-castingu.

5. Wnioski

- Zaimplementowany algorytm Greinera-Hormanna działa poprawnie dla wielokątów zgodnych z przyjętymi założeniami.
- Algorytm zwraca prawidłowy wynik również w przypadku gdy wielokąty nie przecinają się.
- Ponieważ złożoność czasowa algorytmu po wyznaczeniu przecięć jest liniowa, to jest on bardzo efektywnym sposobem na wyznaczenie sumy i iloczynu wielokątów.
- Złożoność czasowa zaimplementowanego algorytmu wynosi O(n*m), gdzie n i m są liczbami boków wielokątów. Efektywność można poprawić stosując szybsze wyznaczanie przecięć odcinków (np. algorytm zamiatania).

6. Dokumentacja

Program został napisany w języku Python. Aby uruchomić program wymagany jest dostęp do bibliotek **NumPy** oraz **Matplotlib**.

6.1. Główne moduły programu

- **Wyznaczanie i wstawianie przecięć krawędzi** – funkcje `intersects`, `segment_intersection`, `param_intersection`, `find_intersections`, `insert_intersections`.
- **Określenie położenia punktu względem wielokąta** - funkcja `in_polygon`.
- **Oznaczanie punktów wejścia i wyjścia** – funkcja `mark_intersections`.
- **Konstruowanie wyników** – funkcje `common_part` (iloczyn) oraz `union` (suma).
- **Zwracanie iloczynu oraz sumy** – funkcja `greiner_hormann`.
- **Interaktywne rysowanie dwóch wielokątów** – funkcja `draw_two_polygons`.
- **Wizualizacja** – `show_common_part`, `show_union`.
- **Animacje etapów algorytmu**:
  - `animate_entry_exit_marking` – animacja oznaczania punktów wejścia i wyjścia na wielokącie,
  - `animate_common_part` – animacja konstruowania iloczynu.
  - `animate_union` – animacja konstruowania sumy.

6.2. Interfejs użytkownika

1. Po uruchomieniu programu pojawia się interaktywne okno rysowania – użytkownik klika lewym przyciskiem myszy, aby dodać wierzchołki pierwszego wielokąta. Po zbliżeniu się do pierwszego punktu wielokąt zamyka się - jest to sygnalizowane zmianą koloru wielokąta na czerwony. Następnie w analogiczny sposób rysowany jest drugi wielokąt.

2. Po zakończeniu rysowania i zamknięciu okna wyświetlane są:
   - obraz przedstawiający iloczyn wielokątów,
   - obraz przedstawiający sumę wielokątów,
   - animacje oznaczania przecięć jako wejście lub wyjście na obu wielokątach,
   - animacja konstruowania iloczynu,
   - animacja konstruowania sumy.

   Jeśli wielokąty się nie przecinają, animacje nie są wyświetlane.

W konsoli wypisywane są zadane wielokąty oraz ich iloczyn i suma. Każda figura wypisywana jest jako lista współrzędnych wierzchołków.

7. Bibliografia

1. Günther Greiner, Kai Hormann (1998). „Efficient clipping of arbitrary polygons”. *ACM Transactions on Graphics*, 17(2), 71–83.
2. https://en.wikipedia.org/wiki/Point_in_polygon#Ray_casting_algorithm