Szymon Budziak
Informatyka II rok, grupa poniedziałek 14:40 - 16:10

Algorytmy geometryczne, laboratorium 3 - sprawozdanie

1.Opis ćwiczenia

Zadaniem które należy wykonać na laboratorium 3 jest implementacja oraz przetestowanie trzech algorytmów:

- algorytm sprawdzający czy dany wielokąt jest y-monotoniczny, czyli monotoniczny względem osi OY,
- algorytm klasyfikujący wierzchołki wielokąta na początkowy, końcowy, łączący, dzielący i prawidłowy,
- algorytm wykonujący triangulację wielokąta y-monotonicznego.

Również należało dostosować aplikację graficzną w taki sposób, aby istniała możliwość zadawania wielokątów przy użyciu myszki, ich zapis oraz odczyt.

2. Środowisko, biblioteki oraz użyte narzędzia

Ćwiczenie zostało wykonane w Jupyter Notebook i napisane w języku Python. Do rysowania wykresów zostało użyte dostarczone na laboratorium narzędzie graficzne, które oparte jest o bibliotekę matplotlib. Umożliwiało to również wspomniane w opisie ćwiczenia zadawanie wielokątów przy pomocy myszki. Wszystko było wykonywane na systemie operacyjnym Linux Ubuntu 20.04 oraz na procesorze Intel Core i5-7300HQ 2.50GHz.

3. Plan, sposób wykonania ćwiczenia, opisy algorytmów oraz wizualizacja ich działania na przykładach

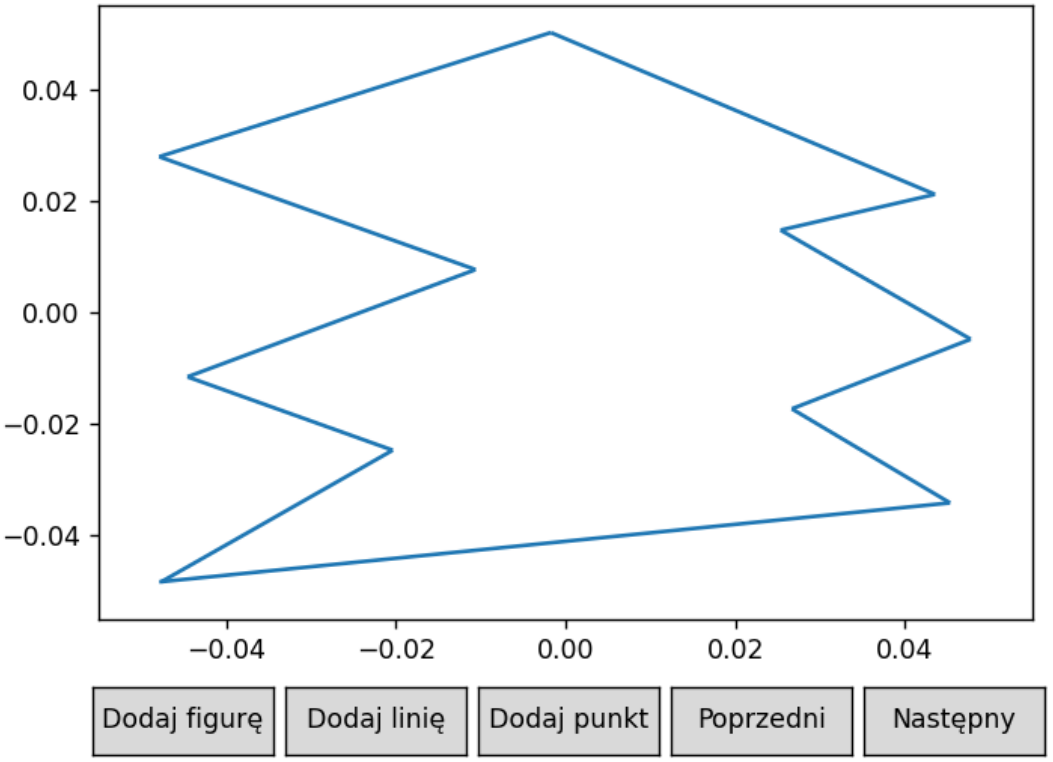
3.1 Algorytm sprawdzający y-monotoniczność wielokąta

Pierwszym algorytmem, który należało zaimplementować jest algorytm sprawdzający czy podany wielokąt jest y-monotoniczny. Sam algorytm, którego implementacja jest w jupyter notebook polega na przejściu po wszystkich punktach (wierzchołkach) przeciwnie do ruchu wskazówek zegara i sprawdzeniu czy każda trójka kolejnych punktów (aktualnie sprawdzany punkt, jego lewy sąsiad i prawy) nie tworzą wierzchołka łączącego lub dzielącego. Jeżeli którakolwiek trójka takowy tworzyła, to znaczy, że zadany wielokąt nie jest wielokątem y-monotonicznym i główna funkcja zwracała fałsz. Jeżeli jednak udało

się przejść przez wszystkie punkty bez znalezienia wierzchołków łączących lub dzielących funkcja zwraca prawdę, czyli wielokąt jest y-monotoniczny.
Algorytm jest poprawny ponieważ, znalezienie w wielokącie chociaż jednego wierzchołka łączącego lub dzielącego sprawia, że nie jest on wielokątem y-monotonicznym. Przejście po wszystkich punktach i sprawdzenie jakim wierzchołkiem jest dany punkt gwarantuje nam poprawność algorytmu.

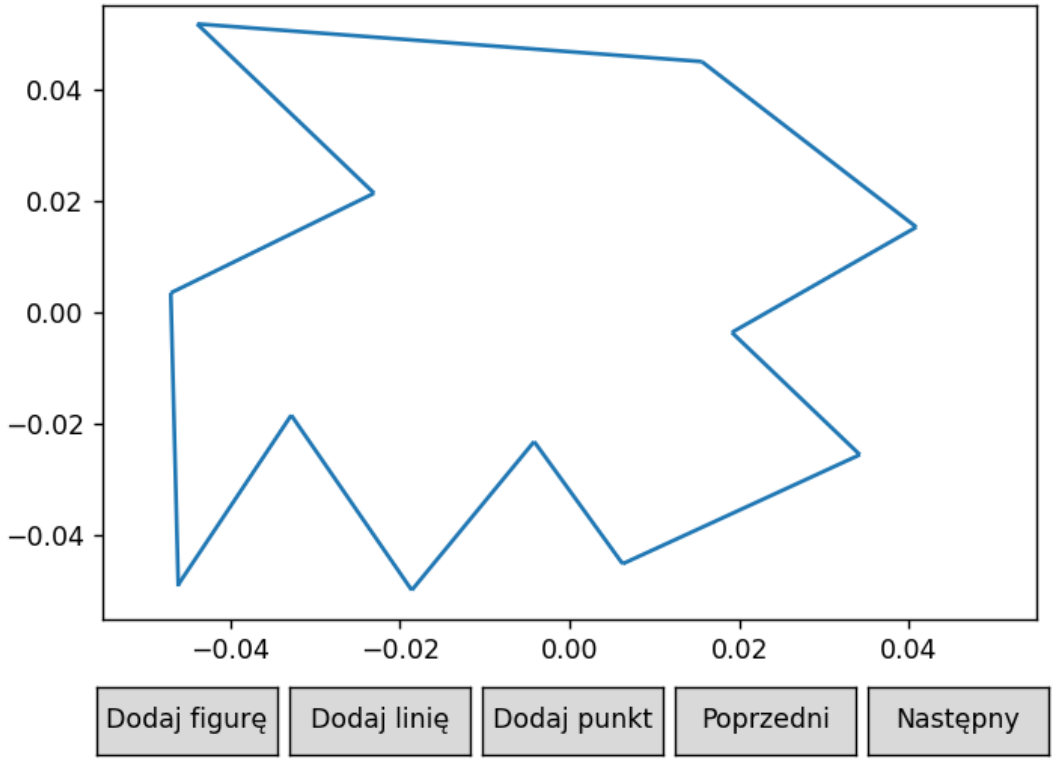
Przykład działania algorytmu:

Wykres_1: Przykład wielokąta y-monotonicznego



Algorytm zakwalifikuje taki wielokąt jako y-monotoniczny i jest to poprawna kwalifikacja.

Wykres_2: Przykład wielokąta nie y-monotonicznego



Algorytm zakwalifikuje taki wielokąt jako nie y-monotoniczny i jest to poprawna kwalifikacja.

3.2 Algorytm klasyfikujący wierzchołki wielokąta

Algorytm polega na przeglądnięciu wszystkich wierzchołków wielokąta i określeniu na podstawie wysokości sąsiadów aktualnie sprawdzanego wierzchołka oraz kąta jaki z nimi tworzy ten wierzchołek czy jest on:

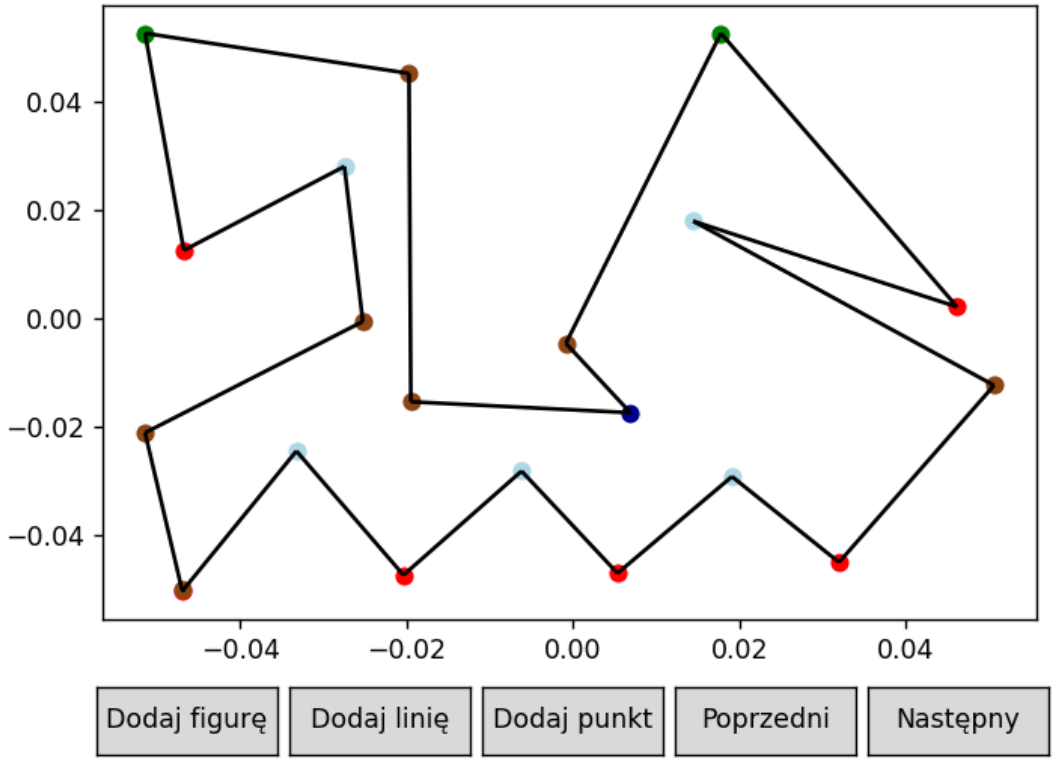
- wierzchołkiem początkowym - obaj sąsiedzi leżą poniżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest $< \pi$,
- wierzchołkiem końcowym - obaj sąsiedzi leżą powyżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest $< \pi$,
- wierzchołkiem łączącym - obaj sąsiedzi leżą powyżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest $> \pi$,
- wierzchołkiem dzielącym - obaj sąsiedzi leżą poniżej i kąt wewnętrzny tworzony przez wierzchołek i jego sąsiadów jest $> \pi$,
- wierzchołkiem prawidłowym - w pozostałych przypadkach.

Algorytm koloruje wierzchołki w sposób:

- **zielony** - wierzchołek początkowy,
- **czerwony** - wierzchołek końcowy,
- **ciemnoniebieski** - wierzchołek łączący,
- **jasnoniebieski** - wierzchołek dzielący,
- **brązowy** - wierzchołek prawidłowy.

Przykład działania algorytmu:

Wykres_3: Przykład wielokąta o wszystkich typach wierzchołków



Algorytm w poprawny sposób określił każdy typ wierzchołka. Na tym wykresie są przedstawione wszystkie typy wierzchołków: początkowy, końcowy, łączący, dzielący i prawidłowy.

3.3 Algorytm wykonujący triangulację wielokąta

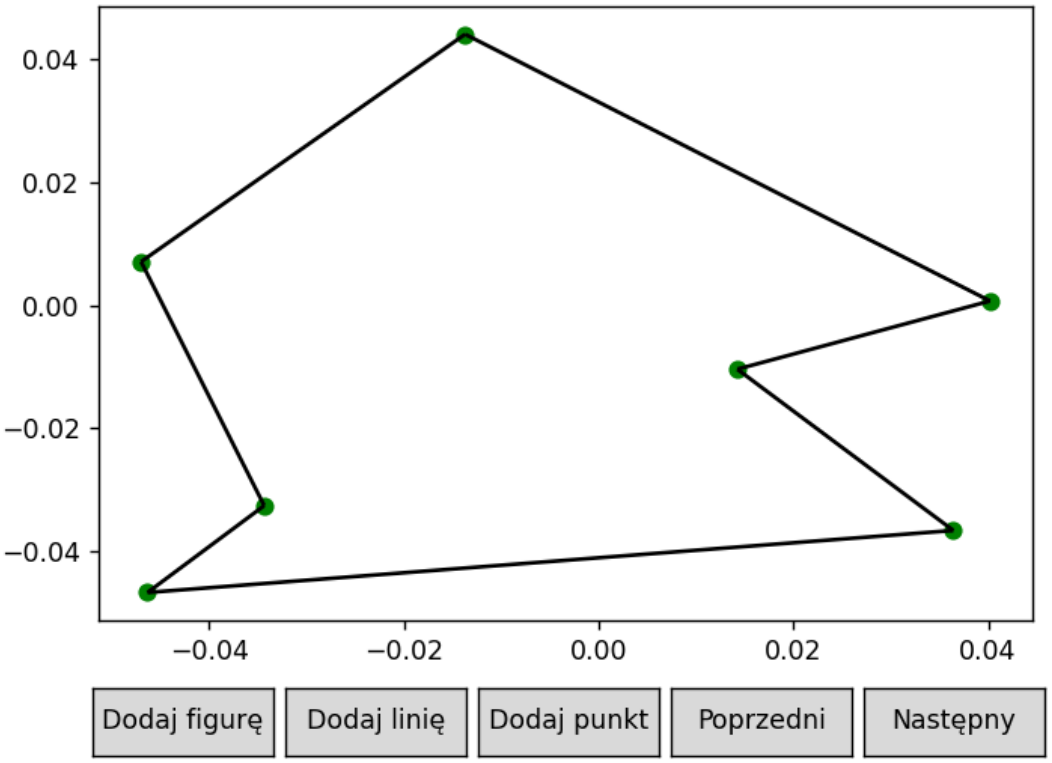
Początek algorytmu polega na sprawdzeniu czy wielokąt jest y-monotoniczny. Sprawdzenie odbywa się w dokładnie taki sam sposób jak jest to opisane w punkcie 3.1. Następnym krokiem jest podzielenie wierzchołków na te które leżą w lewym łańcuchy oraz na te które leżą w prawym łańcuchu. Najniższy wierzchołek zawsze ląduje do prawego łańcucha natomiast najwyższy do lewego. Następnie odbywa się sortowanie wierzchołków po współrzędnej y oraz odwrócenie otrzymanej

listy, tak aby pierwszym elementem listy był wierzchołek o najwyższej współrzędnej y. Pierwszy i drugi element listy po posortowaniu (odpowiednio o najwyższych współrzędnych) lądują na stos.
Przeglądamy wszystkie pozostałe wierzchołki z posortowanej listy i dodajemy krawędzie tworzące trójkąty. Jednak dodawanie krawędzi ma następujące warunki:

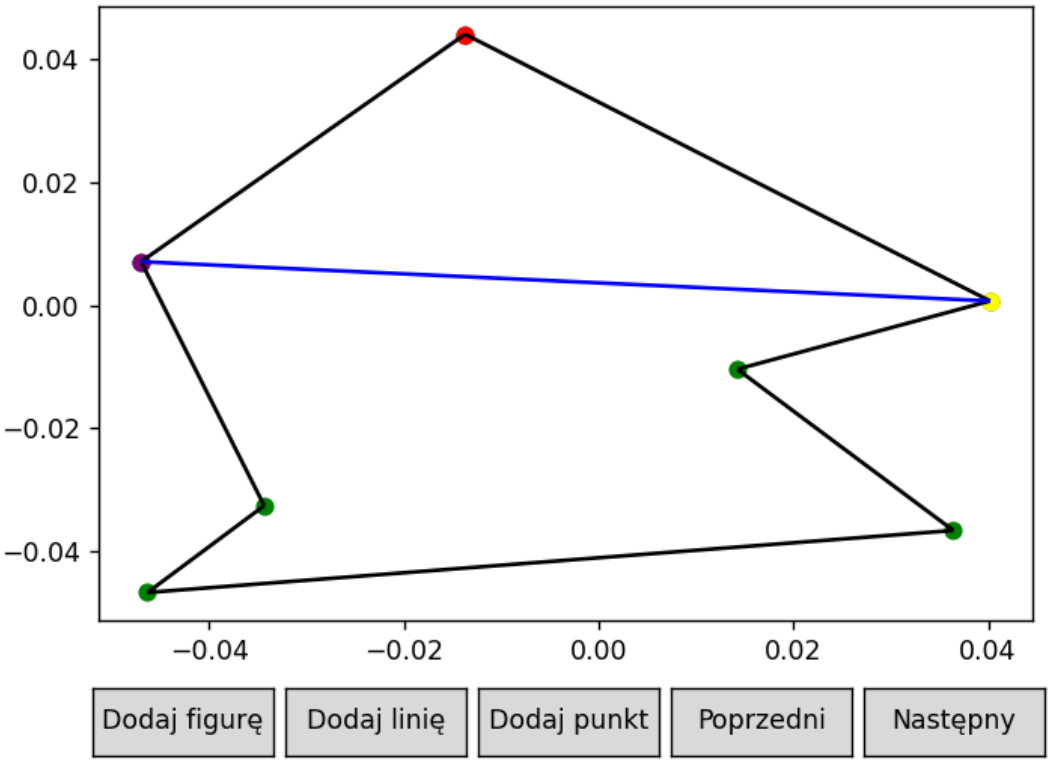
- jeśli aktualnie rozpatrywany wierzchołek znajduje się na innym łańcuchu niż szczyt stosu, to łączymy go z wszystkimi wierzchołkami które znajdują się na stosie i nie są jego sąsiadami, po wykonaniu całej operacji na stosie zostawiamy dwa ostatnio analizowane wierzchołki,
- jeśli aktualnie rozpatrywany wierzchołek znajduje się na tym samym łańcuchu co szczyt stosy, to rozpatrujemy dwa przypadki:
    - utworzony trójkąt należy do wielokąta oraz szczyt stosu nie jest sąsiadem aktualnie rozpatrywanego wierzchołka, to usuwamy wierzchołek ze stosu a wierzchołki łączymy krawędzią,
    - utworzony trójkąt nie należy do wielokąta, to umieszczamy badane wierzchołki na stosie.

Algorytm zwraca dwa elementy: przekątne, które zostały dodane w czasie algorytmu oraz sceny, które umożliwiają odtworzenie wykresu i działania algorytmu.
Dodane przekątne mają na wykresie kolor niebieski, wierzchołki które nie są rozpatrywane kolor zielony, aktualnie rozpatrywany wierzchołek kolor żółty, wierzchołki, które znajdują się aktualnie na stosie kolor czerwony i wierzchołek z którym połączony jest w danym momencie aktualny wierzchołek kolor fioletowy.
Pod pokazanym wykresem znajdują się również informacje czy dany wielokąt jest wielokątem y-monotonicznym, liczba przekątnych jaką posiada wielokąt po działaniu algorytmu oraz same przekątne (ich współrzędne).

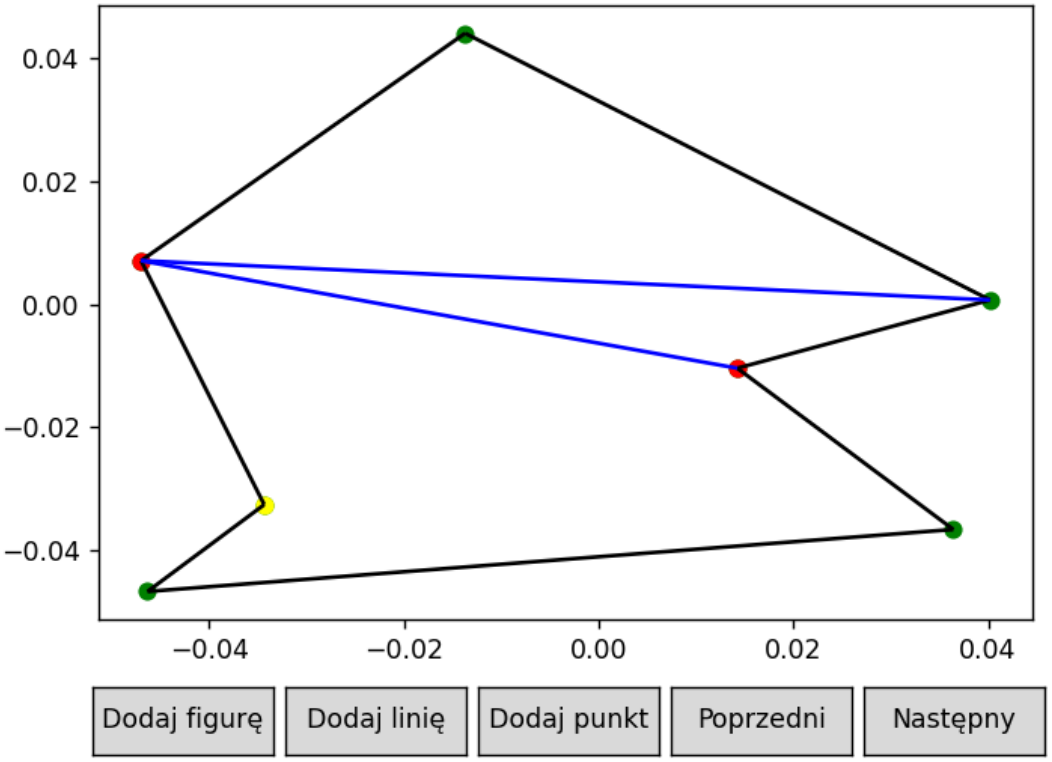
Przykład działania algorytmu (niektóre kroki):
Wykres_4: Przykład wielokąta, wszystkie wierzchołki są koloru zielonego



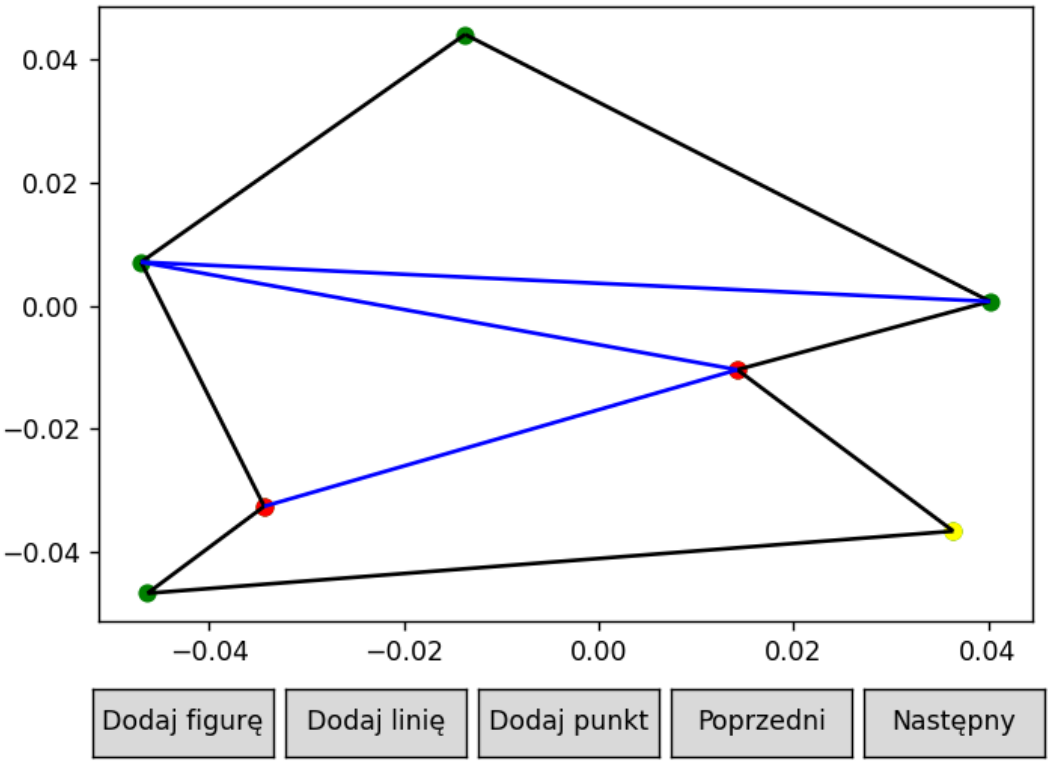
Wykres_5: Triangulacja po kolejnym kroku (nie jest to bezpośredni krok po wcześniejszym wykresie), aktualny wierzchołek - żółty, przekątna - niebieski, wierzchołek na stosie - czerwony, wierzchołek z którym łączony jest aktualny wierzchołek - fioletowy



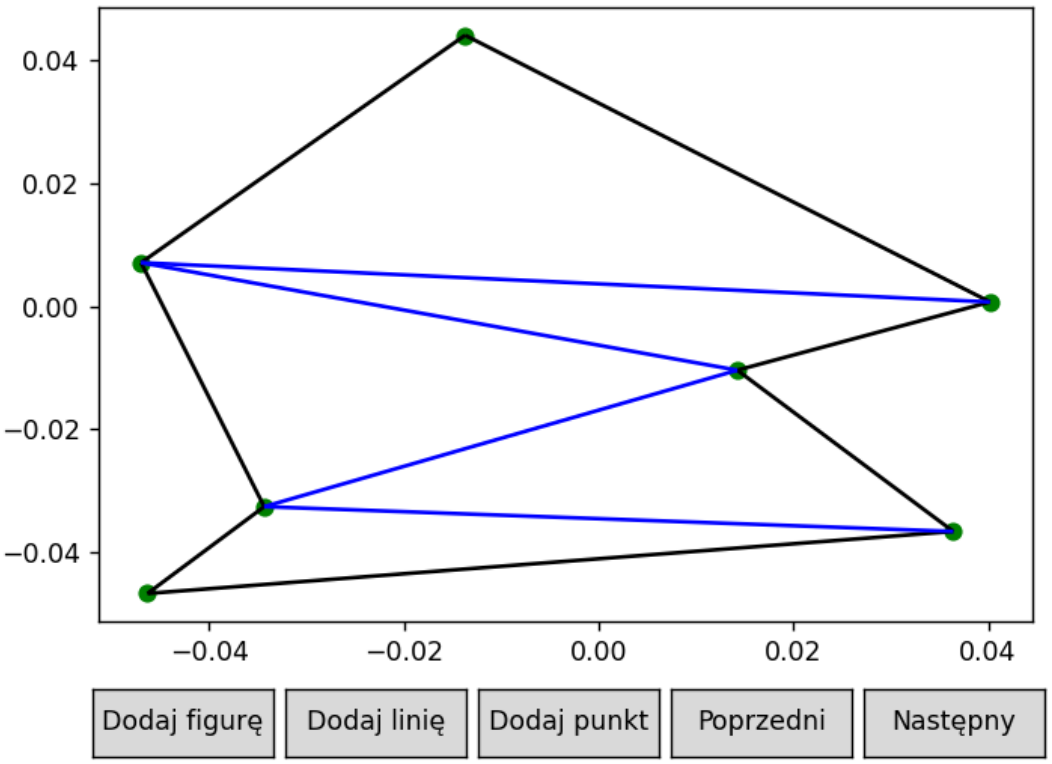
Wykres_6: Triangulacja po kolejnym kroku (nie jest to bezpośredni krok po wcześniejszym wykresie), aktualny wierzchołek - żółty, przekątne - niebieski, wierzchołki na stosie - czerwony,



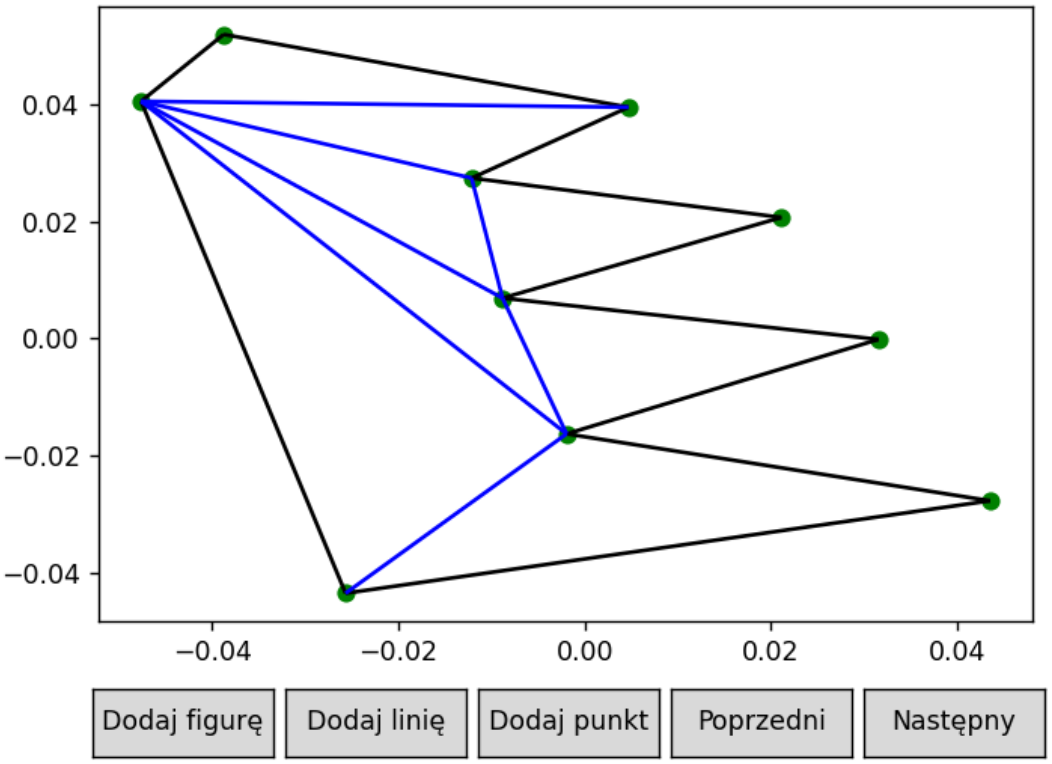
Wykres_7: Triangulacja po kolejnym kroku (nie jest to bezpośredni krok po wcześniejszym wykresie), aktualny wierzchołek - żółty, przekątne - niebieski, wierzchołki na stosie - czerwony,



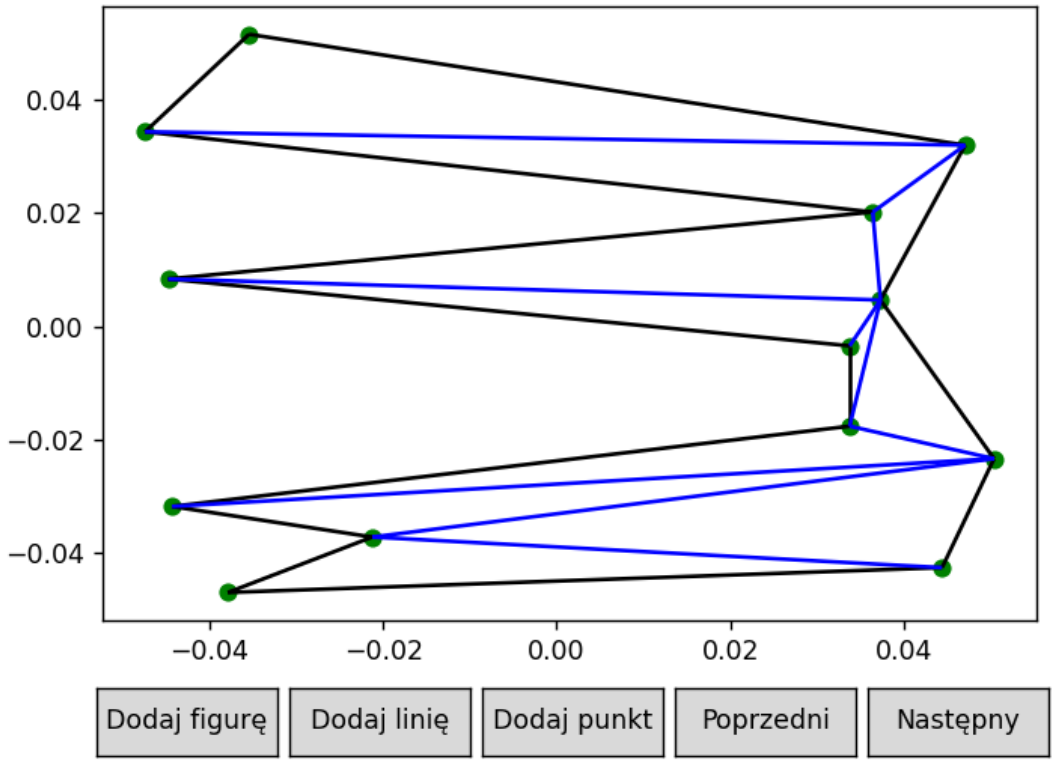
Wykres_8: Wielokąt po skończonej triangulacji, wszystkie wierzchołki - zielony, przekątne - niebieski



Wykres_9: Inna przykładowa triangulacja (efekt końcowy)



Wykres_10: Inna przykładowa triangulacja (efekt końcowy)



4. Wnioski

Wszystkie algorytmu po ich przetestowaniu na powyższych zbiorach (oraz innch, które są w jupyter notebook) działają poprawnie i nie zauważono w nich żadnych błędów w działaniu. Można zatem stwierdzić, że są one zaimplementowane i działają w sposób poprawny.