Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

Algorytmy geometryczne

Ćwiczenie 3

Autor:
Andrzej Zaborniak
Informatyka, rok II gr. 4

24.11.2022 r.

1 Specyfikacja techniczna

Parametry techniczne komputera na którym zostało wykonane ćwiczenie:
Procesor: Intel® Core™ i7-5600U CPU @ 2.60GHz × 4
Karta graficzna: Mesa Intel® HD Graphics 5500 (BDW GT2)
Pamięć RAM: 8,0 GB
System operacyjny: Ubuntu 22.04.1 LTS
Wersja GNOME: 42.4
Użyty język programowania: Python 3.10.6
Wykorzystany program: Jupyter Notebook

1.1 Narzędzie graficzne

W ćwiczeniu użyty został program , który był rekomendowany na laboratoriach. Punkty tworzące wielokąt zostały wprowadzane ręcznie za pomocą myszki. Aby umożliwić późniejsze operacje na wielokącie została w nim dopisana funkcja: get_points_from_plot zapisująca współrzędne w postaci listy krotek umożliwiając późniejsze odczytanie punktów zaznaczonych na płaszczyźnie.

2 Cel ćwiczenia

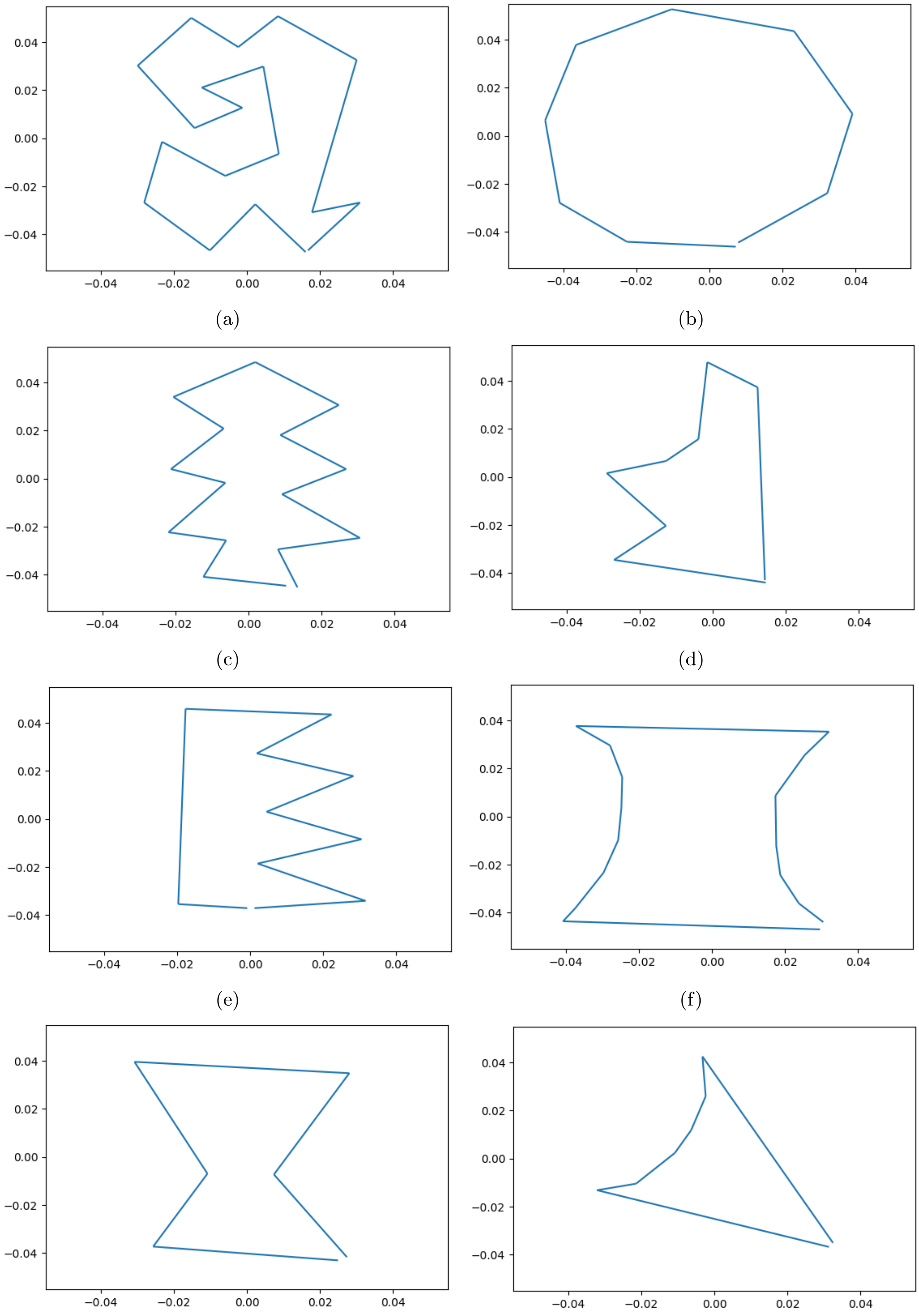
Głównym zadaniem które należało wykonać na laboratorium trzecim, była implementacja algorytmów mających na celu zaznajomienie studenta z pojęciami jak: y-monotoniczność, podział punktów w wielokącie oraz triangulacja wielokąta y-monotonicznego.

Wielokąt jest monotoniczny względem prostej l ,gdy przecięcie dowolnej prostej l' prostopadłej do l z dowolnym łańcuchem jest spójne. Przecięcie wielokąta z l' jest spójne, czyli jest odcinkiem, punktem lub jest puste. Wielokąt monotoniczny względem osi y nazywamy y-monotonicznym.

Triangulacja - technika stosowana w grafice komputerowej polegająca na rozbiciu bardziej złożonych powierzchni obiektów na trójkąty.

3 Wielokąty

Obrazki przedstawiają wielokąty na których zostały uruchomione algorytmy zaimplementowane w ćwiczeniu.



(a) (b) (c) (d) (e) (f) (g) (h)

Rysunek 1

4 Ćwiczenia

4.1 Y-monotoniczność

Pierwsze ćwiczenie polegało na implementacji funkcji sprawdzającej czy wielokąt jest y-monotoniczny. Wszystkie wielokąty testowane w programie zostały wprowadzone za pomocą myszki w kierunku przeciwnym do ruchu wskazówek zegara.

Zaimplementowana procedura sprawdzania y-monotoniczności:

W zbiorze punktów na samym początku szukane są dwa skrajne punkty, jeden z największą wartością Y oraz drugi z najmniejszą wartością Y. Względem nich wielokąt jest dzielony na dwie cześć, przyjmijmy: część prawą oraz lewą. Algorytm przechodzi przez poszczególne punkty w części lewej od najwyżej do najniżej położonego punkty, jeśli podczas tego przejścia następujące po sobie wartości Y są rosnące to znaczy że wielokąty nie jest y-monotoniczny. Analogicznie dla części prawej tylko przechodzimy od najniżej do najwyżej położonego punkty, a następujące po sobie wierzchołki nie mogą być malejące.

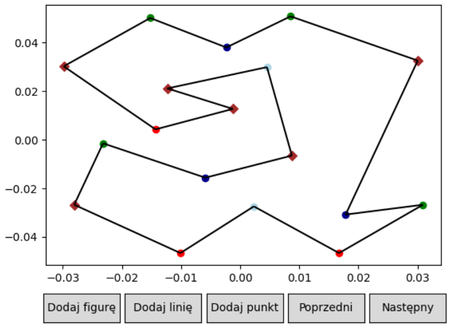
Algorytm dla rysunku 1a) poprawnie stwierdził, iż wielokąt nie jest y-monotoniczny, przy czym dla pozostałych przykładów również popranie sklasyfikował wielokąty do y-monotonicznych. Poprawność algorytmu potwierdza druga część zadania którą jest klasyfikacja punktów, ponieważ dla rysunków od 1b) do 1h) nie zostały znalezione punkty łączące oraz dzielące.

4.2 Podział wierzchołków

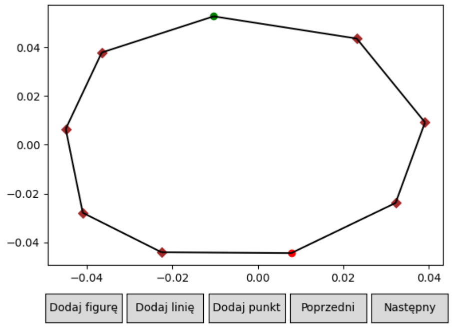
Algorytm miał na celu przeglądnięcie wszystkich wierzchołków wielokąta, a następnie ich sklasyfikowanie na podstawie wysokości jego sąsiadów, a także kąta jaki wraz z nimi tworzy. Podział był następujący:

- wierzchołek początkowy - obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny $< \pi$,
- wierzchołek końcowy - gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny $< \pi$,
- wierzchołek łączący, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny $> \pi$,
- wierzchołek dzielący - obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny $> \pi$,
- wierzchołek prawidłowy - pozostałe przypadki.

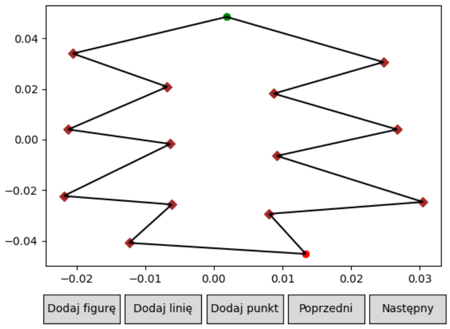
Algorytm koloruje wierzchołki w następujący sposób:
- zielony - wierzchołek początkowy,
- czerwony - wierzchołek końcowy,
- ciemnoniebieski - wierzchołek łączący,
- jasnoniebieski - wierzchołek dzielący,
- brązowy w kształcie diamentu - wierzchołek prawidłowy.



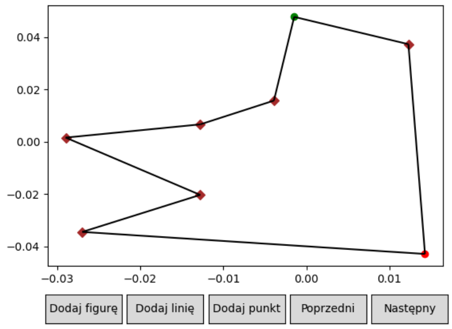
(a)



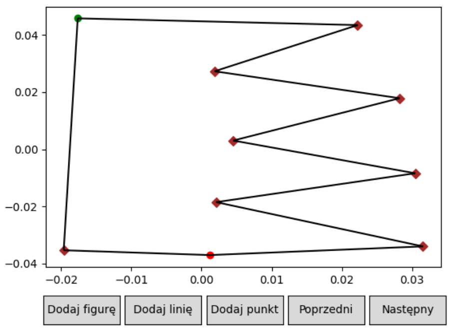
(b)



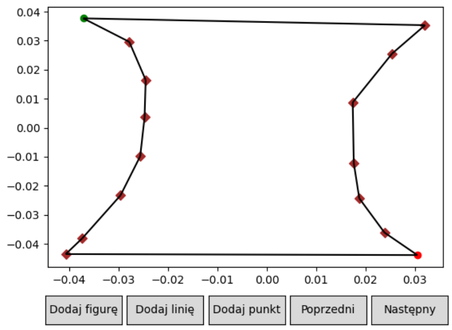
(c)



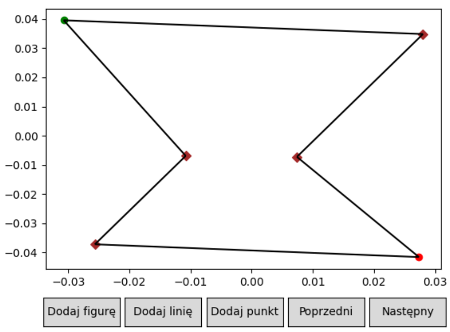
(d)



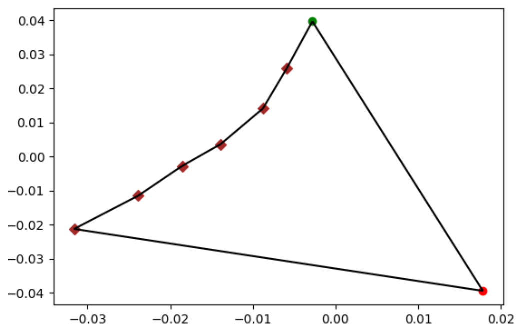
(e)



(f)



(g)



(h)

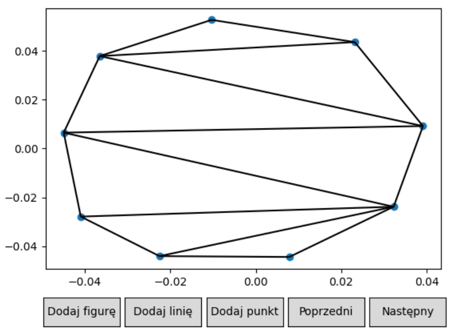
Rysunek 2

Wszystkie wierzchołki figur zostały sklasyfikowane poprawnie, zgodnie z wykładem dla wielokątów y-monotonicznych są tylko pojedyncze wierzchołki początkowe (najwyższe) i końcowe (najniższe) oraz nie ma wierzchołków dzielących i łączących. Jedynym wielokątem w którym zostały one wykryte jest na Rysunku 2a) i jest to wielokąt niemonotoniczny względem osi Y.

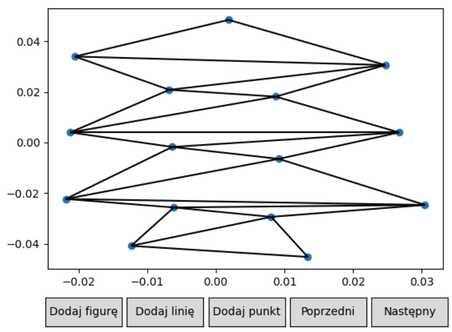
4.3 Triangulacja wielokątów monotonicznych

Triangulacja jest techniką stosowaną w grafice komputerowej, polegającą na rozbiciu bardziej złożonych figur na trójkąty.

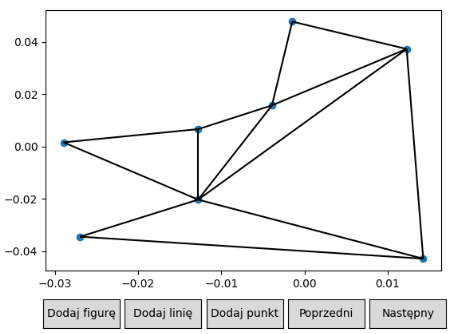
Procedura triangulacji:
1. Znajdujemy najwyżej oraz najniżej położone punkty wielokąta względem których dzielimy go na dwa łańcuchy: prawy, lewy,
2. Sortujemy wierzchołki względem osi Y,
3. Wkładamy dwa pierwsze wierzchołki na stos,
4. Jeśli kolejny wierzchołek należy do innego łańcucha niż wierzchołek stanowiący szczyt stosu, to możemy go połączyć ze wszystkimi wierzchołkami na stosie. Na stosie zostają dwa wierzchołki, które były „zamiatane" ostatnie,
5. Jeśli kolejny wierzchołek należy do tego samego łańcucha co wierzchołek ze szczytu stosu to dodajemy go do listy krawędzi (jest to krawędź zewnętrzna wielokąta), a następnie analizujemy kolejne trójkąty, jakie tworzy dany wierzchołek z wierzchołkami zdejmowanymi ze stosu:
5a) Jeśli trójkąt należy do wielokąta, to usuwamy wierzchołek ze szczytu stosu,
5b) W przeciwnym wypadku umieszczamy badane wierzchołki na stosie.



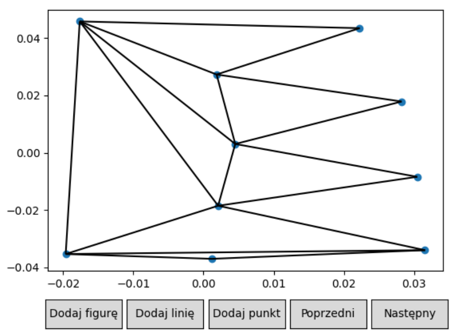
(a)



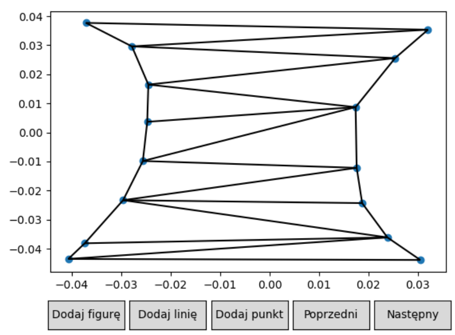
(b)



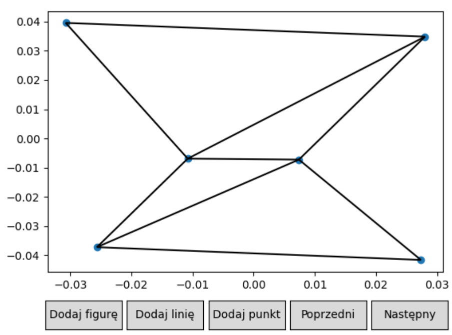
(c)



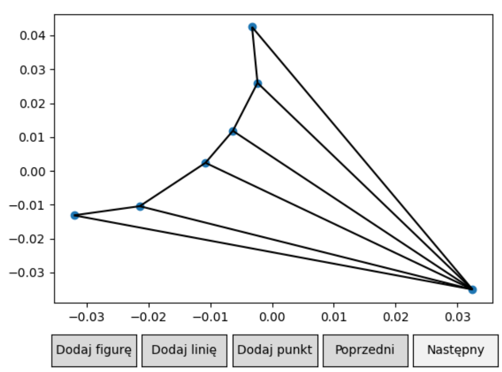
(d)



(e)



(f)

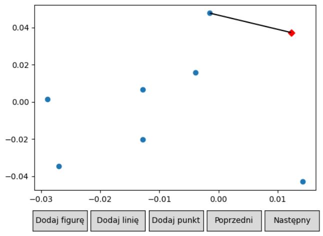


(g)

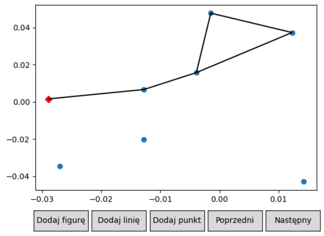
Rysunek 3

4.4 Wizualizacja działania algorytmu

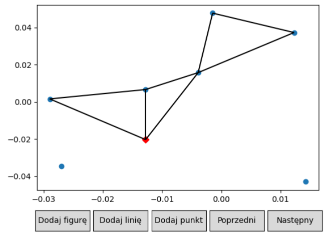
Więcej wizualizacji można znaleźć w pliku Jupyter Notebook, tutaj zostanie przedstawiona jedynie część pokazująca działanie algorytmu.



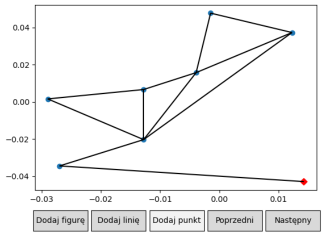
(a)



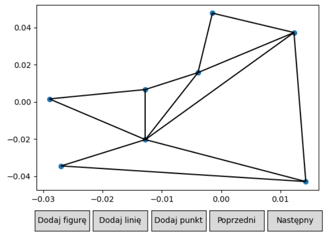
(b)



(c)

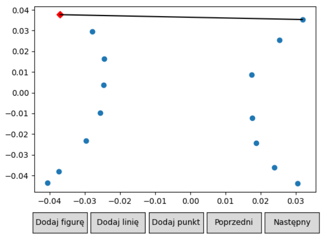


(d)

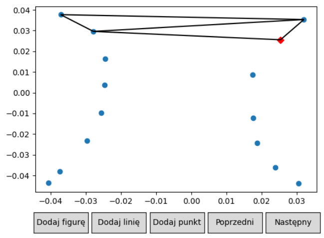


(e)

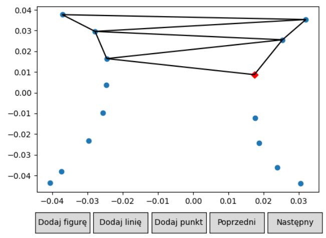
Rysunek 4



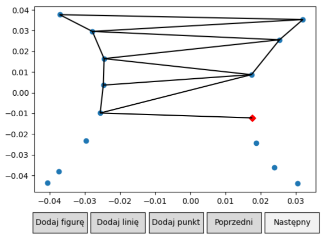
(a)



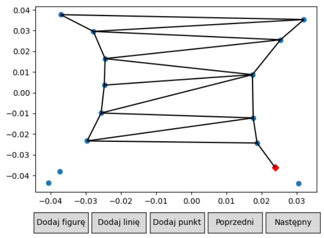
(b)



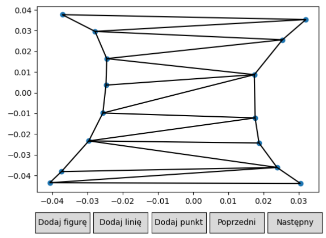
(c)



(d)

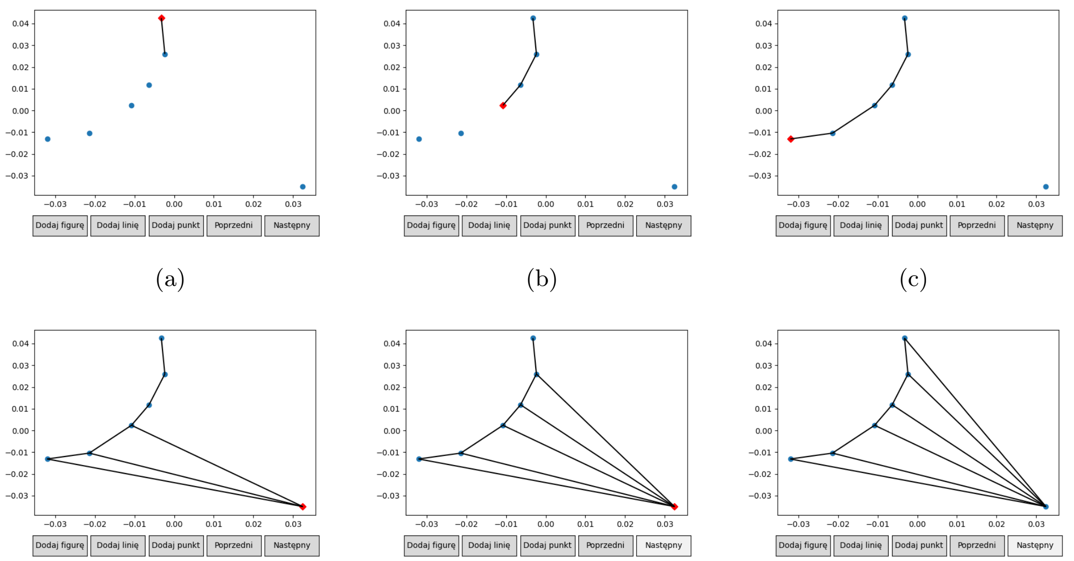


(e)



(f)

Rysunek 5



(a) (b) (c)

(d) (e) (f)

Rysunek 6

Struktura w której przechowany został wielokąt przed triangulacją jest w postaci uporządkowanej listy punktów, których kolejność wskazuje kolejne boki w wielokącie w kierunku odwrotnym do ruchu wskazówek zegara, zaś po triangulacji przekątne jak i boki wielokąta są w postaci nieuporządkowanej listy par krotek. Krotki odpowiadają odpowiednio indeksom punktów na płaszczyźnie. Jedna krotka przechowuje (punkt 1, punkt 2), gdzie punkt 1 jest indeksem wierzchołka w uporządkowanej liście, czyli krotka jest odpowiednio krawędzią w figurze.
Wybrałem tę strukturę z uwagi na prostotę oraz przejrzystość dzięki której można w łatwy sposób przeglądnąć wszystkie krawędzie w figurze po triangulacji oraz jesteśmy w stanie zaoszczędzić pamięć komputera z uwagi iż trzymamy indeksy punktów a nie same punkty.

5 Wnioski

Zaimplementowane algorytmy po przetestowaniu na wyżej przedstawionych wielokątach dały prawidłowe wyniki, dzięki czemu możemy stwierdzić, że działają poprawnie, a użyta struktura danych zaoszczędza ilość użytej pamięci w komputerze potrzebnej do przechowania krawędzi wielokąta.