Algorytmy geometryczne

Sprawozdanie z ćwiczeń 3.

Joanna Kulig, grupa 5

Dane urządzenia, na którym wykonano ćwiczenie:

Procesor: 1,4GHz Czterordzeniowy procesor Intel Core i5
Komputer z systemem macOS Monterey
Środowisko: jupyter notebook
Do rozwiązania ćwiczenia wykorzystano język Python wraz z bibliotekami numpy i matplotlib

1. Cel ćwiczenia.

Ćwiczenie polegało na implementacji algorytmu, który służy do triangulacji wielokąta y-monotonicznego. W tym celu należało również zaimplementować następujące funkcje:

1. sprawdzającą, czy wielokąt jest y-monotoniczny,
2. kolorującą wierzchołki zgodnie z podanymi warunkami.

2. Wygenerowanie danych.

W ćwiczeniu umożliwiono wczytanie dowolnego wielokąta przy wykorzystaniu interfejsu graficznego biblioteki matplotlib.

Na wejściu wczytujemy kolejne punkty wielokąta w kierunku przeciwnym do ruchu wskazówek zegara.

3. Zaimplementowane algorytmy.

Sprawdzanie y-monotoniczności

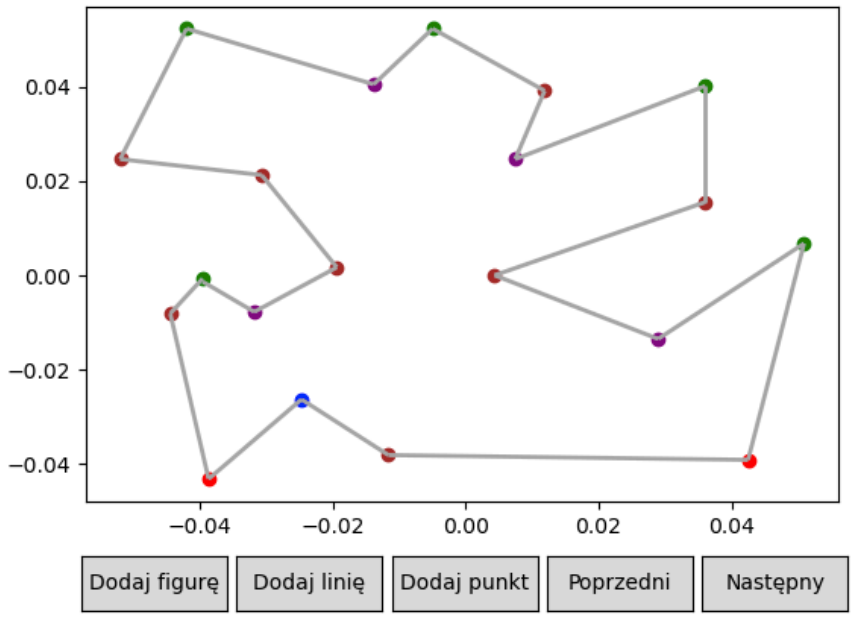
Algorytm polegał na sprawdzeniu kolejnych punktów należących do zadanego wielokąta i zliczeniu takich, których obydwoje sąsiedzi mają mniejszą wartość współrzędnej y. Jeśli istnieje tylko jeden taki wierzchołek, to wielokąt jest y-monotoniczny.

Kolorowanie wierzchołków

Wierzchołki są kolorowane według podanych warunków:

(1) Początkowy (zielony) – obaj sąsiedzi poniżej, kąt wewnętrzny < π,
(2) Końcowy (czerwony) – obaj sąsiedzi powyżej, kąt wewnętrzny < π,
(3) Łączący (purpurowy) – obaj sąsiedzi powyżej, kąt wewnętrzny > π ,
(4) Dzielący (niebieski) – obaj sąsiedzi poniżej, kąt wewnętrzny > π,
(5) Prawidłowy (brązowy) – w pozostałych przypadkach.

Algorytm rozpatruje kolejne trzy punkty należące do wielokąta zgodnie z kolejnością, w której występują. Następnie przypisuje każdemu punktowi kolor zgodnie z powyższymi warunkami. Do obliczania kąta wykorzystano metodę obliczania wyznacznika 3x3 własnej implementacji.

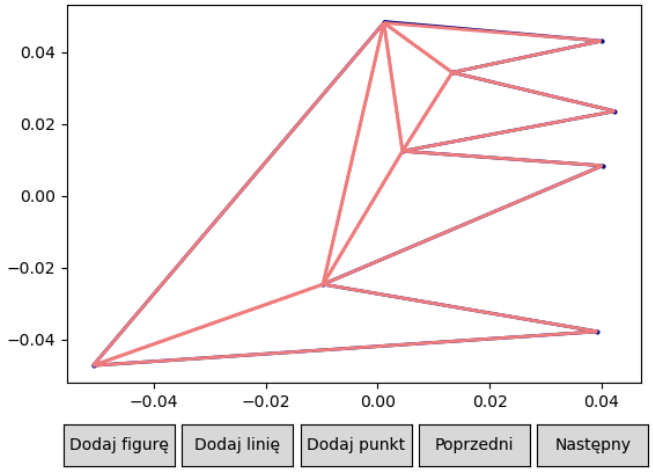


Wykres 1. Wynik działania funkcji przypisującej wierzchołkom określone kolory dla testowego wielokąta

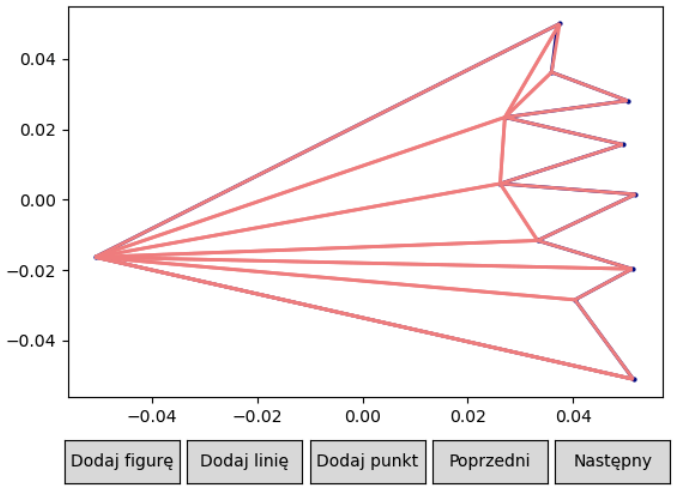
Triangulacja wielokąta monotonicznego

Na początku algorytm weryfikuje, czy zadany wielokąt jest y-monotoniczny. Poźniej wyznacza wierzchołek startowy (najwyżej położony względem współrzędnej y) i dzieli wielokąt na dwa łańcuchy - lewy i prawy. Algorytm ,,krocząc" po tych łańcuchach tworzy tablicę punktów, które są posortowane względem współrzędnej y malejąco. Wykorzystuje do tego algorytm analogiczny do algorytmu scalającego dwie posortowane tablice w Merge Sort. W następnej kolejności przechodzi do głównego algorytmu, tworzy stos i dodaje do niego dwa najwyżej położone wierzchołki, później rozważa każdy kolejny wierzchołek. Jeśli kolejny wierzchołek znajduje się na innej gałęzi niż ostatni ze stosu to łączy go ze wszystkimi wierzchołkami na stosie. W przeciwnym wypadku, gdy znajduje się na tej samej gałęzi, sprawdzamy czy trójkąt utworzony przez aktualny wierzchołek z każdymi dwoma pozostałymi ze stosu znajduje się wewnątrz wielokąta (wykorzystuje do tego wyznacznik). Jeśli tak, to dodaje ten trójkąt. Po całej operacji ze stosu zostaje usunięty ostatni wierzchołek. Algorytm kontroluje, czy dwa wierzchołki między którymi dodaje krawędzie nie jest bokiem wielokąta. Robi to poprzez sprawdzenie różnicy między indeksami wierzchołków w liście przed uporządkowaniem.

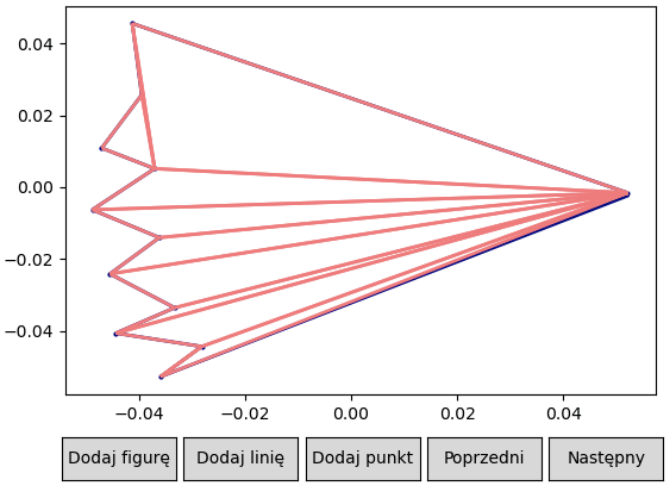
4. Wyznaczone testowe triangulacje.



Wykres 2. Triangulacja wielokąta numer 1

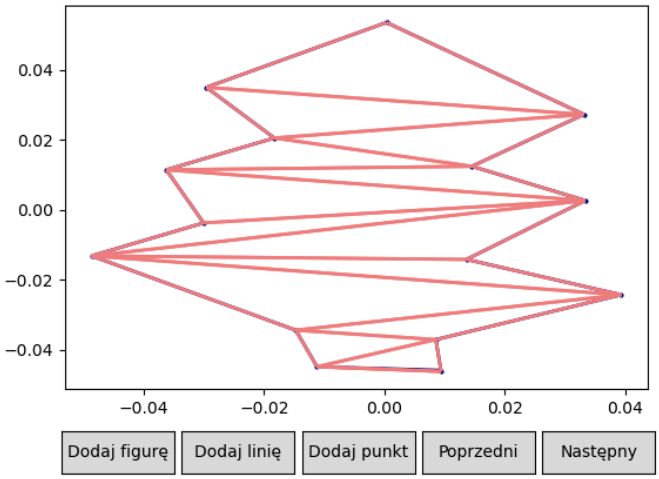


Wykres 3. Triangulacja wielokąta numer 2

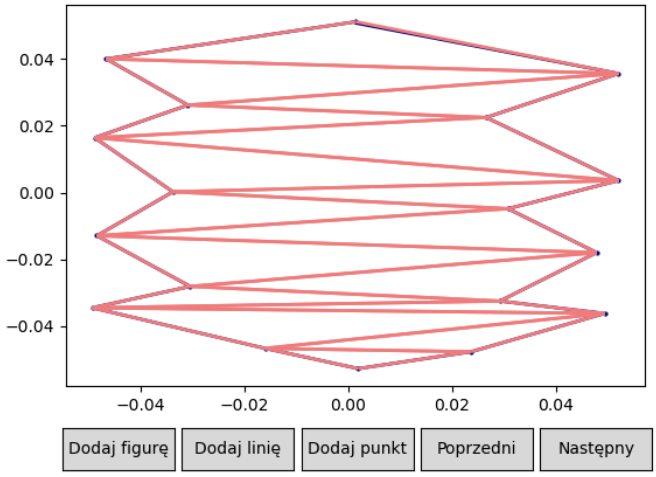


Wykres 4. Triangulacja wielokąta numer 3

Powyższe przykłady sprawdzają poprawność algorytmu w przypadku, gdy istnieje duża dysproporcja pomiędzy liczbą punktów na obu łańcuchach.



Wykres 5. Triangulacja wielokąta numer 4



Wykres 6. Triangulacja wielokąta numer 5

Triangulacje wielokąta numer 4 oraz numer 5 sprawdzają poprawność działania algorytmu w przypadku, gdy analizowane wierzchołki należą naprzemiennie do różnych łańcuchów.

5. Klasa reprezentująca punkt

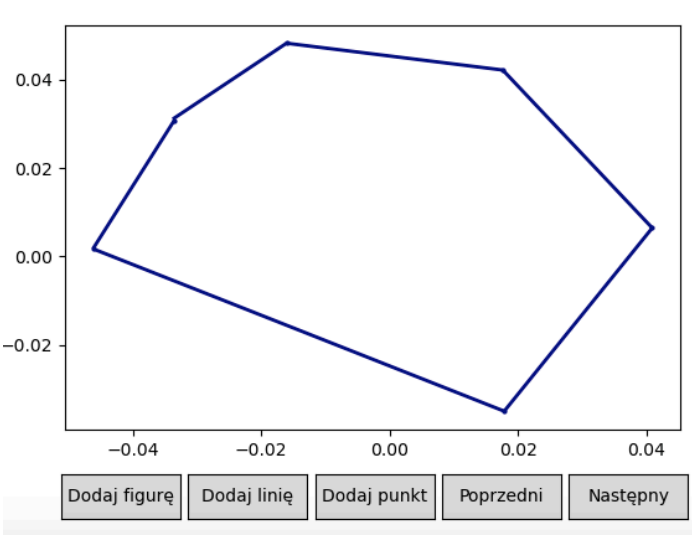
Aby ułatwić dostęp do danych, w algorytmie triangulacji wielokąt przechowywany jest w tablicy obiektów Point. Klasa ta posiada parametry:

1. x - współrzędna x danego punktu
2. y - współrzędna y danego punktu
3. chain - łańcuch, do którego należy dany punkt.
4. index - indeks danego punktu w tablicy przed uporządkowaniem, przeciwnie do ruchu wskazówek zegara

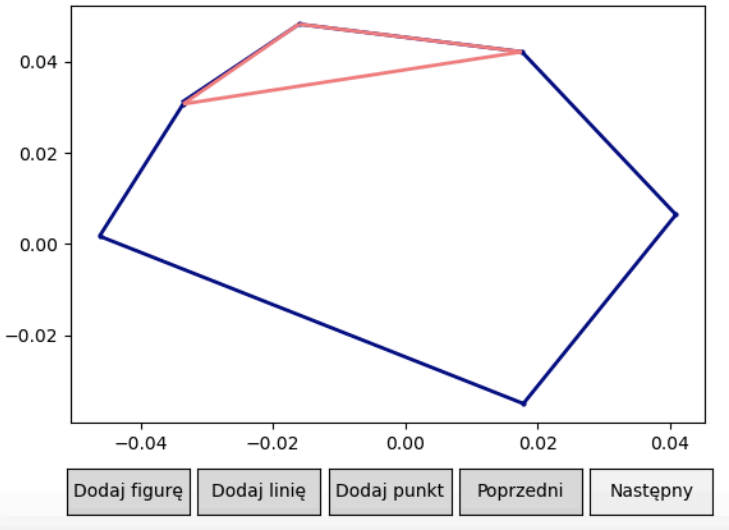
Klasa ta posiada metodę get_coords(), która zwraca współrzędne punktu.

Triangulacja natomiast jest przechowywana w postaci listy zawierającej dwuelementowe krotki. Krotki te zawierają indeksy punktów przed uporządkowaniem, między którymi w wyniku triangulacji powstały przekątne.

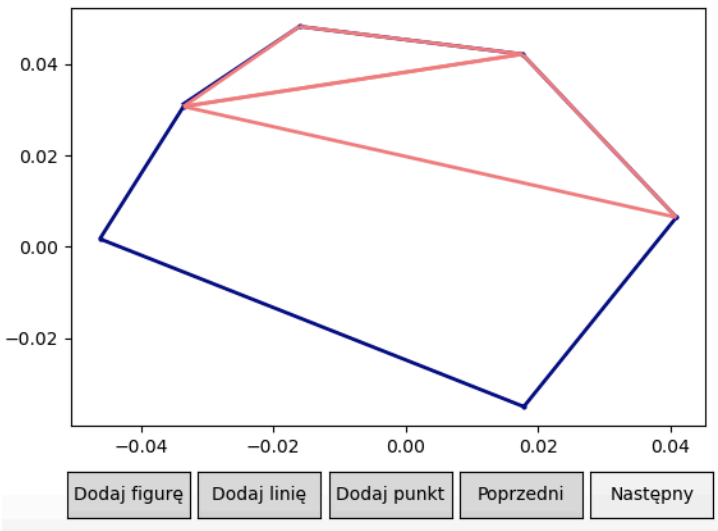
6. Wizualizacja procesu triangulacji dla przykładowego wielokąta.



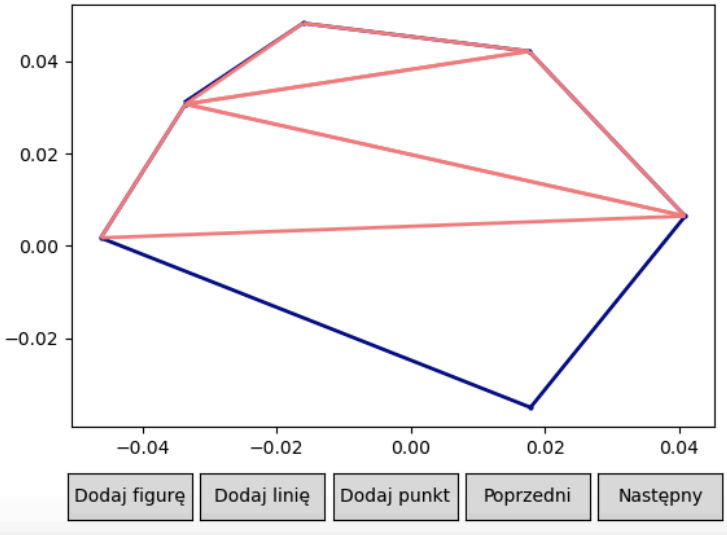
Wykres 7. Wizualizacja triangulacji krok 1



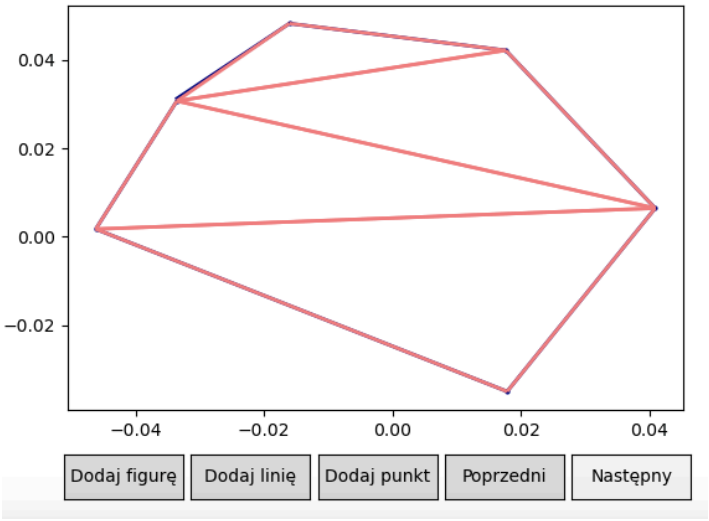
Wykres 8. Wizualizacja triangulacji krok 2



Wykres 9. Wizualizacja triangulacji krok 3



Wykres 10. Wizualizacja triangulacji krok 4



Wykres 11. Wizualizacja triangulacji krok 5

7. Wnioski

Algorytm poprawnie wyznaczał triangulację dla każdego sprawdzanego przypadku i zwrócił poprawną liczbę przekątnych. Zastosowane struktury danych sprawdziły się bardzo dobrze i umożliwiły łatwy i szybki dostęp do potrzebnych informacji.

Elementem algorytmu, na który warto zwrócić uwagę, było porządkowanie wierzchołków. W implementacji skorzystano z porządkowania na zasadzie scalania łańcuchów - lewego i prawego (podobna metoda jest wykorzystywana w algorytmie sortowania Merge Sort). Dzięki temu możliwe było uzyskanie złożoności O(n). Gdyby wierzchołki zostały posortowane tradycyjną metodą sortowania, złożoność osiągnęłaby wtedy O(nlogn).

Zastosowanie klasy Point również przyczyniło się do poprawy złożoności zaimplementowanego algorytmu. Dzięki niej można w łatwy sposób sprawdzić, do którego łańcucha należy dany punkt - operacja ta ma złożoność O(1). Można również sprawdzić przynależność punktu do danego łańcucha korzystając z operatora „in”, jednak złożoność takiej operacji dla list w języku Python wynosi O(n), co jest znacznie gorszym czasowo rozwiązaniem. Podobnie w przypadku indeksu danego punktu przed uporządkowaniem, który dzięki tej klasie możemy sprawdzić szybciej.