Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie

Algorytmy Geometryczne

Laboratorium nr 3

Tomasz Smyda

24 listopada 2023

Spis treści

1 Wstęp

1.1 Opis ćwiczenia

W ćwiczeniu laboratoryjnym nr 3 należało zaimplementować algorytm sprawdzający czy dany wielokąt jest y-monotoniczny, następnie dokonać implementacji algorytmu klasyfikacji wierzchołków dla podanego wielokąta, a na końcu zaimplementować algorytm odpowiedzialny za triangulację wielokąta y-monotonicznego na podstawie algorytmu podanego na wykładzie. Dodatkowo, należało dostosować odpowiednio aplikację graficzną, tak aby można było zadawać wejściowy wielokąt przy pomocy manipulatora stołokulotocznego.

1.2 Biblioteki

Ćwiczenie zostało wykonane przy użyciu narzędzia Jupyter Notebook z wykorzystaniem języka Python w wersji 3.9.18. Do wykonania rysunków oraz narzędzia do zadawania wejściowych wielokątów użyłem biblioteki matplotlib wraz z jej narzędziem PolygonSelector. Podczas kolorowania wierzchołków przyjąłem następującą konwencję:

- Wierzchołek początkowy
- Wierzchołek końcowy
- Wierzchołek łączący
- Wierzchołek dzielący
- Wierzchołek prawidłowy

1.3 Specyfikacja

Wszystkie obliczenia były prowadzone na komputerze Lenovo Legion 5 Pro z systemem operacyjnym Windows 11 Home w wersji 22H2, procesorem Intel Core i7-11800H.

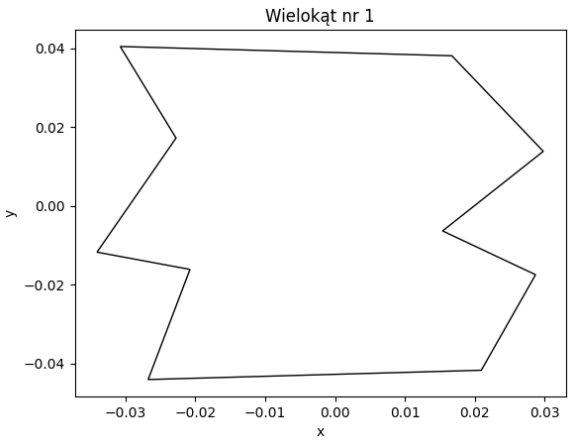
1.4 Plan i sposób wykonania ćwiczenia

1. Przygotowuję 10 wielokątów (niekoniecznie monotonicznych)
2. Implementuję algorytm sprawdzający czy dany na wejściu wielokąt jest y-monotoniczny
3. Implementuję algorytm do klasyfikacji wierzchołków oraz wizualizację rezultatów
4. Implementuję algorytm, który wykonuje traingulację, jeżeli dany na wejściu wielokąt jest y-monotoniczny

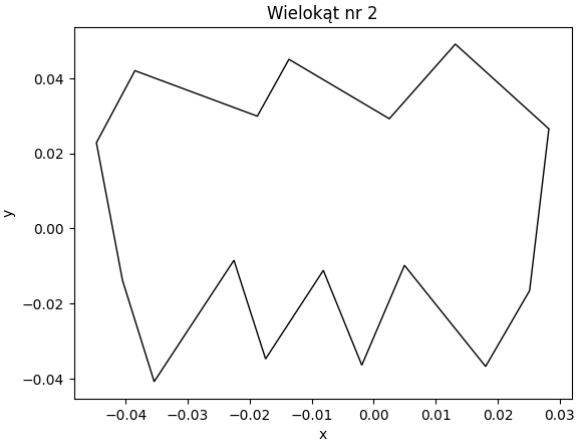
2 Realizacja ćwiczenia

2.1 Przygotowanie wielokątów do testowania

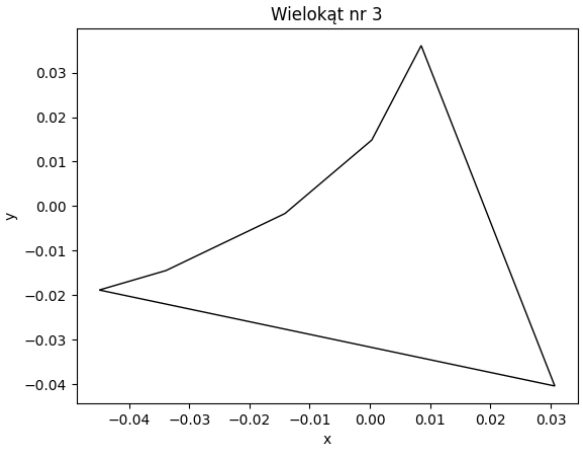
Na poniższych rysunkacj przedstawione są wielokąty, których użyłem do sprawdzenia poprawności algorytmów implementowanych w ramach tego ćwiczenia.



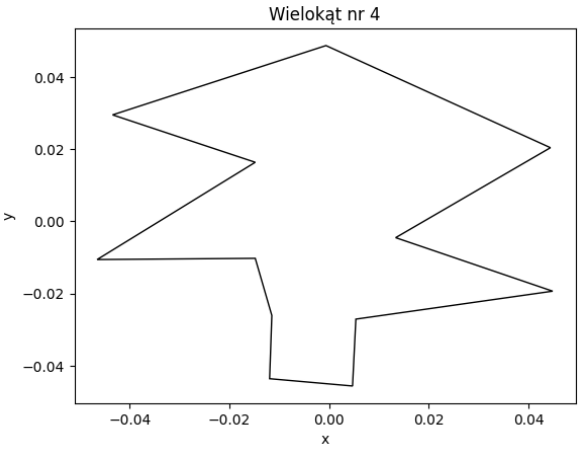
Rysunek 1: Wielokąt nr 1



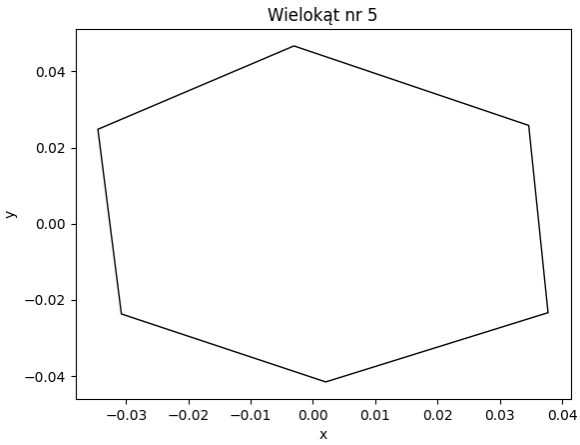
Rysunek 2: Wielokąt nr 2



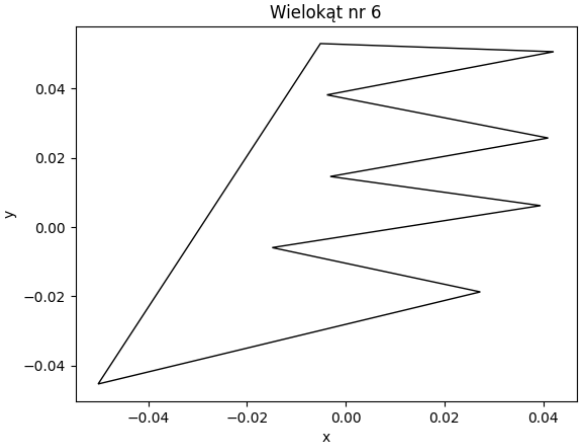
Rysunek 3: Wielokąt nr 3



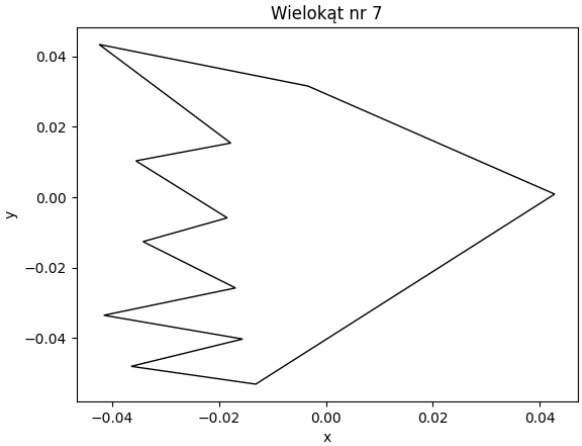
Rysunek 4: Wielokąt nr 4



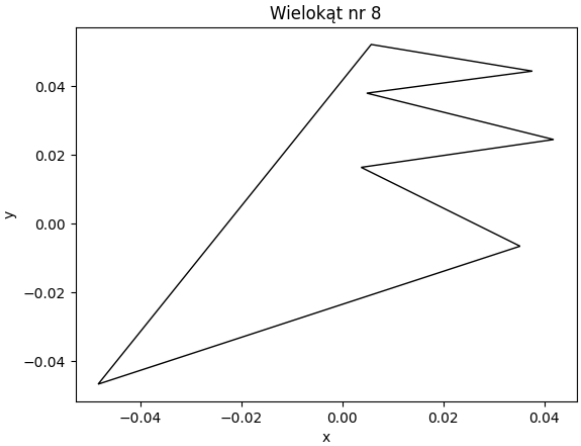
Rysunek 5: Wielokąt nr 5



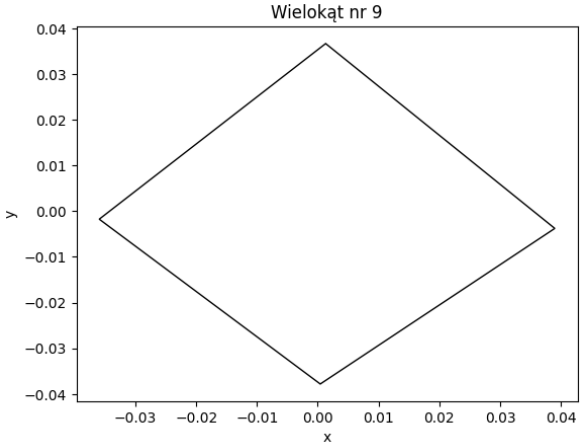
Rysunek 6: Wielokąt nr 6



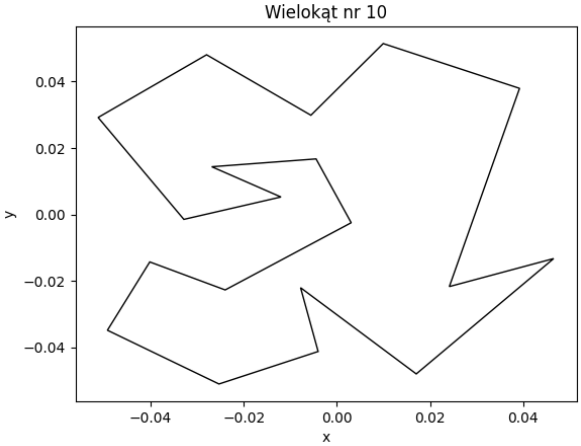
Rysunek 7: Wielokąt nr 7



Rysunek 8: Wielokąt nr 8



Rysunek 9: Wielokąt nr 9



Rysunek 10: Wielokąt nr 10

2.2 Algorytm sprawdzania y-monotoniczności wielokąta

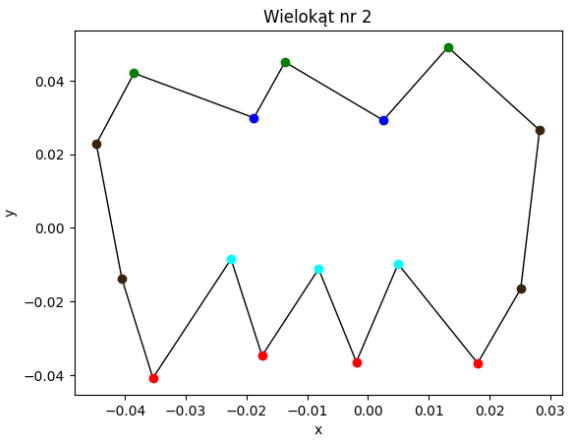
Do sprawdzenia czy podany wielokąt jest y-monotoniczny zaimplemenotwałem funkcję is_y_monotonic, która dla każdych trzech sąsiednich wierzchołków sprawdza, czy nie tworzą wierzchołka dzielącego lub łączącego - jeżeli tak, oznacza to, że podany wielokąt nie jest y-monotoniczny i funkcja zwraca fałsz, w przeciwnym wypadku wartość zostaje zwrócona prawda.

Po wykonaniu funkcji dla wygenerowanych wielokątów, algorytm zakwalifikował wielokąty nr 2, 4 oraz 10 (odpowiednio rysunki 2, 4, 10) jako niey-monotoniczne. Pozostałe wielokąty zostały określone jako y-monotoniczne.

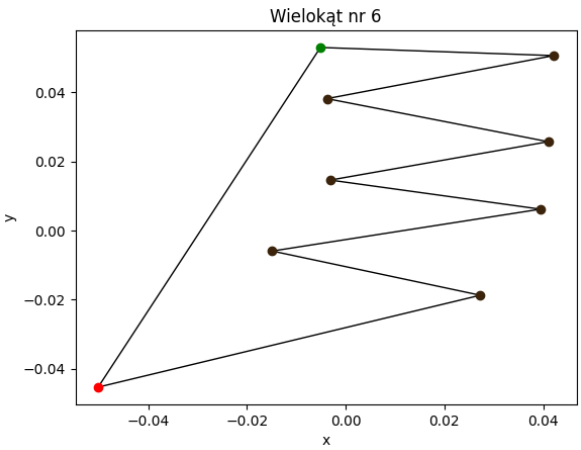
2.3 Algorytm klasyfikacji wierzchołków

Algorytm bazuje na przejściu po wszystkich wierzchołkach podanego wielokąta i określeniu na podstawie drugiej współrzędnej jego sąsiadów oraz kąta wewnętrznego jaki tworzą te wierzchołki. Do implementacji użyłem funkcji wyznacznika 2x2 własnej implementacji, która pomogła mi określić wzajemne położenie trzech punktów na płaszczyźnie. Do klasyfikacji wierzchołków, użyłem następujących kryteriów:

- Wierzchołek początkowy - obaj sądziedzi mają mniejszą drugą współrzędną oraz kąt wewnętrzny jest mniejszy niż π
- Wierzchołek końcowy - obaj sądziedzi mają większą drugą współrzędną oraz kąt wewnętrzny jest mniejszy niż π
- Wierzchołek łączący - obaj sądziedzi mają większą drugą współrzędną oraz kąt wewnętrzny jest większy niż π
- Wierzchołek dzielący - obaj sądziedzi mają mniejszą drugą współrzędną oraz kąt wewnętrzny jest większy niż π
- Wierzchołek prawidłowy - w pozostałych przypadkach



Rysunek 11: Wizualizacja sklasyfikowanych wierzchołków wielokąta nr 2



Rysunek 12: Wizualizacja sklasyfikowanych wierzchołków wielokąta nr 6

2.4 Algorytm triangulacji

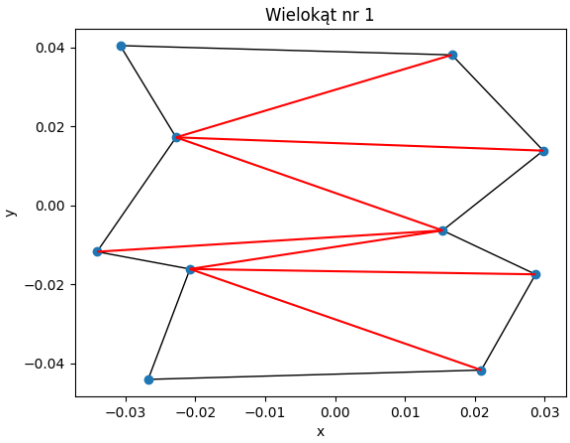
Algorytm na początku sprawdza czy podany na wejściu wielokąt jest ymonotoniczny, jeżeli nie, to kończy pracę zwracając fałsz. Następnie wykonuję podział wierzchołków na dwa łańcuchy: lewy i prawy. Do prawego łańcucha trafia wierzchołek o najniższej drugiej współrzędnej oraz wierzchołki znajdujące się przed wierzchołkiem o najwyższej drugiej współrzędnej idąc przeciwnie do ruchu wskazówek zegara. Po posortowaniu wierzchołków względem drugiej współrzędnej wrzucam pierwsze dwa wierzhołki na stos i przeglądam wszystkie pozostałe i w zależności od przypadku wybieram dwie opcje:

1. Wierzchołek należy do innego łańcucha niż wierzchołek na najwyższym punkcie stosu
   Możemy utworzyć przekątne ze wszystkimi wierzchołkami, które dotychczas znajdują się na stosie. Po kolei zdejmuje każdy wierzchołek ze stosu, sprawdzam czy nie sąsiaduje z rozpatrowanym aktualnie wierzchołkiem, jeżeli nie - dodaję przekątną pomiędzy tymi wierzchołkami. Po zdjęciu wszystkich wierzchołków ze stosu, odkładam na niego wierzchołek, który wcześniej leżał na jego szczycie oraz rozpatrywany wierzchołek.

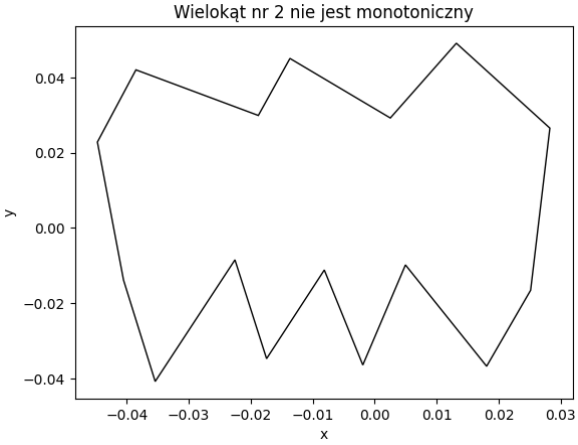
2. Wierzchołek należy do tego samego łańcucha co wierzchołek na najwyższym punkcie stosu
   Ściągam wierzchołek ze stosu i dopóki spełniony jest warunek, że trójkąt tworzony przez wierzchołek aktualnie rozpatrywany, wierzchołek ściągnięty ze stosu oraz wierzchołek na stosie całkowicie zawiera się w wielokącie sprawdzam czy wierzchołki nie są sąsiadującymi, jeżeli nie to dodaje między nimi przekątną. Gdy warunek przestaje być spełniony dodaję na stos początkowo zdjęty wierzchołek oraz aktualnie rozpatrywany wierzchołek.

Podczas realizacji algorytmu triangulacji wielokąta, wierzchołki przechowuję w liście jako krotki współrzędnych wierzchołków, natomiast pod koniec funkcji zamieniam je tak, aby zwracana lista zawierała dwuelementowe listy, w których elementy to indeksy wierzchołków pomiędzy którymi dodaje przekątne.

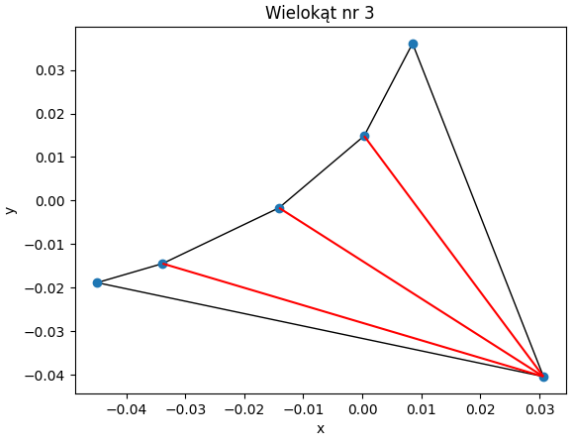
Na kolejnej stronie znajdują się przykłady wizualizacji wyników algorytmu dla podanych wielokątów. (Wszystkie wizualizacje wyników algorytmów dla danych testowych znajdują się w kodzie programu tj. plik z rozszerzeniem .ipynb).



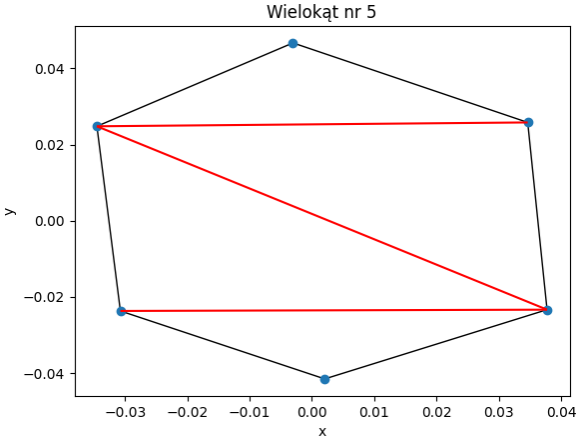
Rysunek 13: Wizualizacja wyniku algorytmu dla wielokąta nr 1



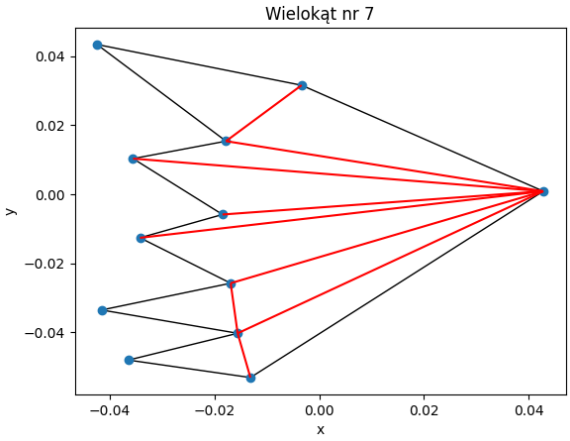
Rysunek 14: Wizualizacja wyniku algorytmu dla wielokąta nr 2



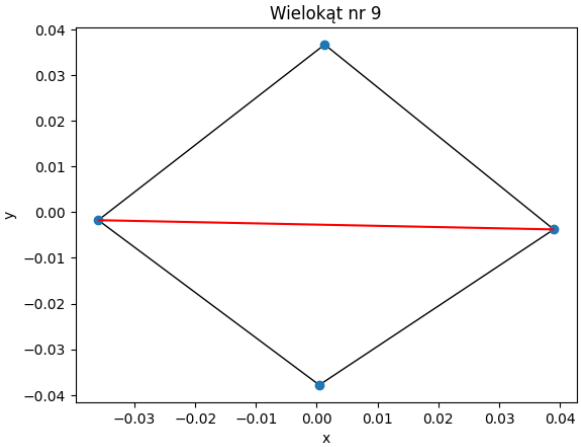
Rysunek 15: Wizualizacja wyniku algorytmu dla wielokąta nr 3



Rysunek 16: Wizualizacja wyniku algorytmu dla wielokąta nr 5



Rysunek 17: Wizualizacja wyniku algorytmu dla wielokąta nr 7



Rysunek 18: Wizualizacja wyniku algorytmu dla wielokąta nr 9

3 Wnioski

Jak możemy zauważyć na przedstawionych rysunkach, algorytmy zaimplementowane podczas wykonywania ćwiczenia działają poprawnie dla podanych wielokątów. Biorąc pod uwagę fakt, że niektóre z tych wielokątów uwzględniają przypadki skrajne, możemy stwierdzić, że algorytmy zostały poprawnie zaimplementowane, a ich rezultaty pokrywają się z oczekiwaniami.