Algorytmy geometryczne
Sprawozdanie z laboratoriów 3

grupa nr.4 Czw_12.20_B

Paweł Surdyka

Dane techniczne urządzenia na którym wykonano ćwiczenie:
Komputer z systemem Windows 10 x64
Procesor: Intel Core I5 9300HF CPU @2.4Ghz
Pamięć RAM: 8GB
Środowisko: Jupyter notebook
Język: Python 3

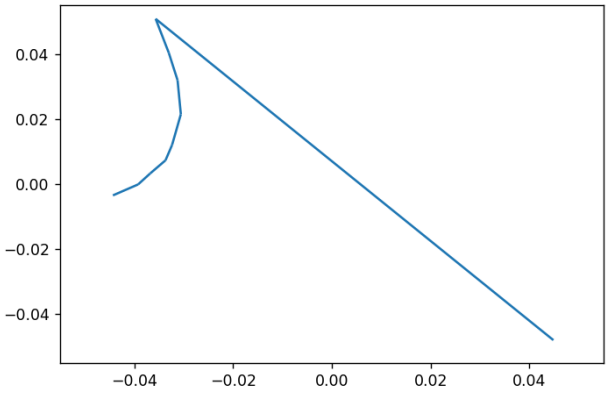
1. Opis realizacji ćwiczenia:

Ćwiczenie polegało na implementacji algorytmów:
- Sprawdzania y-monotoniczności zadanego wielokąta
- Klasyfikacji wierzchołków w dowolnym wielokącie
- Triangulacji wielokąta y-monotonicznego wraz z wizualizacją

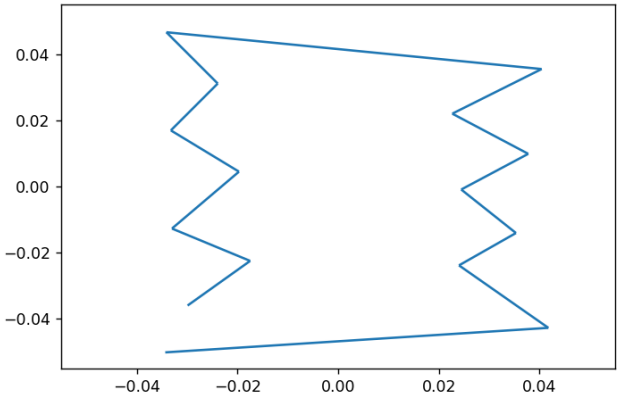
2. Sposób wykonania ćwiczenia, opisy algorytmów oraz wizualizacja ich działania na przykładach

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.

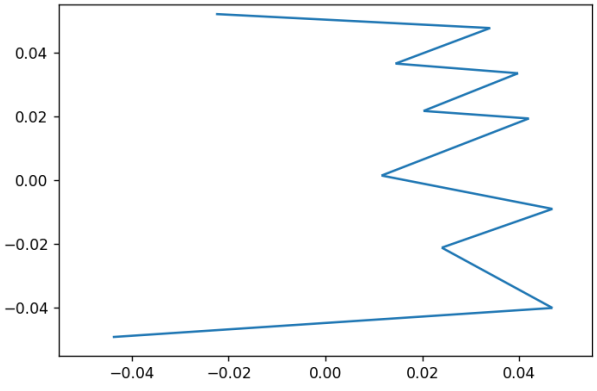
Rysunek 1.1



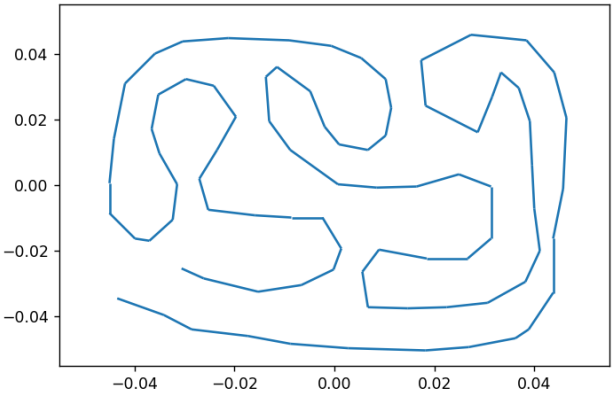
Rysunek 1.2



Rysunek 1.3



Rysunek 1.4



3.Klasyfikacja wierzchołków

Algorytm klasyfikacji wierzchołków wielokąta na początkowe, końcowe, łączące, dzielące i prawidłowe polega na jednokrotnym przejściu po liście wierzchołków i ich klasyfikacji na podstawie ich położenia w relacji do sąsiadów (wierzchołka poprzedniego i następnego). Należało również uwzględnić to, że pierwszy i ostatni wierzchołek na liście również ze sobą sąsiadują.

W celu określenia kąta wewnętrznego jaki tworzą rozpatrywane wierzchołki użyto wyznacznika. Punkty były klasyfikowane na podstawie następujących warunków, z następującymi oznaczeniami:

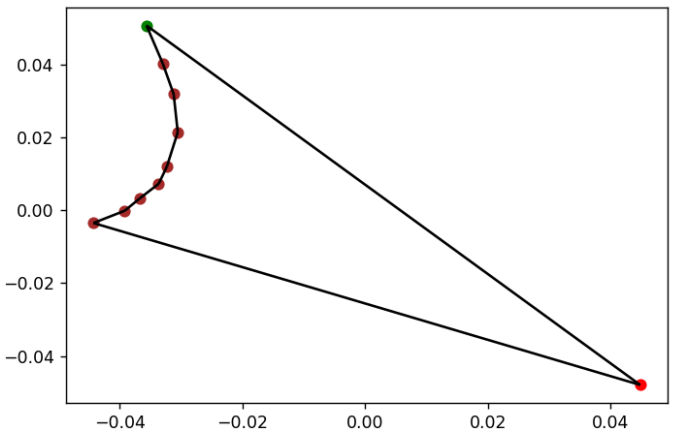
- początkowy, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny <
- końcowy, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny <
- łączący, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny >
- dzielący, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny >
- prawidłowy, w pozostałych przypadkach (ma jednego sąsiada powyżej, drugiego – poniżej). (brązowy)

Dla wszystkich zestawów wierzchołki zostały sklasyfikowane poprawnie. Zgodnie z oczekiwaniami wielokąty monotoniczne zawierały po jednym wierzchołku początkowym (najwyższy), końcowym (najniższym), a pozostałe wierzchołki były prawidłowe.

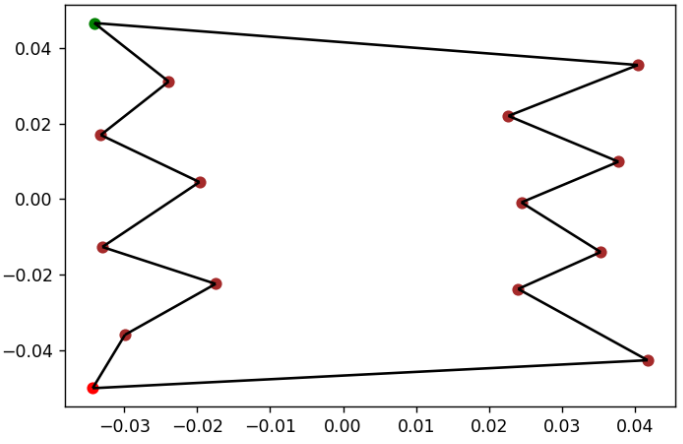
Algorytm koloruje wierzchołki w sposób:

- zielony - wierzchołek początkowy,
- czerwony - wierzchołek końcowy,
- żółty i - wierzchołek łączący,
- pomarańczowy - wierzchołek dzielący,
- brązowy - wierzchołek prawidłowy

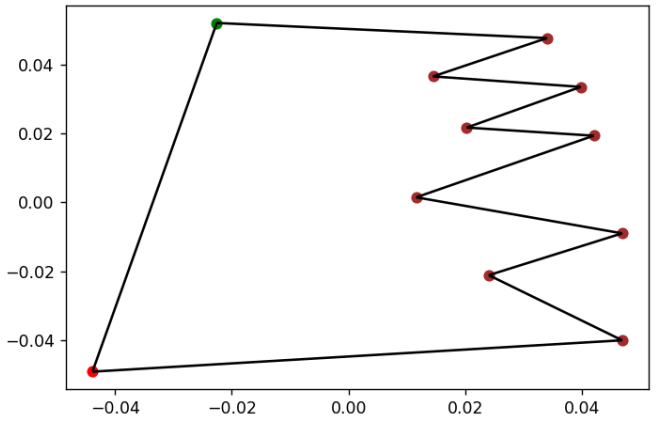
Rysunek 2.1



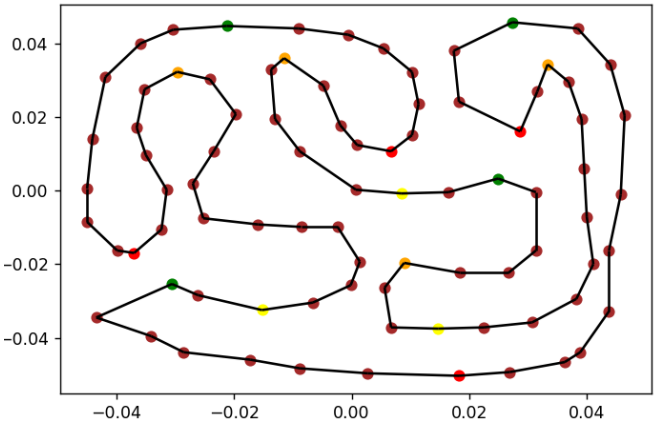
Rysunek 2.2



Rysunek 2.3



Rysunek 2.4



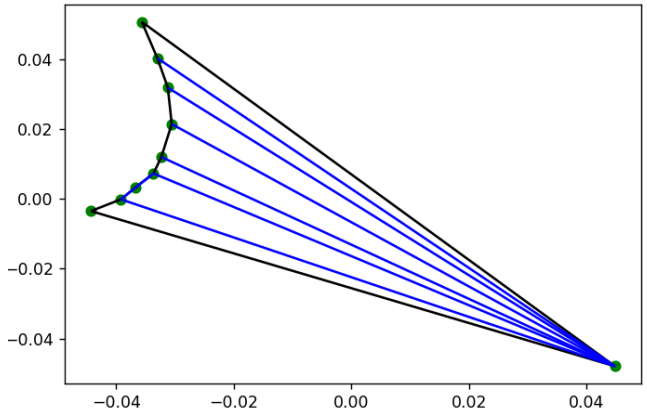
4.Triangulacja

Początek algorytmu polega na sprawdzeniu czy wielokąt jest y-monotoniczny. Sprawdzenie odbywa się w funkcji y_monotonic. 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:

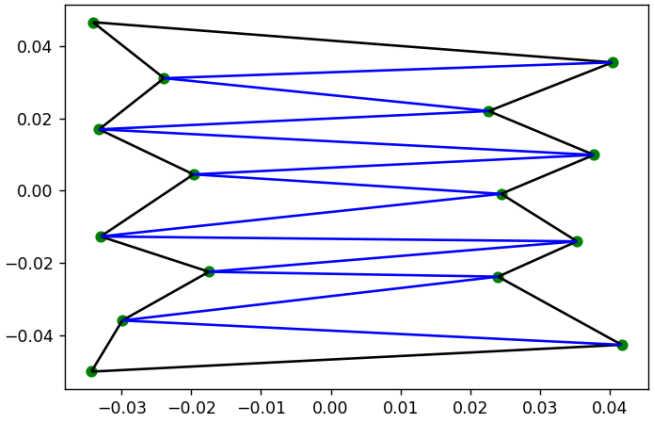
- jeśli aktualnie rozpatrywany wierzchołek znajduje się na innym łańcuchu niż szczyt stosu, to łączymy go z wszystkimi wierzchołkami które znajdują się na stosie i nie są jego sąsiadami, po wykonaniu całej operacji na stosie zostawiamy dwa ostatnio analizowane wierzchołki,
- jeśli aktualnie rozpatrywany 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).

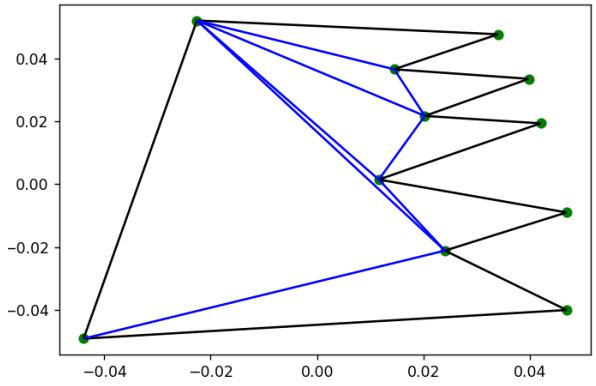
Rysunek 3.1



Rysunek 3.2



Rysunek 3. Rysunek 3.



Wnioski

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