Sprawozdanie Algorytmy Geometryczne

Ćwiczenie 3 triangulacja wielokątów monotonicznych

Norbert Wolniak
Grupa Czw_16.15_A

1. Cel ćwiczenia:
   Celem ćwiczenia jest zapoznanie się oraz zaimplementowanie procedury do sprawdzania czy dany wielokąt jest y-monotoniczny, algorytmu do klasyfikacji wierzchołków w dowolnym wielokącie oraz algorytmu do triangulacji wielokąta y-monotonicznego.

2. Uwagi techniczne:
   Ćwiczenie zostało przeprowadzone w oprogramowaniu Jupyter Notebook, na systemie Windows 10 Home, komputer wyposażony w procesor x64 przy użyciu proponowanego narzędzia graficznego, które napisane jest w języku Python3 i wykorzystuje bibliotekę MatPlotLib. Do liczenia wyznacznika użyto wyznacznika 3x3 zaimplementowanego samemu przy tolerancji dla zera $e=10^{-13}$, który okazał się najlepszy w ćwiczeniu 1.

3. Wielokąty:
   Do zadawania wielokątów przeciwnie do ruchu wskazówek zegara użyto proponowanego narzędzia graficznego, wielokąty przetrzymywane są w zmiennej, przy pomocy funkcji narzędzia graficznego do pozyskiwania narysowanego myszką wielokąta.

4. Klasyfikacja wierzchołków wielokąta:

    Początkowy - gdy oba wierzchołki sąsiednie leżą poniżej i kąt wewnętrzny jest mniejszy niż 180°.

    Końcowy - gdy oba wierzchołki sąsiednie leżą powyżej i kąt wewnętrzny jest mniejszy niż 180°.

    Łączący - gdy oba wierzchołki sąsiednie leżą powyżej i kąt wewnętrzny jest większy niż 180°.

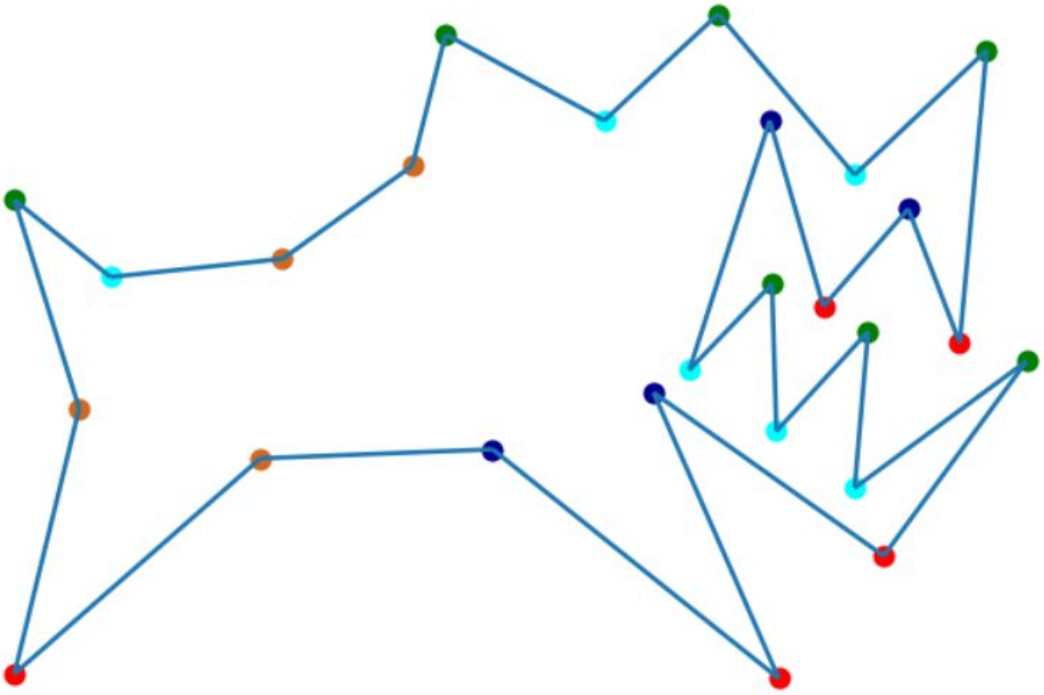
    Dzielący - gdy oba wierzchołki sąsiednie leżą poniżej i kąt wewnętrzny jest większy niż 180°.

    Prawidłowy - gdy ma jeden wierzchołek sąsiedni leży poniżej, a drugi powyżej.

5. Procedura sprawdzająca y-monotoniczność.
   Korzystając z tego, że wielokąt jest y-monotoniczny, gdy nie posiada wierzchołków dzielących i łączących procedura polega na przeiterowaniu wszystkich wierzchołków w wielokącie w kolejności zadawanych wierzchołków (przeciwnie do ruchu wskazówek zegara) oraz sprawdzenia warunków na odpowiednie wierzchołki dzielących i łączących. Do celu tego użyłem dwóch funkcji pomocniczych is_below(q,p) i is_above(q,p) oraz wyznacznika det3x3(p,q,r).

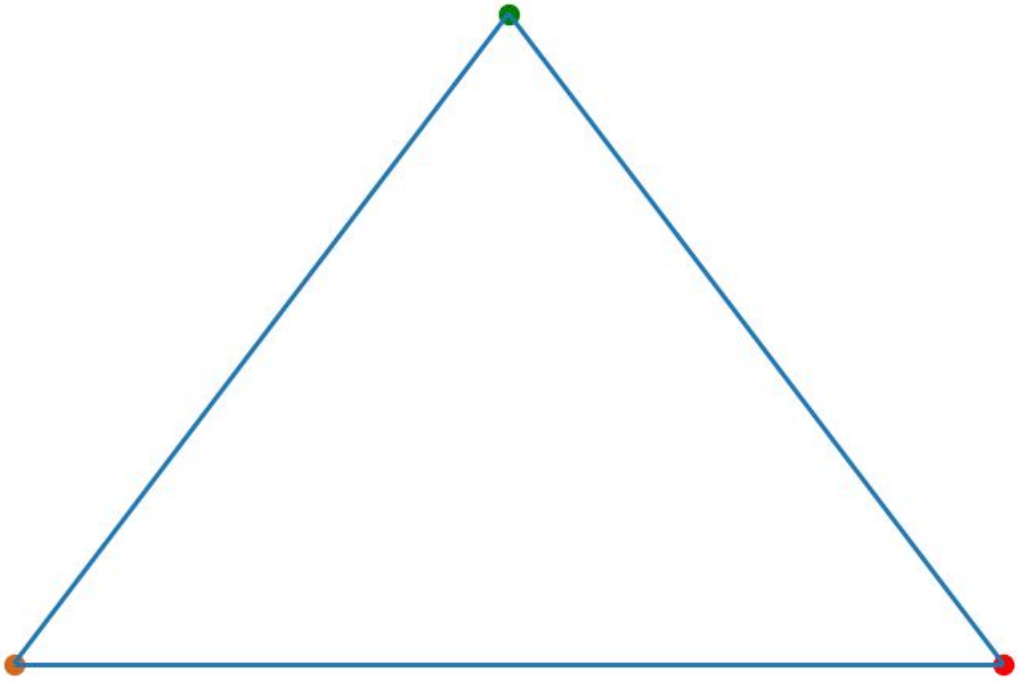
6. Algorytm klasyfikacji wierzchołków.
   polega na przeiterowaniu wszystkich wierzchołków w wielokącie w kolejności zadawanych wierzchołków (przeciwnie do ruchu wskazówek zegara) oraz sprawdzenia warunków na odpowiednie wierzchołki - początkowe, końcowe, łączące, dzielące i prawidłowe. Do celu tego użyłem dwóch funkcji pomocniczych is_below(q,p) i is_above(q,p) oraz wyznacznika det3x3(p,q,r).

   Rysunek nr 1.



Rysunek nr 2.

Rysunek nr 3.



7. Algorytm triangulacji wielokąta y-monotonicznego.

   Opis algorytmu:
   Algorytm polega na podzieleniu wierzchołków na dwa różne łańcuchy - lewy i prawy względem kierunku monotoniczności. Sortujemy wierzchołki wzdłuż kierunku monotoniczności w tym przypadku od największej współrzędnej y do najmniejszej. Tworzymy stos i po kolei iterujemy po wszystkich wierzchołkach w posortowanej kolejności. Występują dwa przypadki przynależności rodzaju łańcucha wierzchołka obecnie odwiedzanego do rodzaju łańcucha wierzchołka znajdującego się na szczycie stosu. Obsługujemy je w odpowiedni sposób opisany na wykładzie. Czas działania samego algorytmu triangulacji to O(n), a zatem jest liniowy gdzie n - liczba wierzchołków wielokąta monotonicznego. Czas działania zaimplementowanego algorytmu to O(nlogn), ponieważ dostarczamy nie posortowane wierzchołki.
   W mojej implementacji używam dodatkowego oznaczenia wierzchołków początkowego i końcowego jako należącego do obu łańcuchów, aby łatwiej i czytelniej można było obsłużyć końcowy wierzchołek podczas działania algorytmu.

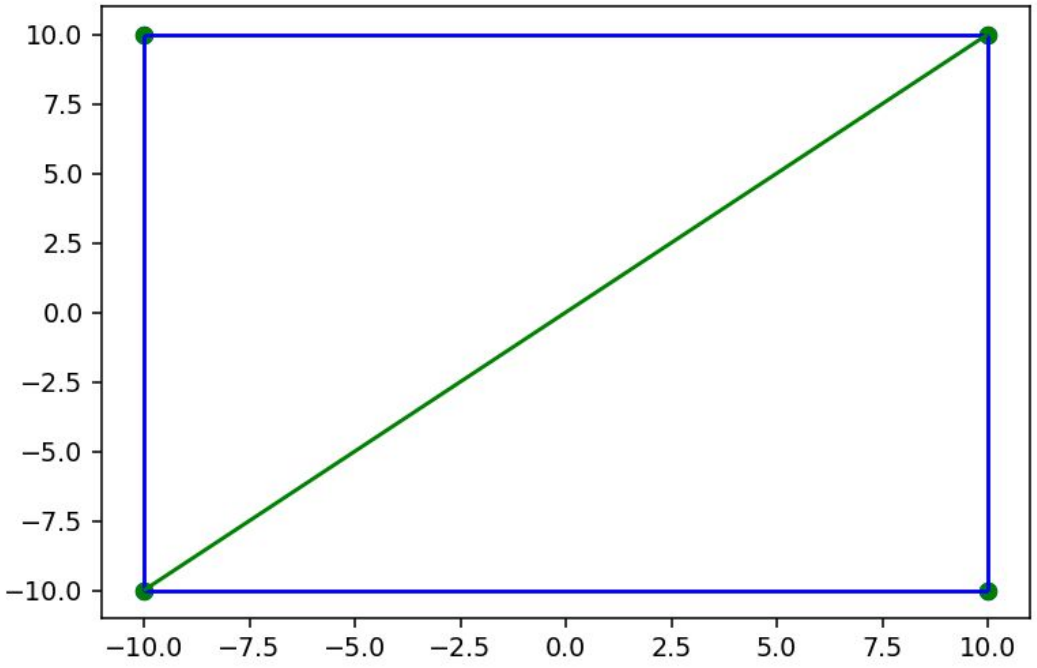
   Zwracana struktura:
   W zaimplementowanym przeze mnie algorytmie zwracana jest lista przechowująca dwie listy - odpowiednio listę wszystkich trójkątów na jakie dzieli się wielokąt, gdzie wierzchołki danego trójkąta znajdują się w kolejności przeciwnej do ruchu wskazówek zegara oraz listę przekątnych, które powstają na skutek działania algorytmu dzieląc dany wielokąt na trójkąty.
   Użyłem takiej struktury danych ponieważ umożliwia ona przetrzymywanie trójkątów jak i przekątnych w jednej liście.

   Wizualizacja algorytmu:
    Wierzchołek odwiedzany.

    Utworzona przekątna.

    Wierzchołek już przetworzony.

Testowane wielokąty :
Kwadrat - poprawna definicja funkcji is_below(q,p) i is_above(q,p) (widoczna również na rysunku nr 2.)



Trójkąt : Przypadek najmniej liczny wierzchołkowo.

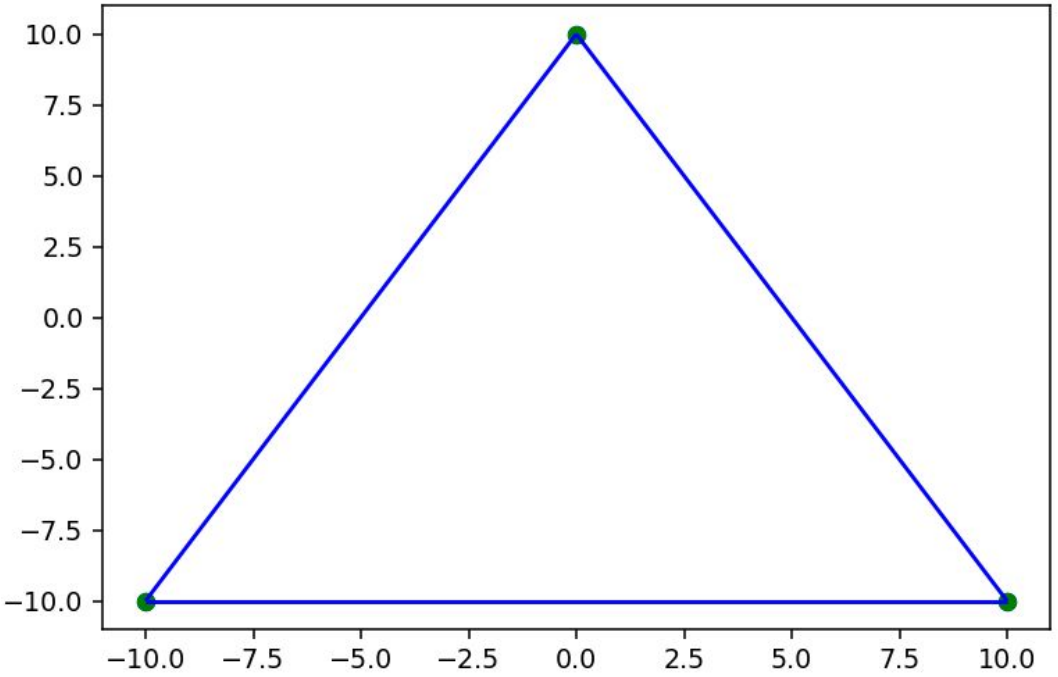


Figura nr 1 : Działanie algorytmu.

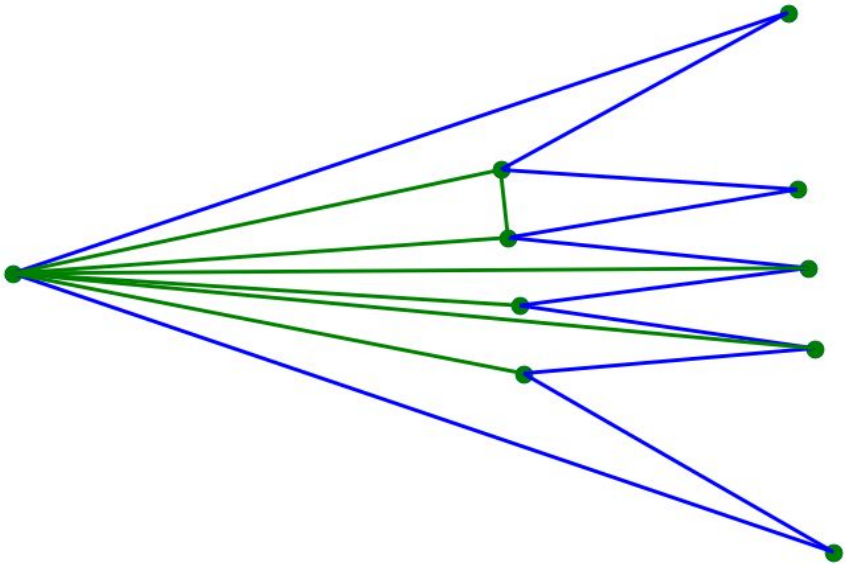


Figura nr 2 : Sprawdzenie czy któraś z przekątnych się przecina.

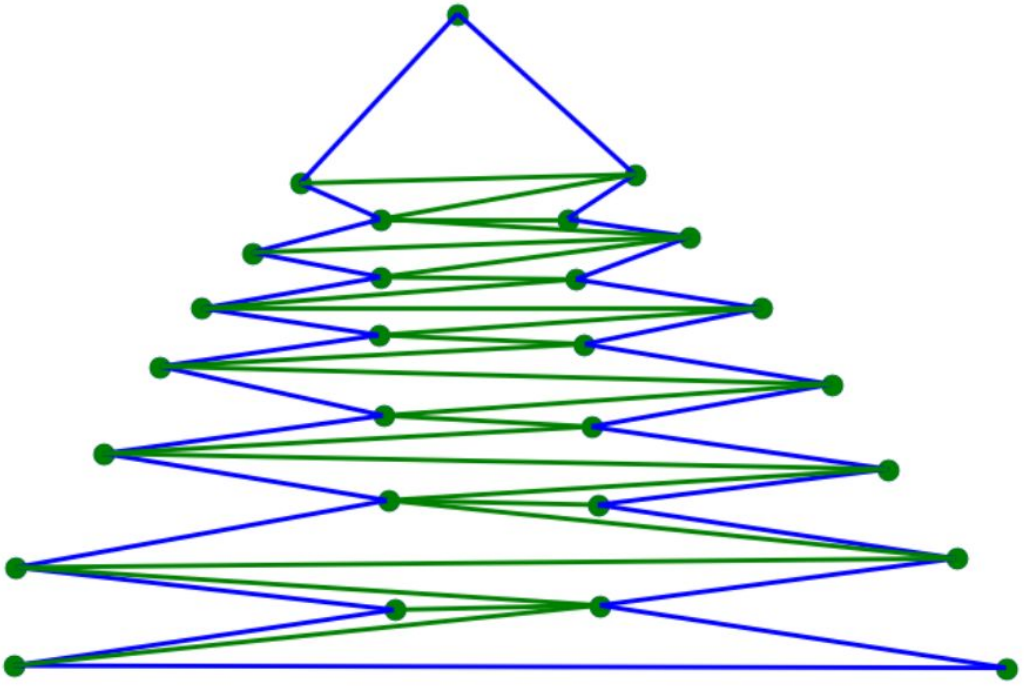


Figura nr 3: Sprawdzenie przypadku gdy wierzchołek początkowy jest połączony z wierzchołkiem końcowym oraz czy widoczne 3 ‘zielone’ trójkąty są zwracane w liście wynikowej.

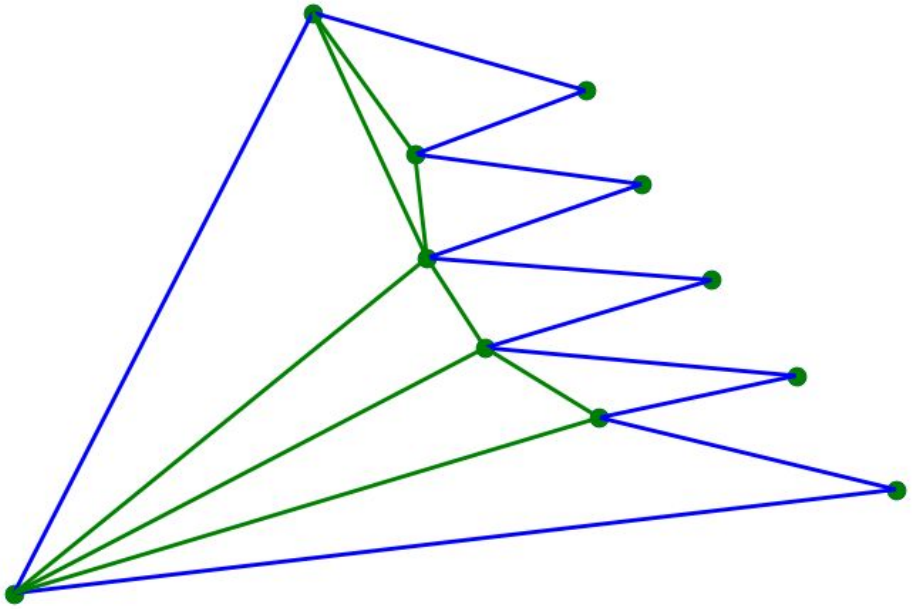


Figura nr 4: Odbicie lustrzane figury nr 3.

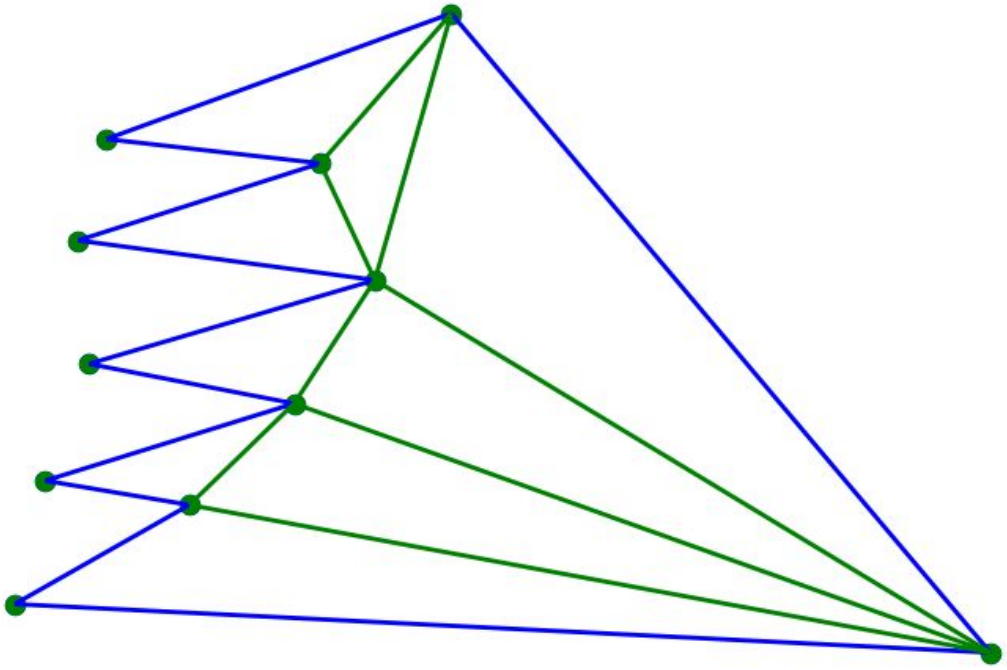
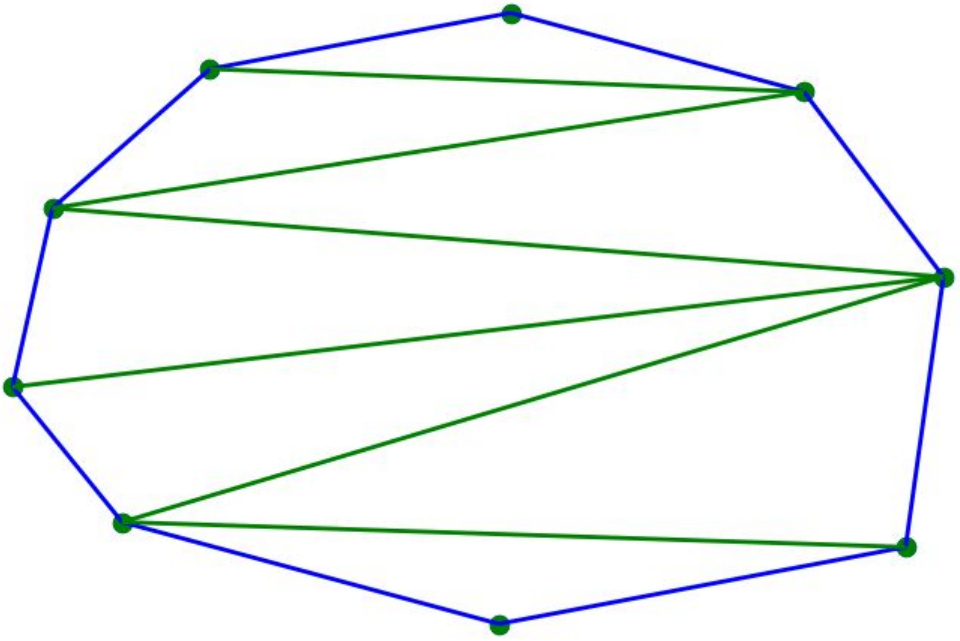


Figura nr 5: Otoczka wypukła.



8. Wnioski:
   Zaimplementowany algorytm triangulacji działa na każdym wielokącie y - monotonicznym, sama złożoność triangulacji jest liniowa zatem jest on szybkim algorytmem do wyznaczania triangulacji wielokąta monotonicznego.