1 Wstęp

Celem ćwiczenia było zapoznanie się z zagadnieniem monotoniczności wielokąta, a w szczególności z algorytmem sprawdzania, czy dany wielokąt jest monotoniczny, algorytmem klasyfikacji wierzchołków w dowolnym wielokącie oraz algorytmem triangulacji wielokąta monotonicznego.

1.1 Wstęp teoretyczny

1.1.1 y-monotoniczność

Wielokąt y-monotoniczny jest definiowany jako wielokąt, którego wierzchołki można podzielić na dwa łańcuchy $L = (l_1, l_2, ..., l_n)$ i $P = (p_1, p_2, ..., p_m)$, dla których zachodzą:

$$\begin{aligned} &\forall l_i = (l_{i_x}, l_{i_y}), l_i = (l_{i+1_x}, l_{i+1_y}) \in L \quad \text{istnieje krawędź } (l_i, l_{i+1}) \wedge l_{i_y} > l_{i+1_y} \\ &\forall p_i = (p_{i_x}, p_{i_y}), p_i = (p_{i+1_x}, p_{i+1_y}) \in P \quad \text{istnieje krawędź } (p_i, p_{i+1}) \wedge p_{i_y} > p_{i+1_y} \end{aligned} \tag{1}$$

L i P będę nazywał kolejno lewym i prawym łańcuchem, przyjmując L jako łańcuch idący w kierunku wbrew wskazówkom zegara od punktu o najwyższej współrzędnej względem osi OY.

Algorytm sprawdzający czy wielokąt jest y-monotoniczny znajduje najwyższy (względem osi OY) wierzchołek i sprawdza czy idąc wbrew ruchowi wskazówek zegara wszystkie sąsiadujące wierzchołki są niżej od poprzedniego, aż do najniższego punktu. Następnie, idąc od najniższego punktu, każdy kolejny wierzchołek powinien być wyżej od poprzedniego, aż do najwyższego punktu.

Monotoniczność wielokąta da się uogólnić do monotoniczności względem dowolnej prostej $k : Ax + By + C = 0$, lecz w tym ćwiczeniu zajmowaliśmy się jedynie przypadkiem dla $k : x = 0$.

1.1.2 Klasyfikacja wierzchołków

W dowolnym wielokącie jesteśmy w stanie rozróżnić 5 kategorii wierzchołków:

1. **Początkowe**
   gdy obaj jego sąsiedzi leżą poniżej i kąt między nimi jest mniejszy od π.
2. **Końcowe**
   gdy obaj jego sąsiedzi leżą powyżej i kąt między nimi jest mniejszy od π.
3. **Łączące**
   gdy obaj jego sąsiedzi leżą powyżej i kąt między nimi jest większy od π.
4. **Dzielące**
   gdy obaj jego sąsiedzi leżą poniżej i kąt między nimi jest większy od π.
5. **Prawidłowe**
   gdy jeden sąsiad jest wyżej, a drugi niżej.

(Kąt między punktami sprawdzamy omawianą w poprzednich ćwiczeniach funkcją `orient`.)

Kategorie te między innymi pozwalają nam na dzielenie wielokątów prostych na wielokąty monotoniczne, w celu prostszej ich triangulacji.

1.1.3 Triangulacja wielokątów monotonicznych

Algorytm przebiega następująco:

1. Dzielimy wielokąt na łańcuchy L i P względem monotoniczności.
2. Łączymy L i P w ciąg wierzchołków $U = (u1, u2, ..., u_n)$ posortowany względem kierunku monotoniczności (sortowanie jesteśmy w stanie wykonać w czasie liniowym, scalając odpowiednio łańcuchy).
3. Tworzymy stos S i wkładamy na niego wierzchołki $u1$ i $u2$. Niech $top(S)$ oznacza wierzhołek na górze stosu.
4. Dla każdego kolejnego wierzchołka u_i (gdzie $i = 3, ..., n$):
   - Jeżeli u_i i $top(S)$ należą do różnych łańcuchów (w mojej implementacji przyjmuję, że najwyższy i najniższy punkt należą do obu łańcuchów jednocześnie), to łączymy u_i z każdym wierzchołkiem w S. Następnie dodajemy do S kolejno wierzchołki u_{i-1} i u_i.
   - Inaczej, sciągamy wierzchołek ze stosu, oznaczając go p. Następnie, dopóki trójkąt $(top(S), p, u_i)$ znajduje się wewnątrz wielokąta, łączymy u_i z $top(S)$ i ściągamy kolejny wierzchołek ze stosu. Po zakończeniu łączenia wierzchołków dodajemy kolejno p i u_i do S.
5. Wynik algorytmu może być przedstawiony na różne sposoby, dwa użyte w mojej implementacji to:
   - Lista par indeksów wierzchołków tworzących przekątne. (w tym przypadku musimy w trakcie działania algorytmu pomijać dodawane odcinki, które jednocześnie są bokami wielokąta).
   - Lista trójek indeksów wierzchołków tworzących trójkąty

Wspomniane w algorytmie sprawdzanie, czy trójkąt zawiera się w wielokącie jesteśmy w stanie zaimplementować za pomocą funkcji `orient`; wierzchołki a, b i c (w tej kolejności), przy c znajdującym się na lewym łańcuchu, tworzą trójkąt wewnątrz wielokąta wtw. ich "skrętność" jest wbrew ruchowi wskazówek zegara i odwrotnie gdy c znajduje się na prawym łańcuchu.

Złożoność obliczeniowa:	$O(n)$	gdzie n oznacza liczbę wierzchołków wielokąta
Złożoność pamięciowa:	$O(n)$	

1.2 Specyfikacja narzędzi i sprzętu

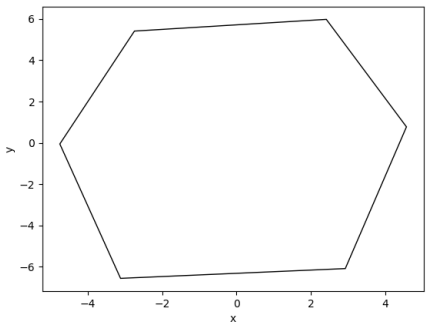
Wszystkie potrzebne obliczenia zostały wykonane przy użyciu interpretera języka Python wersji 3.12. Ponadto, w celu wygenerowania zbiorów punktów użyłem biblioteki `numpy`. Wykresy i wizualizacja wyników została przygotowana za pomocą narzędzia przygotowanego przez koło naukowe Bit. Do implementacji interaktywnego zadawania wielokątów użyłem biblioteki `matplotlib`. Kod znajduje się w załączonym pliku. Przedstawione wyniki zostały wygenerowane na komputerze z systemem operacyjnym Debian 12 i procesorem AMD Ryzen 5 3600 3.6GHz.

2 Realizacja obliczeń

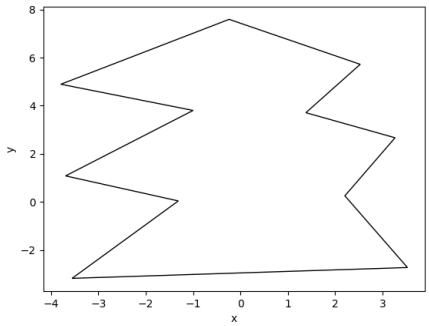
Realizację ćwiczenia zacząłem od implementacji trzech algorytmów: sprawdzania czy wielokąt jest y-monotoniczny, kategoryzowania wierzchołków na te wymienione w sekcji 1.1.2 i triangulacji wielokątów y-monotonicznych. Wyniki algorytmu triangulacji przedstawiłem na dwa sposoby wymienione w sekcji 1.1.3.

Pary indeksów wierzchołków tworzących przekątne umożliwiły mi prostszą wizualizację algorytmu za pomocą użytych przeze mnie narzędzi, a trójki indeksów wierzchołków tworzących trójkąty są wypisywane przez program, ponieważ jest to bardziej uniwersalny sposób na przedstawienie triangulacji, co umożliwia prostsze użycie wyników w innym programie. Zaimplementowane przeze mnie algorytmy zakładają, że wierzchołki wielokąta podane są w kierunku wbrew ruchowi wskazówek zegara.

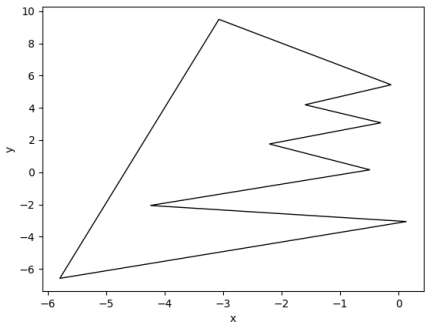
Następnie przygotowałem 5 wielokątów w celu zwizualizowania wyników działania algorytmów. Wielokąty te przedstawione są na Rysunku 1.



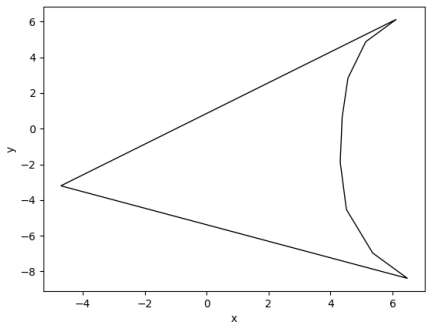
(i) Wielokąt A



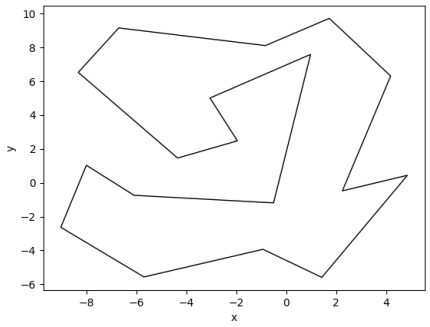
(ii) Wielokąt B



(iii) Wielokąt C



(iv) Wielokąt D



(v) Wielokąt E

Rysunek 1: Wygenerowane zbiory

Ostatecznie przygotowałem aplikację umożliwiającą zadawanie dowolnych wielokątów (w załączonym kodzie) oraz animacje wizualizujące działanie algorytmu triangulacji (animacje są załączone razem ze sprawozdaniem).

3 Wyniki

Przyjmuję następujące oznaczenia dla poszczególnych algorytmów:

- `is_y_monotonic` - sprawdzanie czy wielokąt jest y-monotoniczny. Wynikiem jest `True` (prawda) lub `False` (fałsz).
- `color_vertex` - kategoryzacja wierzchołków. Zwizualizowane wyniki dla poszczególnych wielokątów zawarte są na Rysunkach: 2, 4, 6, 8, 10. Wierzchołki oznaczone są następującymi kolorami:
  1. **Początkowe** - zielony
  2. **Końcowe** - czerwony
  3. **Łączące** - granatowy
  4. **Dzielące** - niebieski
  5. **Prawidłowe** - brązowy
- `triangulate` - triangulacja wielokątów y-monotonicznych. Zwizualizowane wyniki dla poszczególnych wielokątów zawarte są na Rysunkach: 3, 5, 7, 9. (Wielokąt E nie jest monotoniczny, więc jest pominięty). W celu prostszej wizualizacji algorytm zwracał jedynie przekątne, które należy stworzyć (jako pary indeksów wierzchołków), implementacja zwracająca trójkąty (jako trójki indeksów wierzchołków) jest zawarta w załączonym kodzie. Stworzone przez algorytm przekątne oznaczone są kolorem czerwonym.

Załączone ze sprawozdaniem animacje przyjmują następujące oznaczenia:

- Czerwone odcinki - przekątne stworzone przez algorytm.
- Czerwone punkty - wierzchołki znajdujące się na stosie.
- Brązowe punkty - wierzchołki zdjęte ze stosu.
- Niebieskie punkty - wierzchołki jeszcze nie przetworzone przez algorytm.

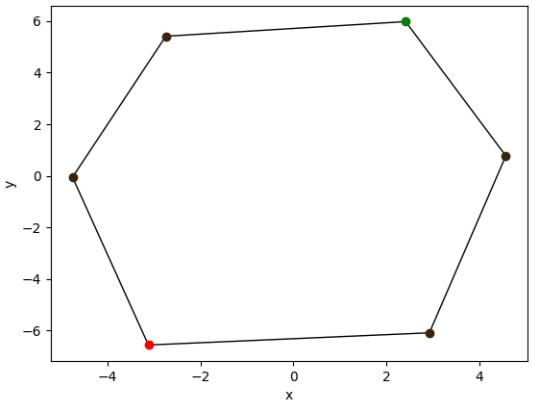
Sekcje od 3.1 do 3.5 zawierają wyniki algorytmów na poszczególnych wielokątach, wraz z krótkim komentarzem na ich temat.

3.1 Wielokąt A

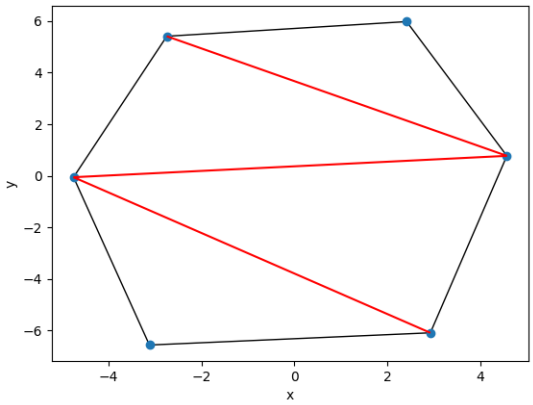
Wielokąt ten pozwala nam sprawdzić czy algorytm poprawnie radzi sobie z odfiltrowywaniem boków wielokąta z wyniku (jest to konieczne jedynie kiedy zwracamy tylko stworzone przekątne). Ponadto pokazuje nam on czy algorytm poprawnie radzi sobie gdy każdy kolejny rozważany punkt należy do innego łańcucha.

Wyniki:

`is_y_monotonic`: `True`



Rysunek 2: Wynik `color_vertex` dla wielokąta A



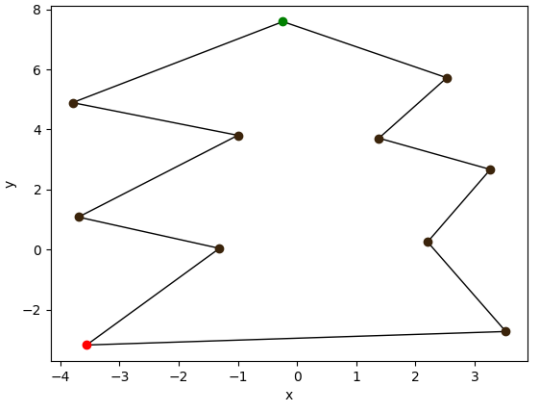
Rysunek 3: Wynik `triangulation` dla wielokąta A

3.2 Wielokąt B

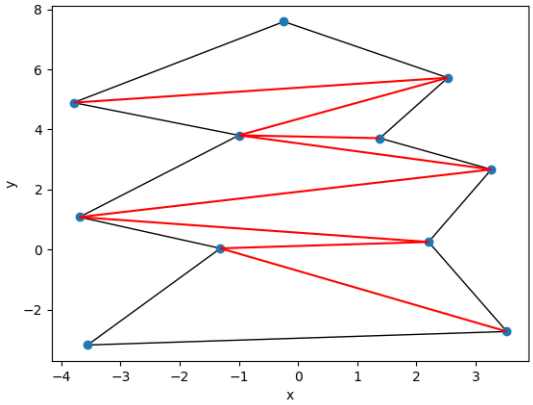
Wielokąt ten pozwala nam sprawdzić czy algorytm triangulacji poprawnie radzi sobie w przypadku, gdy kolejno rozważane wierzchołki należą do różnych łańcuchów.

Wyniki:

`is_y_monotonic`: `True`



Rysunek 4: Wynik `color_vertex` dla wielokąta B



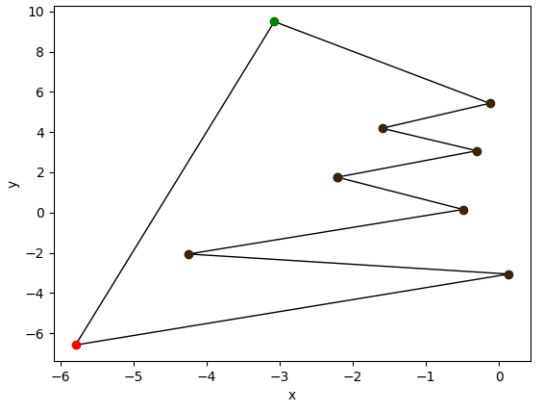
Rysunek 5: Wynik `triangulation` dla wielokąta B

3.3 Wielokąt C

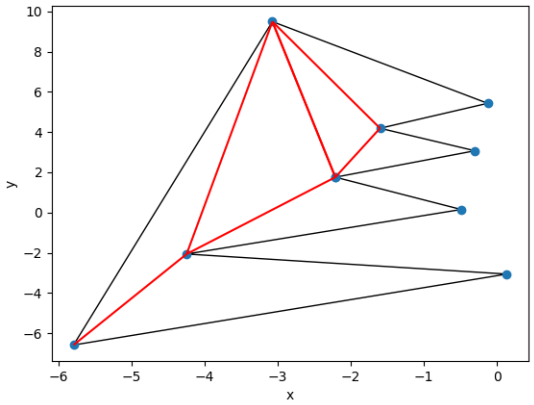
Wielokąt ten pozwala nam sprawdzić czy algorytm triangulacji poprawnie radzi sobie w przypadku, gdy większość wierzchołków leży na jednym łańcuchu, w którym tworzą trójkąty należące do wielokąta.

Wyniki:

`is_y_monotonic`: `True`



Rysunek 6: Wynik `color_vertex` dla wielokąta C



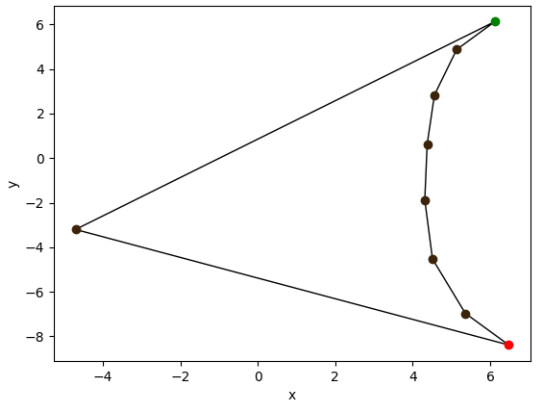
Rysunek 7: Wynik `triangulation` dla wielokąta C

3.4 Wielokąt D

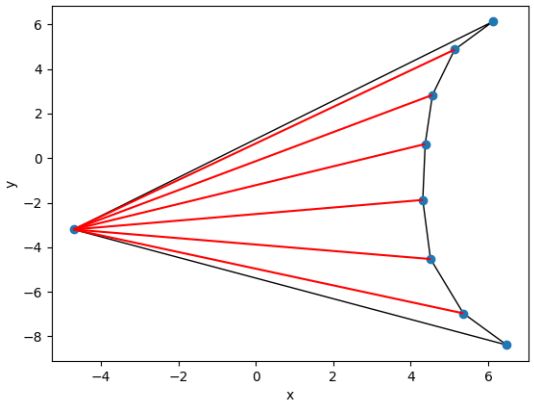
Wielokąt ten pozwala nam sprawdzić czy algorytm triangulacji poprawnie radzi sobie w przypadku, gdy większość wierzchołków leży na jednym łańcuchu, w którym tworzą trójkąty nie należące do wielokąta.

Wyniki:

`is_y_monotonic`: `True`



Rysunek 8: Wynik `color_vertex` dla wielokąta D



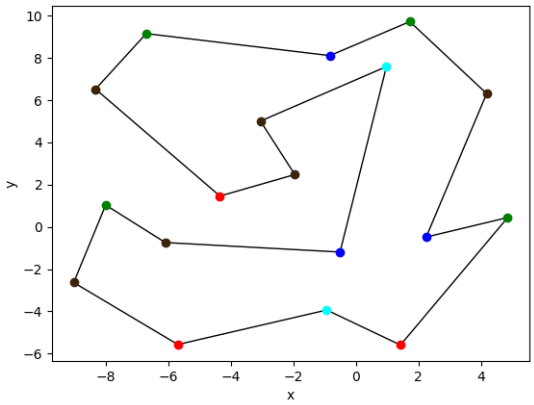
Rysunek 9: Wynik `triangulation` dla wielokąta D

3.5 Wielokąt E

Wielokąt ten pozwala nam sprawdzić czy wielokąt niemonotoniczny zostanie poprawnie sklasyfikowany i czy każdy typ wierzchołka jest odpowiednio rozpoznawany.

Wyniki:

`is_y_monotonic`: `False`



Rysunek 10: Wynik `color_vertex` dla wielokąta E

4 Podsumowanie

Algorytmy zostały zaimplementowane prawidłowo, zarówno przechodząc wszystkie testy jak i pokrywając się z przewidywaniami po wizualizacji. Wielokąty przedstawione w sprawozdaniu zostały wybrane w celu wizualizacji działania algorytmu i sprawdzenia jego poprawności w róznych przypadkach, opisanych w sprawozdaniu. Wygenerowane przez program wyniki zawarte są także w załączonym kodzie, który generuje także listy znalezionych trójkątów dla każdego zbioru (reprezentowanych jako trójki indeksów wierzchołków). Kod zawiera także funkcjonalność zadawania własnych wielokątów myszką i sprawdzania na nich wyników wszystkich opisywanych algorytmów. Wraz ze sprawozdaniem załączone są animacje przedstawiające poszczególne kroki algorytmu triangulacji dla wielokątów A, B, C i D.