Laboratorium 3 - Triangulacja wielokątów monotonicznych

Mateusz Podmokły - II rok Informatyka WI

15 listopad 2023

1 Specyfikacja użytego środowiska

Specyfikacja:

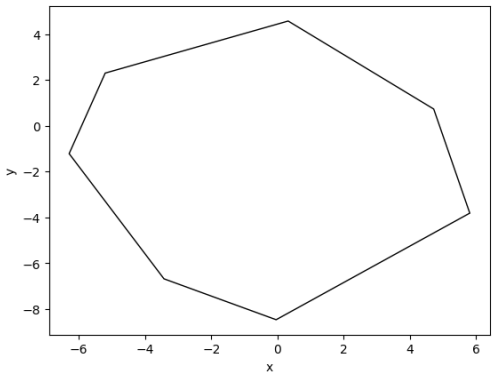
- Środowisko: Jupyter Notebook,
- Język programowania: Python,
- System operacyjny: Microsoft Windows 11,
- Architektura systemu: x64.

2 Przebieg ćwiczenia

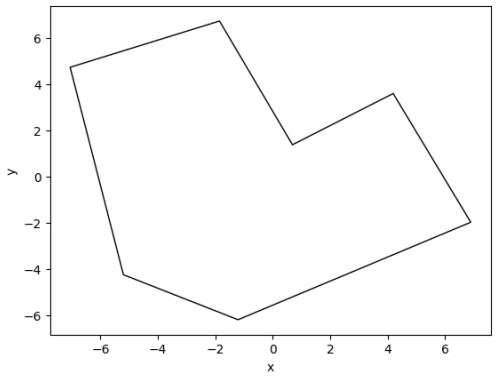
Ćwiczenie polega na zaimplementowaniu algorytmu sprawdzania, czy podany wielokąt jest wielokątem y-monotonicznym oraz algorytmu triangulacji takich wielokątów. Dodatkowo napisane zostało narzędzie pozwalające zadawać wielokąty przy użyciu myszki.

2.1 Sprawdzanie y-monotoniczności

Y-monotoniczność została sprawdzona poprzez analizę kolejnych wierzchołków. Jeżeli więcej niż jeden raz odwrócona była monotoniczność współrzędnej y wierzchołków, znaczyło to, że wielokąt nie jest y-monotoniczny.



Rysunek 1: Wielokąt y-monotoniczny.

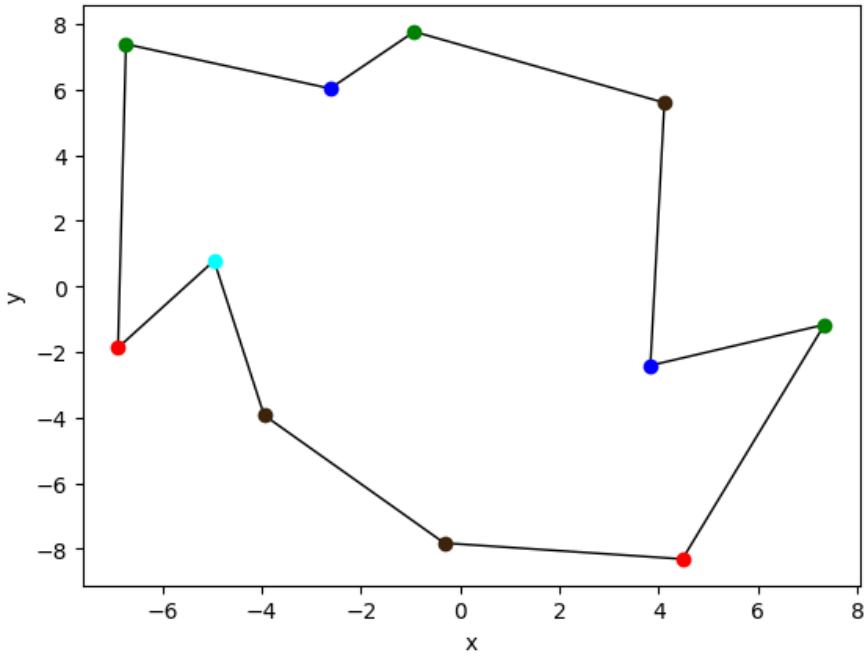


Rysunek 2: Nie jest y-monotoniczny.

2.2 Klasyfikacja wierzchołków wielokąta

Wierzchołki wielokąta zostały podzielone na 5 kategorii:

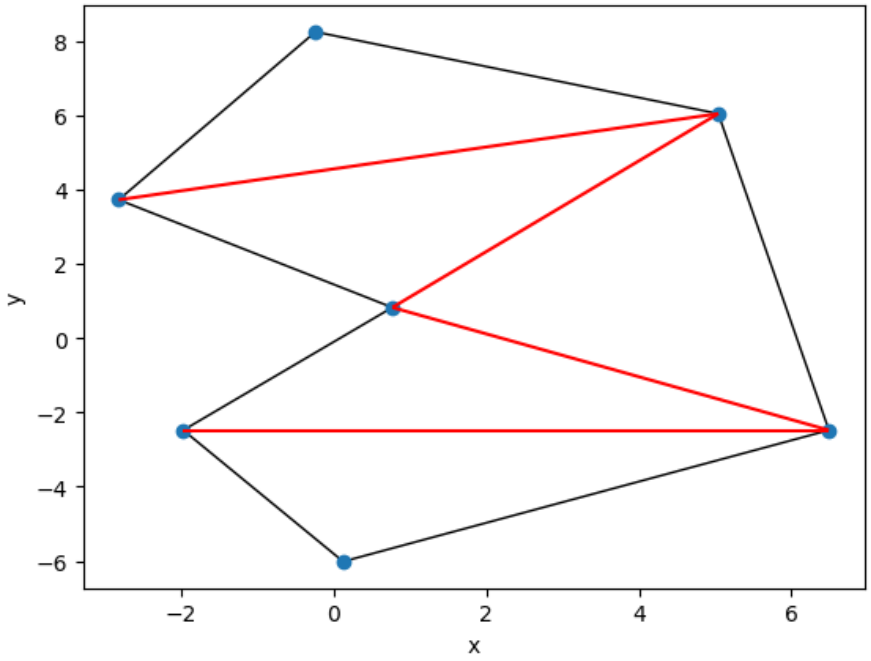
- początkowe, gdy obaj sąsiedzi leżą poniżej i kąt wewnętrzny ma mniej niż 180 stopni
- końcowe, gdy obaj sąsiedzi leżą powyżej i kąt wewnętrzny ma mniej niż 180 stopni
- dzielący, gdy obaj sąsiedzi leżą poniżej i kąt wewnęntrzny ma więcej niż 180 stopni
- łączący, gdy obaj sąsiedzi leżą powyżej i kąt wewnęntrzny ma więcej niż 180 stopni
- prawidłowe, pozostałe przypadki, jeden sąsiad powyżej, drugi poniżej



Rysunek 3: Wielokąt z pokolorowanymi wierzchołkami.

2.3 Triangulacja wielokąta

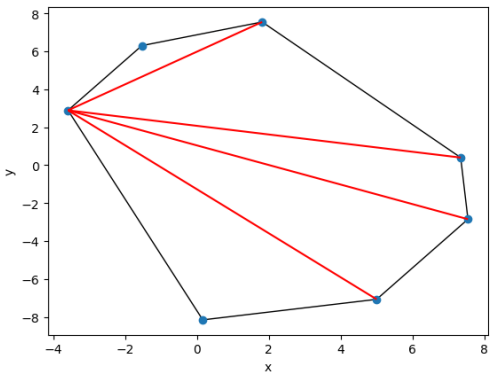
Wielokąt został podzielony na dwa łańcuchy wierzchołków - lewy i prawy. Następnie oba łańcuchy zostały połączone w jeden, co dało ciąg wierzchołków posortowanych według współrzędnej y w czasie $O(n)$, gdzie n to liczba wierzchołków. W celu znajdowania sąsiadów danego wierzchołka utworzona została tablica oryginalnej kolejności wierzchołków w wielokącie (w programie nazwana *order*). Triangulacja jest przechowywana w postaci listy par indeksów wierzchołków między którymi występują połączenia tworzące triangulację. Wybrane struktury przechowujące wielokąt oraz triangulację pozwoliły na uzyskanie złożoności obliczeniowej całości algorytmu $O(n)$.



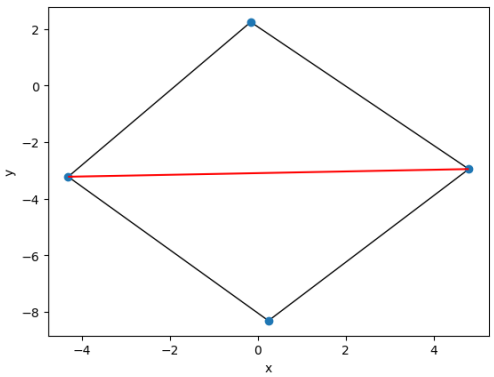
Rysunek 4: Przykładowa triangulacja.

3 Analiza otrzymanych triangulacji

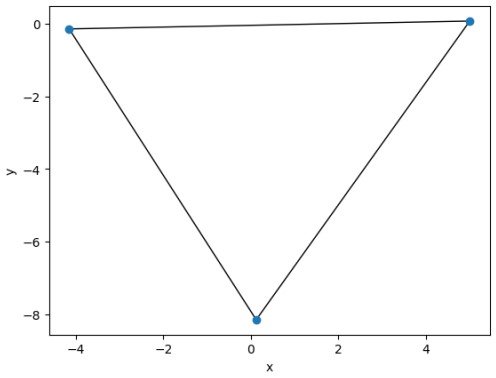
3.1 Wielokąty użyte do testów



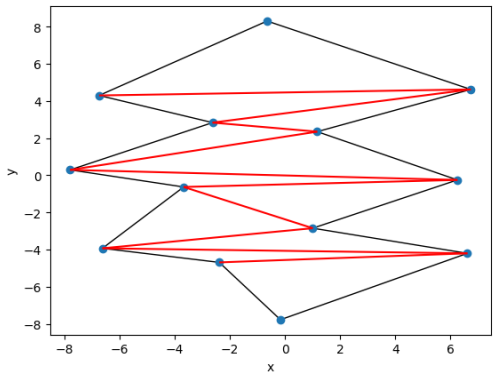
Rysunek 5



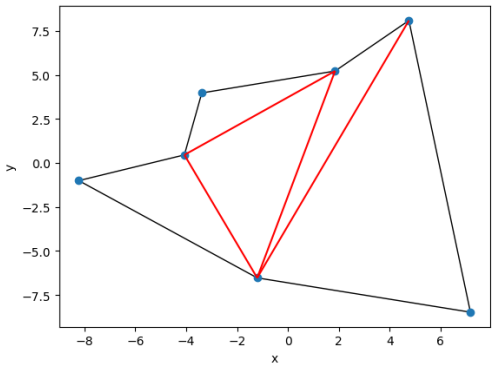
Rysunek 6



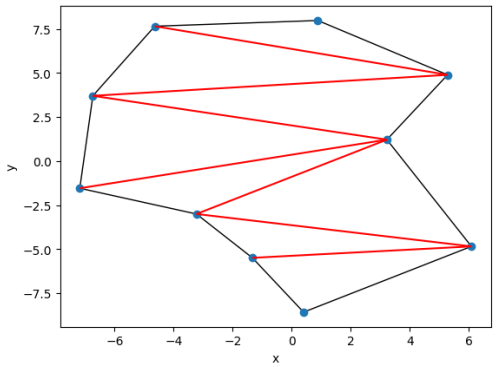
Rysunek 7



Rysunek 8



Rysunek 9



Rysunek 10

3.2 Analiza wyników

Pierwszy wielokąt (rys. 5) posiada wszystkie punkty lewego łańcucha powyżej prawego, dlatego najniższy punkt lewego łańcucha jest połączony ze wszystkimi z prawego. W drugim (rys. 6) znajduje się tylko jeden odcinek podziałowy, ze względu na to, że ma tylko 4 boki. Trzeci wielokąt (rys. 7) jest trójkątem, więc nie wymaga żadnych podziałów. W następnym (rys. 8) kolejne wierzchołki pojawiają się naprzemiennie w łańcuchu prawym oraz lewym, dlatego podziały przechodzą po kolei od góry do dołu. Rysunek 9 zawiera wielokąt z wklęsłościami, który wymaga "odcięcia"wklęsłości. Ostatni (rys. 10) to połączenie naprzemiennych podziałów z wielokrotnymi podziałami do niższych wierzchołków.

4 Wnioski

Użyte wielokąty pozwoliły przetestować algorytm triangulacji na różnych rodzajach wielokątów, dając odmienne efekty podziałów na trójkąty. Warto zaznaczyć, że wszystkie

były y-monotoniczne. Zastosowany algorytm jest wydajny bez względu na rodzaj triangulowanego wielokąta, ponieważ posiada złożoność obliczeniową $O(n)$. Jednak nadaje się on jedynie do wielokątów y-monotonicznych, co może być sporym ograniczeniem.