Wojciech Michaluk grupa nr 6 godz. 13:00-14:30 czw. A

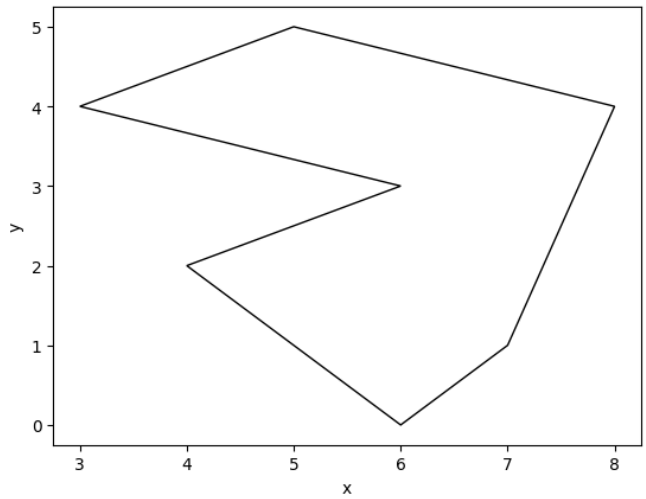
Sprawozdanie z laboratorium nr 3: Triangulacja wielokątów monotonicznych

Data wykonania: 16.11.2023r. Data oddania: 02.12.2023r.

1. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z zagadnieniami monotoniczności wielokątów oraz implementacja algorytmów: sprawdzania, czy wielokąt jest y-monotoniczny; klasyfikacji wierzchołków w dowolnym wielokącie; triangulacji wielokąta monotonicznego, a także wizualizacja i opracowanie wyników.

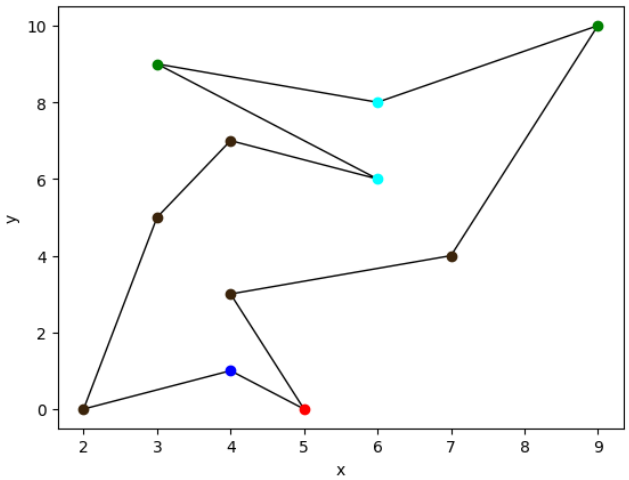
2. Wstęp teoretyczny



Rys. 2.1 Przykładowy wielokąt monotoniczny

Formalnie mówimy o wielokącie monotonicznym względem pewnej prostej l. W tym zadaniu będzie nas interesować **y-monotoniczność** (względem osi OY). Wielokąt nazywamy y-monotonicznym, gdy jego wierzchołki mogą być ułożone w taki sposób, że jego współrzędna y-owa zawsze rośnie lub maleje wzdłuż kolejnych wierzchołków. Innymi słowy, dla każdej pary sąsiednich wierzchołków wielokąta (oprócz wierzchołka startowego i końcowego), jeden z punktów ma większą (lub mniejszą) wartość danej współrzędnej niż drugi.

Kategorie wierzchołków:
1) początkowe, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny ma mniej niż 180^o
2) końcowe, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny ma mniej niż 180^o
3) dzielący, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny ma więcej niż 180^o
4) łączący, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny ma więcej niż 180^o
5) prawidłowy - pozostałe przypadki (jeden sąsiad powyżej, drugi poniżej)

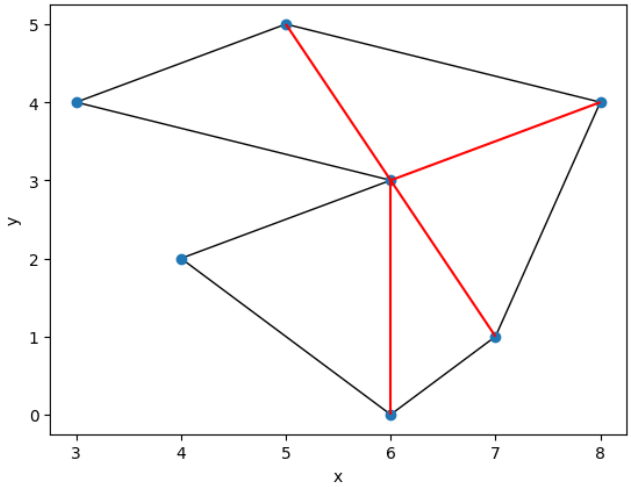


Rys. 2.2 Przykładowe kolorowanie wierzchołków

Zarówno na rysunku 2.2, jak i w pliku z implementacją jest przyjęta następująca konwencja kolorowania w zależności od kategorii wierzchołka: 1 – zielony, 2 – czerwony, 3 – ciemnoniebieski, 4 – jasnoniebieski, 5 – czarny.

Wierzchołki startowe i końcowe są ważne w kontekście algorytmów przetwarzania wielokątów monotonicznych, takich jak algorytmy dziel i zwyciężaj oraz triangulacji. Wierzchołki łączące i dzielące odgrywają kluczową rolę w procesie triangulacji wielokątów, pozwalając na podział figury na trójkąty w sposób bezkolizyjny. Wielokąt monotoniczny charakteryzuje się tym, że nie ma wierzchołków dzielących ani łączących. Czym jest wspomniana triangulacja?

Triangulacja wielokąta monotonicznego to proces podziału wielokąta monotonicznego na trójkąty poprzez dodawanie odpowiednich przekątnych wewnętrznych, które się nie przecinają (poza wspólnymi końcami – wierzchołkami).



Rys 2.3 Triangulacja przykładowego wielokąta monotonicznego

3. Metodologia, specyfikacja narzędzi i sprzętu

Aby móc zadawać wielokąty przy użyciu myszki, skorzystałem z pakietu funkcji oferowanych przez bibliotekę matplotlib. W algorytmie podziału wierzchołków na kategorie oraz triangulacji do sprawdzania, czy trzy punkty tworzą odpowiedni układ (lewo- / prawostronny skręt) używam wyznacznika macierzy 3x3, przedstawionego na laboratorium nr 1. Wykresy przedstawiające przykładowe i „wygenerowane” wielokąty oraz podział ich wierzchołków na kategorie, jak i triangulacja wraz z wizualizacją poszczególnych jej kroków, powstały przy użyciu biblioteki matplotlib oraz dzięki narzędziu przygotowanemu przez koło naukowe Bit.

Wielokąt przechowuję w postaci listy krotek współrzędnych (x,y) jego wierzchołków, uporządkowanych przeciwnie do ruchu wskazówek zegara. Taka reprezentacja jest szczególnie pomocna przy algorytmach implementowanych w ramach tego laboratorium, ponieważ mamy łatwy dostęp do sąsiadów danego wierzchołka, z którymi wykonujemy odpowiednie porównania w zależności od algorytmu.

Pomocniczą strukturą do przechowywania triangulacji, która występuje w pliku z implementacją jest tablica krotek indeksów wierzchołków, pomiędzy które należy dodać przekątne, aby striangulować badany wielokąt monotoniczny. Na przykład [(1,5),(2,3)] oznacza, że w celu striangulowania należy dodać przekątne pomiędzy wierzchołki o indeksach 1 i 5 oraz 2 i 3. Można ją z łatwością wykorzystać, aby wygenerować odpowiedni rysunek przedstawiający triangulację. Sama triangulacja może być przechowywana w następujący sposób: poprzez tablicę wierzchołków (tablica krotek postaci (indeks wierzchołka, krotka współrzędnych)); wspomnianą tablicę dodanych „przekątnych" oraz tablicę krawędzi tworzących trójkąty w triangulacji (krotki trzyelementowe, każdy element to krotka przedstawiająca indeksy wierzchołków, między którymi jest krawędź).

Program (w pliku „michaluk_kod_3.ipynb") jest napisany w języku Python w środowisku Jupyter Notebook. Przedstawione wyniki pochodzą z uruchomienia programu na komputerze z systemem Windows 11 i procesorem Intel Core i5-8300H 2.30 GHz.

4. Program ćwiczenia

Najpierw zaimplementowałem funkcje umożliwiające zadawanie wielokątów za pomocą myszki. Zgodnie z opisem, który znajduje się w pliku z implementacją, wystarczy odpowiednio kliknąć myszką aby dodać nowy wierzchołek we wskazanym punkcie / zakończyć wprowadzanie wielokąta. Wielokąt zadajemy w kierunku przeciwnym do ruchu wskazówek zegara, ponieważ do takiej postaci dostosowane są funkcje. Użytkownik ma możliwość wygenerowania wielokąta i testowania funkcji na nim, ale oprócz tego przygotowałem wielokąty testowe, które opisane są w następnym rozdziale.

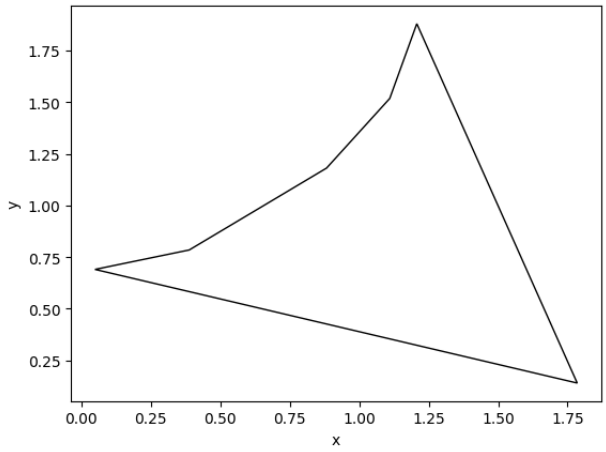
Następnie zaimplementowałem procedurę sprawdzającą, czy podany wielokąt jest y-monotoniczny. Pełna implementacja znajduje się w pliku .ipynb, tutaj krótko opiszę, na czym się ta procedura opiera.

Najpierw wyznaczam lewy i prawy łańcuch monotoniczności wielokąta. W tym celu znajduję skrajne wierzchołki (górny i dolny, dla takich samych współrzędnych y-owych – przyjmuję pierwszy napotkany). Korzystając z faktu, że wierzchołki w wielokącie są podane w kierunku przeciwnym do ruchu wskazówek zegara, wyznaczenie łańcuchów jest stosunkowo proste. Lewy łańcuch zawiera wierzchołki w kierunku przeciwnym do ruchu wskazówek zegara, a prawy – zgodnym (aby potencjalnie współrzędne y-owe malały). Pozostaje przejść po każdym z łańcuchów, sprawdzając, czy cały czas idę w kierunku malejących współrzędnych y-owych, tzn. czy po drodze nie „cofam" się – nie idę nagle znowu w górę. Jeżeli taka sytuacja nastąpi, to wielokąt nie jest y-monotoniczny. Jeżeli nie następuje ani razu, to jest.

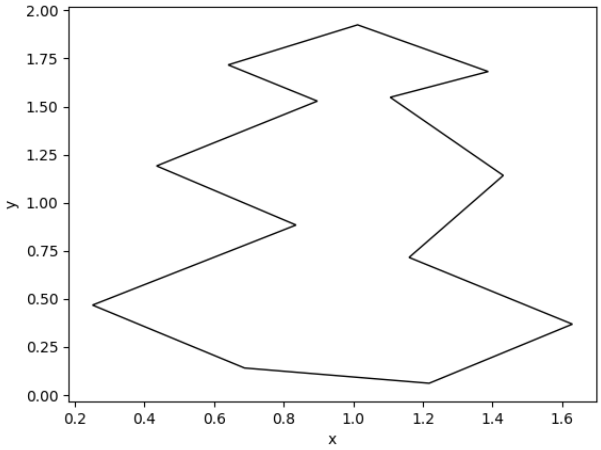
Pozostało zaimplementować funkcję dzielącą wierzchołki na kategorie oraz dokonującą triangulacji. Wszystkie kroki triangulacji są przedstawione w postaci gifów w pliku z implementacją, tutaj umieszczę wybrane „klatki" tego algorytmu dla testowanych wielokątów. Jakiego typu wielokąty będą testowane? Przedstawiam je poniżej.

5. Testowane zbiory danych i wyniki działania algorytmów

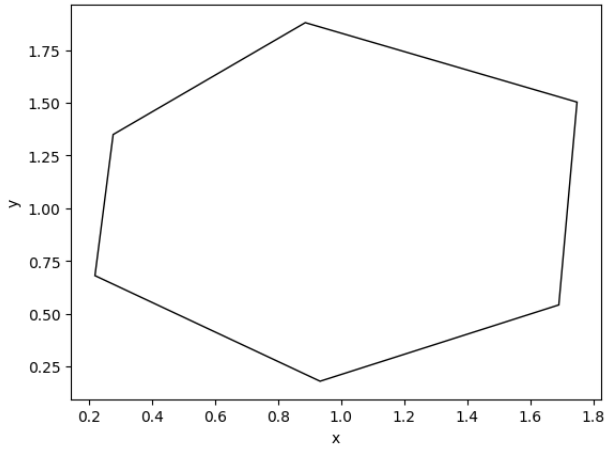
Aby testy były rzetelne, starałem się dobrać wielokąty różnej natury, które mogłyby potencjalnie sprawić problemy programowi / przetestować przypadki brzegowe. Na następnych stronach przedstawiam dobrane **pięć** wielokątów, dla których przetestuję działanie zaimplementowanych w ramach tego laboratorium algorytmów.



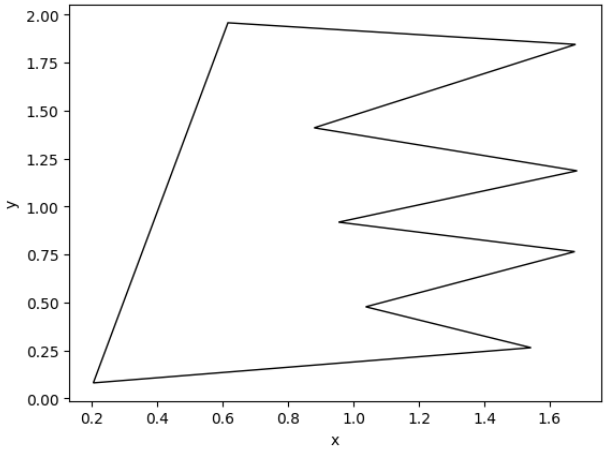
Rys. 5.1 Pierwszy testowany wielokąt – A



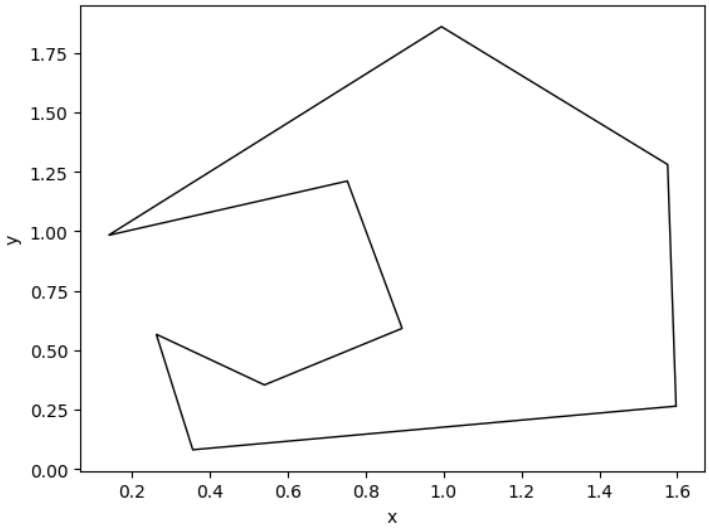
Rys 5.2 Drugi testowany wielokąt - B



Rys. 5.3 Trzeci testowany wielokąt - C



Rys. 5.4 Czwarty testowany wielokąt – D



Rys. 5.5 Piąty testowany wielokąt – E (uwaga, czy jest on monotoniczny?)

W poniższej tabeli przedstawiam wyniki działania funkcji sprawdzającej monotoniczność wielokąta – True, jeśli jest y-monotoniczny, False w przeciwnym przypadku.

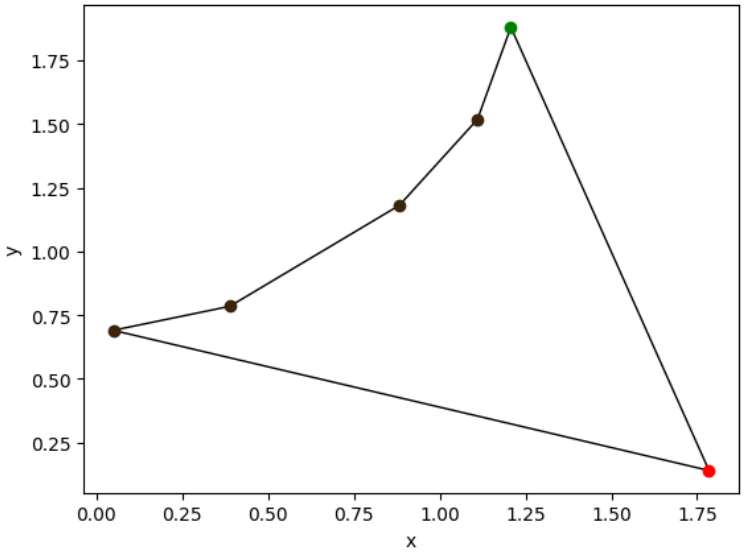
Testowany wielokąt	Wielokąt A	Wielokąt B	Wielokąt C	Wielokąt D	Wielokąt E
Czy monotoniczny?	True	True	True	True	False

Tabela 5.6 Sprawdzenie monotoniczności wielokątów

Jak widać w tabeli 5.6, istotnie ostatni wielokąt nie jest y-monotoniczny, ale pozostałe są.

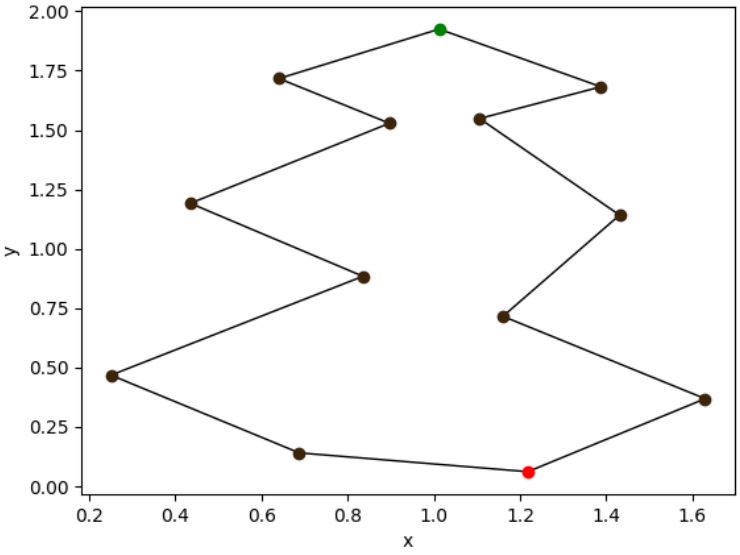
Teraz przedstawię podział wierzchołków na kategorie dla każdego z testowanych wielokątów. Konwencja kolorowania została wspomniana wcześniej, w rozdziale 2.

Wielokąt A



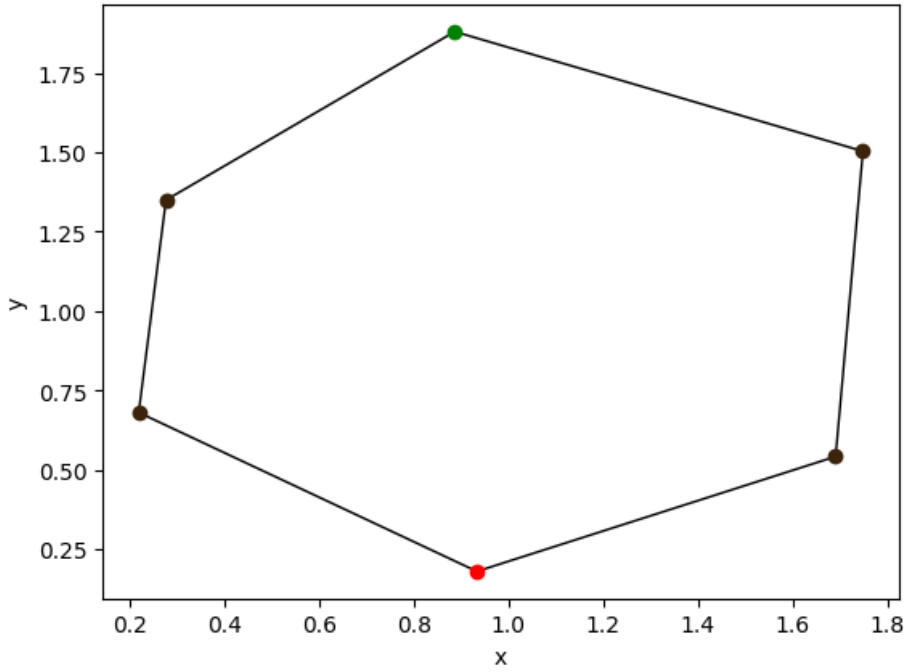
Rys. 5.7 Kolorowanie wierzchołków dla wielokąta A

Wielokąt B



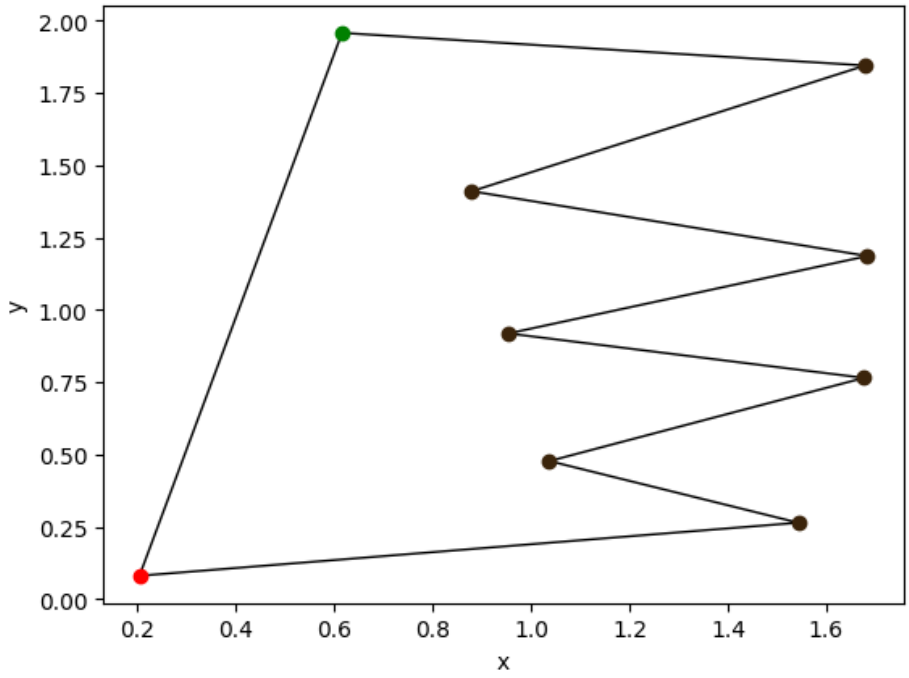
Rys. 5.8 Kolorowanie wierzchołków dla wielokąta B

Wielokąt C



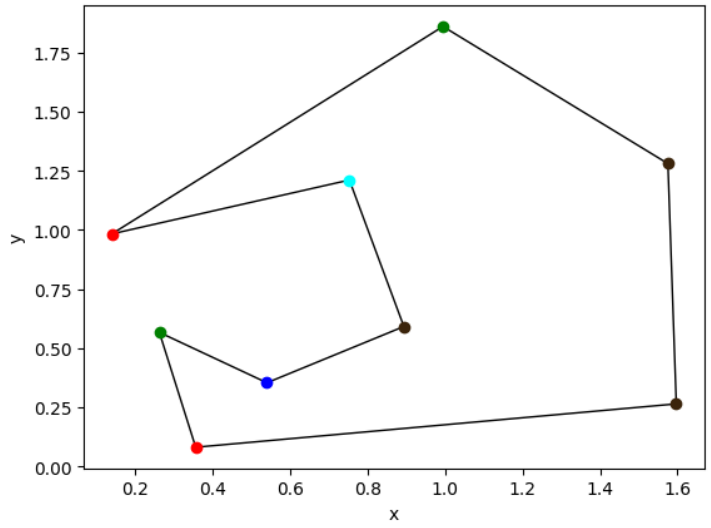
Rys. 5.9 Kolorowanie wierzchołków dla wielokąta C

Wielokąt D



Rys. 5.10 Kolorowanie wierzchołków dla wielokąta D

Wielokąt E

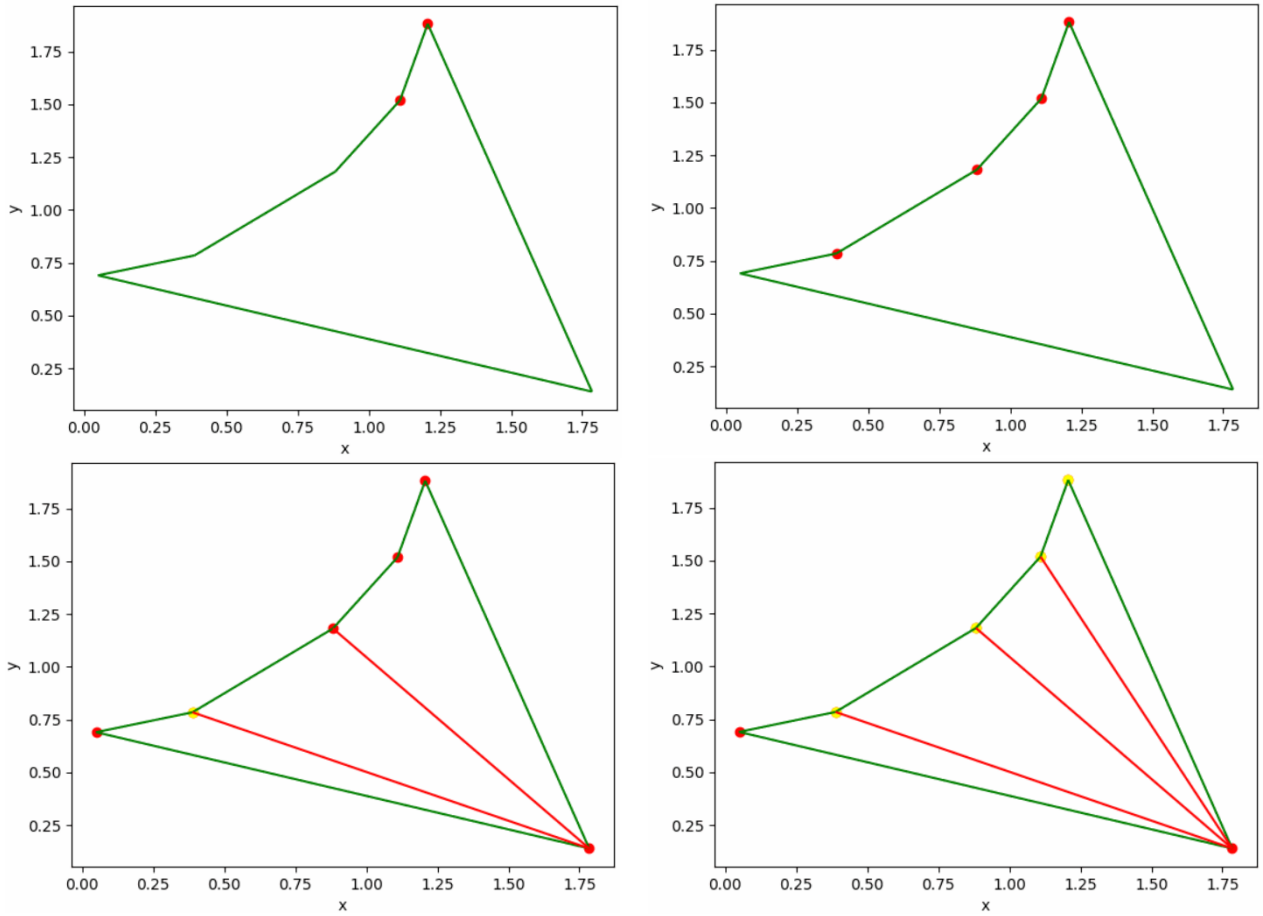


Rys. 5.11 Kolorowanie wierzchołków dla wielokąta E

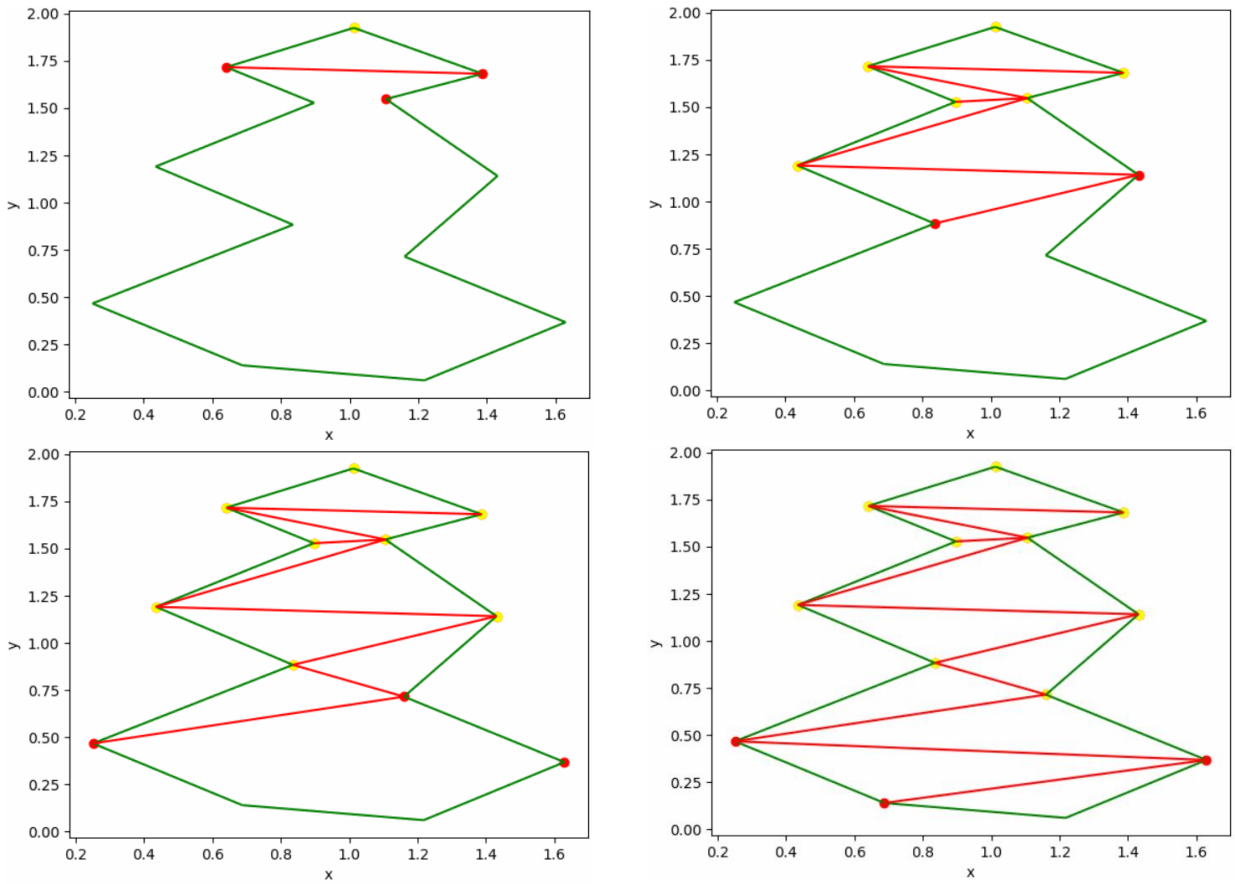
Jak widać, ten wielokąt, jako niemonotoniczny, jest jedynym, który ma wierzchołki dzielące i łączące. Pozostałe wielokąty mają jedynie wierzchołki początkowe, końcowe i prawidłowe.

Triangulacja wielokątów (monotonicznych)

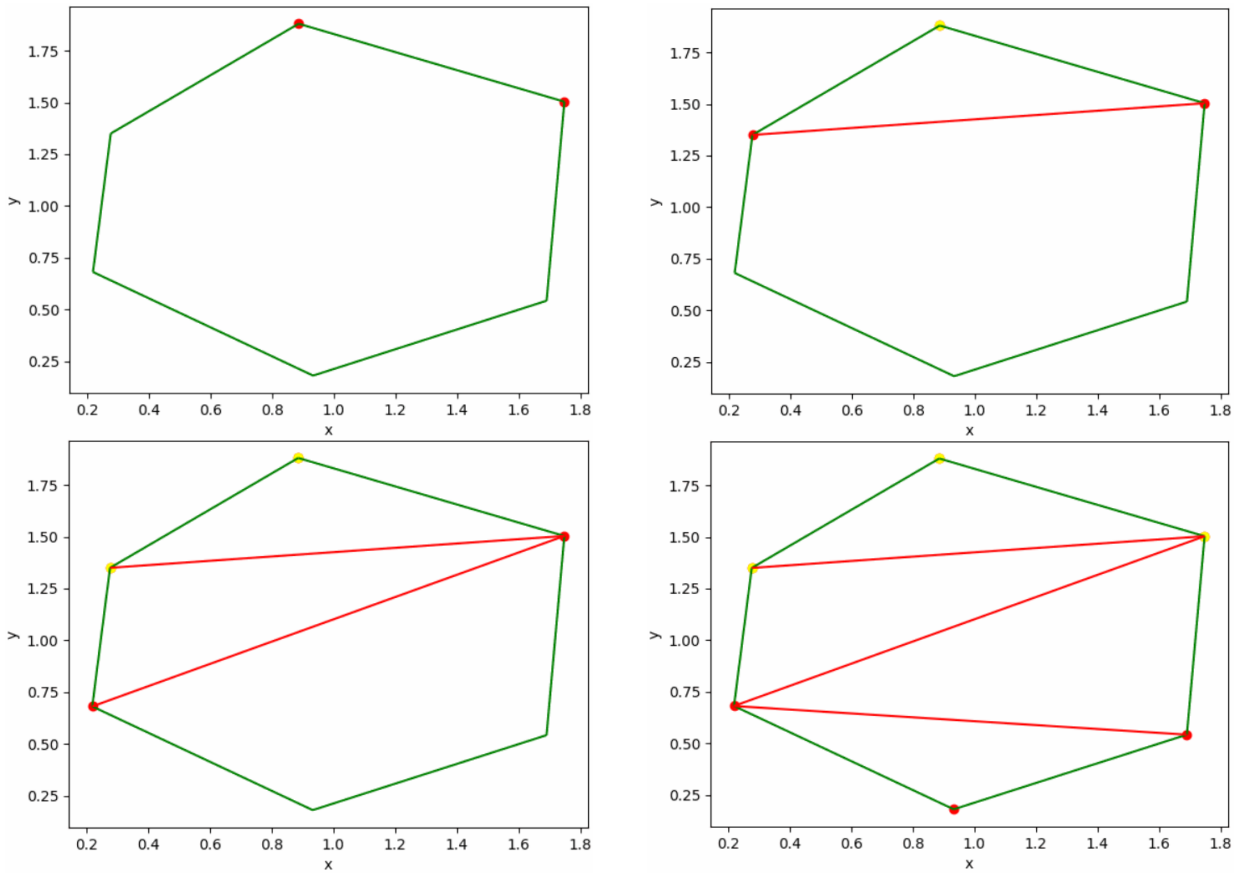
Dla każdego z wielokątów A, B, C oraz D przedstawię cztery wybrane kroki w procesie ich triangulowania. Dla wielokąta E triangulacja nie jest możliwa bez wcześniejszego podziału go na wielokąty monotoniczne, więc tu nie zostanie uwzględniona. Kolorem czerwonym zaznaczone są dodane przekątne oraz wierzchołki znajdujące się aktualnie na stosie, natomiast kolorem żółtym wyróżnione są wierzchołki, które były na stosie, ale już zostały zdjęte.



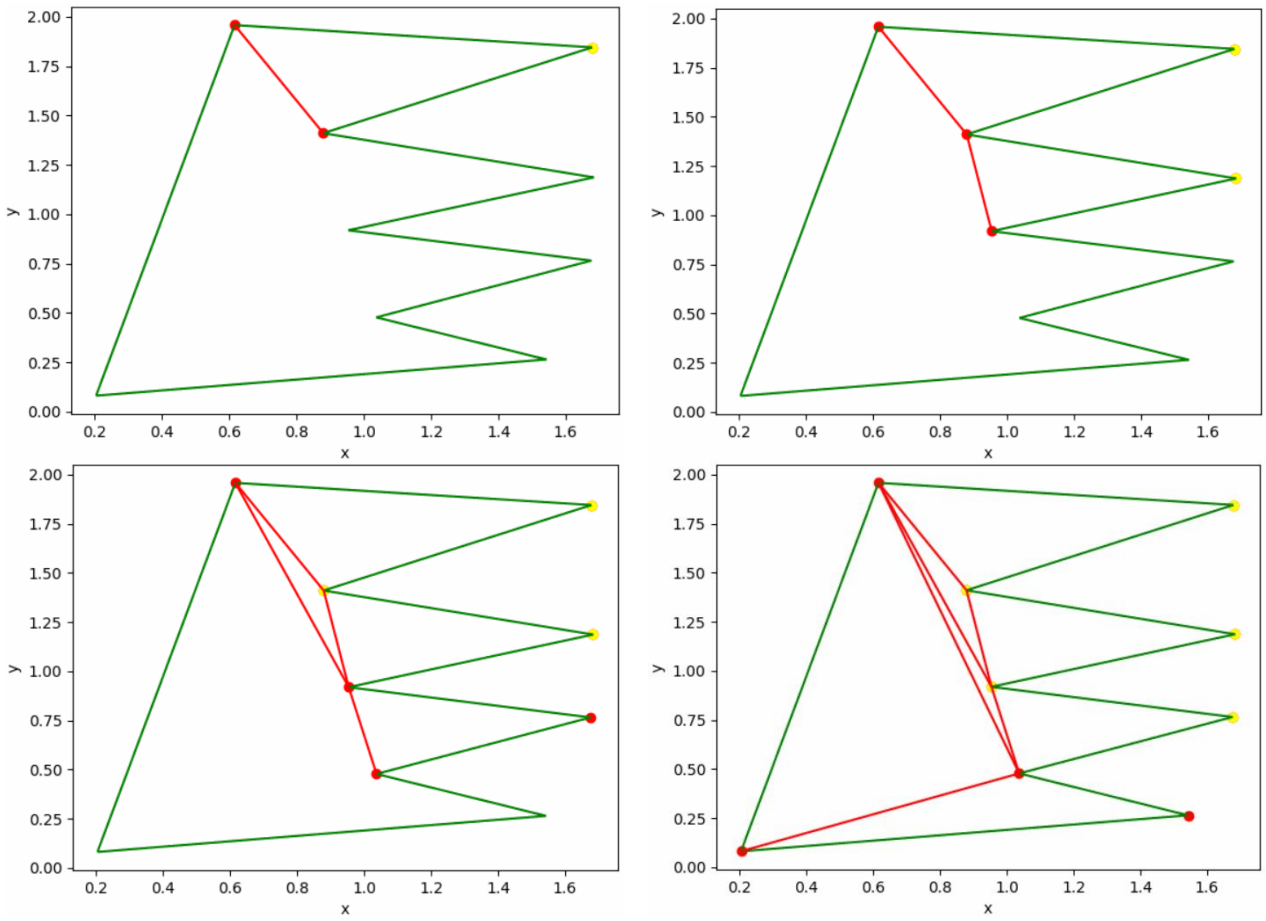
Rys. 5.12 Wybrane etapy triangulacji wielokąta A



Rys. 5.13 Wybrane etapy triangulacji wielokąta B



Rys. 5.14 Wybrane etapy triangulacji wielokąta C



Rys. 5.15 Wybrane etapy triangulacji wielokąta D

Gify przedstawiające kolejno wszystkie etapy („klatka po klatce") triangulacji oraz efekt końcowy znajdują się w pliku z implementacją (.ipynb).

6. Podsumowanie i wnioski

Testowane wielokąty miały różną specyfikę. Dobrałem następujące wielokąty, aby móc wystawić program na próbę – czy w procesie triangulacji nie zostanie dodana jakaś nieprawidłowa przekątna, czyli poprowadzona poza wielokątem, pokrywająca się z bokiem albo przecinająca inną już dodaną przekątną – wtedy nie uzyskamy odpowiedniej triangulacji. Zarówno dla przyjętych wielokątów testowych, jak i dla wielokątów w testach zaproponowanych przez Koło Naukowe AGH Bit, program radził sobie dobrze, co widać także na rysunkach wygenerowanych dla triangulacji, na których można przenalizować utworzone trójkąty. Oczywiście program bazował na algorytmie opisanym na wykładzie, więc nie był dostosowany dla wielokątów niemonotonicznych – dla nich zwracałby nieprawidłową triangulację.

W poprawnym działaniu algorytmu pomogła przyjęta konwencja przechowywania wielokąta oraz pomocnicza struktura do triangulacji – dobrze współpracowała ona z algorytmem triangulowania. Aby triangulacja była transferowalna, należy przechowywać ją w pełnej strukturze, wymienionej w rozdziale 3.