Algorytmy geometryczne

Sprawozdanie z laboratorium 3

Triangulacja wielokątów monotonicznych

Iwo Zowada

Gr. 6 Wtorek 15:00-16:30 A

Dane techniczne

Komputer z systemem Windows 10
Procesor Intel Core i7-3770 CPU 3.40GHz
Pamięć RAM 24GB
Program napisany w języku Python w środowisku Jupyter Notebook z wykorzystaniem bibliotek numpy, random, matplotlib oraz narzędzia przygotowanego przez KN Bit

Opis ćwiczenia

Celem ćwiczenia jest zapoznanie się z zagadnieniami dotyczącymi monotoniczności wielokątów, ze szczególnym uwzględnieniem algorytmu służącego do sprawdzania, czy dany wielokąt jest monotoniczny, algorytmu klasyfikacji wierzchołków w wielokątach, oraz algorytmu triangulacji dla wielokątów monotonicznych, a także wizualizacja otrzymanych wyników.

Realizacja ćwiczenia

W ramach 3 laboratorium należało wykonać cztery niezbędne zadania:

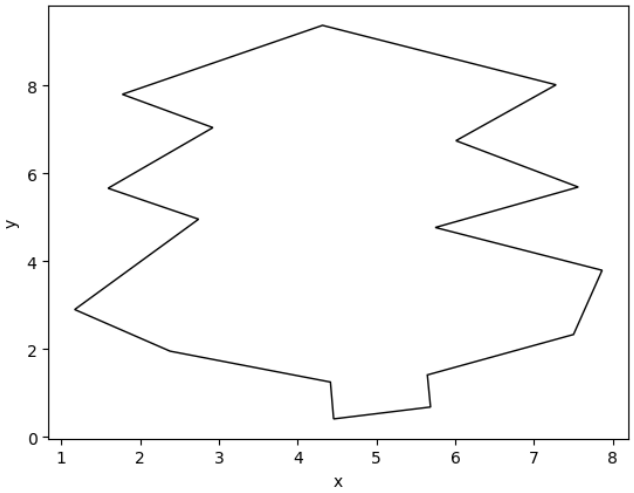
1. Dostosować aplikację graficzną tak, aby można było zadawać proste wielokąty przy użyciu myszki, z dodatkowym zapisem i odczytem podanych wielokątów.
2. Zaimplementować procedurę sprawdzającą, czy podany wielokąt jest y-monotoniczny.
3. Zaimplementować algorytm, który dla zadanego wielokąta będzie wyszukiwał wierzchołki początkowe, końcowe, łączące, dzielące i prawidłowe. Wierzchołki mają zostać odpowiednio pokolorowane zgodnie z klasyfikacją.

4. Zaimplementować procedurę triangulacji wielokąta monotonicznego (zgodnie z algorytmem opisanym na wykładzie). Program powinien pokazywać kolejne kroki algorytmu.

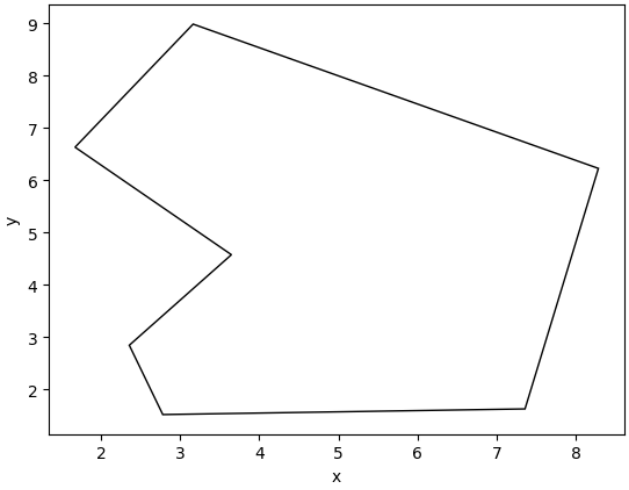
W pliku „Zowada_kod_3.ipynb” zamieszczam kod, realizujący wszystkie podpunkty, natomiast poniżej przedstawię cztery przykładowe wielokąty, które wygenerowałem i opisze dla nich działanie funkcji realizujących zadania z punktów 2,3 i 4.

1. Generowanie wielokątów

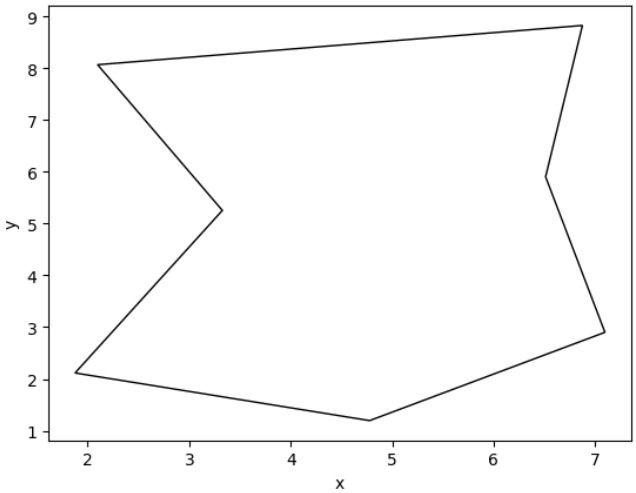
Działanie mojego programu przetestuje na tych czterech wielokątach:



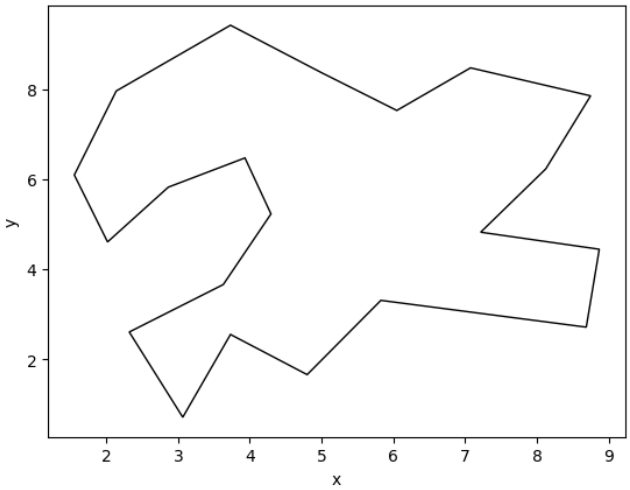
(Rys 1.1): *Wielokąt A*



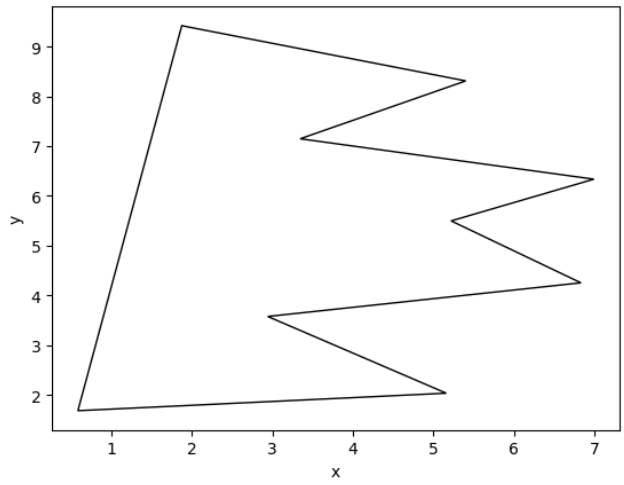
(Rys 1.2): *Wielokąt B*



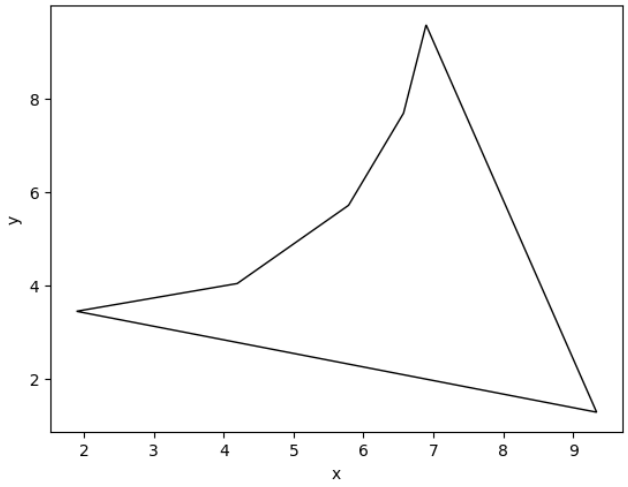
(Rys 1.3): *Wielokąt C*



(Rys 1.4): *Wielokąt D*



(Rys 1.5): *Wielokąt E*



(Rys 1.6): *Wielokąt F*

2. Sprawdzanie y-monotoniczności podanych wielokątów

Dla każdego z tych wielokątów musimy sprawdzić y-monotoniczność. Robimy to wyznaczając łańcuch lewy i prawy (są zadane przez najniższy i najwyższy punkt), a następnie przechodząc po nich sprawdzając czy w którymś momencie nasz łańcuch nie „cofa się" tworząc nam wielokąt niemonotoniczny. W tabelce poniżej przedstawiam klasyfikacje wielokątów względem tego czy są y-monotoniczne.

Wielokąt	A	B	C	D	E	F
Czy y-monotoniczny?	Tak	Tak	Tak	Nie	Tak	Tak

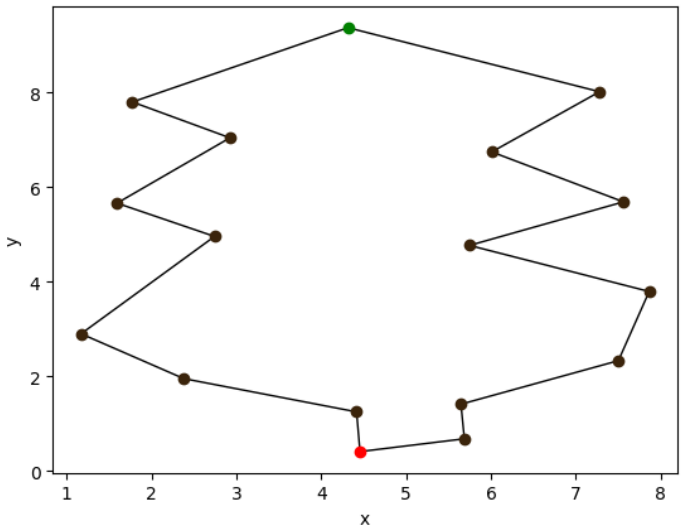
(Tabela 2.1): *Klasyfikacja wielokątów względem y-monotoniczności*

3. Kolorowanie wierzchołków wielokątów

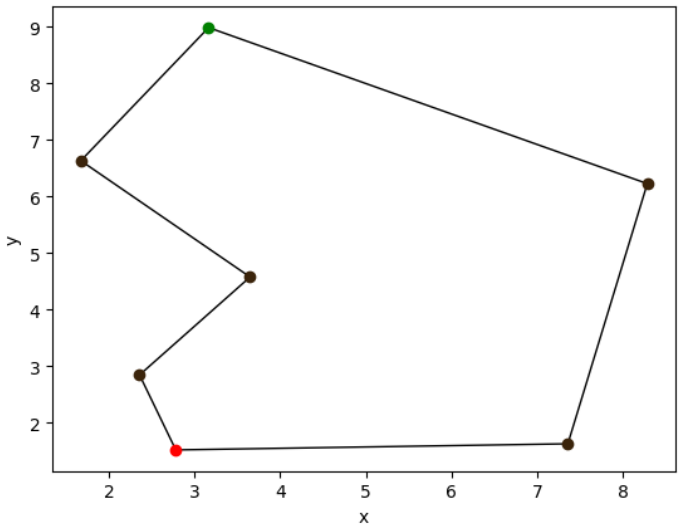
Program ma teraz pokolorować wierzchołki zadanego wielokąta z zależności od położenia jego sąsiadów oraz kąta wewnętrznego jaki z nimi tworzy. Poniżej przedstawiam warunki na których podstawie klasyfikujemy wierzchołki oraz kolory jakimi będziemy kolorować dane punkty.

- Początkowe - gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny ma mniej niż 180 stopni – kolor zielony
- Końcowe - gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny ma mniej niż 180 stopni – kolor czerwony

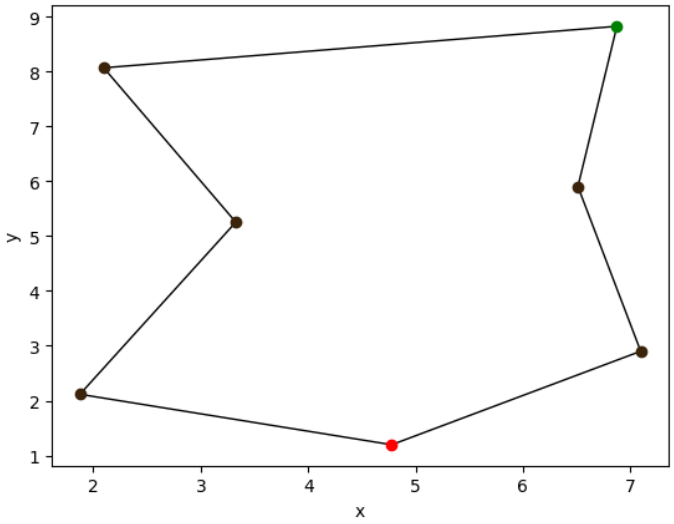
- Dzielące - gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny ma więcej niż 180 stopni – kolor niebieski
- Łączące - gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny ma więcej niż 180 stopni – kolor niebieskozielony
- Prawidłowe - pozostałe przypadki, jeden sąsiad powyżej, drugi poniżej – kolor brązowy



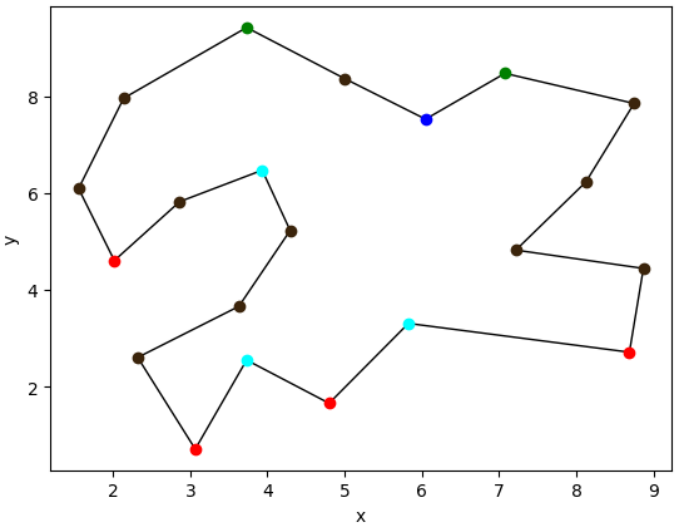
(Rys 3.1): *Kolorowanie Wielokąta A*



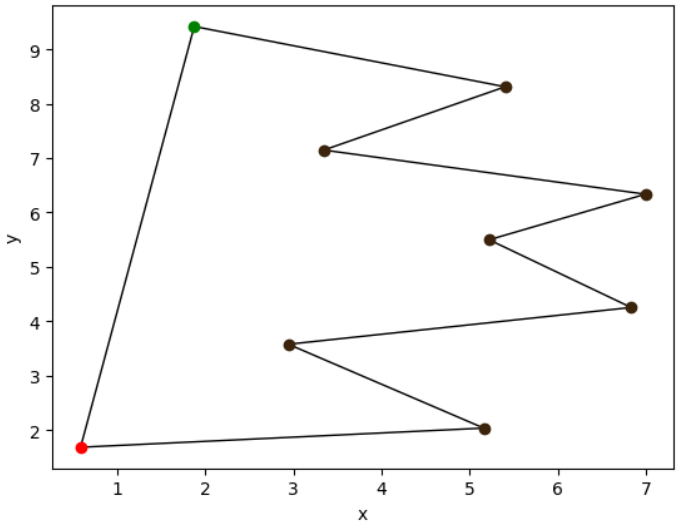
(Rys 3.2): *Kolorowanie Wielokąta B*



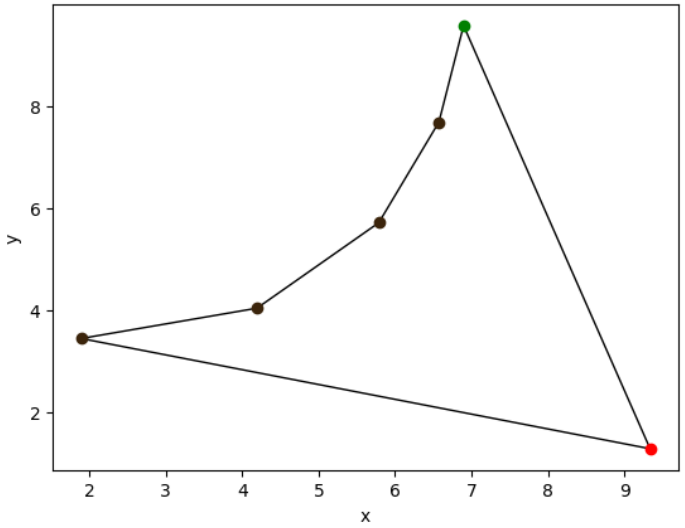
(Rys 3.3): *Kolorowanie Wielokąta C*



(Rys 3.4): *Kolorowanie Wielokąta D*



(Rys 3.5): *Kolorowanie Wielokąta E*



(Rys 3.6): *Kolorowanie Wielokąta F*

4. Triangulacja wielokątów monotonicznych

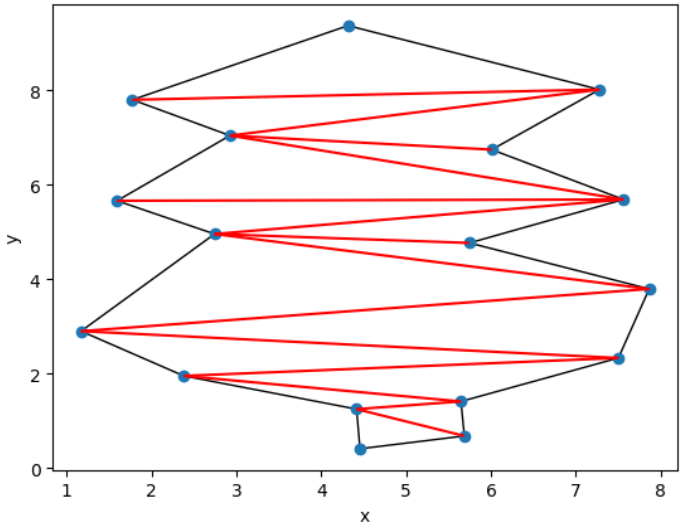
Moja implementacja algorytmu triangulacji wielokąta działa tylko dla wielokątów y-monotonicznych, więc pierwszym krokiem jest sprawdzenie czy taki on właśnie jest. Jeśli tak to postępujemy zgodnie z krokami opisanymi poniżej:

- Określamy lewy i prawy łańcuch wielokąta względem kierunku monotoniczności
- Porządkujemy wierzchołki wzdłuż kierunku monotoniczności
- Wkładamy dwa pierwsze wierzchołki na stos.
    - Jeśli kolejny wierzchołek należy do innego łańcucha niż wierzchołek stanowiący szczyt stosu, to możemy go połączyć ze wszystkimi wierzchołkami na stosie. Na stosie zostają dwa wierzchołki, które były „zamiatane” ostatnie.
    - Jeśli kolejny wierzchołek należy do tego samego łańcucha co wierzchołek ze szczytu stosu, to analizujemy kolejne trójkąty, jakie tworzy dany wierzchołek z wierzchołkami zdejmowanymi ze stosu:
        - Jeśli trójkąt należy do wielokąta, to dodajemy przekątną i usuwamy odpowiedni wierzchołek ze szczytu stosu. Aby można było zaliczyć ten trójkąt, muszą być spełnione dwa warunki:
            - Przekątna nie może leżeć na krawędzi – musimy sprawdzić czy indeksy wierzchołka aktualnego z dwoma które leżą na szczycie stosu nie różnią się o {0, 1, n-1},
            - Przekątna nie może leżeć na zewnątrz wielokąta – obliczając wyznacznik macierzy dla danych trzech punktów mówi nam to jaki skręt robimy (prawo lub lewostronny). Biorąc pod uwagę to na jakim łańcuchu znajduje się dany punkt

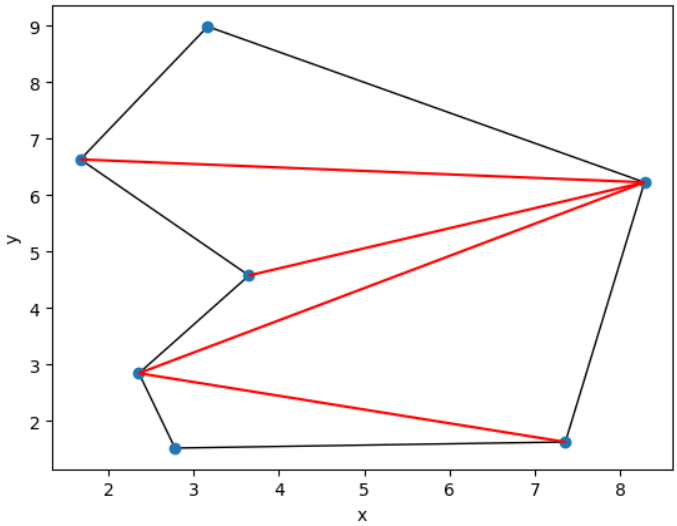
definiujemy czy przekątna będzie leżeć w środku czy na zewnątrz wielokąta.

- Jeśli trójkąt nie należy do wielokąta, to umieszczamy badane wierzchołki na stosie.

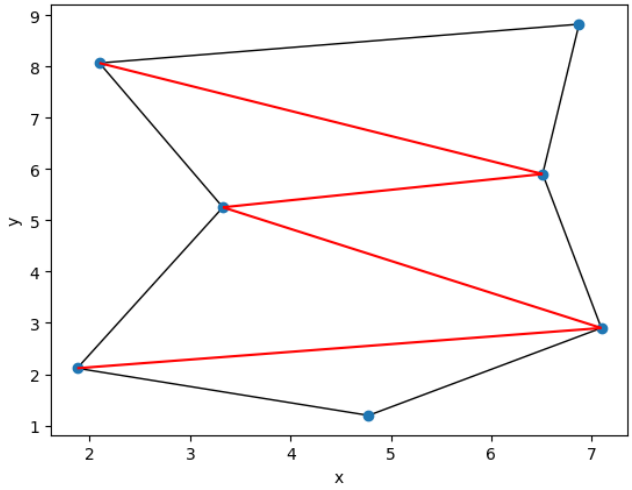
Poniżej przedstawię finalny efekt triangulacji wybranych przeze mnie wielokątów. Kolejne kroki algorytmu triangulacji wielokąta przedstawione są w moim kodzie znajdującym się w pliku „Zowada_kod_3.ipynb”.



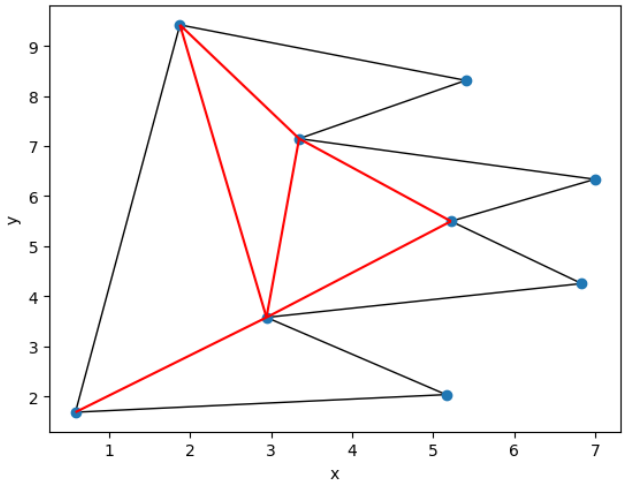
(Rys 4.1): *Triangulacja Wielokąta A*



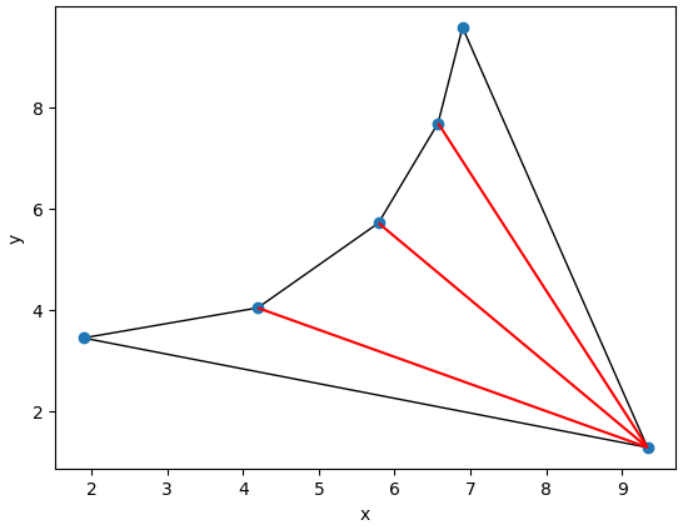
(Rys 4.2): *Triangulacja Wielokąta B*



(Rys 4.3): *Triangulacja Wielokąta C*



(Rys 4.4): *Triangulacja Wielokąta E*



(Rys 4.5): *Triangulacja Wielokąta F*

Wielokąt D jest niemonotoniczny, więc nie przedstawię dla niego triangulacji.

Podsumowanie i wnioski

Implementacja algorytmów oraz narzędzia do zadawania wielokątów za pomocą myszki przebiegła bez problemów. Program działa dla wszystkich wielokątów bez zarzutów i otrzymujemy oczekiwane wyniki.