Algorytmy geometryczne

Sprawozdanie z ćwiczenia 3.

Adam Barański

gr. 1 (pon. 12:50 – 14:20)

Platforma Testowa

Ćwiczenie zostało wykonane na komputerze wyposażonym w procesor Intel Core i7-4790k @4.5GHz, 16 Gigabajtów pamięci RAM DDR3 pracującym pod kontrolą systemu operacyjnego Linux Ubuntu 20.04 (64-bit) w środowisku Jupyter Notebook pochodzącego z pakietu Anaconda z użyciem języka programowania Python w wersji 3.9.7.

Biblioteki języka Python wykorzystane w ćwiczeniu:

- Numpy
- Matplotlib.

Cel ćwiczenia

Celem ćwiczenia jest implementacja algorytmów sprawdzania monotoniczności wielokątów prostych, klasyfikowania ich wierzchołków a także triangulacji wielokąta monotonicznego wraz z wizualizacją.

Sprawdzanie y-monotoniczności wielokąta

Algorytm operuje na posortowanej liście punktów. Iteracja po punktach startuje od punktu o maksymalnej współrzędnej y a następnie przesuwa się w kierunku punktu o minimalnej współrzędnej y przemieszczając się po punktach z jednego łańcucha (punkty połączone ze sobą krawędziami) i sprawdzając, czy kolejny punkt ma mniejszą współrzędna y od poprzedniego. Następnie przechodzi po drugim łańcuchu zaczynając od punktu o najmniejszej współrzędnej y kierując się ‘w górę’ i sprawdzając, czy kolejny wierzchołek ma większą współrzędna y od poprzedniego. Jeśli te warunki były spełnione (wierzchołki cały czas „niżej” potem cały czas „wyżej”) wielokąt jest y-monotoniczny.

Klasyfikacja wierzchołków

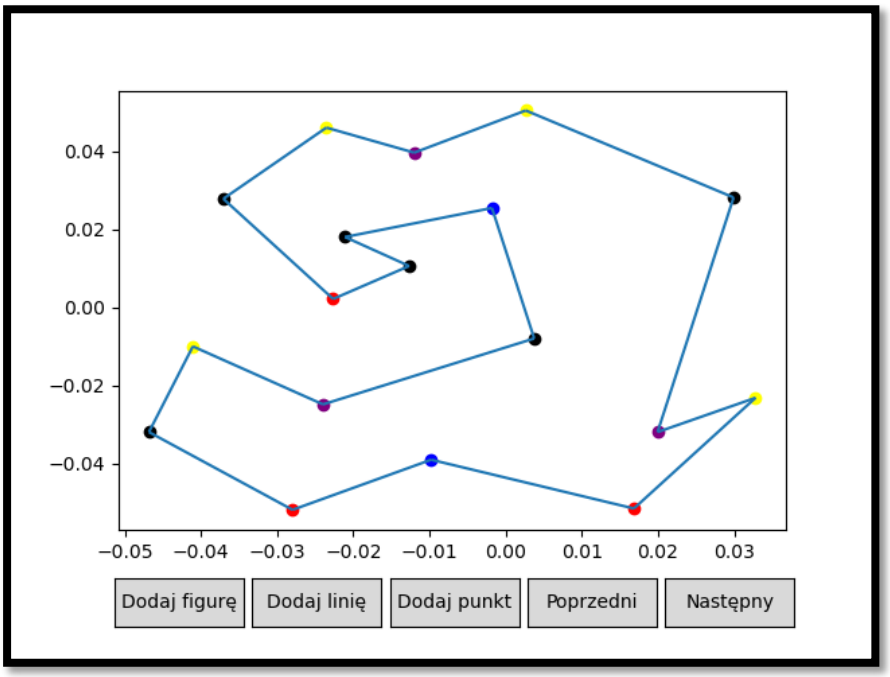
Algorytm sprawdza każdy wierzchołek porównując jego współrzędną y z jego sąsiadami (wierzchołki połączone z nim krawędziami). Jeśli żaden z sąsiadów nie znajduje się „wyżej” od danego wierzchołka to wierzchołek w zależności od kąta wewnętrznego tworzonego przez krawędzie wychodzące z wierzchołka to wierzchołek klasyfikowany jest jako początkowy (kąt wew. < 180°) lub dzielący (w.p.p). W przypadku, gdy obaj sąsiedzi znajdują się „wyżej” to wierzchołek kwalifikowany jest jako końcowy (kąt wew. < 180°) lub

łączący. W każdym innym wypadku wierzchołek jest wierzchołkiem prawidłowym.

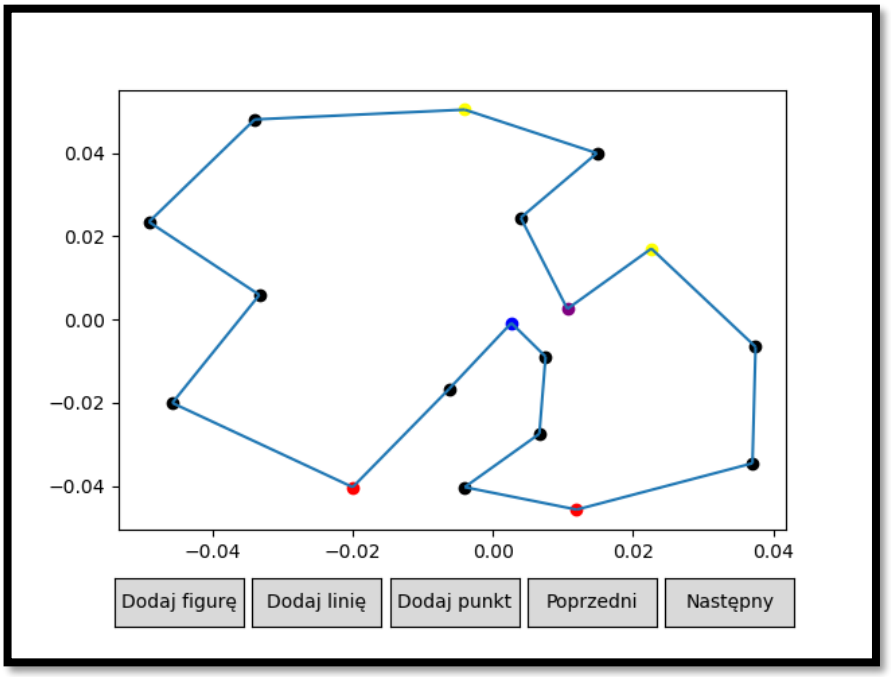
Wyniki takiej klasyfikacji przedstawione są na poniższych obrazkach.

Kolor a typ wierzchołka:

- Żółty – wierzchołek początkowy
- Czerwony – wierzchołek końcowy
- Niebieski – wierzchołek dzielący
- Fioletowy – wierzchołek łączący
- Czarny – wierzchołek prawidłowy



Rysunek 1.1 Kwalifikacja wierzchołków dla przykładowego wielokąta



Rysunek 1.2 Kwalifikacja wierzchołków dla przykładowego wielokąta

Triangulacja wielokąta y-monotonicznego

Algorytm dokonujący triangulacji wielokąta zaczyna swoje działanie od przypisania każdemu wierzchołkowi łańcucha. Do łańcucha 1 przypisane zostają wierzchołki na ścieżce od wierzchołka o max y do wierzchołka o min y (nie włączając go), idąc przeciwnie do wskazówek zegara. Pozostałe wierzchołki zostały przypisane do łańcucha 2.

Następnie algorytm sortuje wszystkie wierzchołki malejąco po współrzędnej y, dodaje na stos pierwsze dwa wierzchołki. Algorytm następnie analizuje wierzchołki po kolei idąc „w dół" po współrzędnej y. W zależności od relacji między wierzchołkiem na szczycie stosu a aktualnie analizowanym algorytm przyjmuje jeden z dwóch wariantów działania:

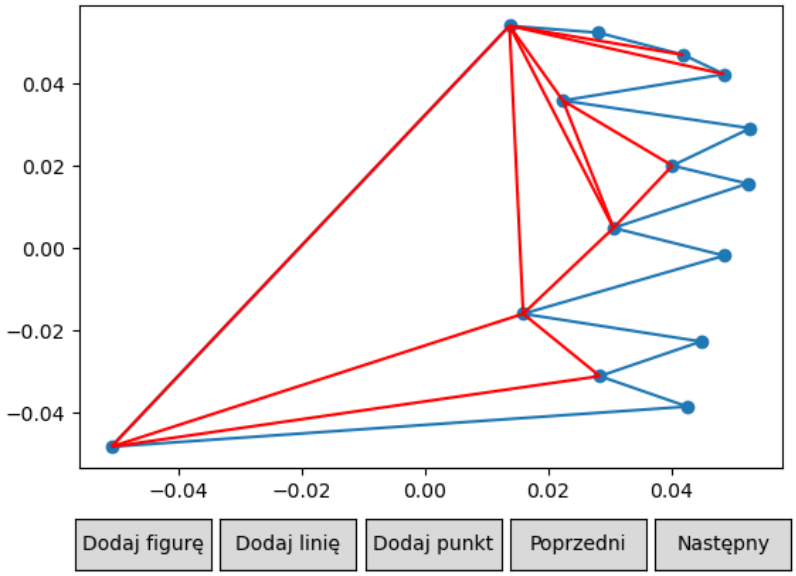
- Wariant I – wierzchołek należy do tego samego łańcucha co wierzchołek z góry stosu. W takiej sytuacji algorytm ściąga ze stosu dwa wierzchołki i analizuje, czy trójkąt należy do wielokąta. Jeśli tak, to jak długo taki stan rzeczy się utrzymuje, ściąga się ze stosu kolejne wierzchołki i analizuje się tworzone przez nich trójkąty, w przeciwnym wypadku odkłada wszystkie 3 analizowane wierzchołki na stos (2 ze stosu i aktualnie analizowany). Jeśli ze stosu zostaną usunięte wszystkie wierzchołki lub któryś kolejny analizowany trójkąt nie należy do wielokąta to algorytm odkłada na stos ostatnio ściągnięty z jego wierzchołek oraz aktualnie analizowany wierzchołek.
- Wariant II – wierzchołek należy do innego łańcucha niż wierzchołek z góry stosu. W takiej sytuacji można utworzyć przekątne ze wszystkimi wierzchołkami znajdującymi się na stosie. Zdejmuje się więc wszystkie, po zakończeniu na stos odkładane są aktualnie analizowany wierzchołek oraz wierzchołek, który leżał na samym szczycie stosu przed rozpoczęciem operacji usuwania wierzchołków ze stosu.

Dodatkowo w obu przypadkach algorytm sprawdza, czy dwa wierzchołki, między którymi chcemy dodać krawędzie nie jest bokiem wielokąta (poprzez sprawdzenie różnicy między indeksami wierzchołków w liście przed posortowaniem)

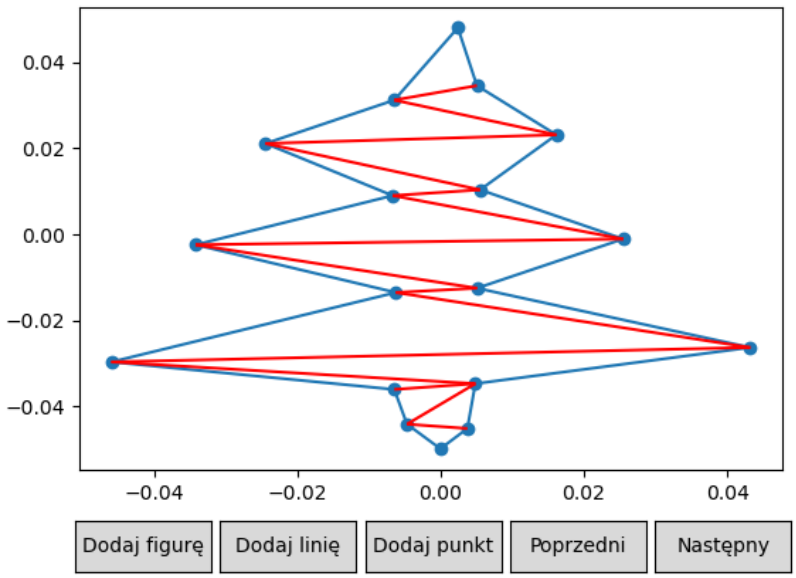
Weryfikacja położenia trójkąta

Weryfikacja tego, czy trójkąt leży wewnątrz wielokąta odbywa się przy pomocy wyznacznika macierzy 3x3, który to weryfikuje wzajemne położenie trzech punktów. Jeśli aktualnie analizowany wierzchołek leży na łańcuchu 1, to wyznacznik ten musi być ujemny, jeśli na łańcuchu nr 2, dodatni. W przypadku, gdy wartość bezwzględna wyznacznika jest mniejsza niż $EPS = 10^{-12}$ punkty uznane są za współliniowe.

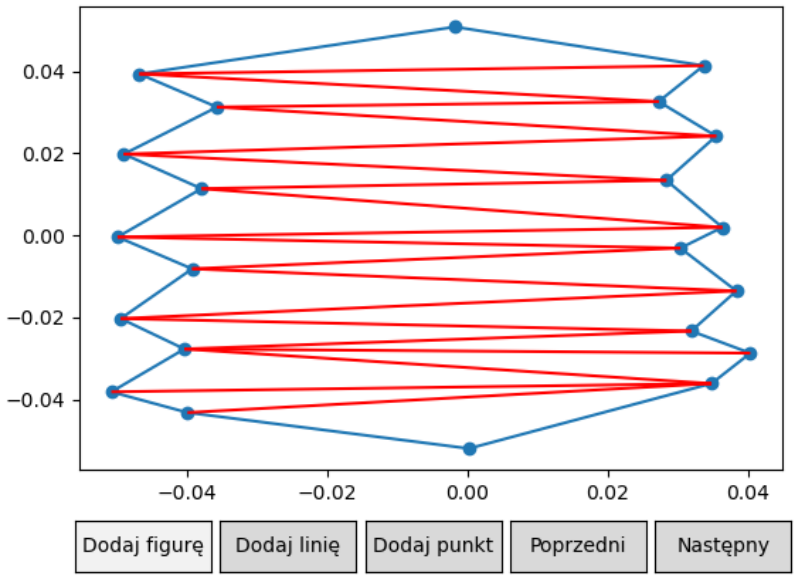
Poniższe obrazki prezentuję utworzone triangulacje kilku wielokątów y-monotonicznych.



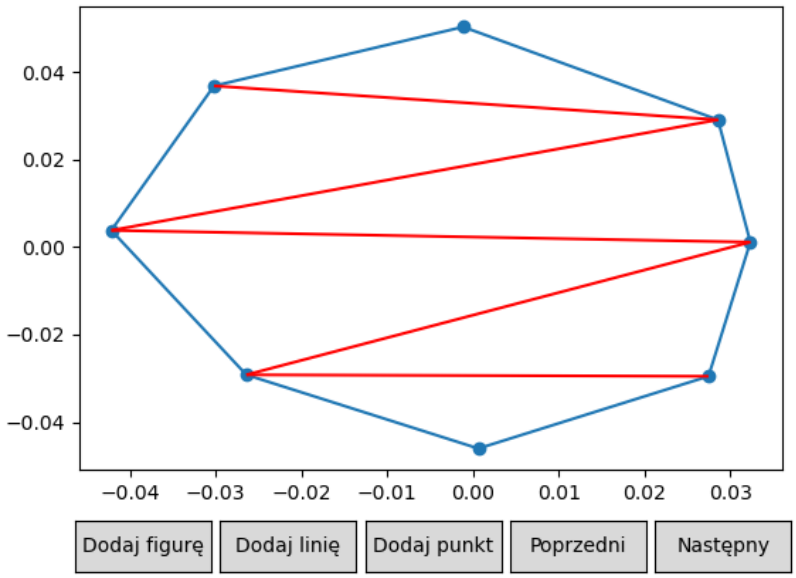
Rysunek 2.1 Triangulacja wielokąta z zestawu testowego 1



Rysunek 2.2 Triangulacja wielokąta z zestawu testowego 2

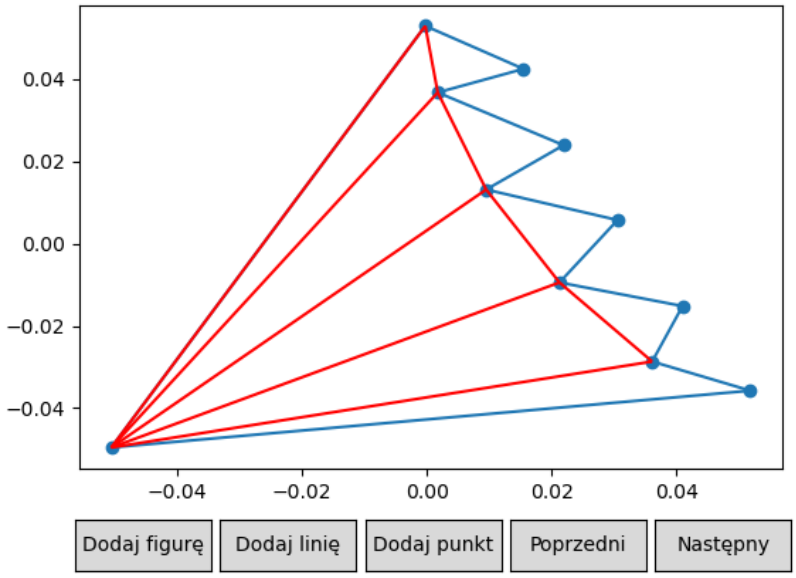


Rysunek 2.3 Triangulacja wielokąta z zestawu testowego 3



Rysunek 2.4 Triangulacja wielokąta z zestawu testowego 4

Rysunek 2.5 Triangulacja wielokąta z zestawu testowego 5



Czerwone odcinki to utworzone w wyniku triangulacji przekątne wielokąta.

Zestawy testowe zostały wybrane tak, by były możliwie najbardziej zróżnicowane. Zestawy 1 i 5 sprawdzają sytuację, kiedy mamy dużo wierzchołków należących do tego samego łańcucha pod rząd.

Zestawy 2, 3 i 4 sprawdzają sytuację, kiedy analizowane wierzchołki należą naprzemiennie do różnych łańcuchów.

Dla każdego testowanego wielokąta algorytm odnajdywał poprawną jego triangulację.

Przechowywanie wielokąta oraz triangulacji

Wielokąt przechowywany został w tablicy obiektów Point, które posiadały następujące parametry:

- x – współrzędna x wierzchołka,
- y – współrzędna y wierzchołka,
- chain – nr łańcucha, do którego należy wierzchołek
- index – numer indeksu, pod jakim przechowywany był wierzchołek w tablicy, przed jej posortowaniem na początku działania triangulacji.

Dodatkowo klasa ta wyposażona została w metodę get_coords(), zwracającą parametry x i y wierzchołka zapakowane w dwuelementowej krotce. Takie

przechowywanie punktów umożliwia łatwy dostęp do potrzebnych informacji i nie utrudnia rysowania przekątnych – wystarczy pobrać współrzędne z użyciem funkcji get_coords().

Triangulacja przechowywana była zaś w liście dwuelementowych krotek, których elementami były kolejne krotki dwuelementowe, zawierające wsp. x i y wierzchołków, między którymi w wyniku triangulacji utworzone zostały nowe przekątne. Przewagą takiego przechowywania triangulacji nad przechowywaniem wszystkich trzech elementów trójkąta ułatwia jej narysowanie – nie musimy zastanawiać się między którą parą wierzchołków rysować przekątną.

Wnioski

Wykonane triangulacje dla każdego ze zbiorów testowych sugerują poprawne działanie zaimplementowanego algorytmu. Algorytm zwrócił poprawne listy przekątnych składających się na triangulacje, nie tworząc duplikatów (zarówno krawędzi wierzchołka jak i przekątnych). Zastosowane struktury danych dobrze sprawdziły się w tym zadaniu, pozwalając na łatwy dostęp do wszystkich informacji o punktach wielokąta.