LABORATORIUM 3

Algorytmy geometryczne

Ćwiczenie 3 – sprawozdanie

Hieronim Koc

23.11.2022

grupa nr 4, czwartek_11_np_4

Dane techniczne specyfikacji komputerowej:

- Procesor : AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
- RAM: 8.00 GB
- Typ system: 64-bitowy system operacyjny
- Środowisko: Jupyter notebook
- System operacyjny: Ubuntu 22.04 LTS Jammy Jellyfish (2022-04-21)

Wszystkie testy i ich implementacje były wykonywane w języku programowania Python 3.9.12, z wykorzystaniem zewnętrznych bibliotek takich jak: *numpy*, *matplotlib.* Dodatkowe zmiany do narzędzia graficznego wprowadził Patryk Klatka.

Cel ćwiczenia

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

Opis ćwiczenia

Ćwiczenie składało się z implementacji trzech algorytmów podanych w celu ćwiczenia, jak i wygenerowaniem przykładowych danych wielokątów i zastosowania na nich danych algorytmów.

Struktury

Dla przejrzystego operowania na danych wielokątach i ich triangulacjach zastosowano przechowywanie ich jako tablica linii, gdzie każda linia składała się z krotki dwu-elementowej zawierająca punkty końca danej linii. Utworzona triangulacja została zapisana w postaci takiej jak wielokąt, ze zmianą że owa tablica zawierała tylko utworzoną triangulację. Wykorzystano taki, a nie inną strukturę przechowania wielokątów, gdyż gwarantuje to bezsprzeczną identyfikację wielokąta razem z szybką analizą Dla szybszego wyszukiwania punktu w łańcuchu prawym czy lewym zastosowano *zbiór* (set()) dla szybszego wyszukiwania czy dany punkt znajduje się w danym zbiorze. Dla obliczania wyznacznika przyjęto epsilon: $\varepsilon = 10e - 12$.

Wielokąt *y*-monotoniczny

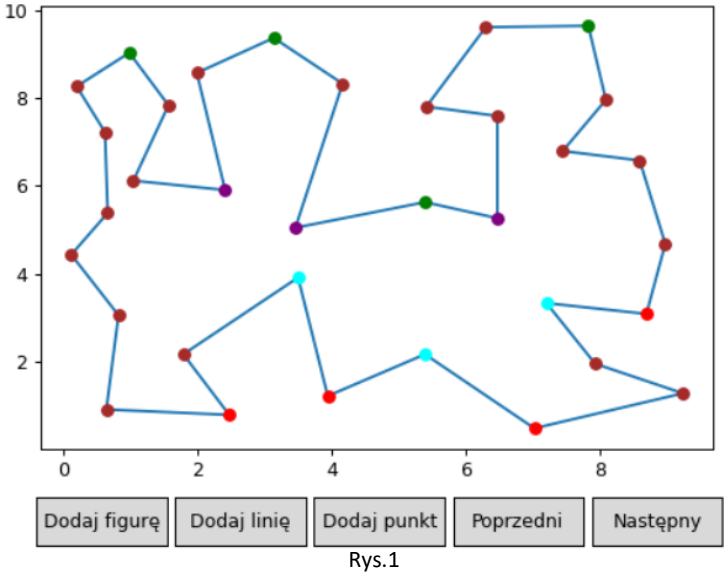
Wyznaczając wielokąt *y*-monotoniczny zastosowano procedurę, gdzie algorytm rozpoczyna się od najwyższego wierzchołka (gdy jest ta sama wysokość, bierze się pod uwagę składową *x*) i dokonuje się podziału na lewy i prawy „łańcuch”. Korzystamy z tego że wierzchołki są ułożone w kierunku przeciwnym do kierunku ruchu zegara, więc wystarczy przejść po dwóch „łańcuchach” i sprawdzić czy składowa *y* kolejnego punktu jest mniejsza, bądź taka sama jak składowa *y* bieżącego punktu. Przeszukiwanie punktów kończy się gdy dojdzie się do najniższego punktu wielokąta.

Klasyfikacja punktów

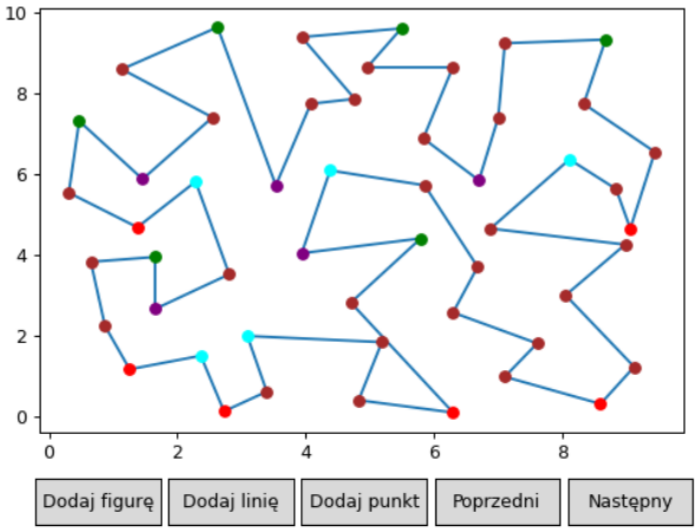
Klasyfikacja punktów została dokonana poprzez wyliczanie wyznacznika kolejnych trzech punktów. Dla danych punktów zostały przypisane konkretne kolory w celu przejrzystszego wyświetlania wyników.

Nazwa wierzchołka	Kolor
Początkowy	Zielony
Końcowy	Czerwony
Łączący	Fioletowy
Dzielący	Cyjanowy
Prawidłowy	Brązowy

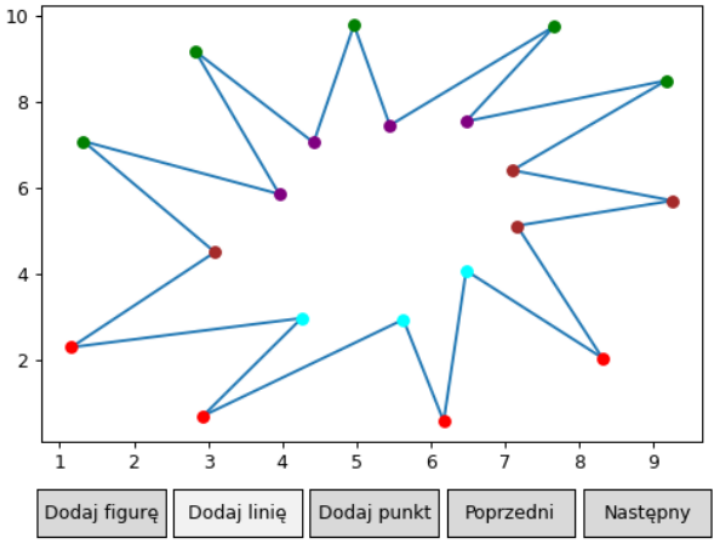
Przykładowe kolorowanie wielokątów



Rys.1



Rys.2



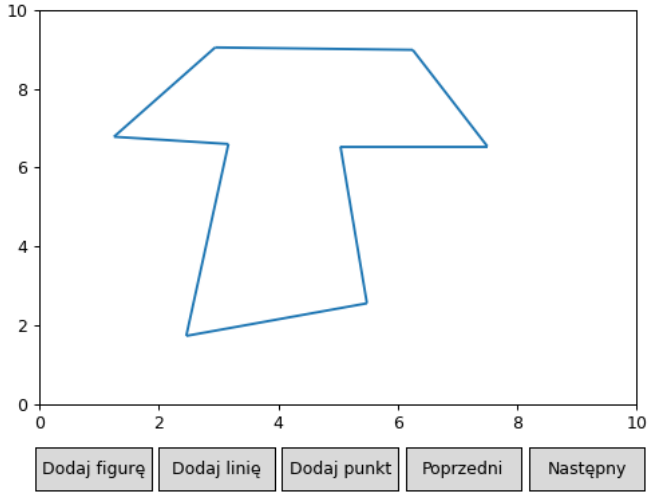
Rys.3

Triangulacja wielokątów monotonicznych

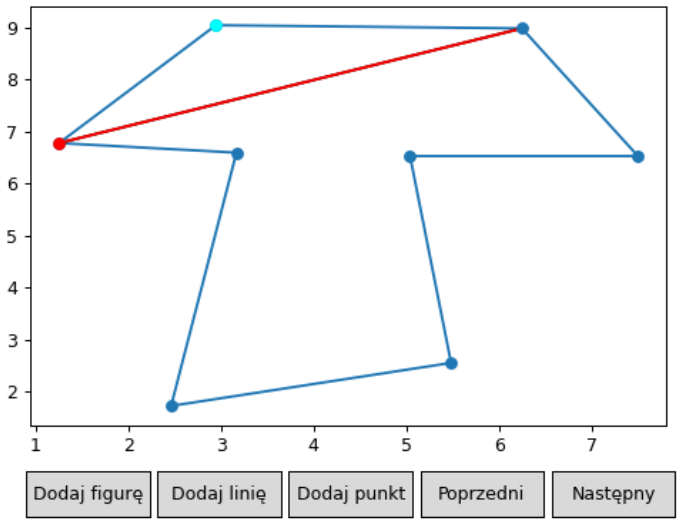
Algorytm został zaimplementowany zgodnie z instrukcją podaną na wykładzie. Przy implementacji dzięki wykorzystaniu *zbiorów* (set()) w *python*-ie stała złożoności obliczeniowej spadła przy wyszukiwaniu danego punktu. Przy wyliczaniu triangulacji pomogły funkcje do wyszukiwania czy dany odcinek należy do danego zbioru oraz tworzenie prawego i lewego łańcucha. Dla przetestowania algorytmów użyto wielokątów z duża ilością wklęsłości, by przetestować go w ekstremalnych warunkach. Ów algorytm powinien działać w czasie *O(n)* (zakładając dobrą implementację i brak obsługi scen, gdyż wzrasta wtedy stała obliczeniowa).

Przykładowy przebieg algorytmu:

Początkowy stan wielokąta (Rys.4)

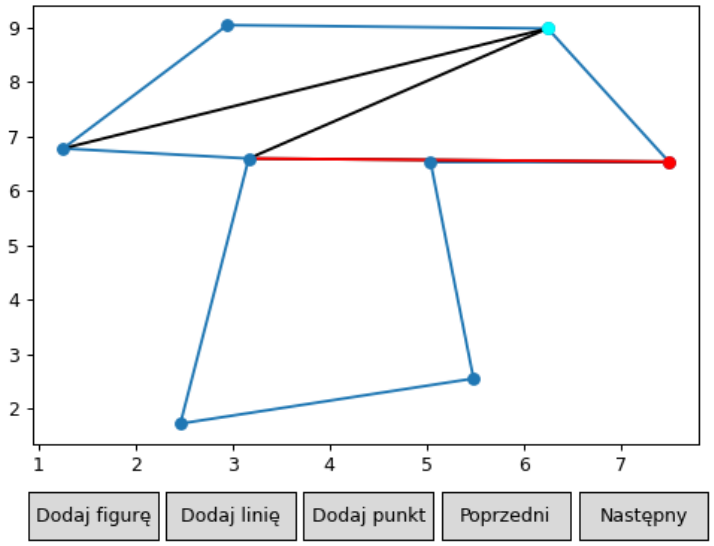


Rys.4



Rys.5

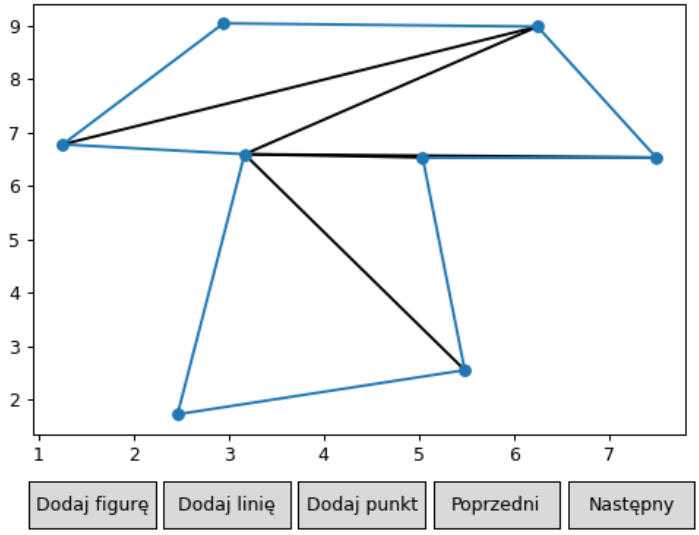
Rys.5 przedstawia aktualnie rozpatrywany wierzchołek (czerwony) oraz wierzchołki na stosie (cyjan) oraz aktualnie rozpatrywaną linię, czy być może należy do triangulacji



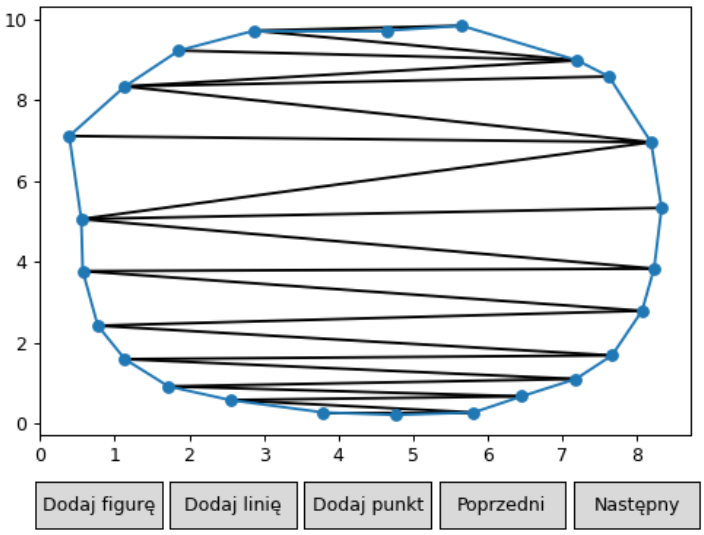
Rys.6

Rys.6 przedstawia aktualnie rozpatrywany wierzchołek (czerwony) oraz wierzchołki na stosie (cyjan), aktualnie rozpatrywaną linię (czerwona linia), czy być może należy do triangulacji oraz linie należące do triangulacji (czarne linie)

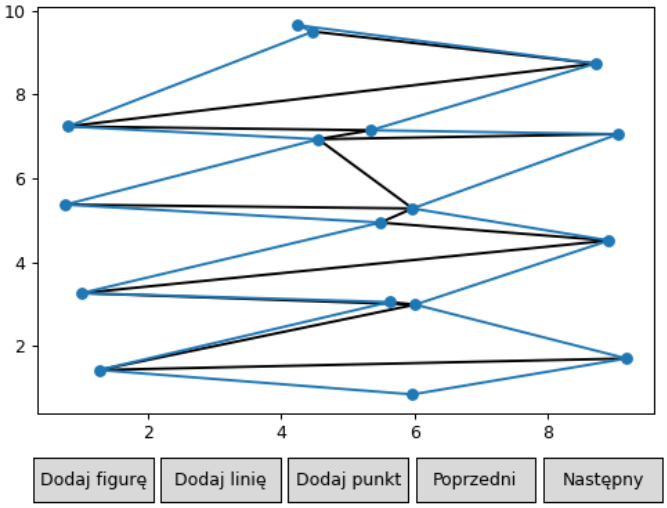
Przykładowe wyniki algorytmu triangulacji wielokątów monotonicznych:



Rys.7



Rys.8



Rys.9

Wnioski

Po wykonaniu ćwiczenia i uruchomieniu wszystkich trzech algorytmów na zbiorach danych można zobaczyć, że wszystkie działały prawidło. Jednakże ze względu na dany epsilon i niedokładność liczb zmiennoprzecinkowych, niektóre wyniki mogły wyjść niedokładne. Wszystkie zbiory były wyznaczane ręcznie. Również trzeba mocno zaznaczyć że wszystkie algorytmy działają wtedy i tylko wtedy, gdy są zadawane przeciwnie do ruchów wskazówek zegara, co może czasami mylnie podać wyniki (jednakże na rzecz danego ćwiczenia – tak jak było podane – wszystkie wielokąty powinny zostać zadawane przeciwnie do ruchów wskazówek zegara).