Algorytmy Geometryczne

Sprawozdanie – Ćwiczenie 3: Triangulacja wielokątów monotonicznych

Krzysztof Chmielewski

Data wykonania: 04.10.2024
Data oddania: 18.11.2024

Spis treści

WSTĘP

CEL ĆWICZENIA

Celem ćwiczenia jest zapoznanie się z pojęciami wielokątów monotonicznych oraz triangulacją. Zadaniem jest także implementacja algorytmów klasyfikujących wierzchołki i algorytmu triangulacji wielokąta monotonicznego.

TEORIA

WIELOKĄT MONOTONICZNY (Y-MONOTONICZNOŚĆ)

Wielokąt monotoniczny to taki wielokąt, dla którego można wskazać prostą wyznaczającą kierunek monotoniczności, dla której każda prosta do niej prostopadła przecina ten wielokąt najwyżej w dwóch punktach, jest to tak zwana silna (ścisła) monotoniczność. Słabą monotoniczność określamy, gdy rozszerzymy tę definicję o wielokąty posiadające krawędzie prostopadłe do prostej wyznaczającej kierunek monotoniczności. Jeżeli kierunek monotoniczności będzie wyznaczać oś OY, to taki wielokąt możemy nazwać Y-monotonicznym.

PODZIAŁ WIERZCHOŁKÓW

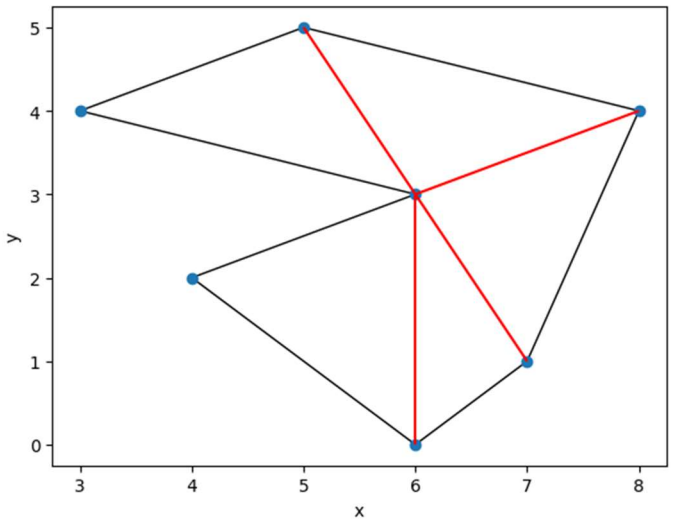
W wielokącie możemy dokonać podziału jego wierzchołków ze względu na jego położenie względem swoich sąsiadów oraz miarę kąta wewnętrznego. Kategoryzujemy wierzchołki:

- Początkowe – jego sąsiedzi leżą poniżej oraz kąt wewnętrzny $< \pi$
- Końcowe – jego sąsiedzi leżą powyżej oraz kąt wewnętrzny $< \pi$
- Łączące – jego sąsiedzi leżą powyżej oraz kąt wewnętrzny $> \pi$
- Dzielące – jego sąsiedzi leżą poniżej oraz kąt wewnętrzny $> \pi$
- Prawidłowe – jeden z jego sąsiadów jest poniżej, drugi powyżej

Tak określone wierzchołki pomagają w określeniu monotoniczności wielokąta. Wielokąt będzie Y-monotoniczny, gdy nie posiada wierzchołków dzielących i łączących.

TRIANGULACJA

Triangulacją nazywamy podział figury geometrycznej na sympleksy, czyli w przypadku figur dwuwymiarowych – trójkąty. Podział ten zakłada, że częścią wspólną dowolnych dwóch sympleksów dwuwymiarowych jest ich wspólny bok, wierzchołek lub zbiór pusty. Przykład wielokąta po triangulacji na Rys. 1. poniżej. Każdy wielokąt prosty można striangulować, każda triangulacja n-kąta składa się dokładnie z n-2 trójkątów.



Rys. 1 Przykładowa triangulacja wielokąta

DANE TECHNICZNE

Ćwiczenie zostało wykonane z użyciem narzędzia graficznego dostarczonego przez Koło Naukowe Bit (**https://github.com/aghbit/Algorytmy-Geometryczne)**, które umożliwia wizualizację wykresów i kształtów geometrycznych wykorzystujące różne biblioteki języka Python (np. numpy, pandas, matplotlib) oraz Anacondę do stworzenia odpowiedniego jądra dla Jupyter Notebook. Kod zawarty w pliku ***chmielewski_kod_3.ipynb*** został napisany w języku Python właśnie przy użyciu Jupyter Notebook.

Obliczenia zostały wykonane z wykorzystaniem sprzętu o następujących parametrach:

Komputer -> wirtualna maszyna VirtualBox:

- **Procesor:** AMD Ryzen 5 5600X 6 rdzeniowy, podstawowe taktowanie: 3,70 GHz -> w wirtualnej maszynie wykorzystano jedynie 5 z 12 wątków
- **Karta Graficzna:** NVIDIA GeForce RTX 3060 Ti 8GB GDDR6X
- **RAM:** 32GB DDR4 3000MHz CL16 -> w wirtualnej maszynie wykorzystano jedynie 16GB
- **OS:** Linux Ubuntu 24.04

Laptop:

- **Procesor:** Intel Core I5-1235u 10 rdzeniowy, podstawowe taktowanie: 1,30 GHz
- **Karta Graficzna:** zintegrowana z procesorem Intel UHD
- **RAM:** 8GB DDR4 3200MHz
- **OS:** Linux Ubuntu 24.04

REALIZACJA ĆWICZENIA

FUNKCJA RYSUJĄCA WIELOKĄTY

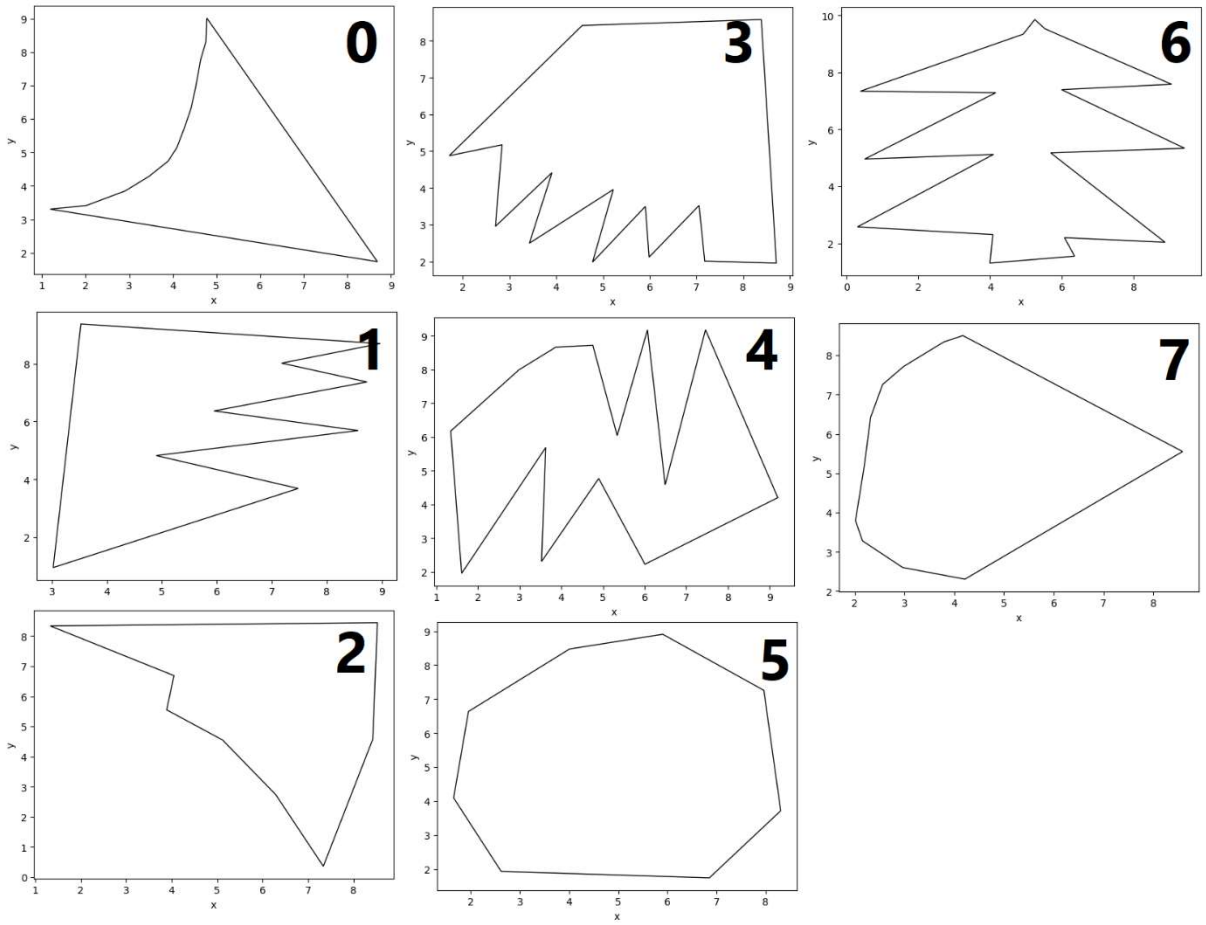
Do realizacji ćwiczenia napisano dodatkową funkcję draw_shape, której zadaniem jest odczytanie wielokąta narysowanego przez myszkę i zapisanie go w postaci listy krotek, gdzie każda krotka to punkt zaznaczony w układzie współrzędnych. W implementacji użyto biblioteki matplotlib, która może różnie współdziałać z notatnikiem Jupyter Notebook, w tym celu w komórce notatnika, gdzie będzie wywoływana funkcja draw_shape należy wcześniej uwzględnić instrukcję pozwalającą na osobne okno do rysowania np. *%matplotlib tk*. Aby powrócić do notatnikowego trybu wyświetlania wykresów i figur należy z powrotem zmienić tryb wyświetlania w następnej komórce instrukcją np. *%matplotlib inline*.

Funkcja draw_shape uwzględnia odpowiednie akcje: **lewy przycisk myszy** - dodanie punktu do wielokąta, przycisk **prawy przycisk myszy** – narysowanie wielokąta z zaznaczonych punktów, przycisk **r** – reset punktów na płaszczyźnie. Aby wielokąt został zapisany należy kliknąć przycisk **spacji**.

UWAGA: Funkcja draw_shape() uwzględnia kierunek zadawania wierzchołków, dlatego do poprawnej analizy należy zadawać wierzchołki w kierunku przeciwnym do wskazówek zegara.

PRZYKŁADY DO ANALIZY

W celu późniejszej kategoryzacji wierzchołków i triangulacji, przygotowano osiem wielokątów przy użyciu wyżej opisanej funkcji draw_shape(). Wielokąty są przedstawione poniżej na Rys. 2. Indeksowanie wielokątów zaczyna się od 0.



Rys. 2 Wygenerowane wielokąty do analizy

Wielokąty dobrano różne, aby były testowane różne etapy algorytmu, w szczególności wielokąty 0 i 1 dają szansę analizy procesu triangulacji dla wielokątów, których podział na prawy i lewy łańcuch jest bardzo widoczny. Wybrano także takie wielokąty jak 3 i 4, aby sprawdzić czy algorytm poprawnie oznaczy je jako nie Y-monotoniczne.

WYNIKI I ANALIZA

OKREŚLENIE Y-MONOTONICZNOŚCI

Wyniki algorytmu, który określa Y-monotoniczność wielokąta wyglądają następująco:

- Polygon 0: True
- Polygon 1: True
- Polygon 2: True
- Polygon 3: False
- Polygon 4: False
- Polygon 5: True
- Polygon 6: False
- Polygon 7: True

Algorytm zatem określił 5 z 8 podanych wielokątów jako Y-monotoniczne. Wyniki te można potwierdzić kategoryzując wierzchołki dla każdego wielokąta jednocześnie sprawdzając, czy wielokąty, dla których wynik był False rzeczywiście mają wierzchołki dzielące lub łączące.

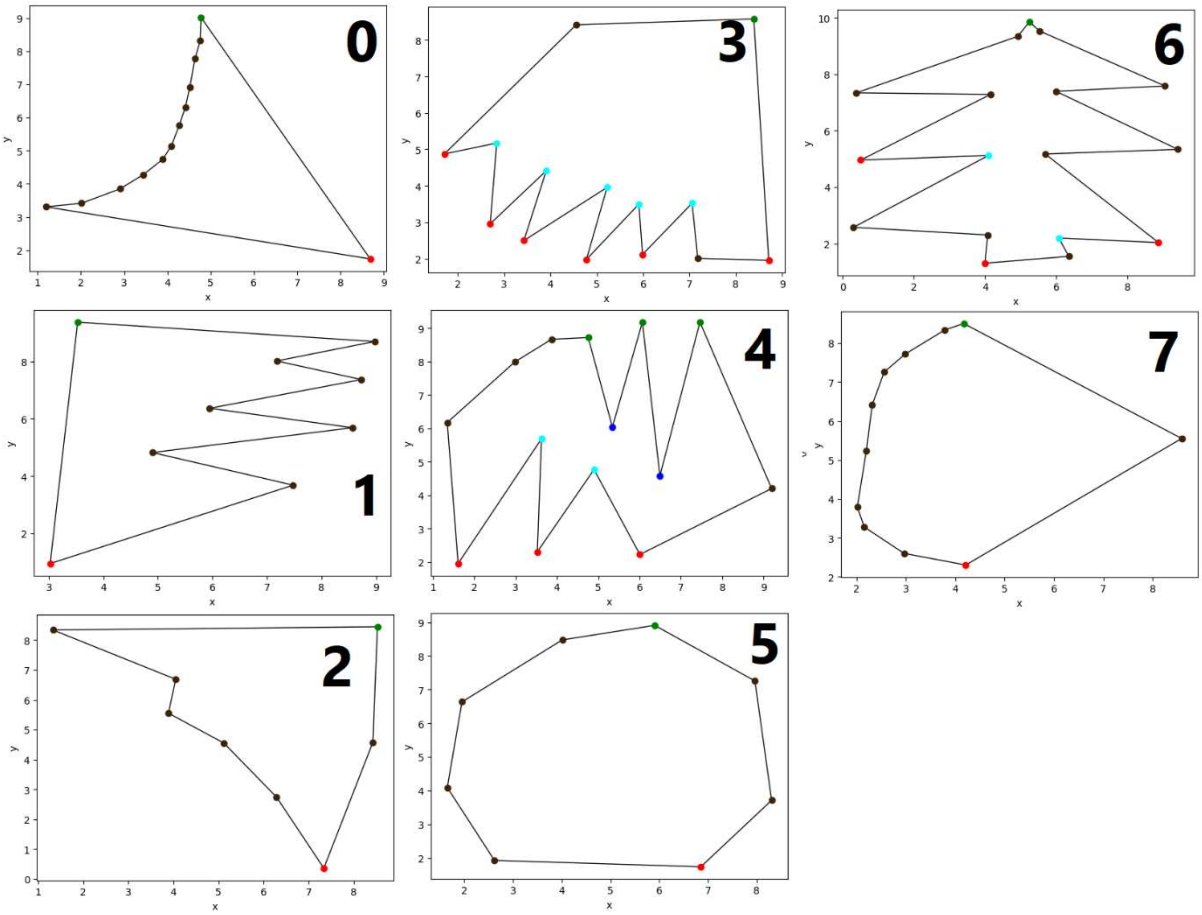
KATEGORYZACJA WIERZCHOŁKÓW

Algorytm kategoryzacji wierzchołków opiera się o badanie trzech kolejnych wierzchołków wielokąta. Algorytm bierze pod uwagę wierzchołek badany – ten, który aktualnie podlega kategoryzacji, wierzchołek poprzedni i wierzchołek następny, względem badanego, utrzymując porządek przeciwny do ruchu wskazówek zegara. Aby określić kategorię wierzchołka badanego należy określić jego położenie na osi OY względem wierzchołka poprzedniego oraz następnego, a także skręt układu tych trzech punktów. Do określenia skrętu algorytm wykorzystuje funkcję orient(a,b,c), która oblicza wyznacznik macierzy 3x3 $\begin{bmatrix} a_x & a_y & 1 \\ b_x & b_y & 1 \\ c_x & c_y & 1 \end{bmatrix}$, gdzie a_x to współrzędna x-owa punktu a, a_y to współrzędna y-owa. Jeżeli wyznacznik jest większy od 0 to układ jest lewoskrętny, jeżeli mniejszy od 0 to układ jest prawoskrętny, jeśli równy 0 to są to punkty współliniowe. Na Rys. 3. poniżej widać wynik działania algorytmu kategoryzacji wierzchołków.

Kolory wierzchołków są przypisywane następująco:

- **Brązowy** – wierzchołek prawidłowy
- **Cyjan** – wierzchołek dzielący
- **Niebieski** – wierzchołek łączący
- **Czerwony** – wierzchołek końcowy
- **Zielony** – wierzchołek początkowy

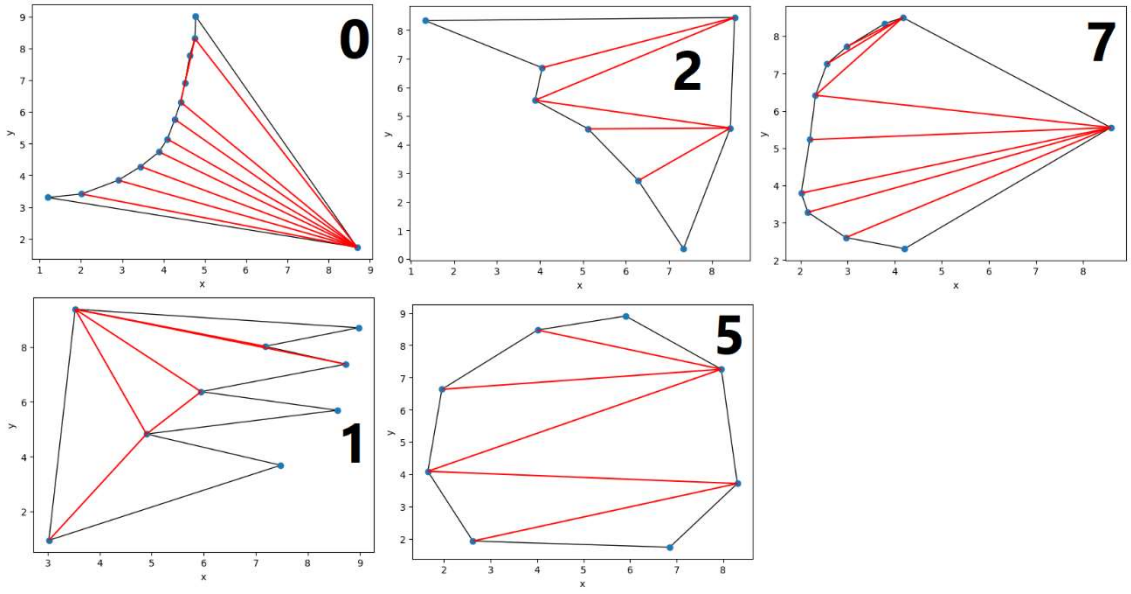
Patrząc na Rys. 3. widać, że w istocie wielokąty 3, 4 i 6 mają wierzchołki albo łączące, albo dzielące, co zgodnie z teorią mówi nam, że nie są to wielokąty Y-monotoniczne.



Rys. 3 Badane wielokąty po kategoryzacji wierzchołków

TRIANGULACJA

Triangulacja obejmuje tylko wielokąty Y-monotoniczne, zatem algorytm odrzucił wielokąty 3, 4 i 6, ponieważ nie spełniają tego założenia. Wyniki procesu triangulacji są widoczne na Rys. 4. poniżej.



Rys. 4 Wyniki triangulacji wielokątów

Do pliku z kodem i sprawozdaniem dołączono również gify przedstawiające proces tworzenia trójkątów w triangulacji. Wierzchołki niebieskie w animacji są wierzchołkami przechowywanymi na stosie, wierzchołki w kolorze cyjanu są to aktualnie rozpatrywane wierzchołki.

WNIOSKI

Do testów wybrano osiem różnych wielokątów, tak, aby sprawdzić kolejne etapy algorytmu. Aby, sprawdzić poprawność należało zadać kilka wielokątów Y-monotonicznych, jak i także kilka takich, które nie są Y-monotoniczne. Algorytm klasyfikacji wierzchołków potwierdził wyniki przedstawione przez algorytm weryfikacji Y-monotoniczności danego wielokąta. Dane zostały dobrane tak, aby przynajmniej połowa z zadanych wielokątów przeszła weryfikację i została poddana triangulacji. Triangulacja została przeprowadzona poprawnie dla zadanych wielokątów.

Przy tworzeniu programu używano takich struktur danych jak tablica krotek, która przetrzymuje zarówno punkty jak i ich indeksy. Użytecznym też był podział wielokąta na dwa łańcuchy – prawy i lewy, które zawierają odpowiednie krotki punktów i indeksów, dzięki temu proces triangulacji można było prowadzić z wykorzystaniem informacji czy rozważany wierzchołek jest w tym samym łańcuchu co wierzchołek na szczycie stosu. Do kategoryzacji punktów użyto funkcji orient opisanej w zakładce KATEGORYZACJA WIERZCHOŁKÓW i kolorowano je zgodnie ze standardem opisanym na wykładzie.

Funkcja traingulation() zwracają listę krotek indeksów, które wskazują jedynie, do punktów o których indeksach należy dołożyć przekątną. Dlatego też ta funkcja nie zwraca całej triangulacji, a jedynie informację o przekątnych które należy dołączyć do wielokąta. Aby uzyskać informację o całej triangulacji napisano funkcję get_whole_triangulation(), która zwraca listę sąsiedztwa indeksów dla danego wielokąta przykładowo dla wielokąta zadanego jako czworokąt zgodnie ze wskazówkami zegara lista może wyglądać następująco: [[1,2],[2],[3],[0]] co oznacza, że istnieje prosta między punktami o indeksach 0-1, 0-2, 1-2, 2-3, 3-0.