Algorytmy Geometryczne

Sprawozdanie z ćwiczenia 3. „Triangulacja wielokątów monotonicznych”

Maciej Wiśniewski
Grupa 3 Poniedziałek 16.45 A

Data wykonania 25.11.2024 Data oddania 27.11.2024

1. Dane techniczne

Specyfikacja komputera: system Ubuntu 24.04.01 Linux 5.15 x64, procesor AMD Ryzen 7 5825U with Radeon 2GHz 8 rdzeni, 16GB pamięci RAM. Ćwiczenie zostało napisane w języku ***Python 3.9.20*** **w** ***Jupyter Notebook*** w środowisku programistycznym ***Visual Studio Code.*** Aby wykonać ćwiczenie posłużono się bibliotekami: ***numpy, itertools, i matplotlib*** . Do wykonania wizualizacji użyto narzędzia graficznego wykonanego przez ***Koło Naukowe BIT*** oraz ***Polygon Selectora*** dostępnego w bibliotece ***matplotlib***.

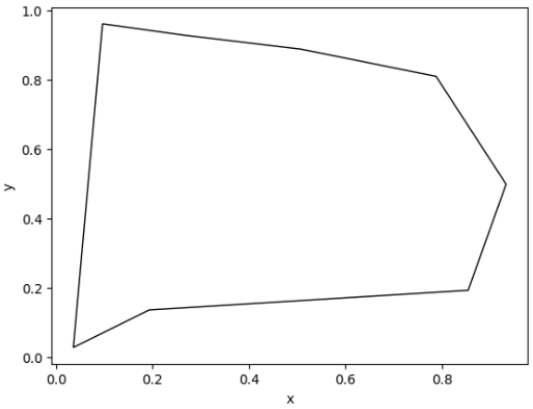
2. Cel ćwiczenia

Celem ćwiczenia jest zapoznanie się z zagadnieniem monotoniczności wielokątów oraz implementacja następujących algorytmów: sprawdzania, czy wielokąt jest y-monotoniczny, klasyfikacji wierzchołków w dowolnym wielokącie, triangulacji wielokąta monotonicznego. Dodatkowo, ćwiczenie obejmuje wizualizację i analizę wyników.

3. Wstęp teoretyczny

Y-monotoniczność

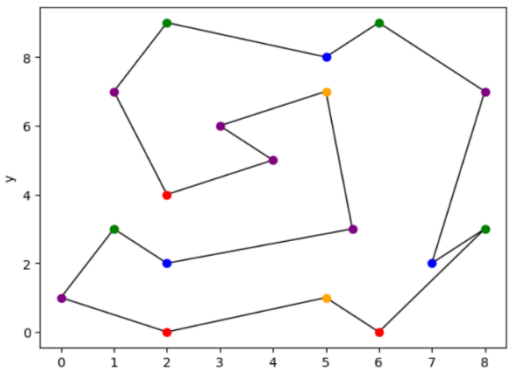
Formalnie, wielokąt nazywamy monotonicznym względem prostej ***l***, jeśli istnieje możliwość uporządkowania jego wierzchołków w taki sposób, że współrzędna wzdłuż tej prostej stale rośnie lub maleje. W kontekście tego zadania interesuje nas **y-monotoniczność**, czyli monotoniczność względem osi ***OY.***
Wielokąt jest **y-monotoniczny**, jeśli jego wierzchołki mogą być ułożone w kolejności, w której współrzędne ***y*** monotonnie rosną lub maleją wzdłuż kolejnych wierzchołków. Oznacza to, że dla każdej pary sąsiednich wierzchołków (z wyłączeniem wierzchołka początkowego i końcowego), jeden z punktów musi mieć większą lub mniejszą wartość współrzędnej ***y*** niż drugi. Przykładowy wielokąt ***y-monotoniczny*** został przedstawiony na Rysunku 1.



Rysunek 1 Przykładowy wielokąt **y-monotoniczny**

Klasyfikacja wierzchołków

Wierzchołki wielokątów zostały podzielone w następujący sposób:
0) ***początkowe***, gdy obaj jego sąsiedzi mają mniejszą wartość współrzędnej ***y*** i kąt wewnętrzny ma mniej niż 180°
1) ***końcowe***, gdy obaj jego sąsiedzi mają większą wartość współrzędnej ***y*** i kąt wewnętrzny ma mniej niż 180°
2) ***dzielący***, gdy obaj jego sąsiedzi mają mniejszą wartość współrzędnej ***y*** i kąt wewnętrzny ma więcej niż 180°
3) ***łączący***, gdy obaj jego sąsiedzi mają większą wartość współrzędnej ***y*** i kąt wewnętrzny ma więcej niż 180°
4) ***prawidłowy*** - pozostałe przypadki (jeden sąsiad ma większą wartość ***y*** ,drugi mniejszą)
Wierzchołkom z danych kategorii przypisano następujące kolorowanie:
0) ***początkowe*** → **zielony**,
1) ***końcowe*** → **czerwony**,
2) ***dzielący*** → **niebieski**,
3) ***łączący*** → **pomarańczowy**,
4) ***prawidłowy*** → **fioletowy**.
Przykładowe kolorowanie zostało zaprezentowane na Rysunku 2.



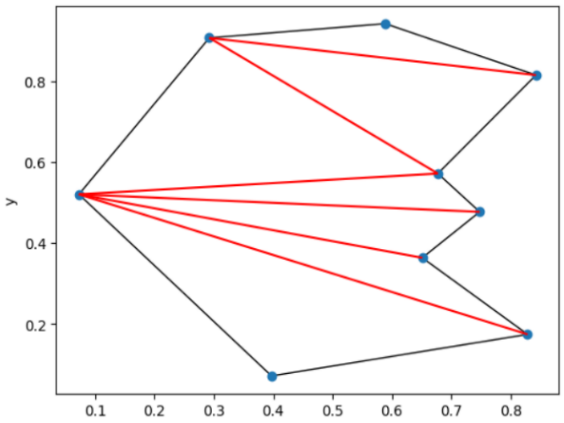
Rysunek 2 Przykładowe kolorowanie wielokąta

Wierzchołki początkowe i końcowe odgrywają istotną rolę w algorytmach przetwarzania wielokątów monotonicznych, takich jak algorytmy typu „dziel i zwyciężaj” oraz algorytmy triangulacji. W szczególności, wierzchołki łączące i dzielące mają kluczowe znaczenie w procesie triangulacji, ponieważ umożliwiają podział wielokąta na trójkąty w sposób gwarantujący brak przecięć krawędzi.

Charakterystyczną cechą wielokątów monotonicznych jest brak wierzchołków dzielących i łączących, co upraszcza ich przetwarzanie i pozwala na zastosowanie bardziej efektywnych metod triangulacji. Dzięki temu analiza takich wielokątów jest nie tylko prostsza, ale także bardziej przewidywalna w kontekście implementacji algorytmicznych.

Triangulacja

Triangulacja wielokąta to proces podziału wnętrza wielokąta na zbiór nieprzecinających się trójkątów za pomocą dodatkowych krawędzi, które łączą wierzchołki wielokąta. Rezultatem triangulacji jest zbiór trójkątów, które razem wypełniają całą powierzchnię wielokąta bez nakładania się i nie pozostawiają pustych obszarów. Zastosowanie triangulacji dla przykładowego wielokąta zostało przedstawione na Rysunku 3.



Rysunek 3 Triangulacja przykładowego wielokąta

4. Opis zastosowanych algorytmów

Na początku zaimplementowana została metoda do interaktywnego wprowadzania wielokątów. Aby móc dodawać wielokąty za pomocą myszki zaimplementowano metodę opartą o bibliotekę ***matplotlib***. Użyto metody bibliotecznej ***Polygon Selector*** do dodawania wierzchołków i interaktywnej wizualizacji, a do wizualizacji utworzonych wielokątów posłużono się narzędziem graficznym wykonanym przez ***Koło Naukowe BIT.***

Y-monotoniczność

Do wyznaczania ***y-monotoniczności*** figur zaimplementowano algorytm ***is-y-monotonic.*** Algorytm ***is-y-monotonic*** sprawdza, czy wielokąt jest ***y-monotoniczny***, czyli czy jego wierzchołki wzdłuż osi ***OY*** układają się monotonicznie po obu stronach, od najniższego do najwyższego punktu. Najpierw identyfikuje wierzchołki o minimalnej i maksymalnej wartości ***y***, które dzielą wielokąt na lewą i prawą stronę. Następnie punkty po obu stronach są porządkowane w zależności od ich położenia względem tych ekstremów.

Dla każdej ze stron algorytm sprawdza, czy współrzędne ***y*** są uporządkowane malejąco. Jeśli zarówno lewa, jak i prawa strona spełniają ten warunek, wielokąt jest ***y-monotoniczny***, a funkcja zwraca **True**. W przeciwnym razie wielokąt nie jest ***y-monotoniczny*** i funkcja zwraca **False**. Algorytm działa w czasie liniowym, przechodząc przez wierzchołki tylko raz.

Klasyfikacja wierzchołków

Do klasyfikowania wierzchołków wielokąta zaimplementowano algorytm ***kolory_wierzchoklow***. Algorytm ***kolory_wierzcholkow*** klasyfikuje wierzchołki wielokąta na podstawie ich charakterystyki geometrycznej i przypisuje każdemu kategorię: **0 (początkowy)**, **1 (końcowy)**, **2 (dzielący)**, **3 (łączący)** lub **4 (prawidłowy)**. Klasyfikacja odbywa się poprzez analizę pozycji każdego wierzchołka względem sąsiednich punktów i ich orientacji obliczanej za pomocą wyznacznika macierzy.

Dla każdego wierzchołka określane są jego sąsiedzi: poprzedni i następny. Algorytm oblicza orientację trójki punktów (poprzedni, bieżący, następny), która pozwala zidentyfikować, czy wierzchołek leży wewnątrz, czy na zewnątrz wielokąta. W zależności od orientacji i porównania współrzędnych ***y*** bieżącego wierzchołka z sąsiednimi, przypisywana jest odpowiednia kategoria.

Funkcja zwraca listę kategorii dla wszystkich wierzchołków w kolejności ich występowania w wielokącie. Algorytm ma złożoność liniową ***O(n),*** ponieważ analizuje każdy wierzchołek dokładnie raz.

Triangulacja

Do wyznaczenia triangulacji zaimplementowany algorytm ***triangulation***. Algorytm ***triangulation*** dzieli wielokąt monotoniczny na trójkąty, tworząc przekątne w sposób systematyczny. Najpierw identyfikuje wierzchołki o minimalnej i maksymalnej wartości ***y***, co pozwala podzielić wielokąt na lewą i prawą stronę. Wierzchołki z każdej strony są porządkowane malejąco względem ***y***, a następnie scalane w jedną monotoniczną sekwencję, w której każdy punkt oznaczony jest jako należący do lewej lub prawej strony. Algorytm przetwarza sekwencję wierzchołków, używając stosu do przechowywania aktualnie analizowanych punktów. Przy zmianie strony dodawane są przekątne łączące bieżący wierzchołek z punktami na stosie, a stos jest resetowany. Dla punktów tej samej strony sprawdzana jest możliwość dodania przekątnej, korzystając z funkcji analizującej geometrię punktów, by uniknąć przecinania się krawędzi. Wynikiem działania algorytmu jest lista przekątnych, reprezentowanych jako pary indeksów wierzchołków ***(i, j)***, gdzie każda para oznacza dodaną linię dzielącą wielokąt na trójkąty. Pomocniczą strukturą danych wykorzystywaną do przechowywania triangulacji jest tablica krotek, w której każda krotka zawiera parę indeksów wierzchołków, między którymi należy dodać przekątną, aby przeprowadzić

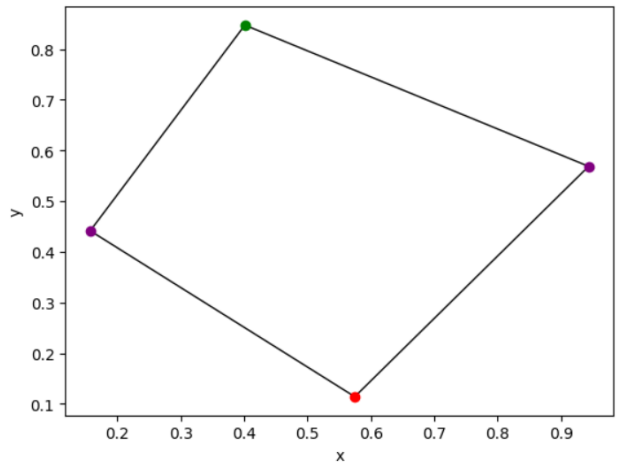
triangulację badanego wielokąta monotonicznego. Struktura ta może być w prosty sposób wykorzystana do generowania wizualizacji triangulacji.

Ostateczna struktura

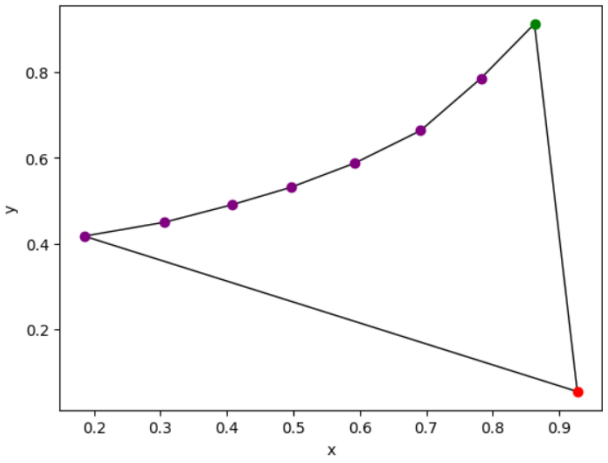
W celu utworzenia ostatecznej struktury przechowującej wielokąt i jego triangulację zaimplementowano funkcję ***znajdz_trojkaty***, która zamienia początkową listę krotek zawierającą pary indeksów wierzchołków ***(i, j)*** na listę krotek ***(i, j, k)*** wierzchołków wielokąta, zawierającą wszystkie trójkąty w triangulacji. Funkcje zmieniającą wierzchołki i przekątne zwrócone przez funkcję ***triangulation*** oparto o metodę biblioteczną ***combinations*** dostępną w bibliotece ***itertools***.

5. *Testowanie algorytmów na wybranych zbiorach danych*

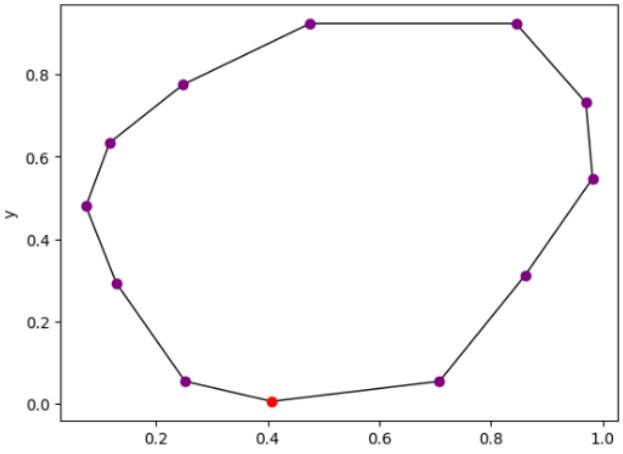
W celu przetestowania algorytmów wybrano różne wielokąty, które miały na celu sprawić programowani problem tudzież sprawdzić działanie programów przy warunkach brzegowych. Na poniższych rysunkach (Rysunki 4 - 13) przedstawiono 10 wielokątów, na których przeprowadzono testy. Przedstawione wielokąty mają już sklasyfikowane wierzchołki, klasyfikacja ta została przedstawiono w punkcie ***3. Wprowadzenie teoretyczne.***



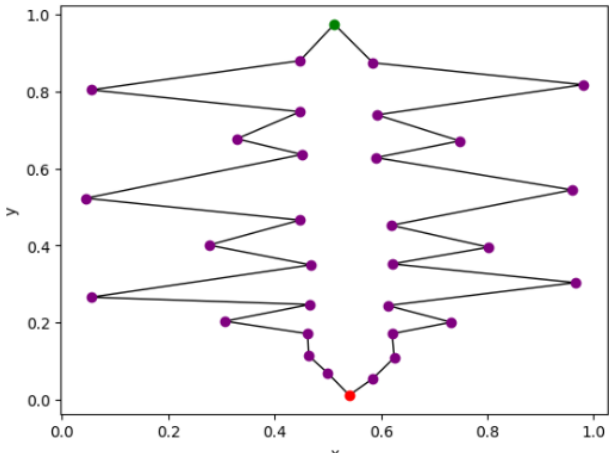
Rysunek 4 Wielokąt 1 po klasyfikacji wierzchołków



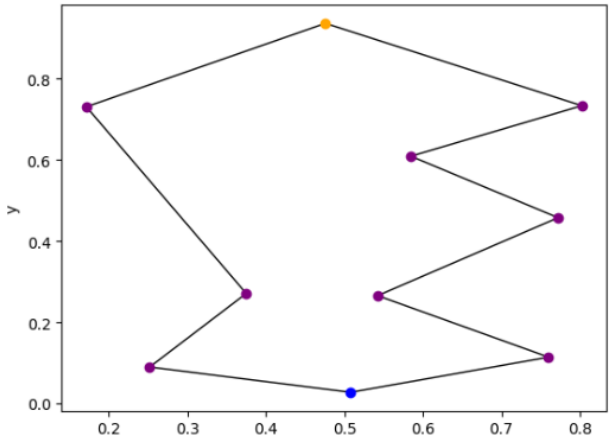
Rysunek 5 Wielokąt 2 po klasyfikacji wierzchołków



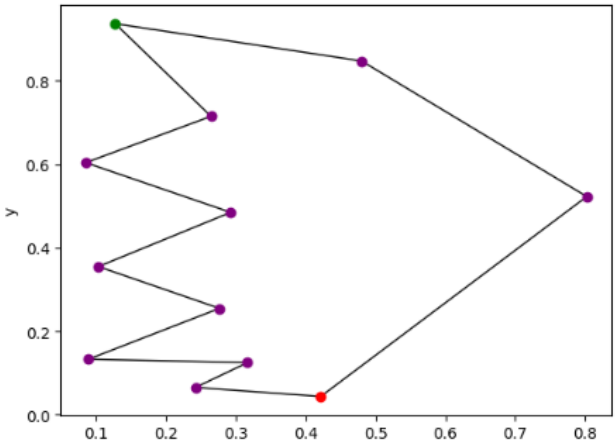
Rysunek 6 Wielokąt 3 po klasyfikacji wierzchołków



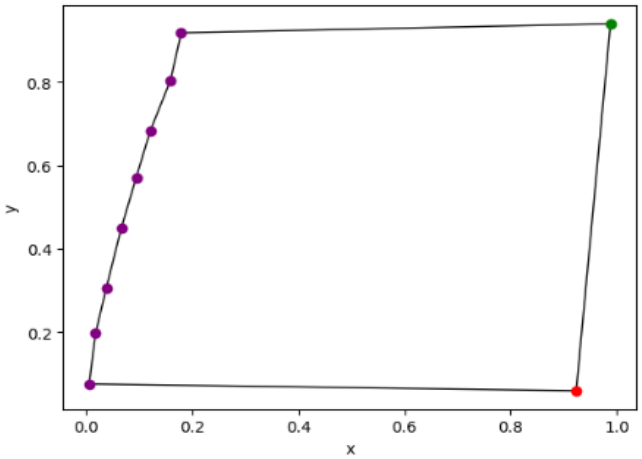
Rysunek 7 Wielokąt 4 po klasyfikacji wierzchołków



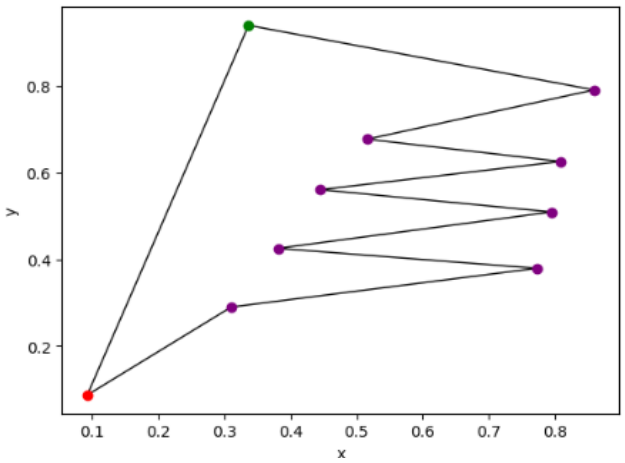
Rysunek 8 Wielokąt 5 po klasyfikacji wierzchołków



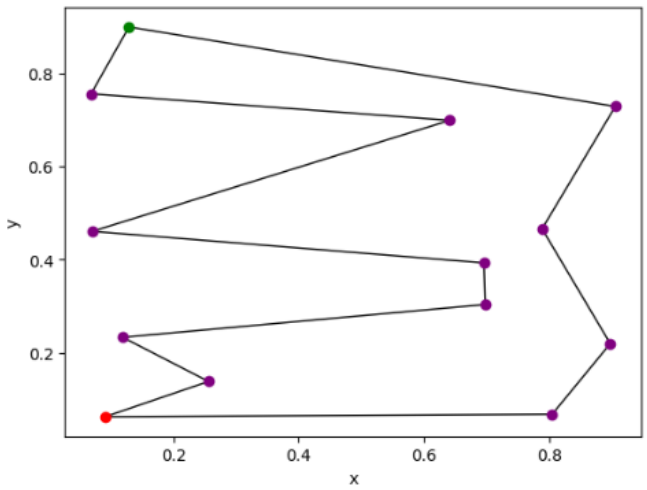
Rysunek 9 Wielokąt 6 po klasyfikacji wierzchołków



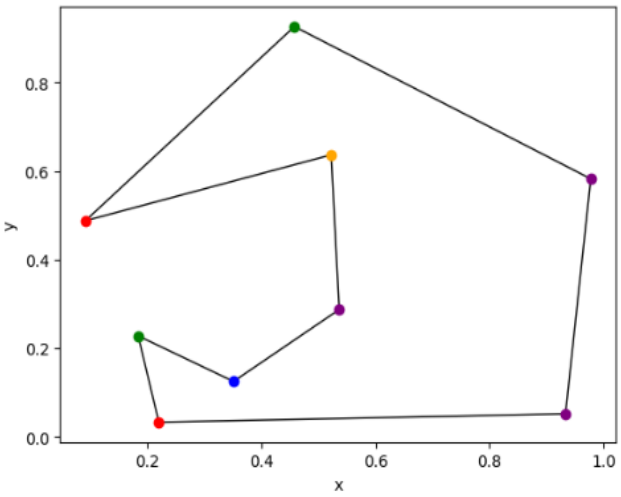
Rysunek 10 Wielokąt 7 po klasyfikacji wierzchołków



Rysunek 11 Wielokąt 8 po klasyfikacji wierzchołków



Rysunek 12 Wielokąt 9 po klasyfikacji wierzchołków

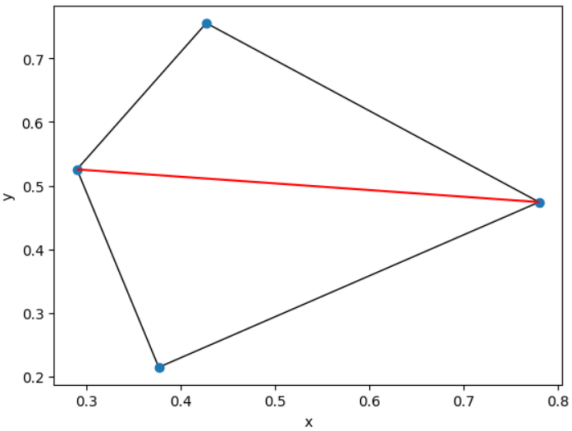


Rysunek 13 Wielokąt 10 po klasyfikacji wierzchołków

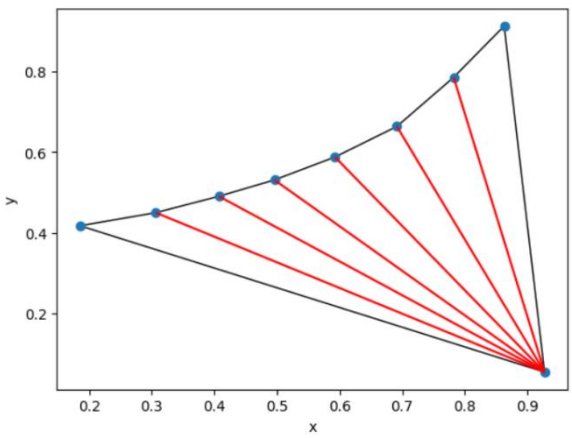
Wielokąt 10, widoczny na Rysunku 13, nie jest wielokątem ***y-monotonicznym***, zamieszczono go w celu ukazania działania algorytmu na wielokątach nie ***y-monotonicznym.***

Triangulacja wielokątów

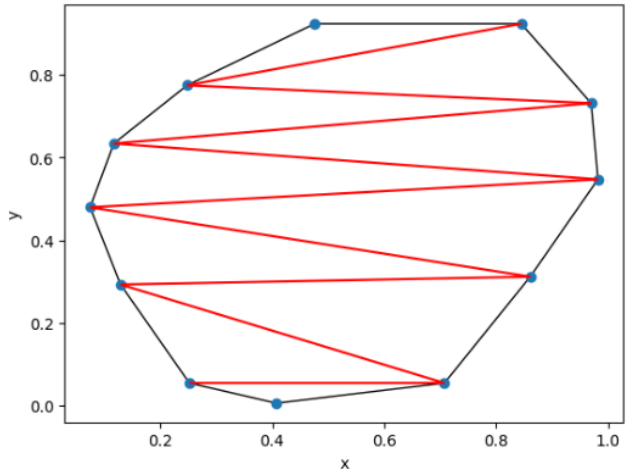
Poniżej przedstawiono wielokąty 1-10 po wykonaniu algorytmu ***triangulacji*** (Rysunki 14-23*)*, dodatkowo, w celu lepszego uwidocznienia działania algorytmu zaprezentowano pliki **.gif** dla każdego z wielokątów 1-10. Pliki gif wizualizujące działanie algorytmu dostępne są w oddzielnym pliku, gdzie ostatnia cyfra w opisie oznacza numer wielokąta, to znaczy plik *vis_gif_2.gif* przedstawia działanie algorytmu ***triangulacji*** dla Wielokąta 2. Na niebiesko oznaczano nierozpatrzone jeszcze punkty, na żółto punkty na stosie, na fioletowo obecnie rozważany trójkąt. Wielokąta 10 nie da się poddać poprawnej triangulacji bez wcześniejszego jego podziału na wielokąty y-monotoniczne, co można zobaczyć na Rysunku 23.



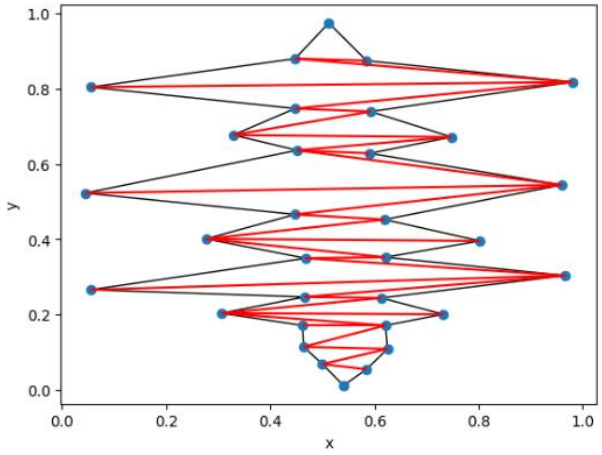
Rysunek 14 Wielokąt 1 po triangulacji



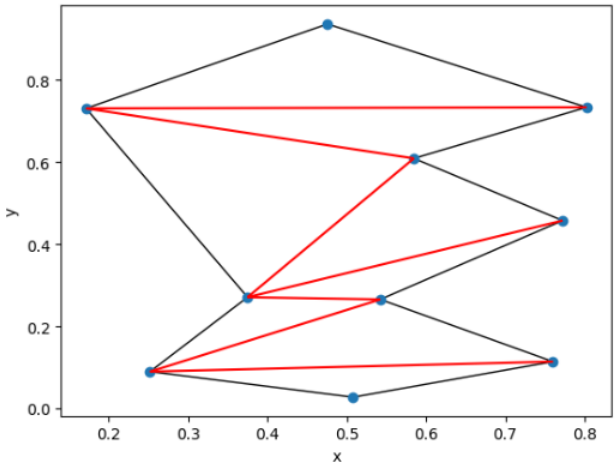
Rysunek 15 Wielokąt 2 po triangulacji



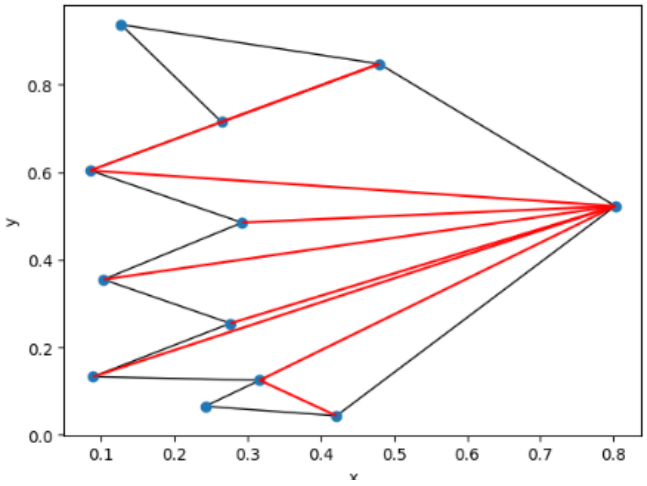
Rysunek 16 Wielokąt 3 po triangulacji



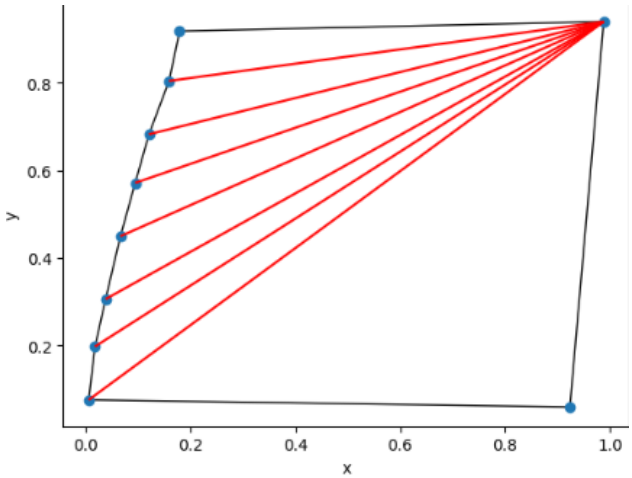
Rysunek 17 Wielokąt 4 po triangulacji



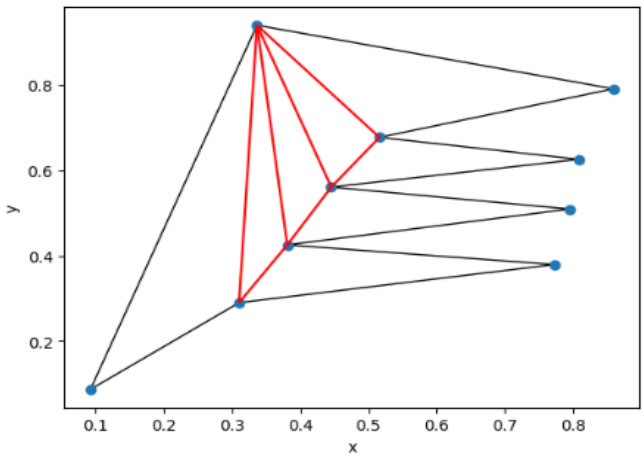
Rysunek 18 Wielokąt 5 po triangulacji



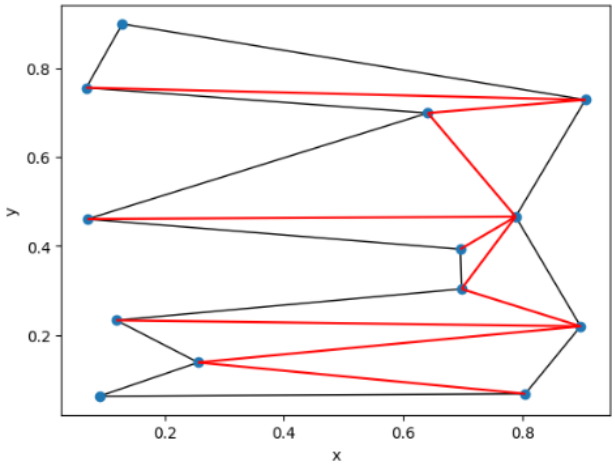
Rysunek 19 Wielokąt 6 po triangulacji



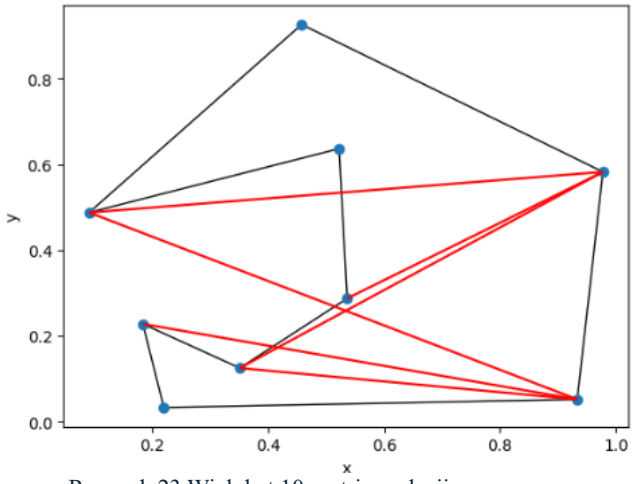
Rysunek 20 Wielokąt 7 po triangulacji



Rysunek 21 Wielokąt 8 po triangulacji



Rysunek 22 Wielokąt 9 po triangulacji



Rysunek 23 Wielokąt 10 po triangulacji

Motywacja wybranych wielokątów jako przypadki testowe

- Wielokąt 1 – podstawowe działanie algorytmu dla prostych wielokątów
- Wielokąt 2 – działanie algorytmu, gdy w pewnym momencie wszystkie punkty były na stosie
- Wielokąt 3 – działanie algorytmu jeśli za każdym razem przechodzimy z jednego łańcuchu na drugi, dodatkowy kiedy wielokąt jest wypukły
- Wielokąt 4 – działanie algorytmu, gdy często, ale nie zawsze, zmieniamy łańcuchy oraz nietypowy kształt
- Wielokąt 5 – działanie algorytmu, gdy na obu łańcuchach jest podobna struktura, ale na jednym łańcuchu jest gęstsza, czyli występuje więcej dynamicznych zmian
- Wielokąt 6 – działanie algorytmu, kiedy większość punktów i skomplikowana struktura jest tylko na jednym z łańcuchów, dodatkowo sprawdzenia czy algorytm poprawnie tworzy wyznacza przekątne, gdy jeden z wierzchołków leży bardzo blisko pewnej wyznaczonej przekątnej, słabo to widać na rysunku, poprawną działalność można uchwycić na przeglądając plik gif z wizualizacją tego wielokąta
- Wielokąt 7 – działanie algorytmu, gdy wszystkie punkty z jednego łańcucha tworzą przekątne z jednym punktem na drugim łańcuchu, inny wielokąt wypukły
- Wielokąt 8 – działanie algorytmu dla punktów leżących na jednym łańcuchu, dodatkowo blisko prostej wyznaczonej między skrajnym wierzchołkami o kącie wklęsłym
- Wielokąt 9 – działanie algorytmu dla często zwężającego i rozszerzającego się wielokąta, gdy zmienne są wybierane punkty z różnych łańcuchów
- Wielokąt 10 – działanie algorytmu dla wielokąta nie ***y-monotonicznego***

Motywacja wybrania struktury końcowej

Jako ostateczną strukturę przetrzymywania rozważono: listę przekątnych, listę trójkątów, Half-Edge Data Structure, DCEL, QuadTree.

Pozytywnego i negatywne cechy rozważanych metod:

- Lista przekątnych
  - Pozytywne cechy: łatwość implementacji, mała złożoność pamięciowa, nie potrzeba dodatkowego algorytmu
  - Negatywy: ciężkie wyciągnięcie jakichkolwiek danych z wyników (nie wszystkie wierzchołki są w wynikach co uniemożliwia odtworzenia początkowego wielokąta)
- Lista trójkątów
  - Pozytywne cechy: prostota reprezentacji, łatwość iteracji, mała złożoność pamięciowa, prostota zapisu i odczytu, prosty algorytm zamieniający dane, kompatybilność
  - Negatywy: redundancja danych, trudność w edycji, brak explicite reprezentowanej topologii, trudności w analizie geometrycznej, brak obsługi bardziej złożonych obiektów
- Half-Edge Data Structure
  - Pozytywne cechy: wspiera dynamiczne zmiany struktury, idealna do operacji topologicznych, nie bardzo trudna do implementacji
  - Negatywy: większe zużycie pamięci, nie znana przez każdego
- DCEL
  - Pozytywne cechy: efektywne rozpatrywanie zapytań przestrzennych, dobrze integruje się z aplikacjami i symulacjami
  - Negatywy: złożona implementacja, nadmiar informacji, duża złożoność pamięciowa
- QuadTree
  - Pozytywne cechy: łatwe i szybkie zapytania o sąsiedztwo, optymalizacja renderowania, łatwa implementacja
  - Negatywy: trudne do utrzymania w przypadku dynamicznych zmian topologicznych, mało intuicyjne dla innych operacji na trójkątach

Wybraną prezentacją jest Lista trójkątów. Listę przekątnych nie rozważono ze względu na jej nieefektywność, a pozostałe ze względu na brak czasu na implementację.

6. Wnioski

Testowane wielokąty miały różnorodne cechy, aby sprawdzić, czy algorytm triangulacji działa poprawnie i nie generuje błędnych przekątnych, takich jak linie wykraczające poza wielokąt, pokrywające się z jego bokami lub przecinające już dodane przekątne. Tego typu błędy uniemożliwiłyby uzyskanie poprawnej triangulacji. Zarówno dla wybranych wielokątów testowych, jak i dla przykładów zaproponowanych przez ***Koło Naukowe AGH Bit***, program działał zgodnie z oczekiwaniami, co potwierdzają wygenerowane rysunki, na których można przeanalizować powstałe trójkąty. Program, oparty na algorytmie przedstawionym podczas wykładu, nie obsługuje wielokątów niemonotonicznych — w ich przypadku zwracałby nieprawidłową triangulację. Poprawność działania algorytmu wspierała przyjęta konwencja przechowywania danych wielokąta oraz zastosowana pomocnicza struktura do triangulacji, która była dobrze dopasowana do używanego algorytmu.