Triangulacja wielokąta prostego – porównanie metod

Bartłomiej Tempka, Radosław Kawa

January 2023

Contents

1 Wprowadzenie

W ramach projektu przygotowano 2 algorytmy triangulacji wielokąta prostego:

- Metoda podziału na wielokąty monotoniczne
- Triangulacja Delaunay

2 Ogólne informacje o programie

Program znajduje się w pliku .ipynb i jest możliwy do odpalenia programem jupyter notebook

2.1 Wymagania techniczne

Program został przygotowany na 2 komputerach:

	System operacyjny	Procesor	Ram
PC 1	Windows 11	AMD Ryzen 5 4500U	8GB
PC 2	macOS	Montery Apple M1	8GB

Table 1: Używane komputery

]

2.2 Biblioteki

- matplotlib
- numpy

2.3 Sposób obsługi

Aby używać programu należy po kolej wywołać moduły a następnie na wykresie zadać wielokąt kliknięciem myszki, po "zamknięciu" wielokąta wywołana zostanie metoda podziału na wielokąty monotoniczne, włączenie następnej komórki spowoduje wywołanie na tym samym wielokącie triangulacji metodą Delaunaya. Uwagi: wielokąt powinien zostać zadany w kierunku przeciwnym do wskazówek zegara oraz powinien nie zawierać krawędzi poziomych pionowych oraz współliniowych

3 Dokumentacja

4 Metoda podziału na wielokąty monotoniczne

4.1 Funckje pomocnicze

4.1.1 plotPoints(points)

funkcja pomocnicza zwracająca zbiór punktów do narysowania wykresu

4.1.2 distance(a, b)

zwraca odległośc punktów a i b

4.1.3 det3x3(a,b,c)

zwraca orientacje 3 punktów

4.1.4 Intersection(l1, l2)

zwraca punkt przecięcia lini l1 i l2

4.1.5 merge(list1,list2,key)

argumenty:
list1, list2 - posortowane listy
key - klucz według którego posortowane są list (bazowo key(x) = x) return - posortowana lista zawierająca list1 ∪ list2

4.2 Interval Tree

Drzewo przchowujące przedziały znajdujące sie w wielokącie.

4.2.1 Node

Zmienne:

- leftEdge- krawędź ograniczająca przedział z lewej strony
- rightEdge- krawędź ograniczająca przedział z prawej strony
- helper- pomocnik przedziału
- left- lewy syn w drzewie
- right- prawy syn w drzewie
- parent - ojciec w drzewie

Metody:

- __init__(leftEdge, rightEdge)- konstruktor

4.2.2 BinarySearchTree

Zmienne:

- root- korzeń drzewa
- broom- pozycja miotły używan
- key- klucz na którego podstawie dodawane są przedziały do drzewa
- points- wielkokąt którego dotyczy drzewo

Metody:

- __init__(poinst)- konstruktor drzewa, przyjmuje wielokątk którego dotyczy
- insert(node, value)- wkłada Node do Drzewa, porównuje przedziały to wartosci value
- delete(node)- usuwa node z drzewa
- findEv(value)- znajduje przedział do którego należy value
- findNext(node)- znajduje następna Node w drzewie

4.3 Funkcje Algorytmu

4.3.1 assignVertices(points)

Przyporządkowuje wierzchołką w zbiorze points ich typ (początkowy, końcowy,łączący, dzielący, prawidłowy)

4.3.2 findDiagonals(points)

Funkcja zwraca przekątne dzielące wielokąt niemonotoniczny- points na wielokąty monotoniczne

4.3.3 makeMonotone(points, diagonals)

Przyjmuje niemonotoniczny wielokąt- points, i przekątne dzielące na wielokąty monotoniczne- diagonals
Zwraca 2 wymiarową tablice zawierającą wielokąty monotoniczne

4.3.4 triangulate(points)

Przyjmuje wielokąt monotoniczny.
Zwraca triangulacje.

5 Triangulacja Delaunaya

5.1 Funkcje Pomocnicze

5.1.1 Orientation(a,b,c)

zwraca orientacje punktów w jaki sposób zostały zadane (względem przeciwnym do wskazówek zegara)

5.1.2 Det(a,b,c)

zwraca wyznacznik

5.1.3 Orient(a,b,c,epsilon)

zwraca orientacje punktów dla przecinania się niektórych odcinków

5.1.4 doIntersect(p1,q1,p2,q2)

zwraca informacje czy istnieje przecięcie dwóch zadanych odcinków

5.2 Klasy

5.2.1 Point

Zmienne:

- X – współrzedna x punktu
- Y – współrzedna y punktu

Metody:

- __init__(x,y)- inicjuje wspolrzedne punktów

- __repr__(self) – reprezentacja punktu jako string
- cross(self,point) – metoda do obliczania wyznacznika do okręgu opisanego na trójkącie

5.2.2 Edge

Zmienne:

- pt1- punkt nr 1
- pt2- punkt nr 2

Metody:

- __init__(pt1,pt2)- inicjuje krawędź

5.2.3 Triangle

Zmienne:

- pt1- punkt nr 1
- pt2- punkt nr 2
- pt3- punkt nr 3
- edges- tablica krawędzi

Metody:

- __init__(pt1,pt2,pt3)- inicjuje trójkąt
- • in_circumcircle(self, point) – metoda sprawdzająca czy dany punkt znajduje się okręgu opisanym na tym trójkącie
- has_vertex(self,vertex) – metoda sprawdzająca czy trójkąt zawiera podany wierzchołek

5.3 Funkcje Algorytmu

5.3.1 supertriangle(point_list)

funkcja przyjmują wierzchołki zadane, zwraca supertrójkąt (trójkąt który zawiera wewnątrz wszystkie zadane punkty)

5.3.2 shared_edge(edge1,edge2)

funkcja sprawdzajaca czy dane 2 krawędzie sa te same

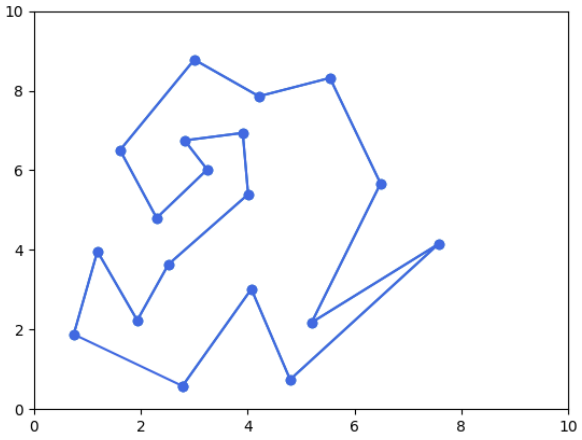
5.3.3 delaunay(P) funkcja triangulacji, przyjmuje zadane wierzchołki, wyświetla wykres z triangulacją zadanego wielokąta

6 Dodatkowe funckje

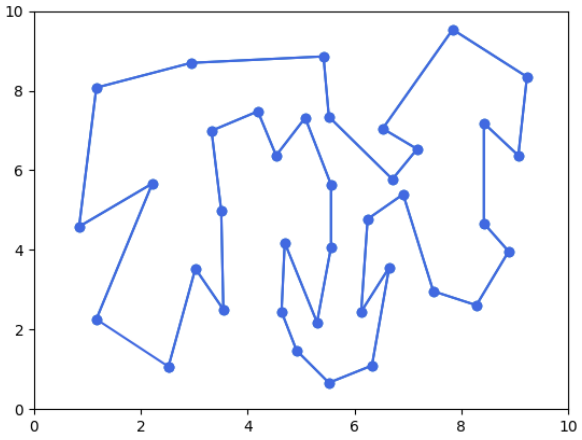
Funkcje ze słowem 'Fast' w nazwie nie zawierają kodu odpowiedzialnego za wizualizacje używane do pomiaru czasu działania algorytmów

7 Sprawozdanie

7.1 Wielokąty



(a) Wielokąt 1



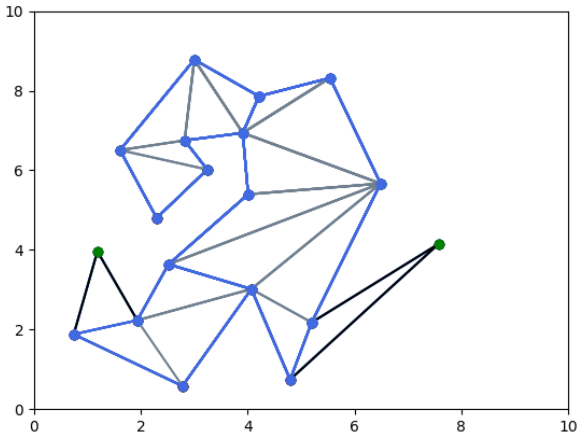
(b) Wielokąt 2

7.2 Czasy działania

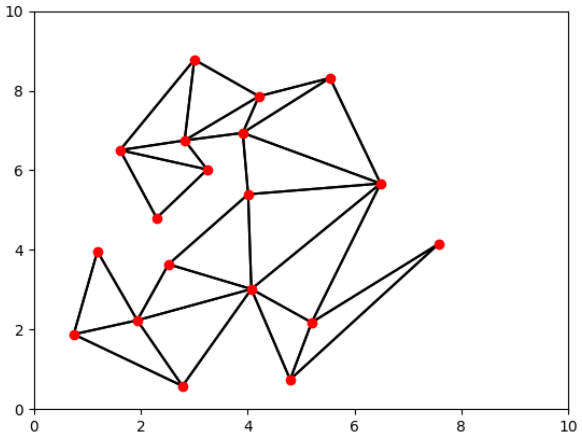
Algorytm	Wielokąt 1	Wielokąt 2
Zamiana na wielokąty monotoniczne	0.0054s	0.0232
Triangulacja Delaunaya	0.0063s	0.0239

Table 2: Czasy działania algorytmów

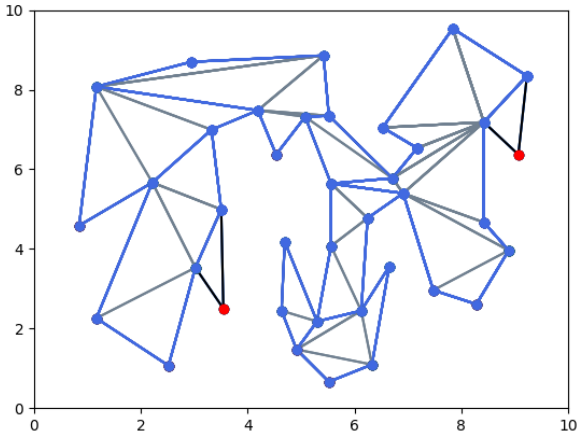
7.3 Wygląd Triangulacji



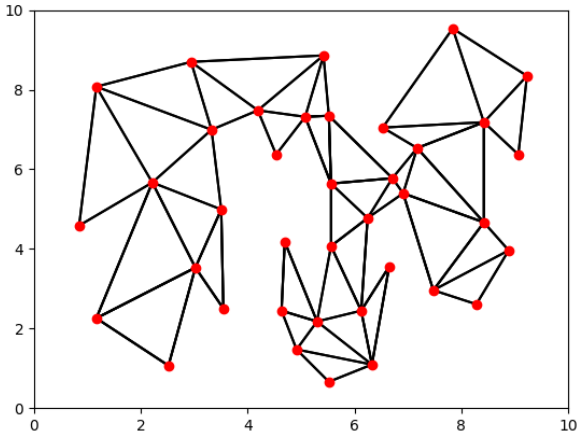
(c) Triangulacja przez zmianę na wielokąty monotoniczne wielokąta 1



(d) Triangulacja Delaunaya wielokąta 1



(e) Triangulacja przez zmianę na wielokąty monotoniczne wielokąta 2



(f) Triangulacja Delaunaya wielokąta 2

7.4 Wnioski

- Algorytmy są porównywalne szybkościowo.

- Triangulacja uzyskana algorytmem Delauanaya nie posiada trójkątów o bardzo małych kątach wewnętrznych.