28.11.2023 r.

Mateusz Król

Algorytmy geometryczne
Triangulacja wielokątów monotonicznych
Ćwiczenie nr 3

Spis treści

1. Realizacja ćwiczenia

W ramach trzeciego ćwiczenia, zaimplementowałem następujące algorytmy:

- algorytm sprawdzający czy podany na wejściu wielokąt jest y-monotoniczny,
- algorytm przypisujący wierzchołkom wielokąta odpowiednia wartość numeryczną symbolizującą kolor zależnie od tego czy w danym wielokącie jest to wierzchołek: początkowy, końcowy, łączący, dzielący czy prawidłowy,
- algorytm obliczający dla zadanego wielokąta y-monotonicznego jego triangulację.

Korzystając ze znalezionego fragmentu kodu umożliwiłem również ręczne (za pomocą kliknięć myszki) wprowadzanie wielokątów, które są następnie zapisywane do tablicy krotek symbolizujących wierzchołki wielokąta – odpowiada za to funkcja **createPolygon()**.

2. Algorytm sprawdzający czy wielokąt jest y-monotoniczny

```python
def is_y_monotonic(polygon):
    n = len(polygon)
    direction = polygon[1][1] -polygon[0][1] # + gdy idziemy w gore, - gdy idziemy w dol
    i = 2
    while direction==0:
        direction = polygon[i][1] -polygon[i-1][1]
        i+=1
    while i<n and direction*(polygon[i][1]-polygon[i-1][1])>=0:
        i+=1
    direction*=-1
    while i<n and direction*(polygon[i][1]-polygon[i-1][1])>=0:
        i+=1
    direction*=-1
    while i<n and direction*(polygon[i][1]-polygon[i-1][1])>=0:
        i+=1
    return i==n
```

Algorytm polega na przejściu po wszystkich wierzchołkach i sprawdzanie dla każdej pary sąsiadujących wierzchołków, czy zmienił się kierunek monotoniczności – czy różnica drugich współrzędnych zmieniła znak w stosunku do poprzedniej pary wierzchołków. W zależności od pierwszego wierzchołka tablicy, kierunek monotoniczności może się zmienić trójkrotnie – możemy zacząć między wierzchołkami maksymalnym i minimalnym w tym względzie.

Kierunek jest przechowywany pod postacią zmiennej ***direction*** i zmienia swój znak na przeciwny dla każdej zmiany kierunku monotoniczności. Na początku algorytmu szukam dwóch punktów, dla których wartości drugiej współrzędnej się różnią, żeby nie otrzymać sytuacji, w której ***direction*** = 0.

Algorytmy kończy swoje działanie, gdy ‘damy szansę’ wielokątowi na zmienienie kierunku trzy razy, a iterator nie dojdzie do końca tablicy.

Algorytm przechodzi wszystkie dostarczone testy, posiada złożoność rzędu O(n).

3. Procedura triangulacji wielokąta y-monotonicznego

Przy implementacji algorytmu korzystałem ze słownika ***edges*** utrzymującego informacje o triangulacji: kluczami są indeksy wierzchołków, a wartościami tablice indeksów wierzchołków, z którymi klucze tworzą krawędź wielokąta lub krawędź triangulacji.

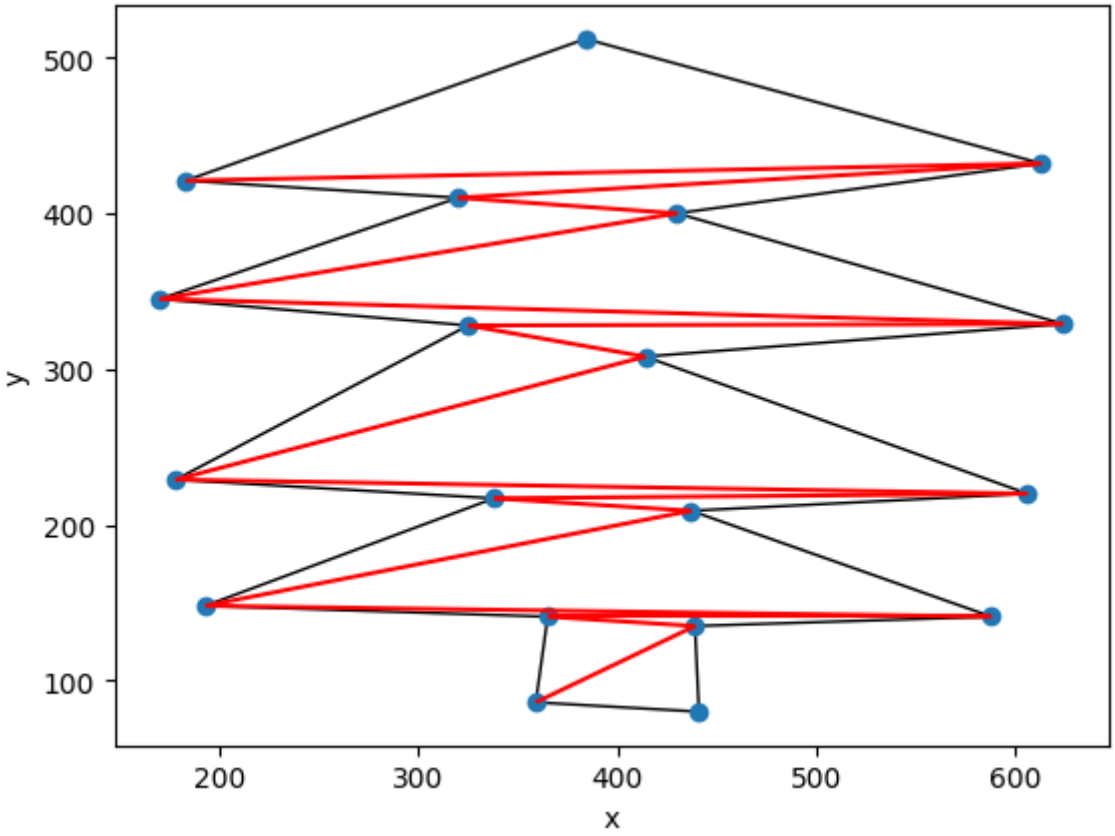
Dodatkowo indeksy wierzchołków tworzących krawędzie triangulacji wielokąta przechowywane są w tablicy ***diagonals***, w celu późniejszego przeformatowania danych i zwrócenia wyniku.

Zdecydowałem się na wybranie powyżej podanych struktur, ponieważ dzięki temu, że wielokąty zadawane są przeciwnie do ruchu wskazówek zegara, jesteśmy w stanie na podstawie samych indeksów stwierdzić czy między wierzchołkami istnieje krawędź. Przechowywanie i porównywanie samych indeksów jest o wiele bardziej efektywne niż wykonywanie porównań i obliczeń na prawdziwych danych. Daje nam to również gwarancję poprawności takowych obliczeń.

4. Testy algorytmu na wygenerowanych zbiorach danych

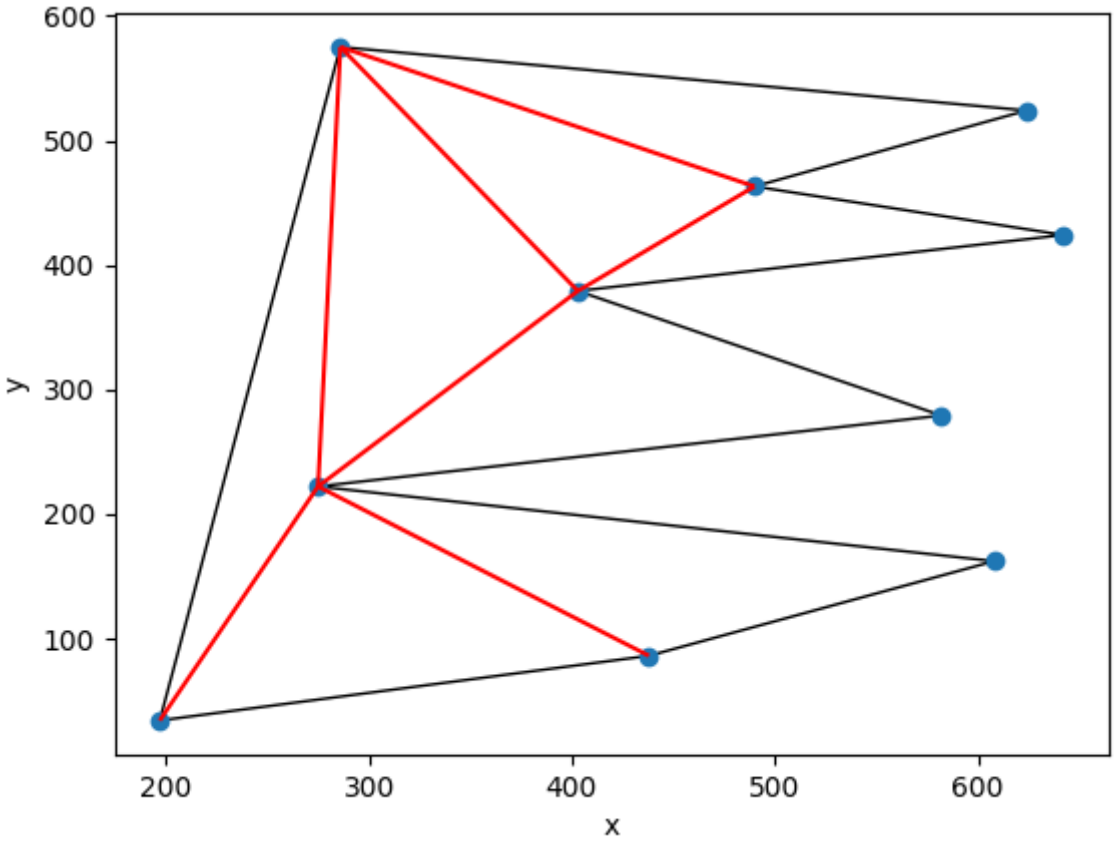
W celu analizy przebiegu działania algorytmu triangulacji wielokąta y-monotonicznego, korzystam z funkcji **draw_triangulation()**, która wykorzystuje ten sam algorytm, ale przy okazji umożliwia wizualizację rozważanych odcinków, odcinków, które zostały zatwierdzone jako poprawne i dodane do triangulacji, obecnie rozważanego punktu oraz punktów obecnie znajdujących się na stosie.

- Zbiór 1 – „choinka”:



Zgodnie z przewidywaniami, algorytm dla każdego nowego rozpatrywanego wierzchołka, łączy go z wierzchołkiem ze szczytu stosu.

- Zbiór 2:



Zgodnie z przewidywaniami, w momencie, w którym algorytm rozpatruje wierzchołki umieszczone najbardziej po prawej stronie wielokąta, decyduje się je pominąć (dodać na stos) i w kolejnym kroku „odciąć” trójkąty utworzone z właśnie tego wierzchołka, nowego rozpatrywanego wierzchołka ***current*** oraz wcześniejszych wierzchołków znajdujących się na stosie.

5. Wnioski

Zaimplementowane przeze mnie trzy algorytmy przechodzą wszystkie testy zadane w pliku z laboratorium. Na podstawie animacji działania algorytmu dla podanych oraz dla różnorakich własnoręcznie wygenerowanych zbiorów danych, jestem w stanie stwierdzić, że algorytm wyznaczania triangulacji dla wielokątów y-monotonicznych funkcjonuje poprawnie.