Łukasz Stępień gr. IV

Algorytmy geometryczne

sprawozdanie z ćw. 3

1. Cel ćwiczenia:

Implementacja algorytmów sprawdzania y-monotoniczności zadanego wielokąta, klasyfikacja wierzchołków w dowolnym wielokącie, triangulacja wielokąta y-monotonicznego wraz z wizualizacją.

2. Dane techniczne:

Język implementacji: Python

Środowisko programistyczne: Jupyter Notebook

System operacyjny: Microsoft Windows 10 Pro x64

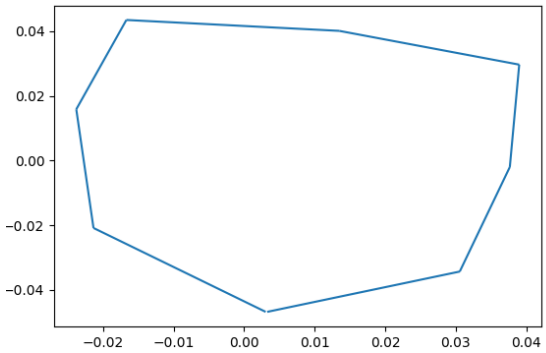
Procesor: Intel(R) Core(TM) i5-10400F CPU @ 2.90GHz, 2904 MHz

3. Zestawy danych i ich wizualizacja:

Aplikacja pozwala na tworzenie figur za pomocą dostarczonego narzędzia graficznego. Za pomocą myszki należy wprowadzić kolejne punkty należące do figury. Po narysowaniu figura jest zapisywana do pliku *json* w postaci listy krawędzi, w kolejności ich dodania. Dodatkowo przygotowane zostały funkcje pomocnicze zamieniające figury na listę linii oraz listę linii na listę kolejnych punktów. Figury można wczytywać z wcześniej utworzonych plików i pobierać z nich potrzebne dane oraz wyświetlać.

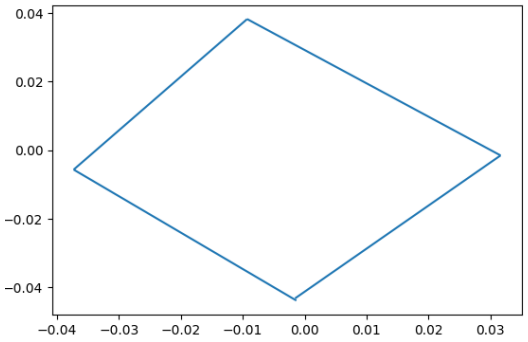
W celu przetestowania algorytmów zostały utworzone następujące figury:

- Zestaw A - przeciętny wielokąt („figura.json”):



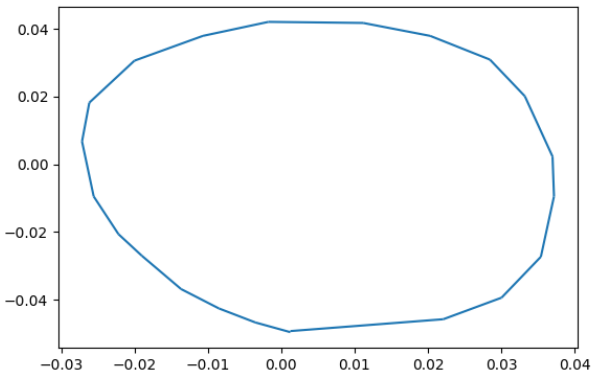
Wykres 1 Zestaw A

- Zestaw B - czworokąt („czworokat.json”) :



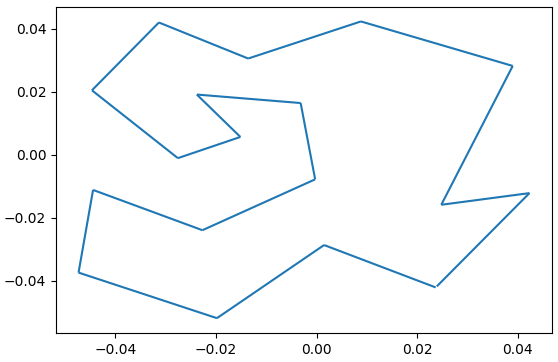
Wykres 2 Zestaw B

- Zestaw C - okrąg („okrag.json”):



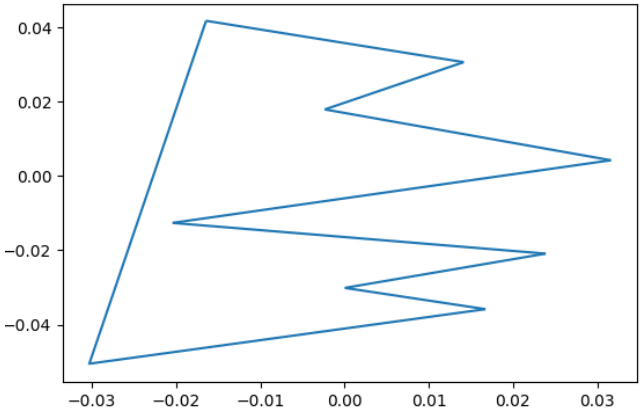
Wykres 3 Zestaw C

- Zestaw D - figura niemonotoniczna („non_monotonous.json”):



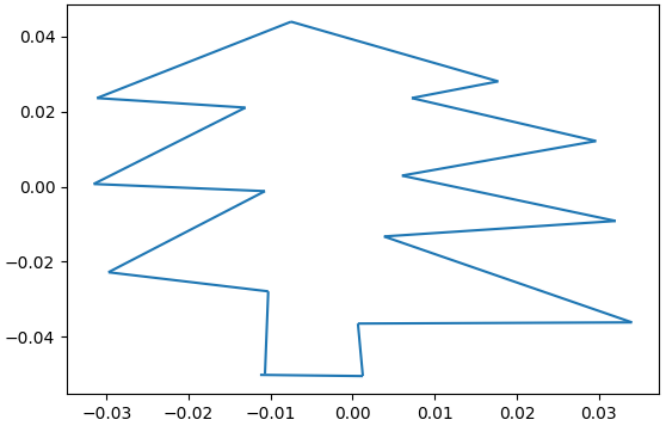
Wykres 4 Zestaw D

- Zestaw E - Flaga („flag.json”) :



Wykres 5 Zestaw E

- Zestaw F - Choinka („choinka.json”):



Wykres 6 Zestaw F

Figury „czworokąt” i „okrąg” i „figura” testują podstawowe przypadki (figury wypukłe, każdy punkt “widoczny” z każdego innego). Figura niemonotoniczna służy sprawdzeniu procedury określania y-monotoniczności. Pozostałe figury nie są wypukłe i testują „odcinanie” krawędzi z wierzchołków, niezależność działania algorytmu triangulacji od orientacji figury, zrównoważenia wielkości obu gałęzi oraz położenia punktów “ekstremalnych”, czyli takich z którymi połączonych będzie wiele innych punktów.

Wszystkie zaimplementowane w ramach tego ćwiczenia algorytmy korzystają z reprezentacji wielokątów w postaci listy wierzchołków w kolejności ich dodawania. Pozwala to na szybkie znajdowanie sąsiadów wierzchołków oraz lewej i prawej gałęzi figury.

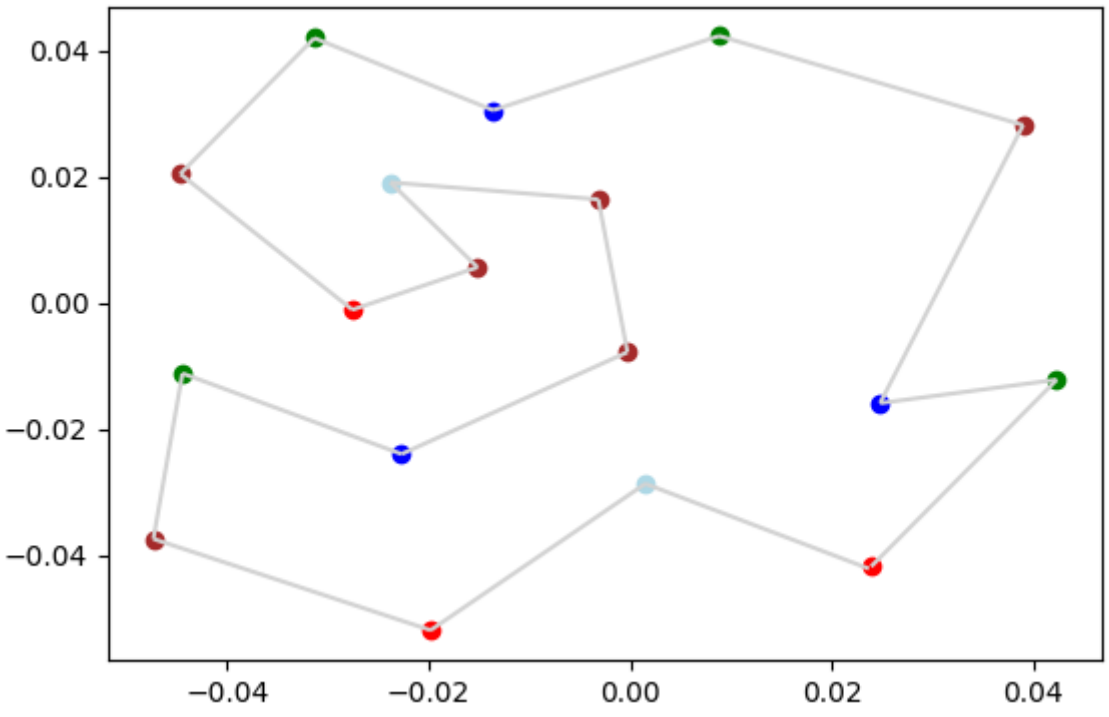
4. Sprawdzanie monotoniczności

W celu sprawdzenia monotoniczności zadanego wielokąta wykorzystano kolejność punktów go tworzących. Wielokąt został zamieniony na listę kolejnych punktów z których się składa, następnie znaleziono indeks punktu o największej i najmniejszej współrzędnej y. Kolejnym krokiem było sprawdzenie, czy każdy kolejny punkt idąc od najwyższego do najniższego leży poniżej wcześniejszego punktu. Warunek ten należało sprawdzić dla obydwóch gałęzi (po obu stronach wierzchołka najwyższego). W przypadku choć jednego odstępstwa od tej zasady można stwierdzić, że figura nie jest monotoniczna.

5. Klasyfikacja wierzchołków

Algorytm klasyfikacji wierzchołków wielokąta na początkowe, końcowe, łączące, dzielące i prawidłowe polega na jednokrotnym przejściu po liście wierzchołków i ich klasyfikacji na podstawie ich położenia w relacji do sąsiadów (wierzchołka poprzedniego i następnego). Należało również uwzględnić to, że pierwszy i ostatni wierzchołek na liście również ze sobą sąsiadują. W celu określenia kąta wewnętrznego jaki tworzą rozpatrywane wierzchołki użyto wyznacznika. Punkty były klasyfikowane na podstawie następujących warunków, z następującymi oznaczeniami:

- *początkowy*, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny < π (zielony),
- *końcowy*, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny < π (czerwony),
- *łączący*, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny > π (ciemny niebieski),
- *dzielący*, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny > π (jasnoniebieski),
- *prawidłowy* w pozostałych przypadkach (brązowy).



Wykres 7 Wizualizacja algorytmu kategoryzowania wierzchołków dla zestawu D

6.1. Triangulacja – opis algorytmu

Algorytm triangulacji na wejściu przyjmuje listę punktów, będących wierzchołkami tworzącymi dany wielokąt w kolejności ich wprowadzenia, zgodnie z ruchem przeciwnym do ruchu wskazówek zegara. Algorytm generuje listę krawędzi, która zawiera zarówno krawędzie należące do wielokąta oraz przekątne wewnętrzne, będące rezultatem triangulacji. Użycie listy krawędzi ma następujące zalety: Krawędzie są najprostszym typem obiektów które definiują rezultat triangulacji - zapis w postaci listy trójek punktów reprezentujących trójkąty byłby trudniejszy w przeglądaniu, rysując go musielibyśmy uważać na duplikaty krawędzi, które w tym rozwiązaniu nie mają miejsca; W prosty sposób możemy znaleźć krawędzie (oraz ich ilość), które wychodzą z zadanego wierzchołka. Ponadto taka reprezentacja nie wymaga deklarowania dodatkowych struktur, jest przejrzysta oraz kompatybilna z zadanym narzędziem graficznych bez żadnych konwersji.

Poza listą krawędzi dodatkowo zwracana jest lista scen zawierających kolejne kroki procesu triangulacji (kolejno dodawane krawędzie) w celach wizualizacji.

Na początku algorytm dokonuje podziału punktów na dwie gałęzie - lewą i prawą. Jest to dokonane poprzez znalezienie indeksów punktów najniższego i najwyższego. Dzięki uporządkowaniu punktów w liście w kolejności ich wstawiania punkty znajdujące się w przedziale od indeksu punktu najniższego do najwyższego stanowią jedną gałąź (w tym przypadku prawą), natomiast pozostałe drugą (lewą). Punkt najniższy znajduje się w prawej gałęzi, natomiast najwyższy w lewej. Dla każdego punktu przechowujemy informacje do której gałęzi przynależy. Następnie punkty są sortowane od najwyższego do najniższego (wg. współrzędnej y).

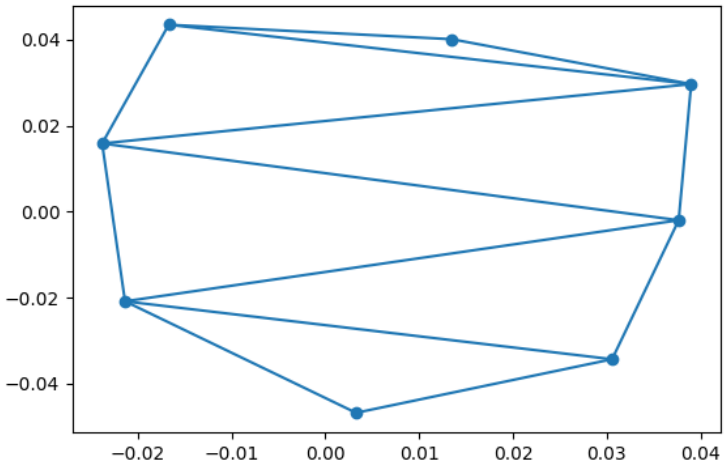
Przed rozpoczęciem głównej pętli na stos dodajemy dwa najwyższe wierzchołki. Następnie wykonywana jest główna pętla, w której rozważamy pojedynczo każdy kolejny wierzchołek. Jeśli rozważany wierzchołek znajduje się na innej gałęzi niż ostatni ze stosu, to łączymy go ze wszystkimi wierzchołkami na stosie (podczas łączenia są one zdejmowane ze stosu), po wykonaniu tej operacji na stosie umieszczane są dwa ostatnio rozważane wierzchołki.

W przypadku gdy rozważany wierzchołek znajduje się na tej samej gałęzi co wierzchołek ze szczytu stosu dodajemy krawędź między tymi dwoma wierzchołkami (jest to krawędź zewnętrzna wielokąta), a następnie podobnie jak w poprzednim przypadku łączymy rozważany punkt ze wszystkimi znajdującymi się na stosie, ale tylko pod warunkiem, że krawędź je łącząca będzie zawierała się we wnętrzu wielokąta (co sprawdzamy przy pomocy wyznacznika).

Po wykonaniu tej operacji na stosie umieszczany jest rozważany wierzchołek oraz ostatni wierzchołek zdjęty ze stosu. Po wykonaniu się głównej pętli otrzymujemy rezultat w postaci listy krawędzi wielokąta oraz utworzonych przekątnych. Poniżej przedstawiono wyniki działania algorytmu dla testowanych figur.

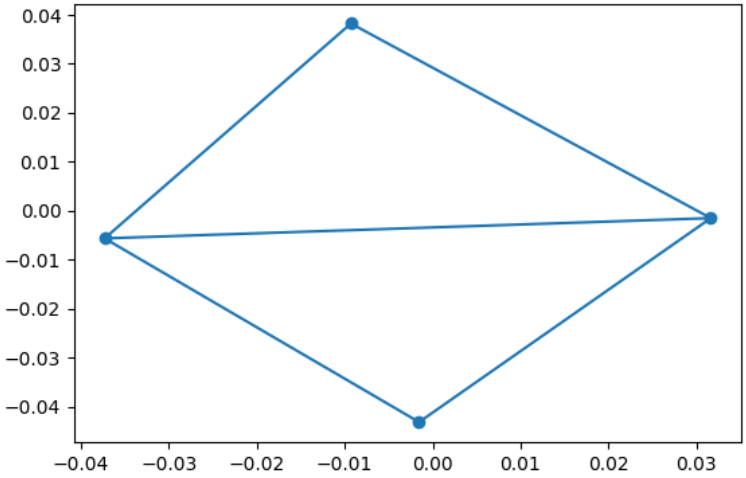
6.2. Triangulacja – wizualizacja dla zestawów A-D

- Zestaw A:



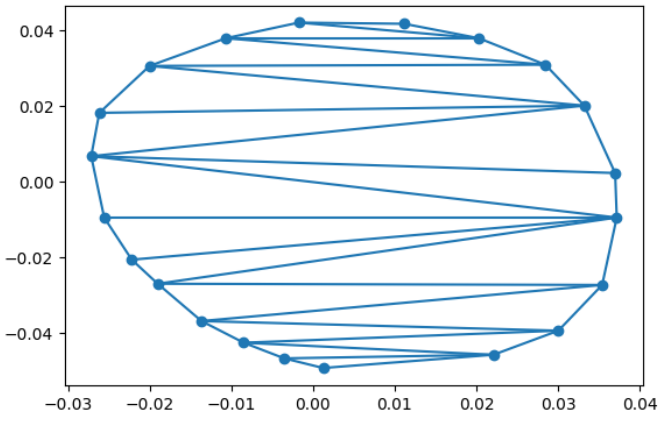
Wykres 8

- Zestaw D:



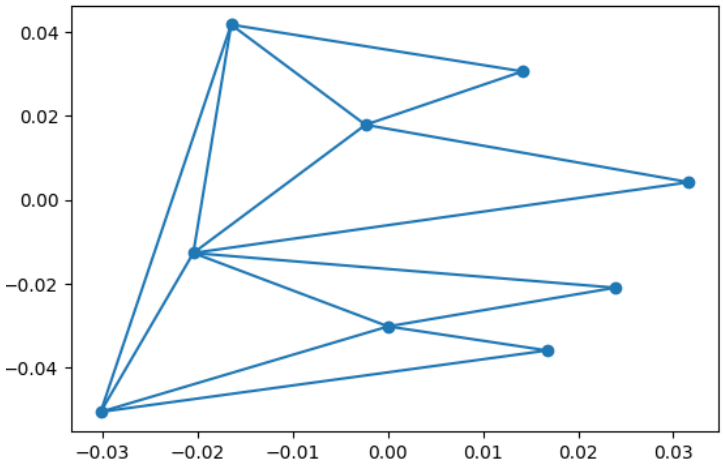
Wykres 9

- Zestaw C:



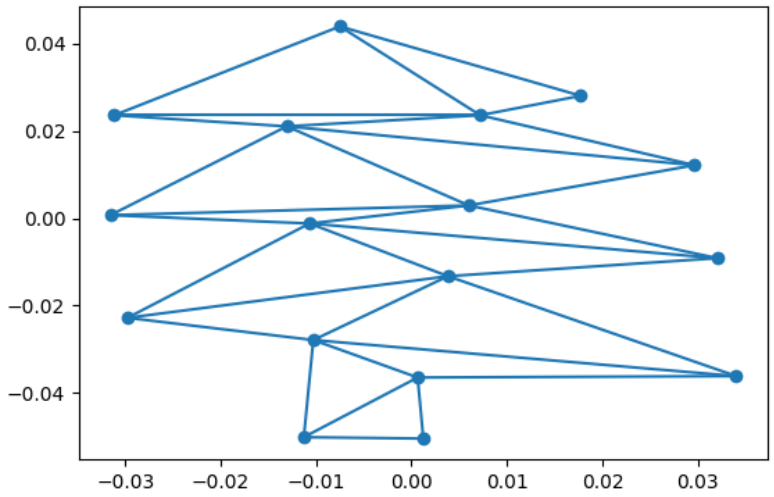
Wykres 10

- Zestaw E:



Wykres 11

- Zestaw F:



Wykres 12

7. Podsumowanie i wnioski:

Wszystkie zaimplementowane procedury i algorytmy zadziałały poprawnie dla testowanych wielokątów. Poprawność triangulacji można było łatwo zweryfikować korzystając z własności wielokątów prostych, mówiącej że dla wielokąta o n wierzchołkach efektem triangulacji będzie n-2 trójkątów. Ponadto algorytm poprawnie nie generuje duplikatów krawędzi. Użyte struktury danych doskonale pasowały do zadanych problemów - pozwalały na szybki dostęp do potrzebnych elementów, nie wymagały dodatkowych konwersji (poza pierwotną - z narysowanej figury) oraz były bezpośrednio kompatybilne z narzędziem do wizualizacji. Możliwość zapisu i odczytu figur z pliku w postaci listy krawędzi znacząco ułatwia korzystanie i testowanie aplikacji dla różnych przypadków.