Dariusz Marecik gr. 4, Pon. godz. 15:00 A

Data wykonania: 04.11.2024 Data oddania: 20.11.2024

Algorytmy Geometryczne - laboratorium 3

Triangulacja wielokątów monotonicznych

1. Cel ćwiczenia

Celem ćwiczenia było zgłębienie zagadnień związanych z monotonicznością wielokątów oraz implementacja kluczowych algorytmów: sprawdzania y-monotoniczności wielokąta, klasyfikacji wierzchołków w dowolnym wielokącie oraz triangulacji wielokąta monotonicznego. Dodatkowo przeprowadzono wizualizację i analizę uzyskanych wyników, co umożliwiło dokładne zrozumienie działania i jakości zastosowanych metod.

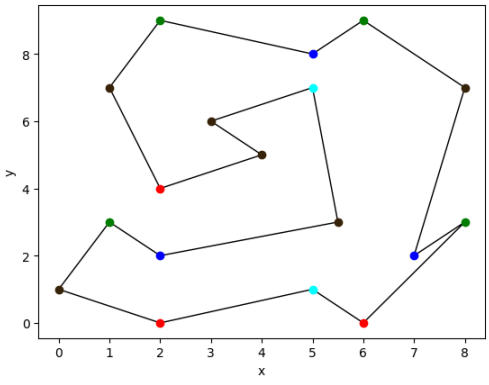
2. Wstęp teoretyczny

Wielokąt prosty nazywamy ściśle monotonicznym względem prostej l (która wyznacza kierunek monotoniczności), gdy jego brzeg można podzielić na dwa spójne łańcuchy, przy czym każda prosta l' prostopadła do l przecina każdy z tych łańcuchów w co najwyżej jednym punkcie. W tym ćwiczeniu rozpatrujemy y-monotoniczność, czyli przyjmujemy, że naszą prostą l jest oś OY

W przypadku y-monotoniczności oznacza to, że wierzchołki wielokąta są rozmieszczone tak, że każdy z nich (poza wierzchołkiem o największej i najmniejszej współrzędnej y) ma sąsiadów — jednego powyżej (z większą współrzędną y) i jednego poniżej (z mniejszą współrzędną y).

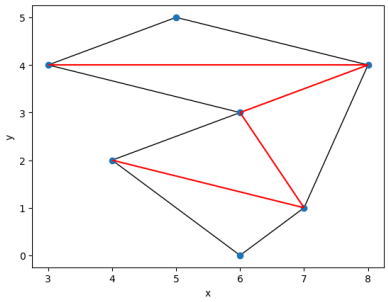
Wierzchołki danego wielokąta prostego mogą zostać podzielone na 5 kategorii. Kolory czcionki nazw punktów odpowiadają kolorom wierzchołków na wizualizacjach (Rysunek 1).

- ***wierzchołek początkowy***, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny $< \pi$,
- ***wierzchołek końcowy***, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny $< \pi$,
- ***wierzchołek łączący***, gdy obaj jego sąsiedzi leżą powyżej i kąt wewnętrzny $> \pi$,
- ***wierzchołek dzielący***, gdy obaj jego sąsiedzi leżą poniżej i kąt wewnętrzny $> \pi$,
- ***wierzchołek prawidłowy***, w pozostałych przypadkach (ma jednego sąsiada powyżej, drugiego – poniżej).



Rysunek 1: Przykładowy wielokąt z podzielonymi wierzchołkami na kategorie

Wielokąt monotoniczny charakteryzuje się tym, że nie posiada wierzchołków łączących i dzielących.



Rysunek 2: Przykładowa triangulacja wielokąta y-monotonicznego

Triangulacja wielokąta monotonicznego to podział wnętrza wielokąta na trójkąty za pomocą odcinków (zwanych przekątnymi), które łączą pary wierzchołków, przy czym przekątne nie przecinają się w obrębie wielokąta. Przykład jednej z możliwych triangulacji wielokąta został przedstawiony na Rysunku 2 obok.

Łańcuch w wielokącie jest definiowany jako ciąg punktów o monotonicznej strukturze względem współrzędnej y.

3. Dane techniczne

3.1. Specyfikacja narzędzi i sprzętu

Do generowania obrazów przedstawiających wielokąty i gifów przedstawiających krok po kroku działanie poszczególnych algorytmów została wykorzystana biblioteka *matplotlib* i *pandas* oraz narzędzie przygotowane przez *koło naukowe Bit*. Program został napisany w języku Python 3.10 w środowisku *Jupiter Notebook*. Przy pomocy tego narzędzia zaimplementowałem program umożliwiający zadawanie prostych wielokątów przy użyciu myszki, z zapisem i odczytem podanych wielokątów. Wykorzystany procesor do zebrania danych to Intel Core i5-12500H 4.5 GHz pracujący w systemie Linux Mint 21.4.

3.2. Metodologia

3.2.1. Przechowywanie danych

Wielokąty są reprezentowane jako lista krotek współrzędnych (x,y) wierzchołków zadanych przeciwnie do ruchu wskazówek zegara. Strukturą do przechowywania przekątnych potrzebnych do triangulacji jest tablica krotek indeksów wierzchołków, które należy połączyć, aby podzielić badany wielokąt na trójkąty. Struktura ta jest efektywna i redukuje ryzyko błędów arytmetyki komputerowej w trakcie operacji na niej z uwagi trzymania samych indeksów punktów. Wyznaczniki użyte w algorytmach zostały wyliczone na bazie macierzy 3x3 z toleracją zera $\varepsilon = 10^{-24}$. Dodatkowa funkcja `create_full_triangulation_form()` zwraca tablice wszystkich krawędzi wiloboku po triangulacji.

3.2.2. Algorytm sprawdzania y-monotoniczności wielokąta

Algorytm ten polega na weryfikacji, czy dany wielokąt spełnia definicję y-monotoniczności. W pierwszej kolejności odnajduje najwyższy i najniższy wierzchołek, a następnie przegląda kolejne punkty ułożone zgodnie z ruchem wskazówek zegara, począwszy od najwyższego. W każdym kroku sprawdza, czy jeden z sąsiednich wierzchołków znajduje się powyżej (poniżej) bieżącego wierzchołka, a drugi poniżej (powyżej) (z wyjątkiem najwyższego i najniższego punktu).

3.2.3. Algorytm kategoryzacji wierzchołków wielokąta

Algorytm realizuje metodę klasyfikacji wierzchołków opisaną w rozdziale 2, przetwarzając każdy punkt wielokąta i określając jego typ na podstawie położenia sąsiednich punktów oraz kąta wewnętrznego. Na początku algorytm analizuje relacje między współrzędnymi y sąsiadujących punktów — poprzedniego i następnego względem danego wierzchołka — aby ustalić, czy znajduje się on powyżej, poniżej lub pomiędzy sąsiadami. To pozwala na wstępną klasyfikację punktu. Następnie, aby określić orientację kąta wewnętrznego, algorytm oblicza wyznacznik macierzy utworzonej przez współrzędne trzech kolejnych punktów (poprzedniego, aktualnego i następnego). Dodatnia wartość wyznacznika oznacza lewostronny skręt, co wskazuje na ostry kąt wewnętrzny. Wynik obliczeń służy do ostatecznej klasyfikacji wierzchołka jako początkowego, końcowego, łączącego, dzielącego lub prawidłowego.

3.2.4. Algorytm znajdowania triangulacji wielokąta

W początkowej fazie algorytmu wyznaczane są lewy i prawy łańcuch wielokąta. W tej fazie do każdego punktu jest dopisywany jego indeks w tablicy początkowej. Następnie algorytm tworzy listę wszystkich punktów, posortowanych malejąco według współrzędnej y łącząc oba łańcuchy element po elemencie, przy czym do każdego punktu dołączona jest informacja o przynależności do konkretnego łańcucha. Na początku na stos są dodawane dwa pierwsze punkty z posortowanej listy, po czym algorytm przechodzi do głównej pętli. W niej, iterując po kolejnych punktach listy, odczytuje, z którego łańcucha pochodzi aktualnie rozpatrywany punkt.

- Dla wierzchołka pochodzącego z innego łańcucha niż wierzchołek na szczycie stosu, algorytm dodaje przekątne łączące aktualny punkt z każdym kolejnym elementem stosu. Przy każdej potencjalnej przekątnej sprawdza, czy nie jest ona przypadkiem bokiem wielokąta — weryfikacja odbywa się przez sprawdzenie, czy analizowane punkty sąsiadują ze sobą w tablicy punktów podanej na wejściu. Na zakończenie tej operacji na stosie pozostają jedynie dwa wierzchołki: aktualnie rozpatrywany punkt oraz wierzchołek ze szczytu stosu.
- Dla wierzchołka należącego do tego samego łańcucha co wierzchołek na szczycie stosu, algorytm analizuje trójkąty powstałe z bieżącego punktu, wierzchołka na szczycie stosu oraz kolejnych wierzchołków na stosie. Z uwagi na y-monotoniczność, bieżący wierzchołek sąsiaduje po tej samej stronie z wierzchołkiem na szczycie stosu. W takim przypadku algorytm poszukuje jedynie trzeciego wierzchołka, który połączy oba punkty w obrębie wielokąta. Jeśli utworzony trójkąt leży całkowicie wewnątrz wielokąta, a jego przekątna nie jest linią prostą (czyli trójkąt nie jest zdegenerowany), algorytm przechodzi do operacji usuwania punktów ze stosu.

  Algorytm usuwa wierzchołki ze stosu od jego szczytu aż do punktu należącego do aktualnie rozpatrywanego trójkąta (nie obejmując tego punktu). Jest to możliwe, ponieważ każdy trójkąt utworzony z bieżącego wierzchołka i usuwanych punktów nie spełnia warunków poprawnej triangulacji. Po dokonaniu tych usunięć krawędź łącząca bieżący punkt z wierzchołkiem stosu zostaje uwzględniona w triangulacji, tworząc jeden z boków poszukiwanego kolejnego trójkąta. Algorytm następnie próbuje wyznaczyć trzeci wierzchołek, aby zakończyć budowę trójkąta. Dzięki temu podejściu można liniowo przejrzeć stos i usunąć zbędne punkty, eliminując konieczność wielokrotnego analizowania trójkątów, które nie spełniają kryteriów poprawnej triangulacji.

  Weryfikacja zawierania się danego trójkąta w wielokącie polega na sprawdzeniu, czy trójkąt o zadanej kolejności wierzchołków jest prawoskrętny (dla aktualnie przetwarzanego punktu z lewego łańcucha) lub lewoskrętny (dla punktu z prawego łańcucha) . Sprawdzane jest zaimplementowane na bazie wyznacznika.

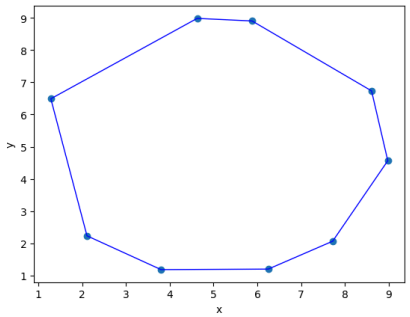
Algorytm zwraca tablicę krotek indeksów wierzchołków, które należy połączyć przekątne, aby podzielić badany wielokąt na trójkąty. Na jej bazie tworzona jest lista wszystkich krawędzi wieloboku.

4. Program ćwiczeń

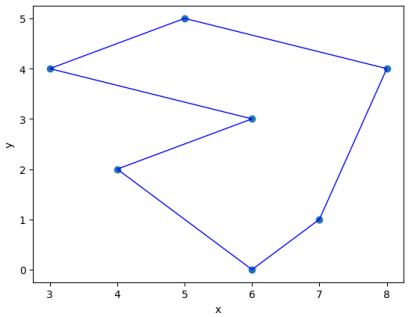
Na początku przystąpiłem do stworzenia narzędzia graficznego pozwalającego na tworzenie wielokątów za pomocą myszki. Następnie za pomocą tego narzędzia stworzyłem 7 wielokątów do przetestowania podanych wyżej algorytmów. Kolejność wierzchołków wprowadzałem przeciwnie do ruchu wskazówek zegara. Są to:

- **Figura A** - składa się z 9 punktów na okręgu i jest wielokątem wypukłym (Rysunek 3)
- **Figura B** - posiada 7 punktów, w tym jeden wierzchołek wklęsły (Rysunek 4)
- **Figura C** - posiada 20 punktów i nie jest y-monotoniczna (Rysunek 5)
- **Figura D** - składa się z 8 punktów tworzących „harmonijkę” po jednej stronie (Rysunek 6)

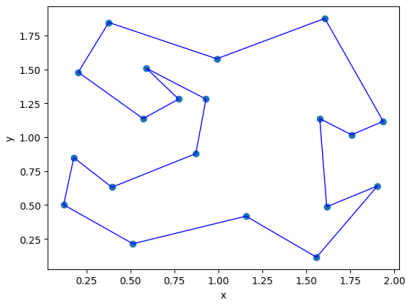
- **Figura E** - podobna do wielokąta D, ale ma dodatkowy, skrajnie górny punkt oraz większość punktów jest na przeciwnym łańcuchu niż w C. (Rysunek 7)
- **Figura F** - figura złożona z 6 punktów na łuku oraz dwóch na zewnątrz (Rysunek 8)
- **Figura G** - dwie asymetryczne „harmonijki" o łącznej liczności punktów równej 22 (Rysunek 9)



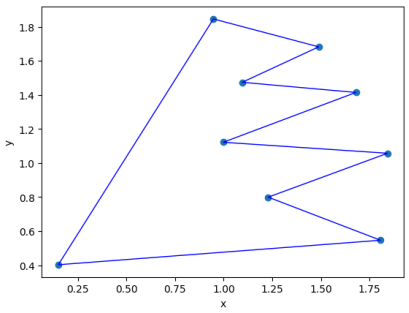
Rysunek 3: Figura A



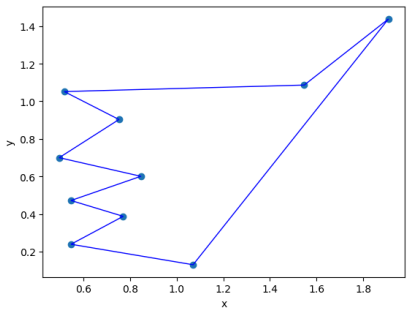
Rysunek 4: Figura B



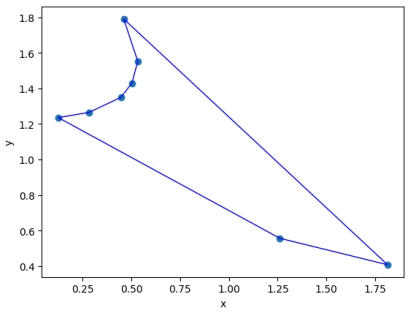
Rysunek 5: Figura C



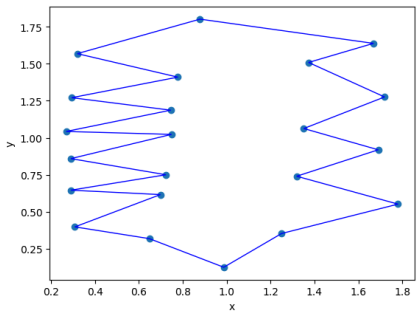
Rysunek 6: Figura D



Rysunek 7: Figura E



Rysunek 8: Figura F



Rysunek 9: Figura G

Figury A i B testują podstawowe przypadki wielokąta wypukłego. Figura C sprawdza algorytm y-monotoniczności. Dla figur D i E większość punktów przynależy do jednego łańcucha, więc testują algorytm w sytuacji dynamicznych modyfikacji stosu.
Punkty na łuku w wielokącie F nie mogą być ze sobą połączone w triangulacji, więc powinny odkładać się na stosie. Dla wielokąta G algorytm będzie musiał poradzić sobie z częstą obsługą punktów z innego łańcucha niż jest aktualnie na stosie.

Następnie uruchomiłem algorytmy omówione w rozdziale 3 i zebrałem wyniki.

5. Analiza wyników

5.1. Algorytm sprawdzania y-monotoniczności wielokąta

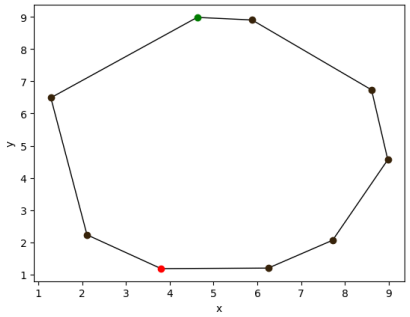
Testowany wielokąt	Figura A	Figura B	Figura C	Figura D	Figura E	Figura F	Figura G
Czy y-monotoniczny	True	True	False	True	True	True	True

Tabela 1: Wyniki działania algorytmu dla wielokątów testowych

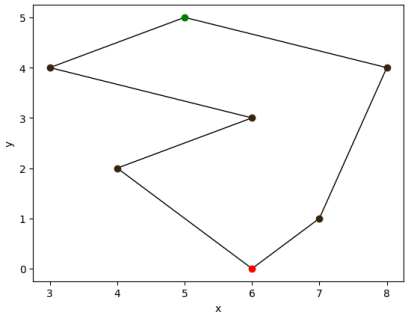
W tabeli Tabela 1 przedstawiono wyniki działania algorytmu sprawdzającego y-monotoniczność wielokąta dla figur testowych. Wyniki są zgodne z oczekiwaniami – algorytm poprawnie zidentyfikował figurę C jako niespełniającą warunków y-monotoniczności. Pozostałe wielokąty spełniają tę definicję, co zostało potwierdzone przez algorytm.

5.2. Algorytm kategoryzacji wierzchołków wielokąta

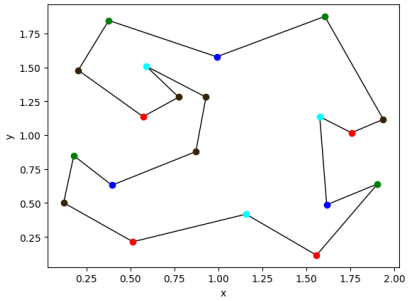
Na poniższych rysunkach zamieszczam wizualizacje zwróconych przez algorytm kategoryzacji wierzchołków wieloboku dla figur testowych. Kolory wierzchołków są zgodne z definicjami zamieszczonymi w rozdziale 1.



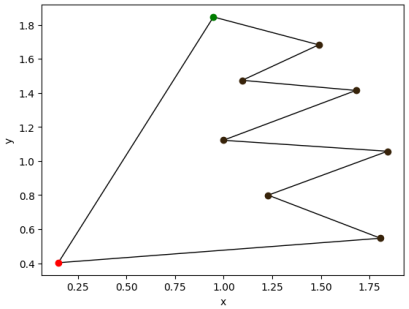
Rysunek 10: Pokolorowanie wierzchołków figury A



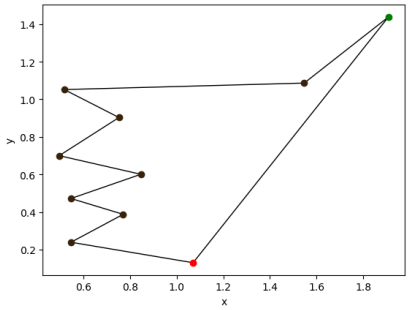
Rysunek 11: Pokolorowanie wierzchołków figury B



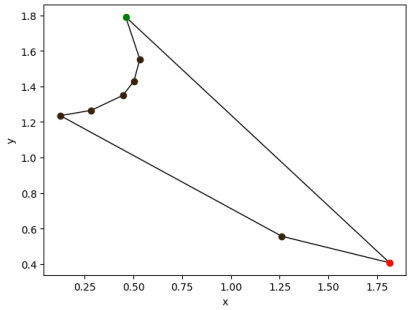
Rysunek 12: Pokolorowanie wierzchołków figury C



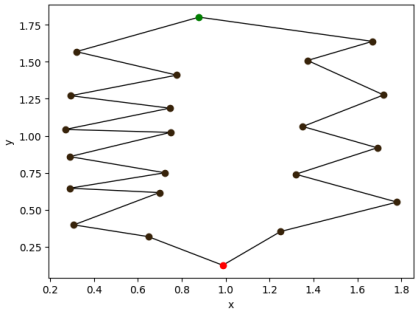
Rysunek 13: Pokolorowanie wierzchołków figury D



Rysunek 14: Pokolorowanie wierzchołków figury E



Rysunek 15: Pokolorowanie wierzchołków figury F



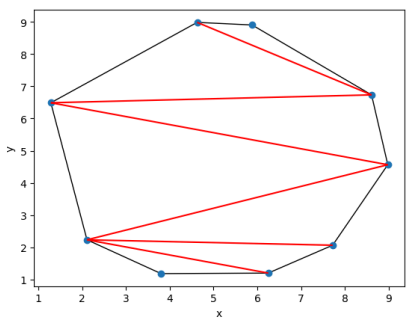
Rysunek 16: Pokolorowanie wierzchołków figury G

Jak pokazano na Rysunek 10, Rysunek 11, Rysunek 13, Rysunek 14, Rysunek 15 oraz Rysunek 16, figury y-monotoniczne (czyli wszystkie testowane wielokąty poza figurą C) posiadają po jednym wierzchołku początkowym i końcowym, natomiast pozostałe wierzchołki są klasyfikowane jako poprawne. Wyniki te są zgodne z oczekiwaniami, ponieważ teoria przewiduje, że figury y-monotoniczne nie mogą mieć ani wierzchołków dzielących, ani dodatkowych wierzchołków łączących – co jest widoczne na powyższych diagramach. Algorytm skutecznie podzielił wierzchołki na kategorie dla każej figury.

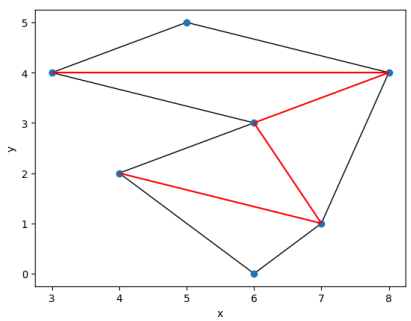
Rysunek 12 również potwierdza teorię, ponieważ, jak omówiono w podrozdziale 5.1, figura C nie spełnia założeń y-monotoniczności, więc zawiera wierzchołki łączące albo dzielące, co zostało zilustrowane na wizualizacji klasyfikacji wierzchołków tego wielokąta.

5.3. Algorytm triangulacji wielokąta

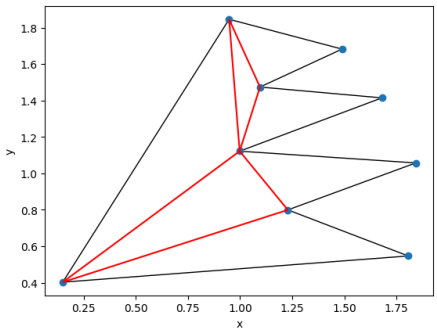
Na poniższych rysunkach zamieszczam wizualizacje zwróconych przez algorytm triangulacji wieloboku dla figur testowych:



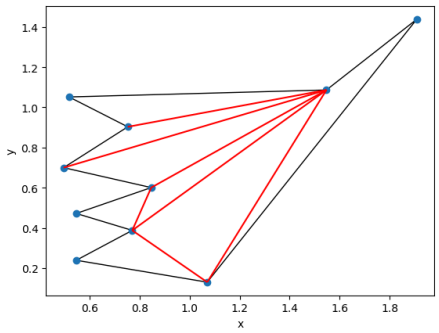
Rysunek 17: Triangulacja figury A



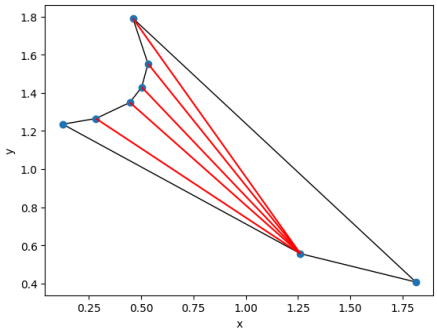
Rysunek 18: Triangulacja figury B



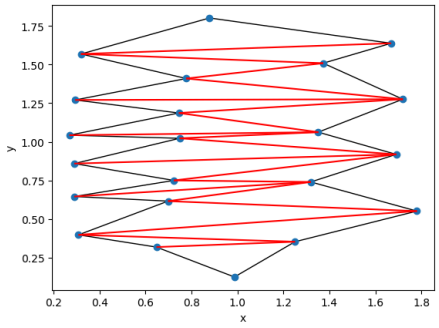
Rysunek 19: Triangulacja figury D



Rysunek 20: Triangulacja figury E



Rysunek 21: Triangulacja figury F



Rysunek 22: Triangulacja figury G

Zwizualizowane triangulacje, które zwrócił algorytm są poprawne. Triangulacja wielokąta C nie została przeprowadzona ponieważ nie jest on y-monotoniczny.

Figury A (Rysunek 17) nie były problemem dla algorytmu z uwagi na prostotę tych wieloboków. Wielobok B (Rysunek 18) zweryfikował, czy bok nie zostanie dodany do listy przekątnych.

W przypadku figur D i E (Rysunek 19 i Rysunek 20) algorytm skutecznie zarządzał operacjami na stosie zarówno dla wierzchołków z prawego, jak i lewego łańcucha, poprawnie reagując podczas analizy trójkątów wykraczających poza obszar wielokąta lub współliniowych z jego bokami. Ponadto, dla figury D ostatni rozpatrywany wierzchołek znajdował się po przeciwnej stronie łańcucha niż wierzchołki dotychczas umieszczone na stosie, co umożliwia również analizę operacji wykonywanych dla punktów leżących po różnych stronach łańcucha.

Proces triangulacji figury F (Rysunek 21) ładnie obrazuje budowanie się stosu tego algorytmu. Dopiero w ostatnim kroku zostały dodane przekątne wieloboku do triangulacji. Ten przykład zweryfikował również poprawne odrzucanie trójkątów zdegenerowanych do odcinka jednego z boków wielokąta oraz trójkątów niezawierających się w wielokącie.

Podczas triangulacji figury G (Rysunek 22) algorytm skutecznie radził sobie z częstym opróżnianiem stosu w przypadku wierzchołków należących do różnych łańcuchów.

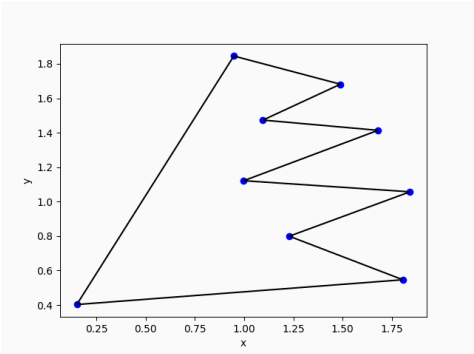
Dodatkowo, przypadek ten ujawnia problem jakości triangulacji generowanej przez algorytm. Zamiast preferować przekątne przebiegające przez bardziej centralne punkty figury, algorytm wybiera przekątne tworzące trójkąty rozwartokątne o dużych kątach rozwarcia, co wynika z mechanizmu ciągłego łączenia rozpatrywanego wierzchołka z punktami na stosie. Takie wybory prowadzą do mniej regularnej struktury triangulacji, co obniża jakość uzyskanej siatki i wskazuje na potrzebę zastosowania algorytmu ulepszającego. Algorytm poprawiający mógłby preferować bardziej wewnętrzne przekątne, umożliwiając uzyskanie bardziej równomiernej i estetycznej siatki triangulacyjnej.

5.4. Wizualizacja krok po kroku działania algorytmu znajdowania triangulacji wielokąta

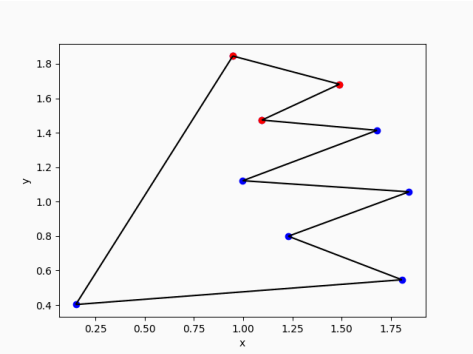
Wszystkie gify przedstawiające wizualizacje krok po kroku działania algorytmu znajdowania triangulacji wielokąta znajdują się w pliku *Jupyter*. Poniżej przedstawię działanie algorytmu dla figury D.

Legenda obrazów:

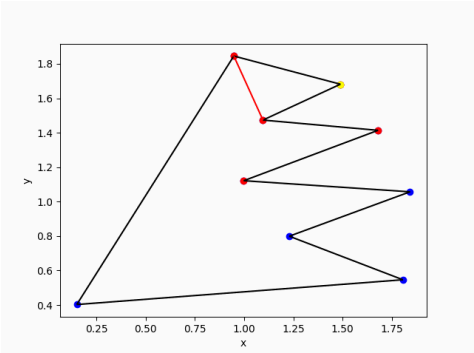
- *kolor czarny* - boki wielokąta
- *kolor niebieski* - punkty należące do figury
- *kolor czerwony* - punkty aktualnie na stosie oraz przekątna przynależna do aktualnej triangulacji
- *kolor żółty* - punkty, które zostały usunięte ze stosu



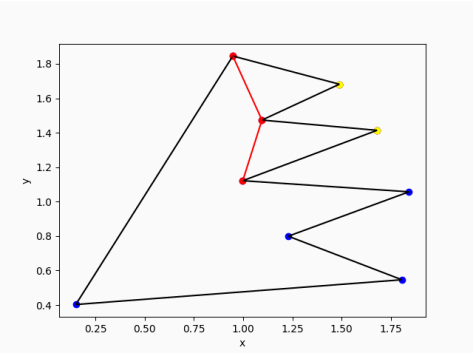
Rysunek 23: Triangulacja figury D
Etap 1



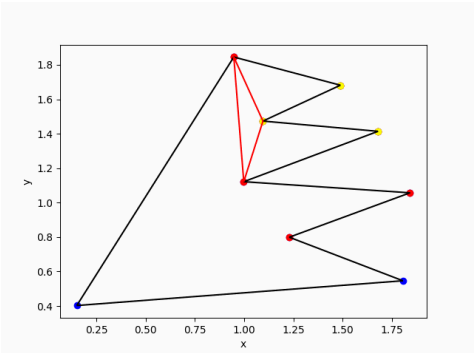
Rysunek 24: Triangulacja figury D
Etap 2



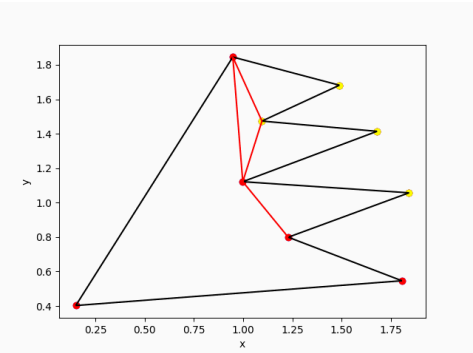
Rysunek 25: Triangulacja figury D
Etap 3



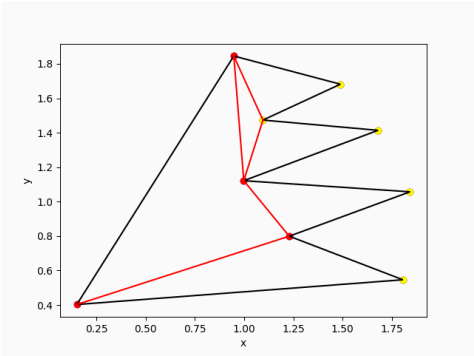
Rysunek 26: Triangulacja figury D
Etap 4



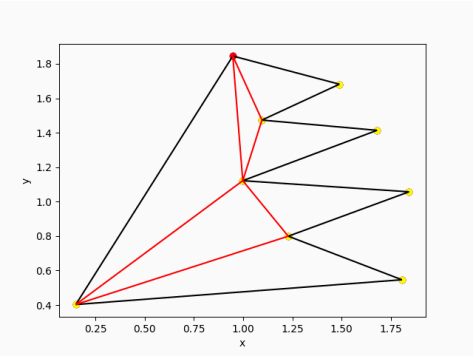
Rysunek 27: Triangulacja figury D
Etap 5



Rysunek 28: Triangulacja figury D
Etap 6



Rysunek 29: Triangulacja figury D
Etap 7



Rysunek 30: Triangulacja figury D
Etap 8

- **Etap 1** (Rysunek 23) - obraz przedstawia wielokąt D
- **Etap 2** (Rysunek 24) - do stosu dodano 3 pierwsze punkty
- **Etap 3** (Rysunek 25) - rozpatrzono trójkąt stworzony z pierwszych trzech punktów stosu i dodano przekątną do triangulacji usuwając punkt ze stosu. Następne punkty są dodawane do stosu z braku możliwości stworzenia po drodze odpowiedniego trójkąta
- **Etap 4** (Rysunek 26) - został rozpatrzone pierwszy trójkąt utworzony z punktów na stosie i dodano odpowiednią przekątną do triangulacji usuwając wierzchołek ze stosu.
- **Etap 5** (Rysunek 27) - kontynuowane było sprawdzanie trójkątów utworzonych z punktów na stosie i dodano odpowiednią przekątną do triangulacji usuwając wierzchołek ze stosu. Następnie rozpatrzono kolejne dwa wierzchołki, które dodano do stosu.
- **Etap 6** (Rysunek 28) - Ponownie został rozpatrzone pierwszy trójkąt utworzony z punktów na stosie i dodano odpowiednią przekątną do triangulacji usuwając wierzchołek ze stosu. Następnie rozpatrzono kolejne dwa wierzchołki, które dodano do stosu. W ten sposób nie ma już punktów w wielokącie, które nie są lub nie były na stosie
- **Etap 7** (Rysunek 29) - Wierzchołek ostatnio dodany do stosu należy do innego łańcucha, więc algorytm dodaje przekątne łączące ten punkt z każdym na stosie (o ile nie tworzą boku)
- **Etap 8** (Rysunek 30) - Otrzymano pełną triangulacje

6. Wnioski

- Implementacja algorytmu prawidłowo rozpoznaje y-monotoniczność dla różnych przypadków wielokątów, co potwierdza jego skuteczność i niezawodność w identyfikacji tego typu struktury. Testy na wielokątach o złożonej geometrii wykazały, że algorytm działa zgodnie z przewidywaniami, wykrywając przypadki niespełniające warunków monotoniczności.
- Algorytm kategoryzacji wierzchołków poprawnie klasyfikuje punkty wielokątów na podstawie ich relacji z sąsiednimi wierzchołkami oraz kąta wewnętrznego, co jest kluczowe dla późniejszych etapów triangulacji. Zgodnie z teorią, wielokąty y-monotoniczne nie zawierają wierzchołków dzielących ani łączących, co zostało potwierdzone przez wyniki wizualizacji.
- Algorytm triangulacji dobrze radzi sobie z podstawowymi i bardziej złożonymi wielokątami monotonicznymi, tworząc poprawne podziały na trójkąty. W przypadku wielokątów z wierzchołkami po obu stronach łańcucha, algorytm prawidłowo zarządza stosowaniem przekątnych, jednak pojawiają się pewne niedoskonałości w regularności generowanej siatki trójkątów.
- Dla bardziej asymetrycznych lub skomplikowanych kształtów, takich jak wielokąt G, algorytm często generuje trójkąty rozwartokątne o dużych kątach, co prowadzi do mniej równomiernej struktury siatki. Ujawnia to konieczność zastosowania algorytmów poprawiających, które mogłyby preferować przekątne przez bardziej centralne punkty figury, zwiększając estetykę i regularność siatki.