Jacek Łoboda gr 6, Pon. 16:45

Algorytmy geometryczne, ćwiczenie 3

1. Dane techniczne

- Procesor: Intel Core i7-7700HQ 2.80GHz
- RAM: 16GB 2133 MHz
- System operacyjny: Windows 10 Home x64
- Środowisko: Visual Studio Code
- Język: Python 3.13.5

Użyto następujących bibliotek: numpy, pandas, matplotlib. Do graficznej wizualizacji wykorzystano narzędzie koła naukowego BIT.

2. Cel ćwiczenia

Celem ćwiczenia było poznanie i implementacja algorytmów: sprawdzania y-monotoniczności wielokąta, klasyfikacji wierzchołków w wielokącie, triangulacji wielokąta y-monotonicznego oraz analiza uzyskanych danych.

3. Wstęp teoretyczny

3.1 Wielokąty monotoniczne

Wielokąt jest ściśle monotoniczny względem pewnej prostej l (kierunku), jeśli jego brzeg można podzielić na dwa spójne łańcuchy (“górny” i “dolny”). Warunek monotoniczności polega na tym, że każda prosta l' prostopadła do kierunku l może przeciąć wielokąt w co najwyżej dwóch odcinkach (lub w jednym, lub wcale).

Wielokąt y-monotoniczny to taki, który jest monotoniczny względem osi Y (kierunek pionowy). Pozioma linia przecina wielokąt w co najwyżej dwóch punktach. Patrząc na wierzchołki: jeśli przechodzimy od najwyższego wierzchołka ($p_{\{\max\}}$) do najniższego ($p_{\{\min\}}$) wzdłuż krawędzi (łańcucha), współrzędna y musi zawsze maleć lub pozostawać stała (nigdy nie rośnie).

3.2 Klasyfikacja wierzchołków wielokąta

Wierzchołki w wielokącie można podzielić na 5 kategorii:

- początkowe: obaj sąsiedzi leżą poniżej i kąt wewnętrzny ma mniej niż 180°.
- końcowe: obaj sąsiedzi leżą powyżej i kąt wewnętrzny ma mniej niż 180°.
- dzielące: obaj sąsiedzi leżą poniżej i kąt wewnętrzny ma więcej niż 180°.
- łączące: obaj sąsiedzi leżą powyżej i kąt wewnętrzny ma więcej niż 180°.
- prawidłowe: jeden sąsiad leży poniżej, a drugi powyżej.

Wielokąt y-monotoniczny będzie miał jeden wierzchołek początkowy, jeden końcowy i resztę prawidłowych.

3.3 Triangulacja wielokąta

Triangulacja wielokąta w płaszczyźnie 2D to jego podział na rozłączne trójkąty, których wierzchołkami są wierzchołki oryginalnego wielokąta. Proces ten polega na dodaniu nieprzecinających się przekątnych, które łączą wierzchołki.

Właściwości Triangulacji:
- Pokrycie: suma wszystkich trójkątów pokrywa całe wnętrze wielokąta.
- Rozłączność: wnętrza różnych trójkątów muszą być rozłączne.
- Wspólne Krawędzie: przecięcie dwóch różnych trójkątów jest albo puste, albo zredukowane do wspólnego wierzchołka, albo do wspólnej krawędzi (przekątnej).
- Liczba Elementów: dla wielokąta o n wierzchołkach, triangulacja zawsze składa się z $n-2$ trójkątów połączonych za pomocą $n-3$ przekątnych.

4. Implementacja

4.1 Algorytm sprawdzający y-monotoniczność wielokąta

Algorytm znajduje w wielokącie wierzchołki o największej i najmniejszej współrzędnej y. Następnie, korzystając z formatu danych (punkty w wielokącie ustawione są w kolejności CCW), przechodzi po nowo powstałym lewym i prawym łańcuchu, sprawdzając monotoniczność sąsiednich punktów. Jeśli oba łańcuchy są y-monotoniczne, to cały wielokąt jest y-monotoniczny. Złożoność czasowa algorytmu wynosi $O(n)$, gdzie n to liczba wierzchołków wielokąta.

4.2 Algorytm klasyfikujący wierzchołki wielokąta

Algorytm przypisuje wierzchołki do 5 kategorii zgodnie z opisem klasyfikacji (sekcja 3.2). Kąt wewnętrzny jest oceniany za pomocą wyznacznika macierzy 3x3. Złożoność czasowa algorytmu wynosi $O(n)$, gdzie n to liczba wierzchołków wielokąta.

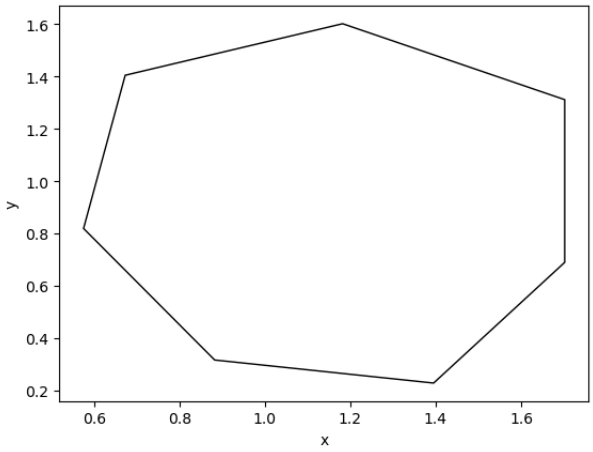
4.3 Algorytm triangulacji wielokątów monotonicznych

Algorytm sortuje wierzchołki w wielokącie względem współrzędnej y malejąco. Wykorzystuje do tego format wejściowy wielokąta, czyli tablicę krotek (float, float) symbolizujących kolejne wierzchołki na płaszczyźnie; dodatkowo punkty te ustawione są w kolejności CCW, co wykorzystywane jest do liniowego ich posortowania. Najpierw dzieli go na dwa od razu posortowane łańcuchy (na potrzeby implementacji wierzchołek początkowy i końcowy umieszcza tylko w prawym łańcuchu), a następnie scala je w jeden. Posortowany wielokąt jest przechowywany jako lista punktów, co pozwala na łatwe przejście po wszystkich punktach z góry do dołu. Następnie algorytm inicjalizuje stos, który będzie używany do budowania nowych przekątnych i wkłada na niego dwa pierwsze wierzchołki posortowanego wielokąta. Z kolejnymi wierzchołkami postępuje w następujący sposób:
- Jeśli należy do innego łańcucha niż góra stosu, to łączy każdy wierzchołek ze stosu przekątną do aktualnie przetwarzanego wierzchołka (o ile nie są sąsiadami w oryginalnym wielokącie) i czyści przy tym stos. Na koniec do pustego stosu dodaje z powrotem górę stosu oraz obecnie przetwarzany punkt.
- Jeśli należy do tego samego łańcucha co góra stosu, to analizuje trójkąty tworzone z 2 ostatnimi wierzchołkami na stosie. Jeśli trójkąt należy do wielokąta (określa to za pomocą wyznacznika 3x3), to dodaje odpowiednią przekątną i usuwa górę ze stosu. Robi to dopóki na stosie znajdują się co najmniej 2 punkty i powstałe trójkąty należą do wielokąta. Na koniec dodaje przetwarzany punkt na stos.

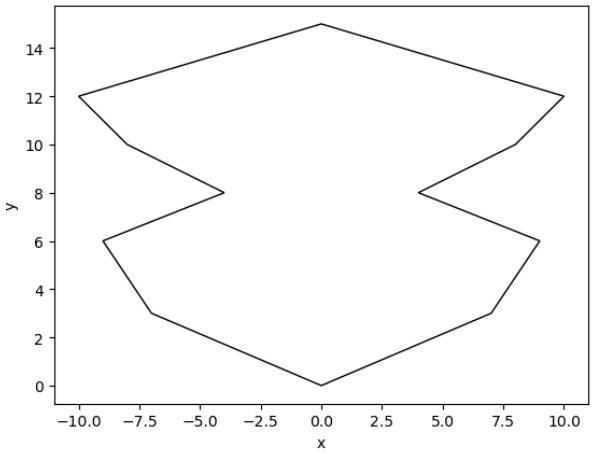
Algorytm zwraca triangulację w dwóch formatach: jako listę krotek (int, int) symbolizujących przekątne, które trzeba dodać między punktami o danych indeksach (format ten używany jest podczas wizualizacji) oraz jako listę krotek (point, point, point) symbolizujących trójkąty powstałe w wyniku triangulacji (alternatywny format ułatwiający przechowywanie całej figury). Złożoność czasowa algorytmu wynosi $O(n)$ dla n wierzchołków w wielokącie.

5. Zbiory danych

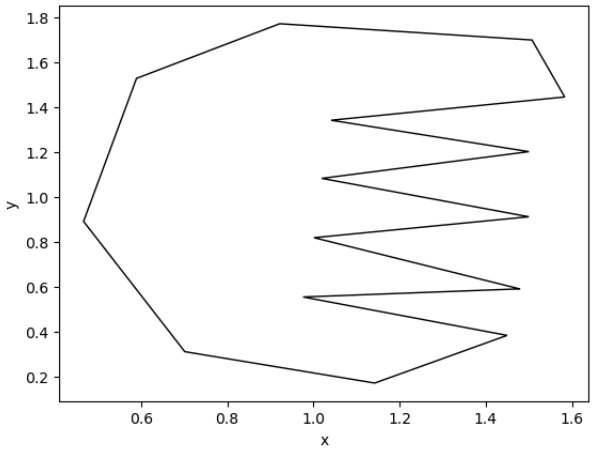
Do przetestowania poprawności implementacji stworzono poniższe zbiory testowe. Użyto aplikacji graficznej korzystającej z biblioteki matplotlib (Wierzchołki są ustawione w kolejności CCW).



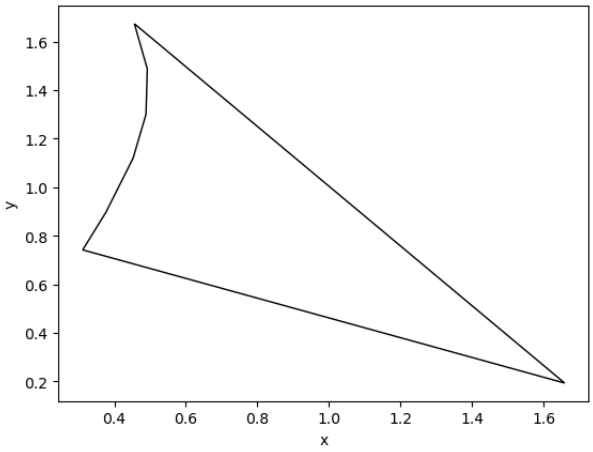
Rys. 1: Zbiór A, wielokąt wypukły zawierający 7 wierzchołków.



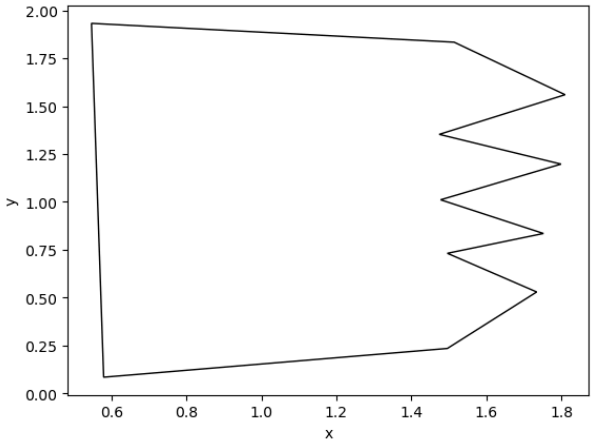
Rys. 2: Zbiór B, wielokąt posiada 2 kąty wklęsłe i 12 wierzchołków.



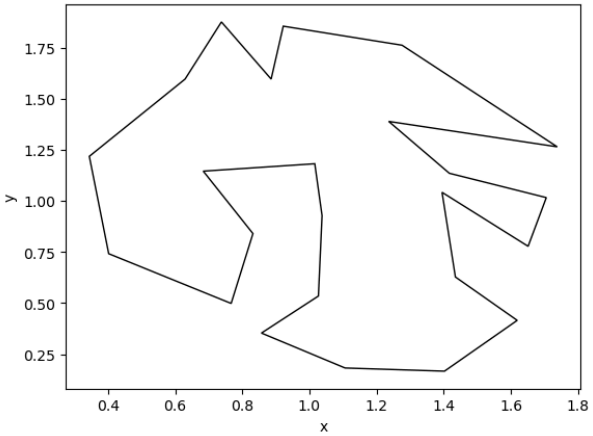
Rys. 3: Zbiór C, wielokąt posiada 4 kąty wklęsłe i 16 wierzchołków.



Rys. 4: Zbiór D, wielokąt posiada wiele kątów wklęsłych i 7 wierzchołków.



Rys. 5: Zbiór E, wielokąt posiadający 3 kąty wklęsłe i 11 wierzchołków.



Rys. 6: Zbiór F, wielokąt niemonotoniczny posiadający 24 wierzchołki.

Powyższe zbiory zostały tak dobrane, aby sprawdzały różne aspekty działania zaimplementowanych algorytmów. A i B to podstawowe przypadki sprawdzające działanie na prostych zestawach danych. C, D i E to bardziej skomplikowane wielokąty, które sprawdzą działanie algorytmu triangulacji na

mniej sprzyjających danych. F to wielokąt niemonotoniczny, więc triangulacja nie zadziała, ale jest dobrym zbiorem do testowania klasyfikacji wierzchołków.

6. Analiza wyników

6.1 Algorytm sprawdzający y-monotoniczność wielokąta

Zbiór	A	B	C	D	E	F
wynik algorytmu	True	True	True	True	True	False

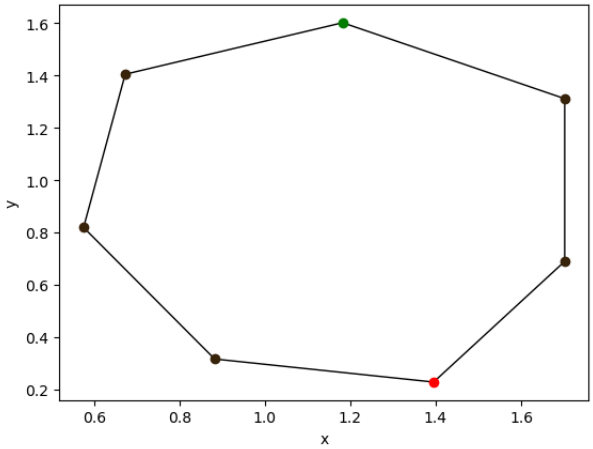
Tabela 1: Wyniki działania algorytmu sprawdzania y-monotoniczności dla danych zbiorów.

Jak widać w Tabeli 1, tylko zbiór F został rozpoznany jako nie y-monotoniczny, co jest zgodne z oczekiwaniami.

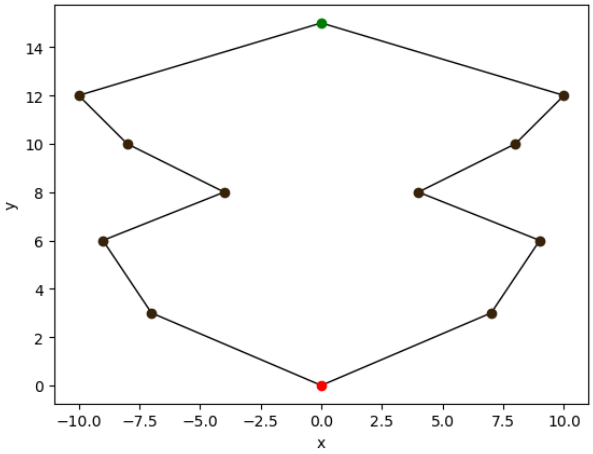
6.2 Algorytm klasyfikacji wierzchołków wielokąta

Rysunki przedstawiające wyniki działania algorytmu na zbiorach grupują punkty kolorami:

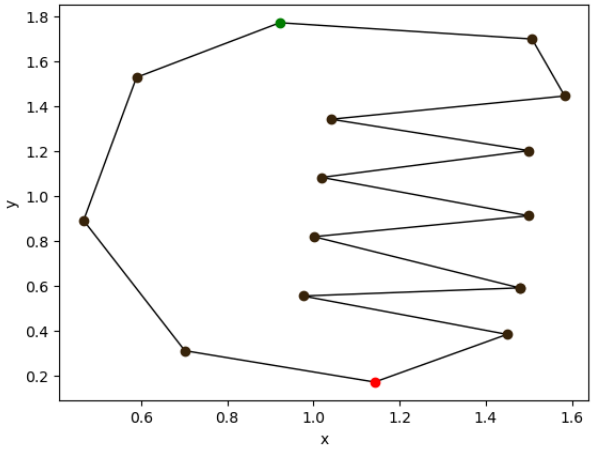
- zielony: punkt początkowy.
- czerwony: punkt końcowy.
- granatowy: punkt łączący.
- niebieski: punkt dzielący.
- czarny: punkt prawidłowy.



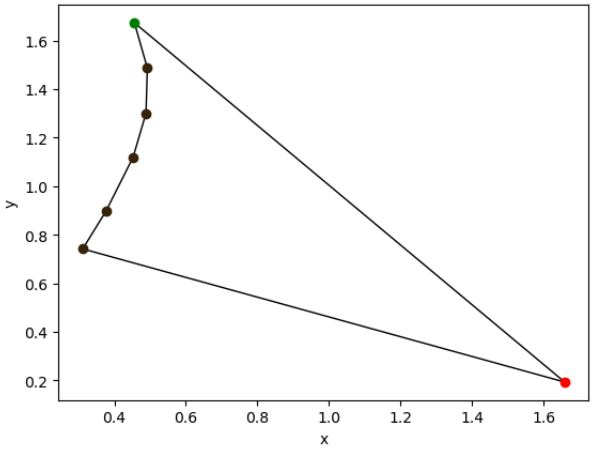
Rys. 7: Wizualizacja klasyfikacji na zbiorze A



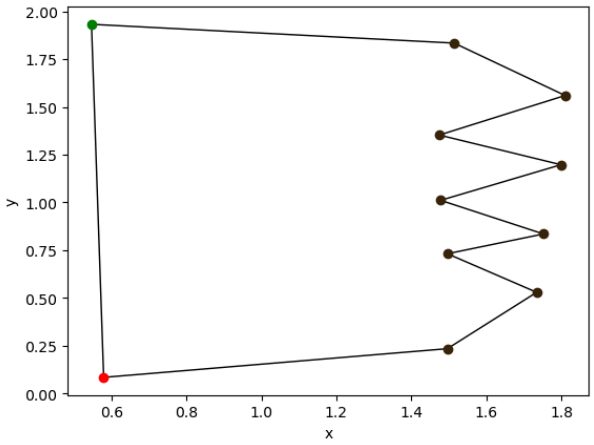
Rys. 8: Wizualizacja klasyfikacji na zbiorze B



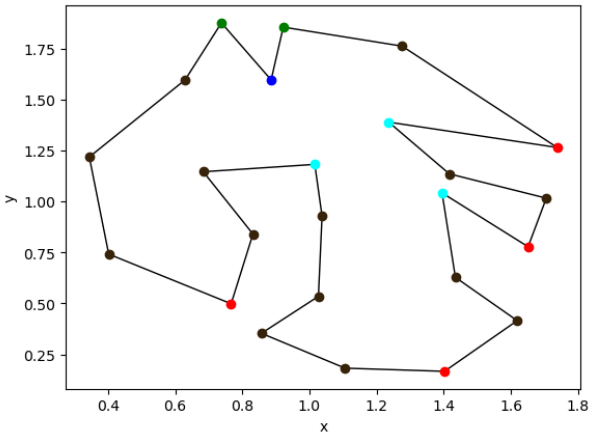
Rys. 9: Wizualizacja klasyfikacji na zbiorze C



Rys. 10: Wizualizacja klasyfikacji na zbiorze D



Rys. 11: Wizualizacja klasyfikacji na zbiorze E

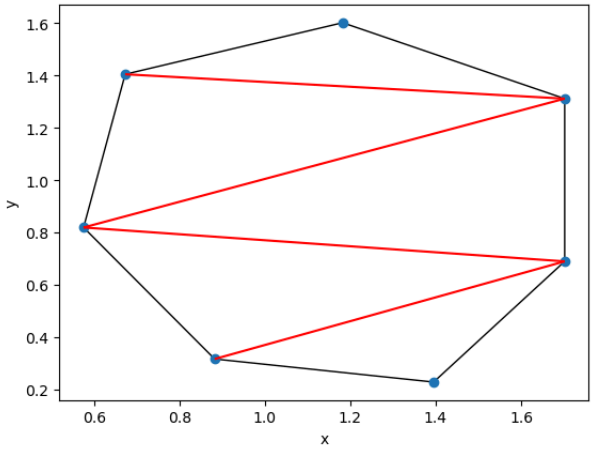


Rys. 12: Wizualizacja klasyfikacji na zbiorze F

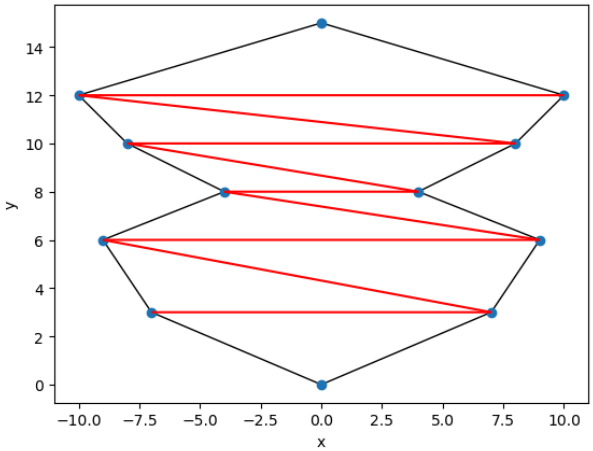
Jak widać na Rys 7 - 11, algorytm podzielił wierzchołki wielokątów monotonicznych na jeden początkowy, jeden końcowy oraz resztę prawidłowych (odpowiada to twierdzeniu zawartym w sekcji 3.2). Ciekawszy wynik algorytm przedstawił na zbiorze F; jak widać na Rys. 12, zakwalifikował on: 2 punkty do początkowych, 4 do kończących, 1 do łączących, 3 do dzielących i 14 do prawidłowych. Z wizualizacji można wywnioskować, że algorytm prawidłowo zakwalifikował punkty we wszystkich zbiorach testowych.

6.3 Algorytm triangulacji wielokąta y-monotonicznego

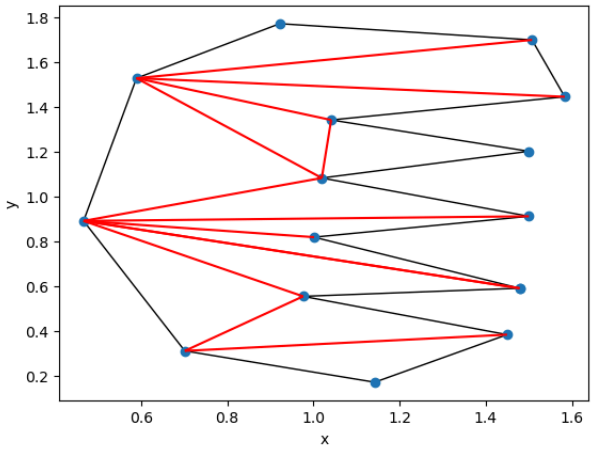
Rysunki przedstawiają wyniki działania algorytmu na zbiorach testowych. Kolorem czarnym oznaczone są krawędzie wielokąta, niebieskim wierzchołki wielokąta, a czerwonym przekątne dodane podczas triangulacji.



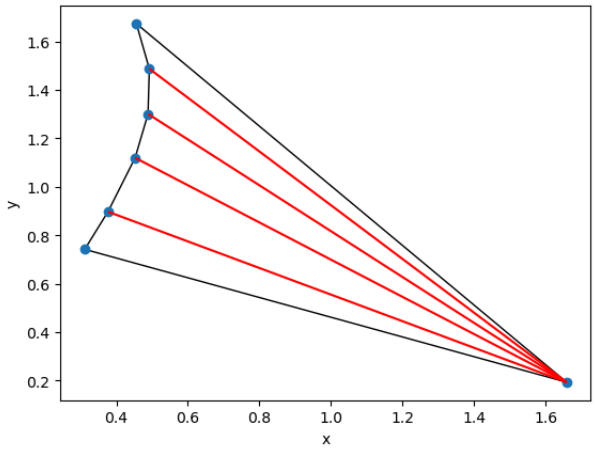
Rys. 13: Wizualizacja triangulacji na zbiorze A



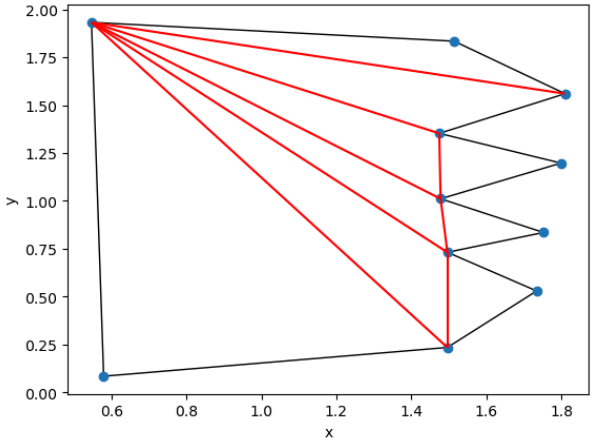
Rys. 14: Wizualizacja triangulacji na zbiorze B



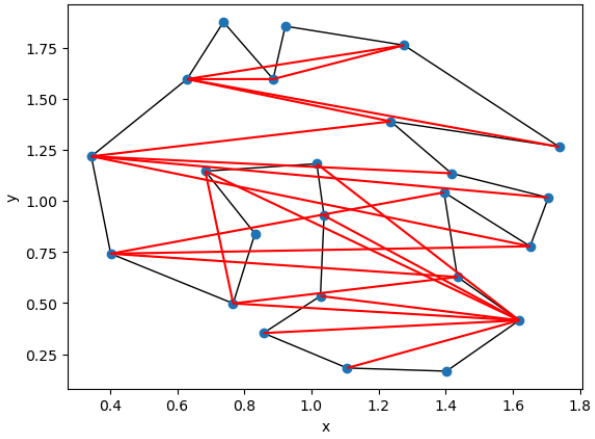
Rys. 15: Wizualizacja triangulacji na zbiorze C



Rys. 16: Wizualizacja triangulacji na zbiorze D



Rys. 17: Wizualizacja triangulacji na zbiorze E



Rys. 18: Wizualizacja triangulacji na zbiorze F

Zbiór	A	B	C	D	E	F
Liczba wierzchołków	7	12	16	7	11	24
Liczba trójkątów	5	10	14	5	9	22
Liczba przekątnych	4	9	13	4	8	21

Tabela 2: Porównanie liczby wierzchołków wielokąta oraz liczby powstałych trójkątów i przekątnych dodanych przez algorytm triangulacji.

Z powyższych rysunków można wysunąć następujące wnioski:

- Zbiór A (Rys. 13): Algorytm działa prawidłowo na prostych wielokątach wypukłych. Aktualnie przetwarzany punkt zawsze znajduje się na innym łańcuchu niż góra stosu, co za tym idzie, podczas triangulacji użyta zostanie tylko jedna z dwóch logik przetwarzania punktów.
- Zbiór B (Rys. 14): Pomimo dwóch kątów wklęsłych punkty dobrane są w taki sposób, że tak jak w Zbiorze A, użyta zostaje tylko logika łączenia przetwarzanego punktu z całym stosem, co w tym przypadku sprowadza się do łączenia przekątną sąsiadów ze zbioru posortowanego.
- Zbiór C (Rys. 15): Algorytm dobrze poradził sobie z wielokątem, którego jeden łańcuch składa się z kątów wklęsłych, a drugi z wypukłych. Jak widać, użyte zostały obie logiki dodawania przekątnych do wielokąta.
- Zbiór D (Rys. 16): Algorytm prawidłowo pogrupował wielokąt, w którym wszystkie punkty (oprócz początkowego i końcowego) znajdują się na lewym łańcuchu. Podczas przetwarzania

kolejnych punktów algorytm po kolei odrzucał trójkąty zewnętrzne, a na koniec połączył cały stos z wierzchołkiem końcowym.

- Zbiór E (Rys. 17): Algorytm zadziałał na zbiorze, w którym lewy łańcuch jest pusty, a prawy składa się z punktów tworzących poszarpany kształt. Algorytm użył tylko logiki drugiej, ponieważ wszystkie punkty wylądowały na jednym łańcuchu.
- Zbiór F (Rys. 18): Algorytm nie poradził sobie z przetworzeniem wielokąta niemonotonicznego, co jest zgodne z oczekiwaniami. Przekątne przecinają się, wiele z powstałych trójkątów nawet nie należy do wielokąta. Poprawne działanie algorytmu na tym zbiorze wymagałoby wcześniejszego podzielenia wielokąta na monotoniczne składowe.

Dodatkowo dane w Tabeli 2 zgadzają się z własnością triangulacji opisaną w sekcji 3.3, mówiącą o tym, że dla wielokąta o n wierzchołkach liczba powstałych trójkątów wynosi $n - 2$, a liczba dodanych przekątnych $n - 3$.

7. Wnioski

7.1 Weryfikacja monotoniczności

Algorytm sprawdzający y-monotoniczność okazał się skutecznym narzędziem filtracji danych wejściowych. Poprawnie zidentyfikował on zbiory A-E jako monotoniczne, a zbiór F jako niemonotoniczny (Tabela 1), co zapewnia podstawę do dalszego działania na zbiorach.

7.2 Klasyfikacja wierzchołków

Algorytm klasyfikacji poprawnie przypisał wierzchołki wielokątów do kategorii. Zgodnie z oczekiwaniami wielokąty monotoniczne składały się z punktów prawidłowych oraz po jednym początkowym i końcowym (Rys. 7 - 11). Pozostałe typy wierzchołków zostały zidentyfikowane na niemonotonicznym Zbiorze F (Rys. 12).

7.3 Triangulacja wielokątów y-monotonicznych

Algorytm triangulacji poprawnie podzielił monotoniczne zbiory danych na trójkąty niezależnie od ich kształtów (Rys. 13 - 17). Jedyną figurą, z którą sobie nie poradził, był zbiór F (Rys. 18), co jest zachowaniem oczekiwanym, biorąc pod uwagę, że algorytm nie jest przystosowany do działania na wielokątach niemonotonicznych. Warto dodać, że wyniki algorytmu zgadzały się z teoretycznymi założeniami triangulacji.