Algorytmy geometryczne
Labolatorium nr 3

Patryk Klatka
Grupa nr 4

24 listopada 2022

1 Opis ćwiczenia

Głównym celem ćwiczenia było zaimplementowanie algorytmów związanych z zagadnieniami monotoniczności oraz triangulacji wielokątów. Została zaimplementowana procedura sprawdzająca czy wielokąt jest y-monotoniczny, algorytm triangulacji wielokątów y-monotonicznych oraz klasyfikacji punktów początkowych, końcowych, łączących, dzielących i prawidłowych. Algorytmy zostały zaimplementowane za pomocą pseudokodu z wykładu oraz literatury zawartej w sylabusie przedmiotu.

1.1 Algorytm klasyfikacji punktów

Punkt poniżej drugiego punktu nazywamy punkt, który ma mniejszą współrzędną y. Kątem wewnętrznym punktu nazywamy kąt, który znajduje się wewnątrz wielokąta oraz tworzą go odcinki wychodzące z badanego punktu i kończące się w punktach sąsiednich (krawędzie sąsiednie badanego punktu). Punkty w dowolnym wielokącie możemy podzielić na pięć kategorii:

1. Początkowe - sąsiedzi wierzchołka leżą poniżej oraz kąt wewnętrzny jest mniejszy od π.
2. Końcowe - sąsiedzi wierzchołka leżą powyżej oraz kąt wewnętrzny jest mniejszy od π.
3. Łączące - sąsiedzi wierzchołka leżą powyżej oraz kąt wewnętrzny jest większy od π.
4. Dzielące - sąsiedzi wierzchołka leżą poniżej oraz kąt wewnętrzny jest większy od π.
5. Prawidłowe - w przeciwnych przypadkach, dokładnie: jeden sąsiad leży powyżej, drugi poniżej.

Algorytm sprowadza się do przejściu po figurze w zadanym kierunku i sprawdzenie kątów wewnętrznych oraz położenia sąsiadów badanego wierzchołka w czasie $O(n)$. Wyznacznik 3x3 jest wykorzystywany w przypadku sprawdzania, czy kąt jest większy lub mniejszy od π biorąc sąsiada poprzedniego (za badanym wierzchołkiem, zgodnie z zadanym kierunkiem figury) i sprawdzając czy sąsiad następny leży po lewej stronie odcinka (kąt wewnętrzny jest mniejszy od π), czy po prawej (kąt większy od π).

1.2 Algorytm sprawdzania y-monotoniczności

Figura jest y-monotoniczna, gdy idąc z najwyżej położonego wierzchołka w kierunku najniżej położonego wierzchołka po lewym lub prawym łańcuchu figury, nigdy nie poruszamy się do góry (tzn. wierzchołek poprzedni, zgodnie z zadanym kierunkiem figury, ma zawsze współrzędną y większą niż wierzchołek następny). Łatwo zauważyć, że taka figura będzie spełniała powyższe warunki, gdy nie będzie posiadała wierzchołków łączących i dzielących (głównie przez to, że kąt wewnętrzny jest większy od π sprawia, że będziemy się poruszać w stronę dodatnich wartości osi OY). Zatem algorytm sprowadza się do sprawdzenia, czy figura posiada takie wierzchołki, jeżeli tak to nie jest y-monotoniczna.

1.3 Algorytm triangulacji wielokąta y-monotonicznego

1. Określamy lewy i prawy łańcuch (punkty leżące na lewo/prawo względem współrzędnej x od punktu najwyżej oraz najniżej położonego względem osi y) wielokąta względem kierunku monotoniczności. Wierzchołek o największej współrzędnej y dodajemy do lewego łańcucha, a wierzchołek o najmniejszej współrzędnej y dodajemy do prawego łańcucha.
2. Porządkujemy wierzchołki wzdłuż kierunku monotoniczności - w naszym przypadku sortujemy punkty po współrzędnej y malejąco.
3. Wkładamy dwa pierwsze wierzchołki na stos (z posortowanej listy).
4. Dla każdego wierzchołka z posortowanej listy wierzchołków, oprócz ostatniego wierzchołka:
    4.1. Jeśli badany wierzchołek należy do innego łańcucha niż wierzchołek stanowiący szczyt stosu, to możemy go połączyć ze wszystkimi wierzchołkami na stosie, oprócz ostatniego wierzchołka. Czyścimy stos i dodajemy dwa wierzchołki, które były „zamiatane" ostatnie (czyli wierzchołek który aktualnie badany i wierzchołek poprzedni względem kierunku monotoniczności figury).
    4.2. W przeciwnym wypadku analizujemy kolejne trójkąty, jakie tworzy dany wierzchołek z wierzchołkami zdejmowanymi ze stosu. Dopóki na stosie są punkty oraz utworzony trójkąt należy do wielokąta to usuwamy wierzchołek ze szczytu stosu i tworzymy przekątną. W przeciwnym wypadku do stosu dodajemy ostatni usunięty wierzchołek i aktualnie badany.
5. Dla ostatniego wierzchołka, z listy posortowanej, tworzymy przekątne do każdego wierzchołka ze stosu oprócz pierwszego i ostatniego.

Do sprawdzania, czy przekątna zawiera się w wielokącie (dokładniej: czy trójkąt tworzony przez 3 badane wierzchołki zawiera się w figurze), wykorzystywany jest wyznacznik 3x3 własnej implementacji, zaimplementowany wykorzystując metodę Sarrusa z tolerancją dla zera `1e-10`. Metoda jest bardzo prosta. Tworzymy wektor $\vec{ab}$, gdzie b, to pierwszy element stosu, a - następny element za wierzchołkiem b. Gdy b leży w lewym łańcuchu figury musimy sprawdzić czy aktualnie badany punkt leży po lewej stronie wektora. W przeciwnym wypadku sprawdzamy czy leży on po prawej stronie.

1.4 Struktury przechowujące wielokąt

W celu usprawnienia implementacji została stworzona klasa `Polygon`, która przechowuje m.in. krawędzie wielokąta oraz jego punkty. Do przechowywania krawędzi oraz punktów są wykorzystywane słowniki oraz zbiory (`dict` oraz `set`), które mają amortyzowane złożoności odczytu/zapisu $O(1)$. Punkty oraz krawędzie są zapisywane w postaci krotek, ponieważ ten typ w Pythonie może zostać zakodowany jako hash, w przeciwieństwie do list. Takie przechowywanie figur pozwala na minimalizację czasu algorytmu oraz pozwala na łatwiejsze manipulowanie danymi wielokąta, np. dzięki użyciu `set` możemy w szybki, prosty sposób sprawdzić, czy dana krawędź znajduje się wielokącie.

2 Użyte narzędzia i środowisko pracy

Cały program został napisany w języku Python za pomocą popularnej, interaktywnej platformy Jupyter Notebook. W narzędziu graficznym zostały wykorzystywane biblioteki takie jak `matplotlib` (do graficznego przedstawiania danych) oraz `numpy` (do generowania zestawów danych). W otrzymanym narzędziu graficznym został zmieniony backend biblioteki matplotlib (`%matplotlib widget`) poprzez doinstalowanie biblioteki `ipympl`, aby narzędzie to mogło obsługiwać nowsze edytory, m.in. `VSCode`. Dodatkowo, dodana została możliwość tworzenia tytułów wykresów/podwykresów, zmiana zakresów osi OX oraz OY i zostały poprawione występujące błędy.

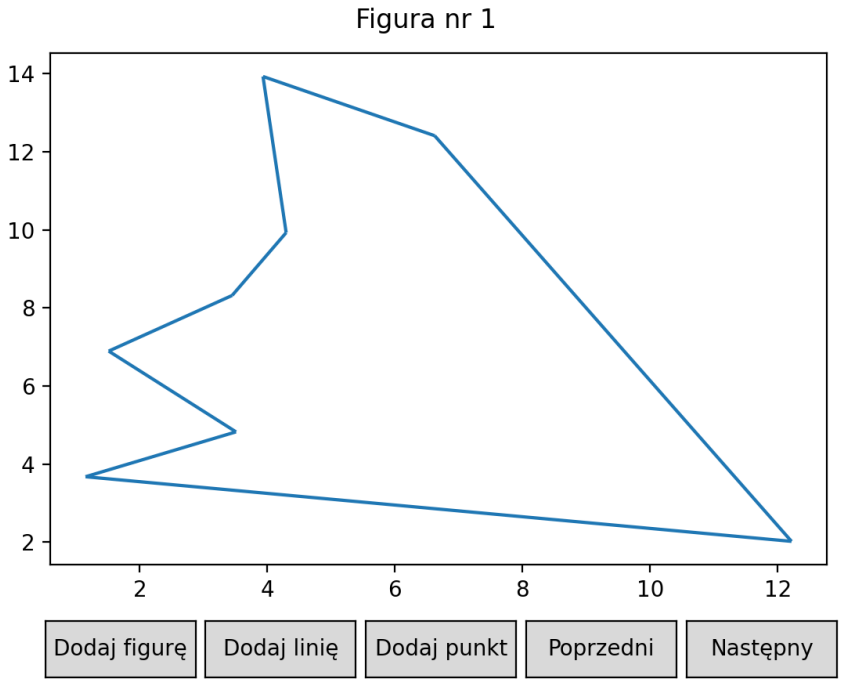
Dane sprzętowe:

- System Windows 10 x64
- Procesor Intel Core i5-8250U

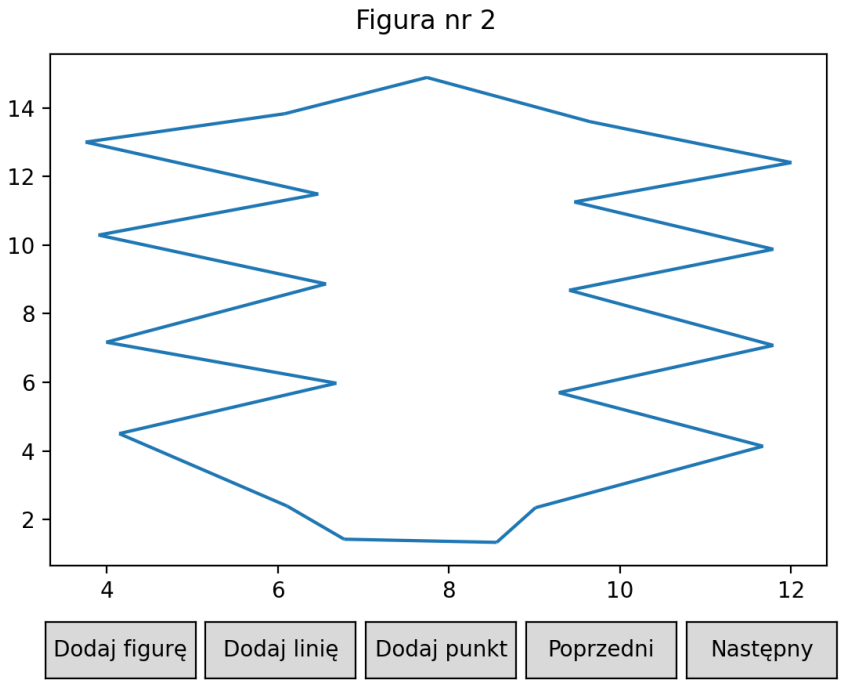
- RAM 8GB
- Wersja języka Python: 3.10

3 Generowanie figur oraz testowanie y-monotoniczności

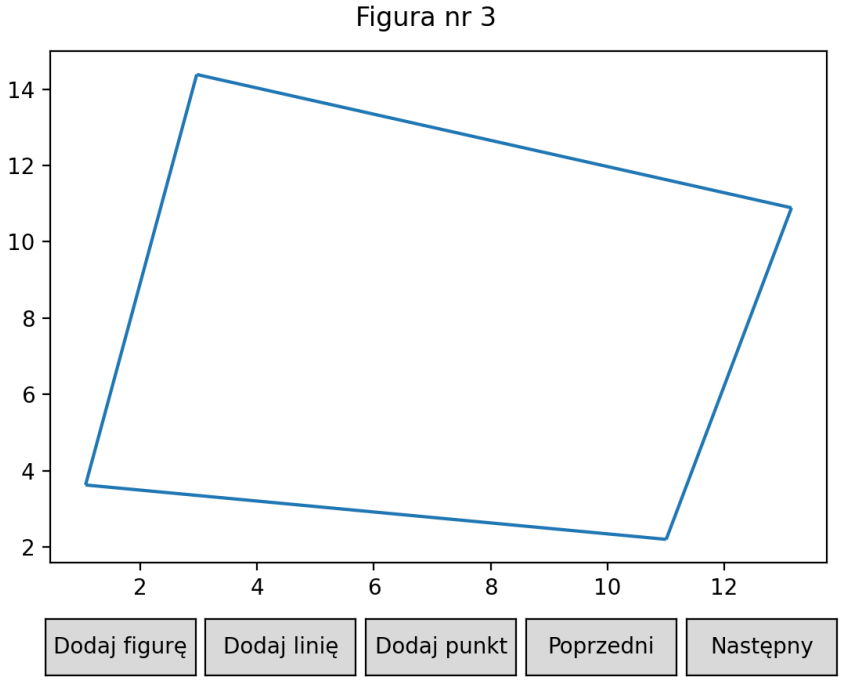
W celu przetestowania algorytmów zostało wygenerowane 14 figur, z czego dwie nie są monotoniczne. Figury te mogą zostać poddane procesowi triangulacji wtedy, gdy wielokąt niemonotoniczny zostanie podzielony na mniejsze, y-monotoniczne wielokąty. Wszystkie figury zostały prawidłowo sklasyfikowane przez algorytm sprawdzający monotoniczność wielokąta.



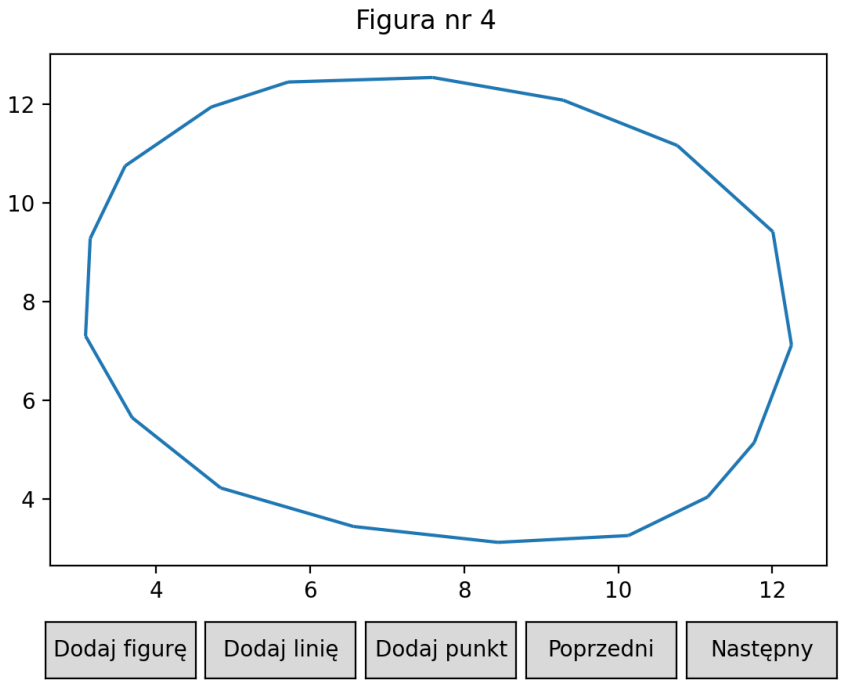
Rysunek 1: Figura nr 1



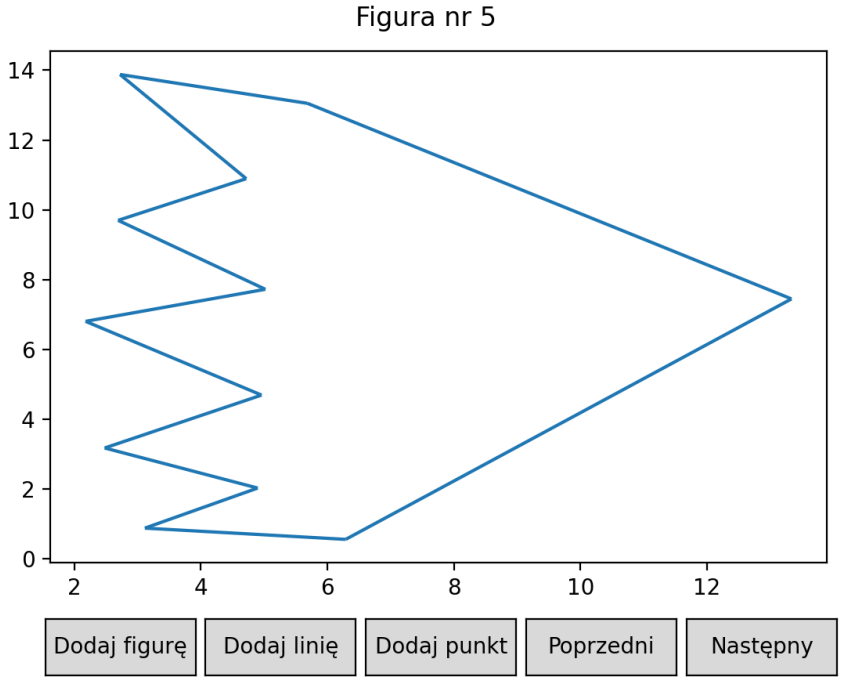
Rysunek 2: Figura nr 2



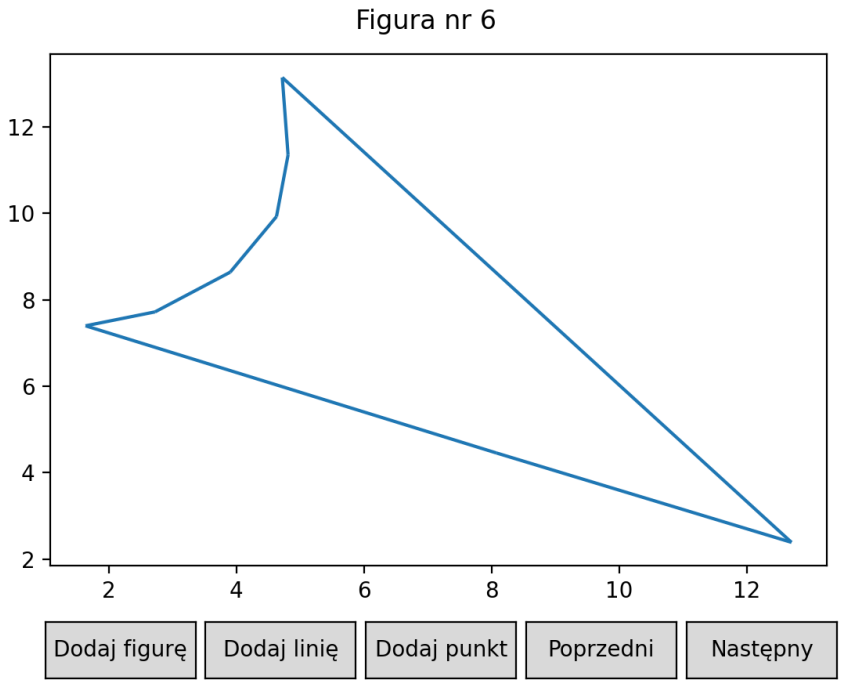
Rysunek 3: Figura nr 3



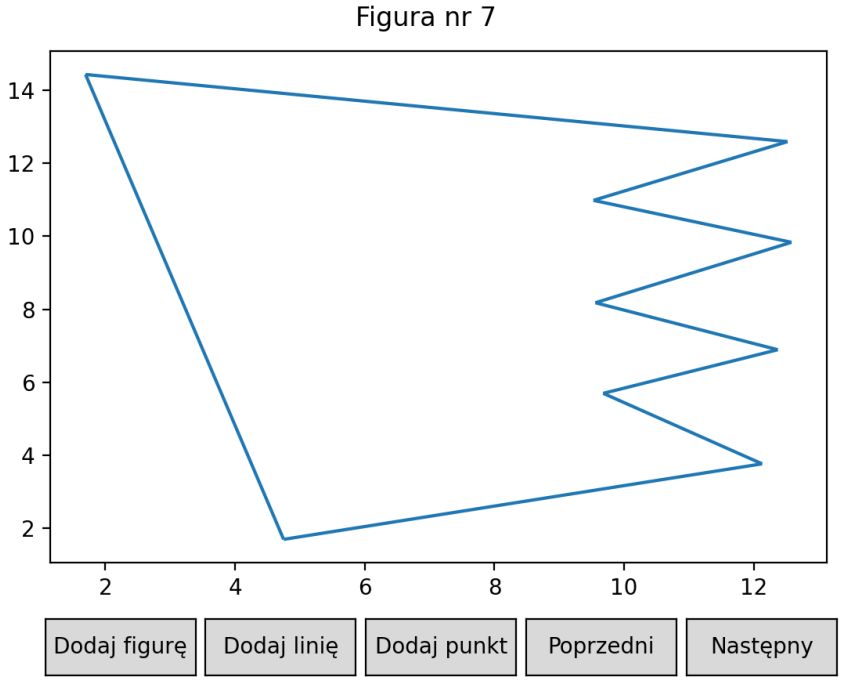
Rysunek 4: Figura nr 4



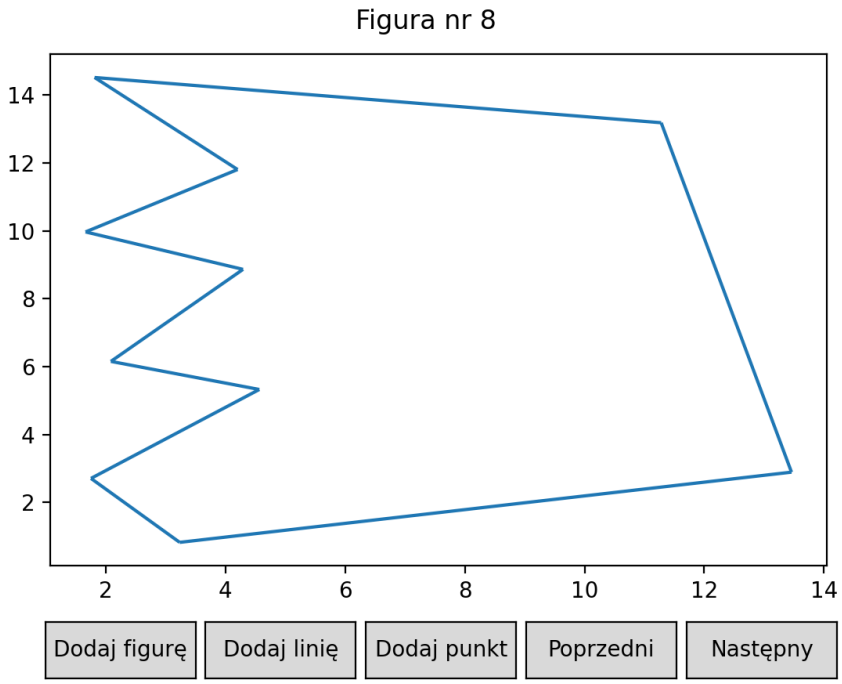
Rysunek 5: Figura nr 5



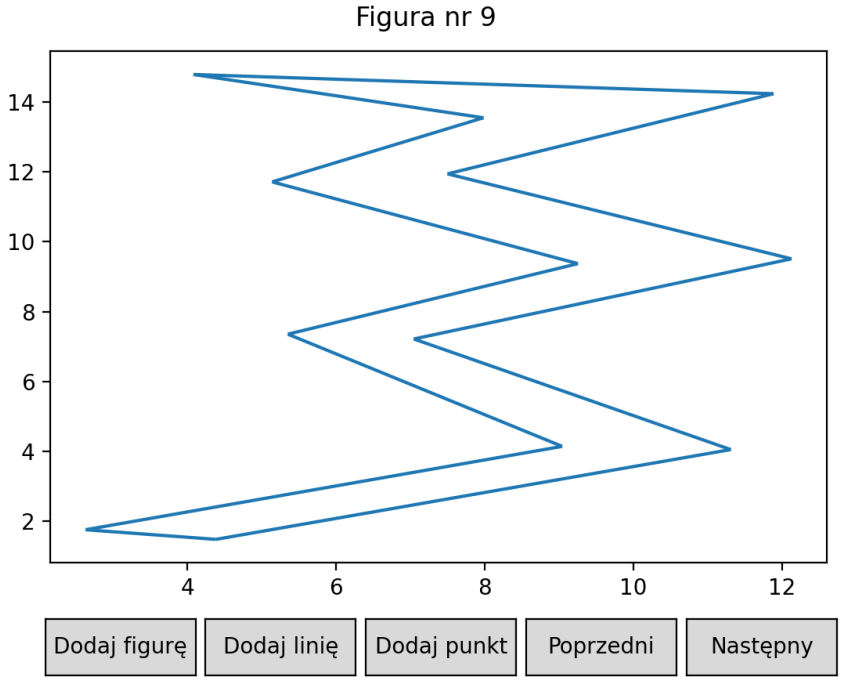
Rysunek 6: Figura nr 6



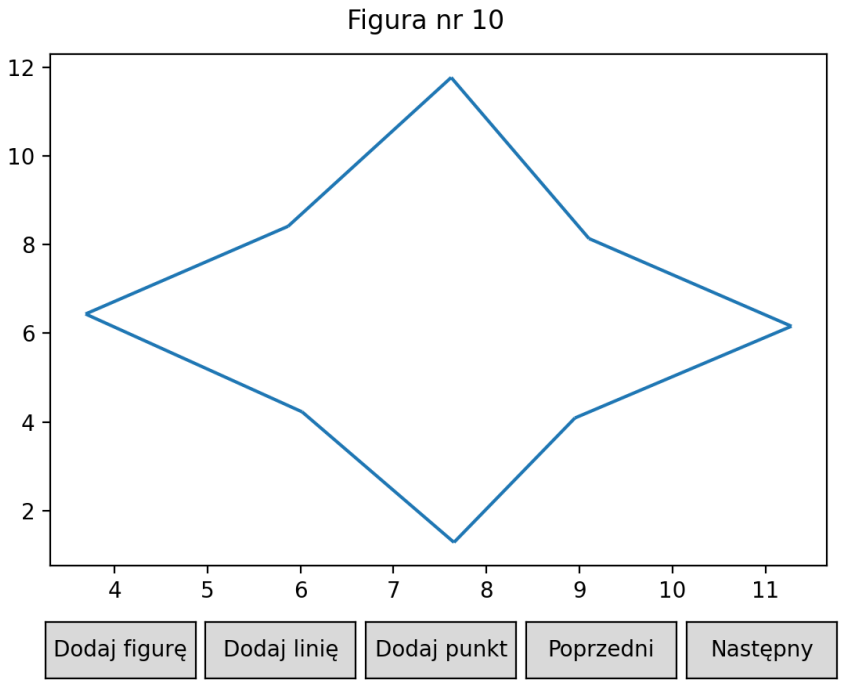
Rysunek 7: Figura nr 7



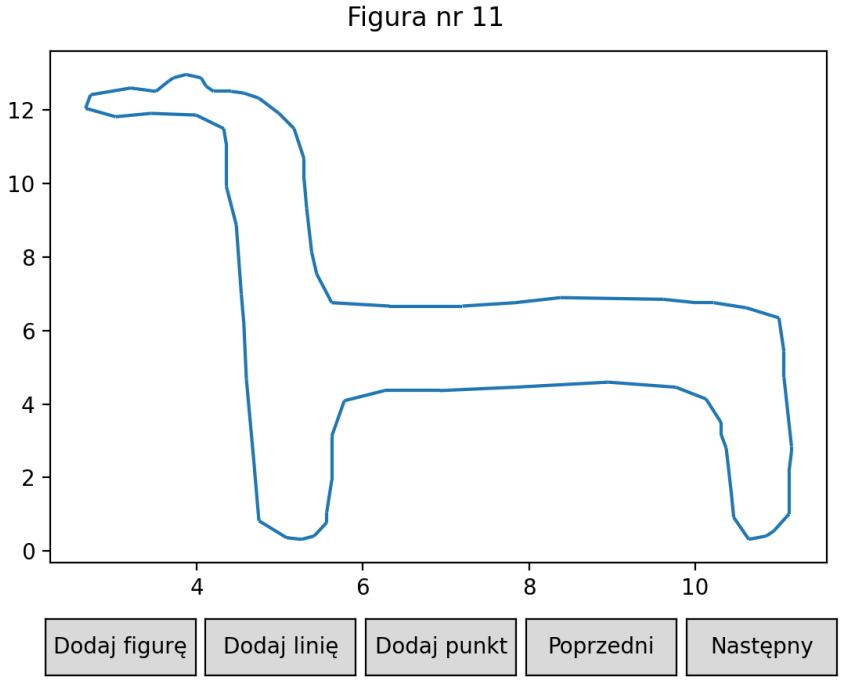
Rysunek 8: Figura nr 8



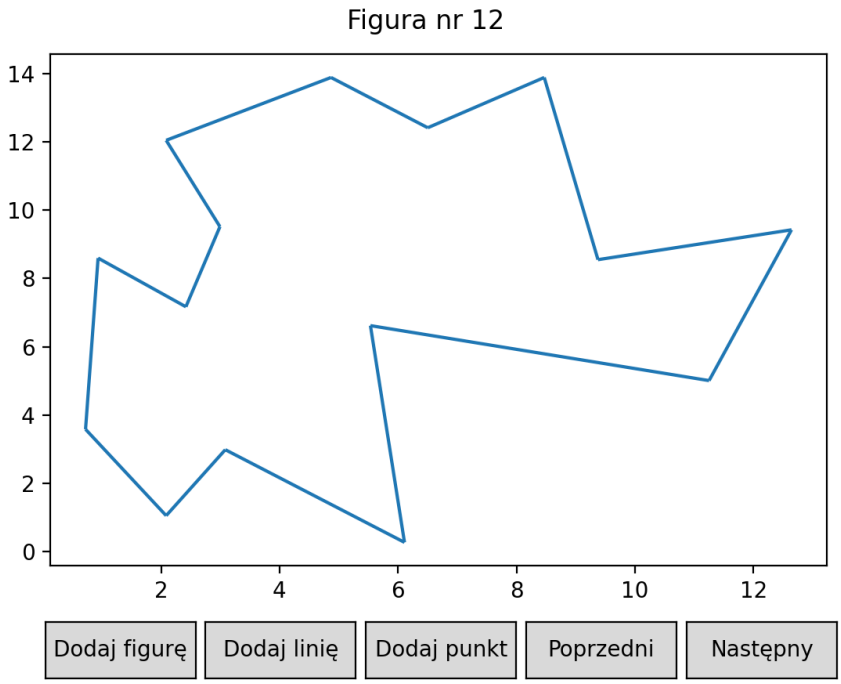
Rysunek 9: Figura nr 9



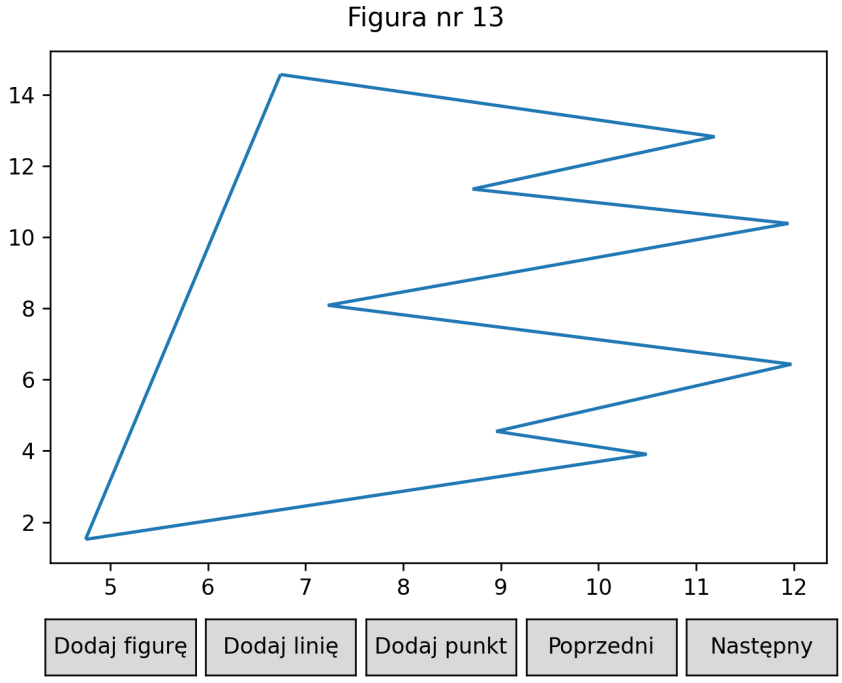
Rysunek 10: Figura nr 10



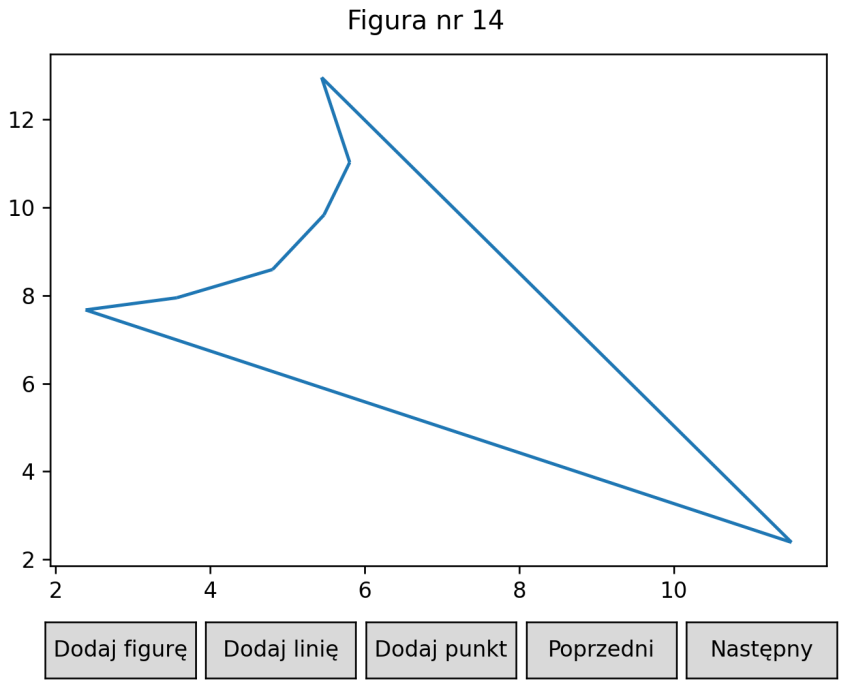
Rysunek 11: Figura nr 11



Rysunek 12: Figura nr 12



Rysunek 13: Figura nr 13



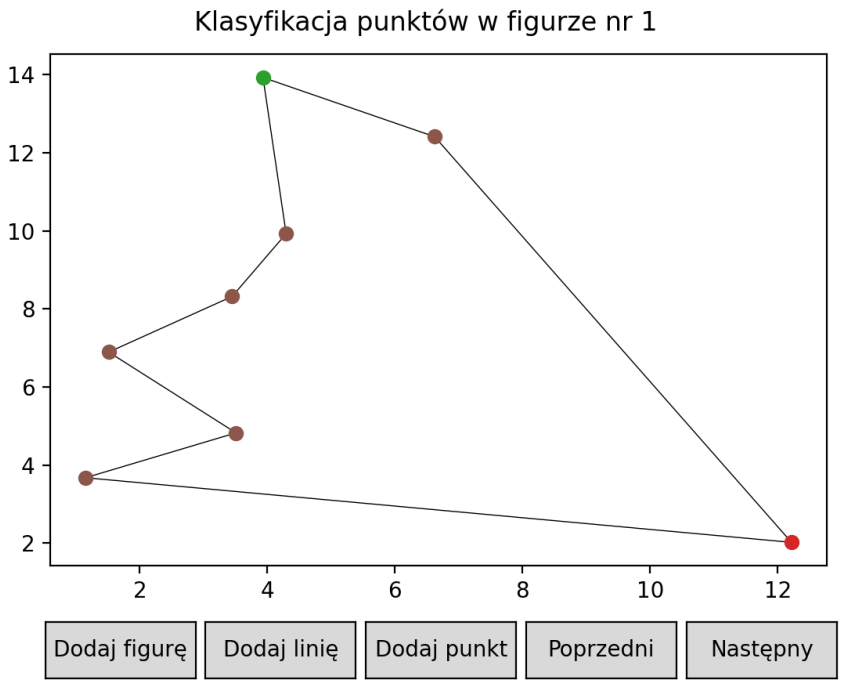
Rysunek 14: Figura nr 14

4 Klasyfikacja wierzchołków

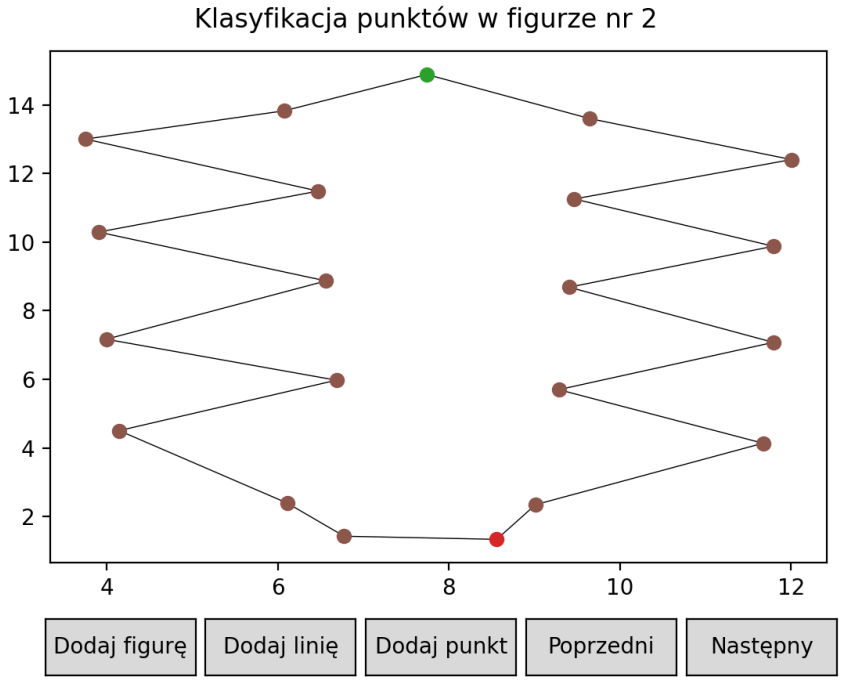
Każdy typ wierzchołka ma odpowiedni kolor:

1. Początkowy - kolor zielony.
2. Końcowy - kolor czerwony.
3. Łączący - kolor ciemnoniebieski.
4. Dzielący - kolor jasnoniebieski.
5. Prawidłowy - kolor brązowy.

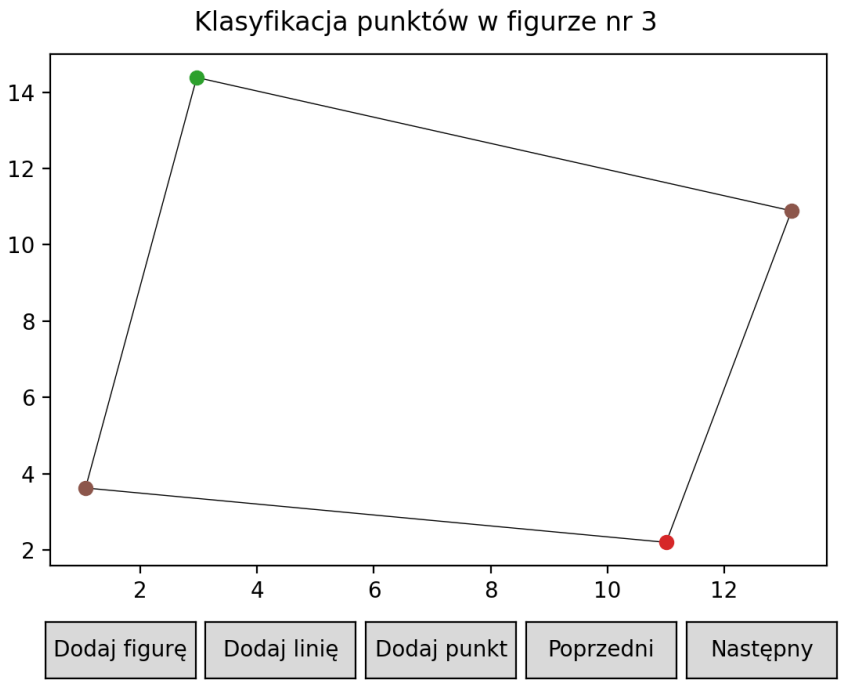
Zgodnie z własnością figur y-monotonicznych, żaden taki wielokąt nie posiada wierzchołka łączącego lub dzielącego, co potwierdza poprawność działania algorytmu do sprawdzania y-monotoniczności wielokąta.



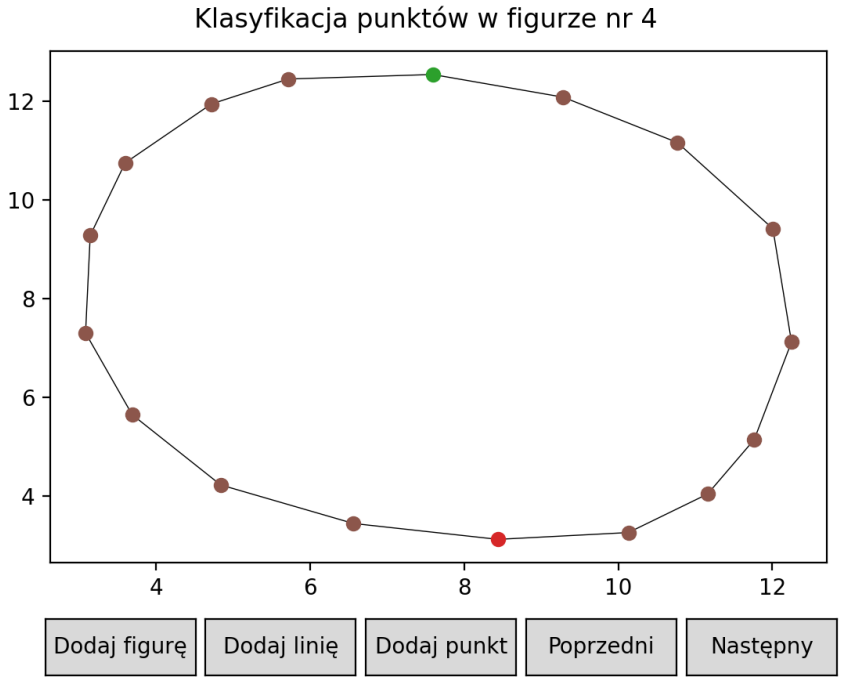
Rysunek 15: Klasyfikacja wierzchołków dla figury nr 1



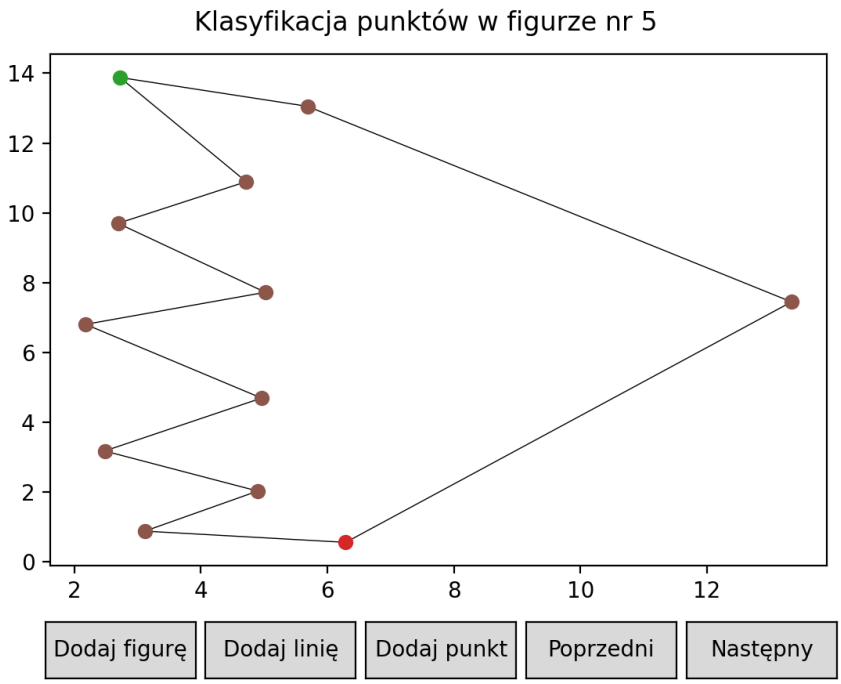
Rysunek 16: Klasyfikacja wierzchołków dla figury nr 2



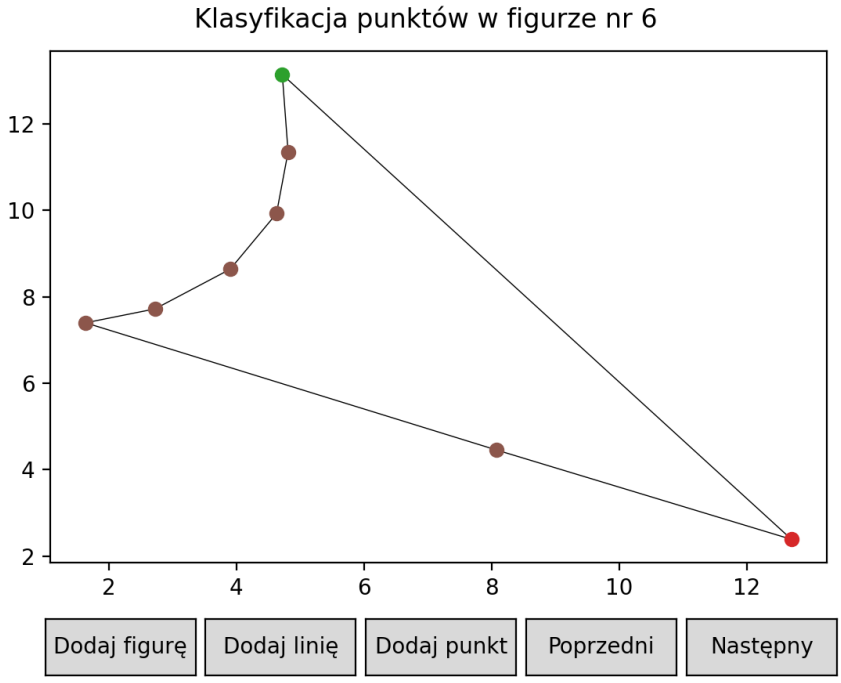
Rysunek 17: Klasyfikacja wierzchołków dla figury nr 3



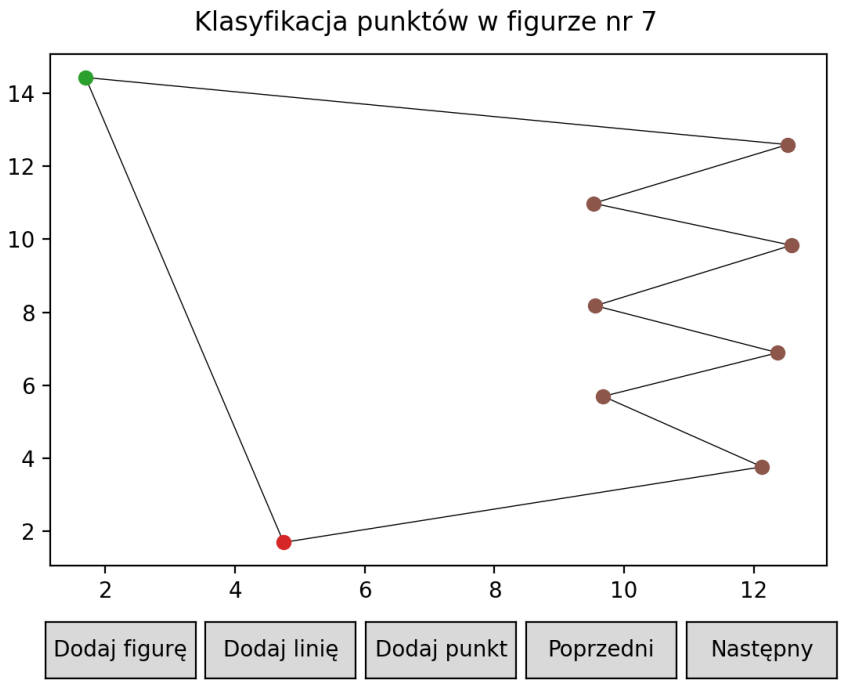
Rysunek 18: Klasyfikacja wierzchołków dla figury nr 4



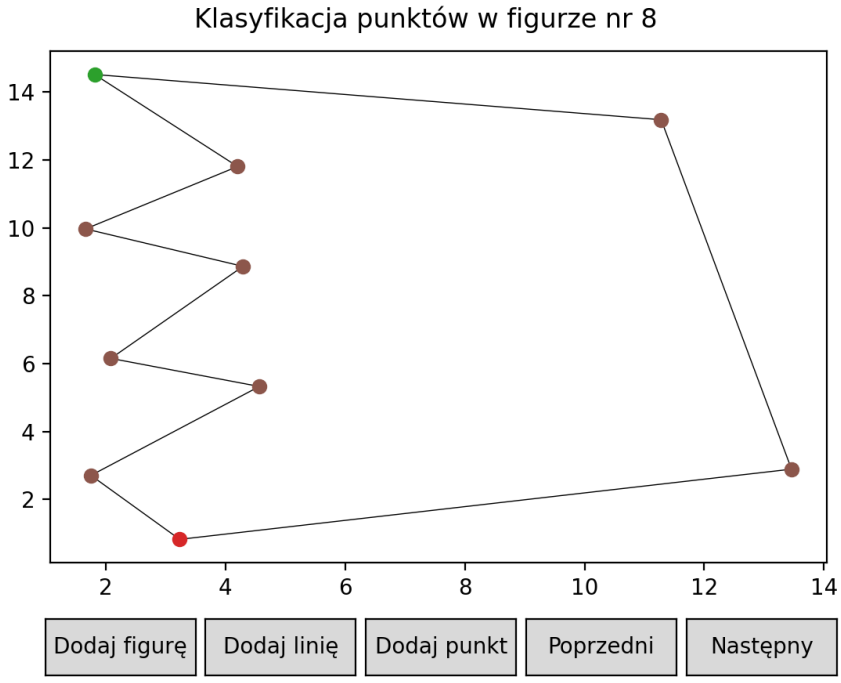
Rysunek 19: Klasyfikacja wierzchołków dla figury nr 5



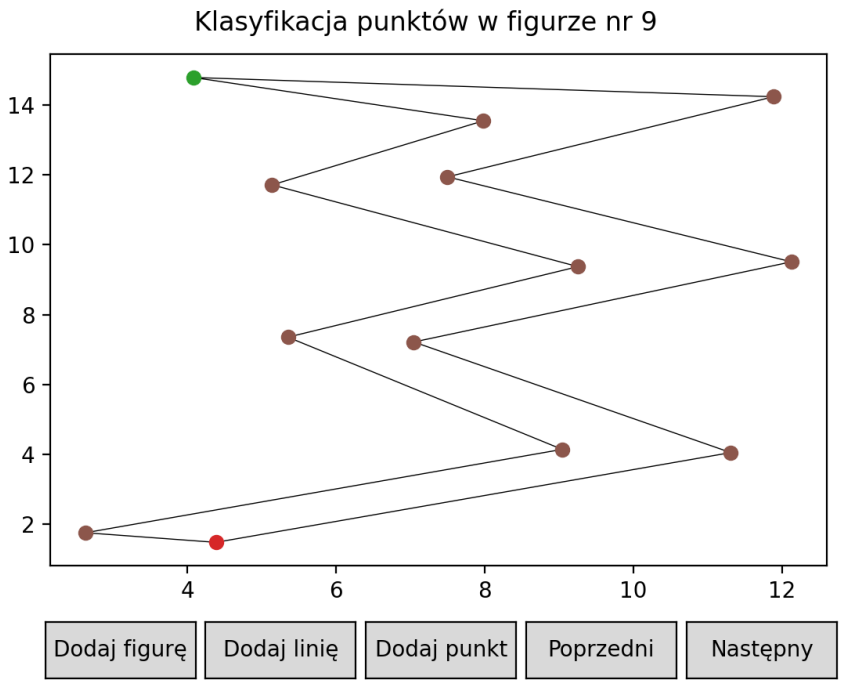
Rysunek 20: Klasyfikacja wierzchołków dla figury nr 6



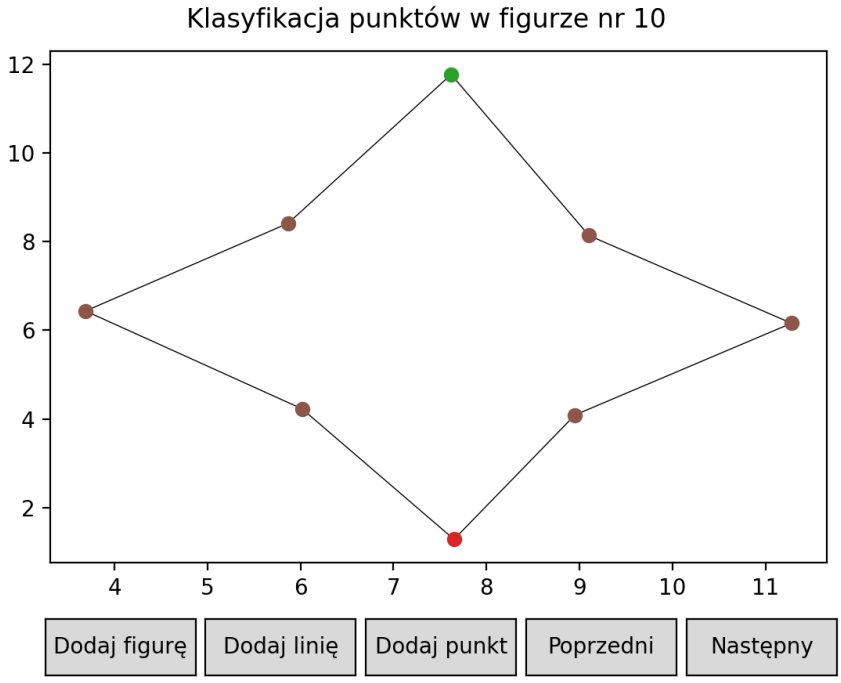
Rysunek 21: Klasyfikacja wierzchołków dla figury nr 7



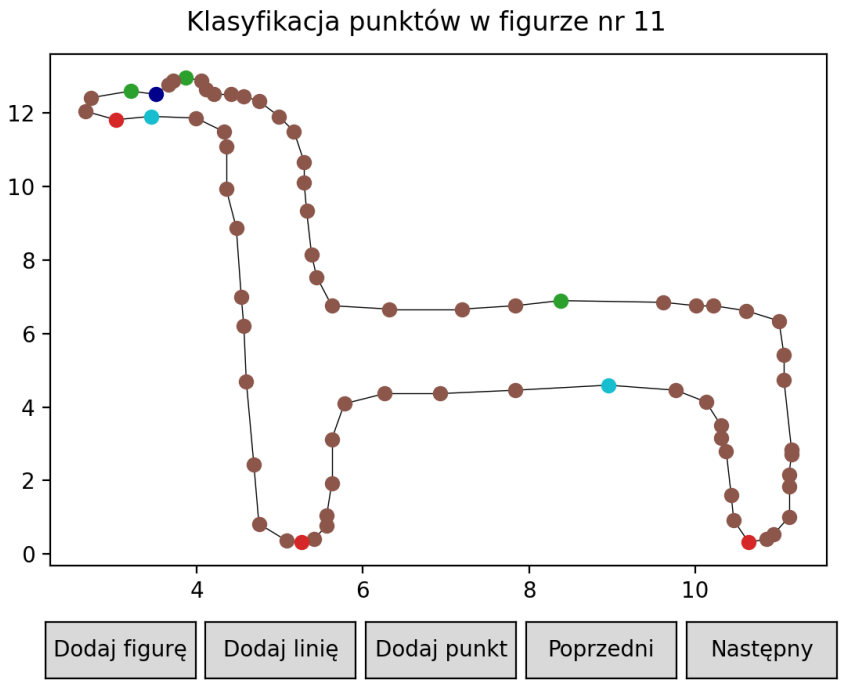
Rysunek 22: Klasyfikacja wierzchołków dla figury nr 8



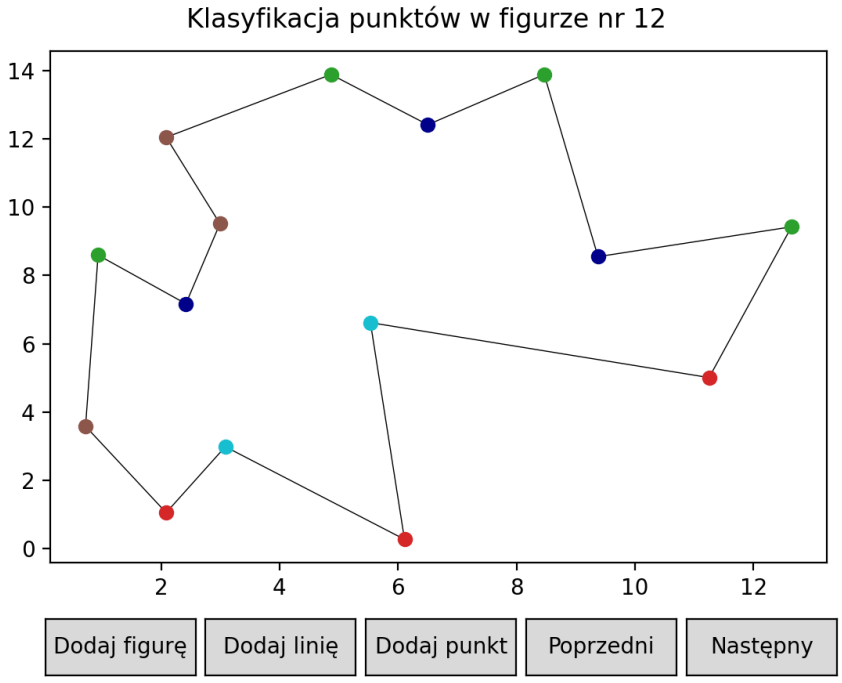
Rysunek 23: Klasyfikacja wierzchołków dla figury nr 9



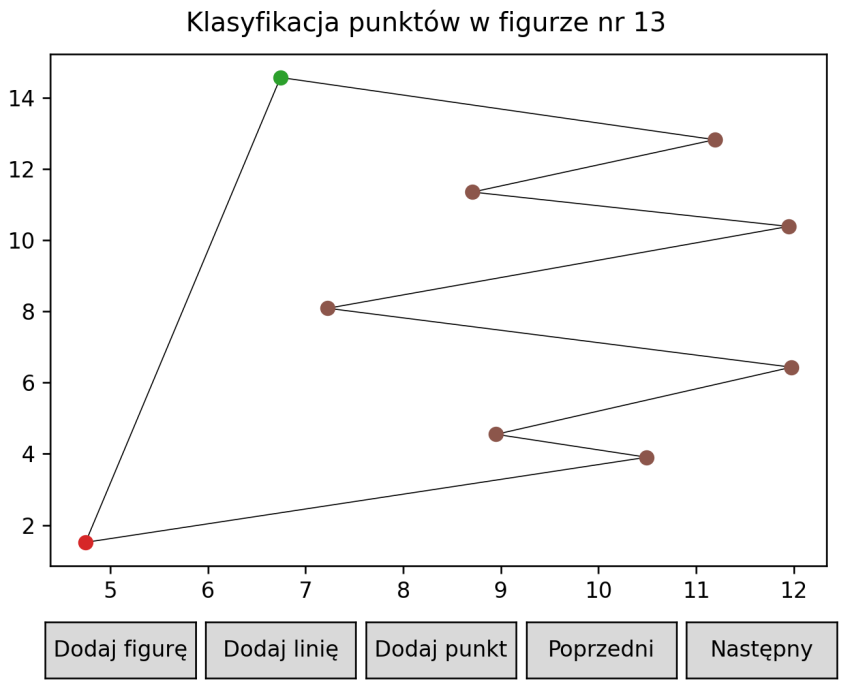
Rysunek 24: Klasyfikacja wierzchołków dla figury nr 10



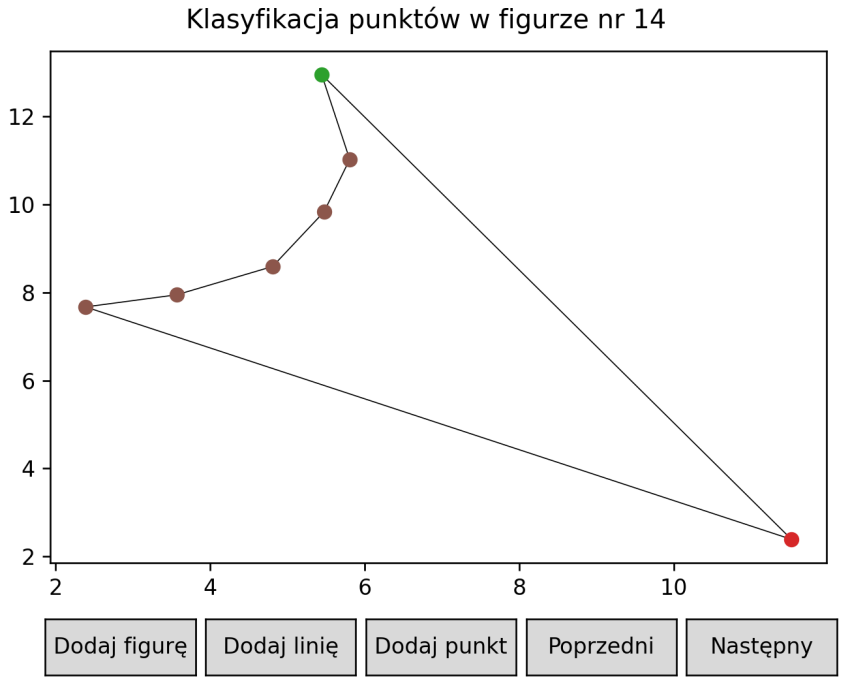
Rysunek 25: Klasyfikacja wierzchołków dla figury nr 11



Rysunek 26: Klasyfikacja wierzchołków dla figury nr 12



Rysunek 27: Klasyfikacja wierzchołków dla figury nr 13

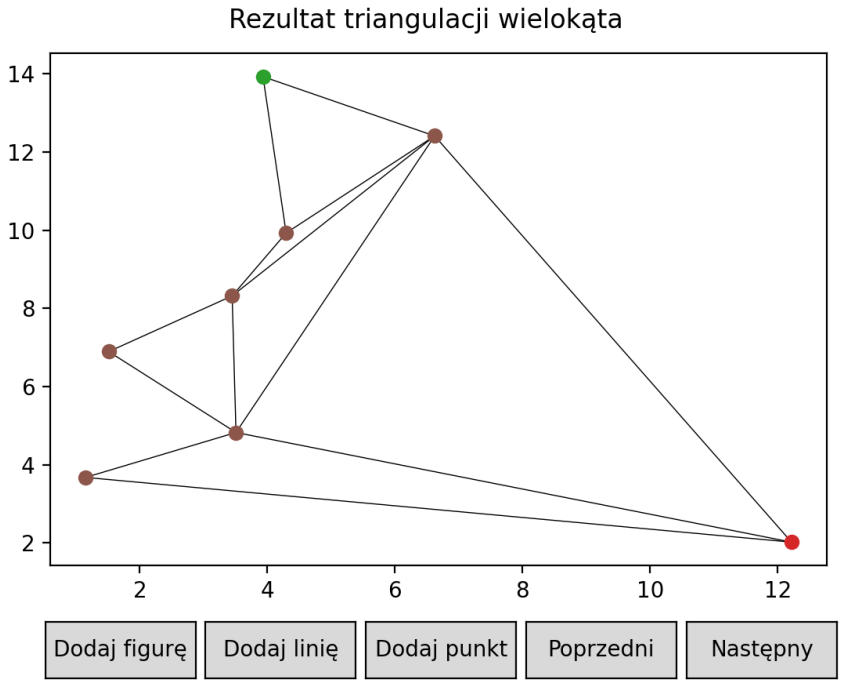


Rysunek 28: Klasyfikacja wierzchołków dla figury nr 14

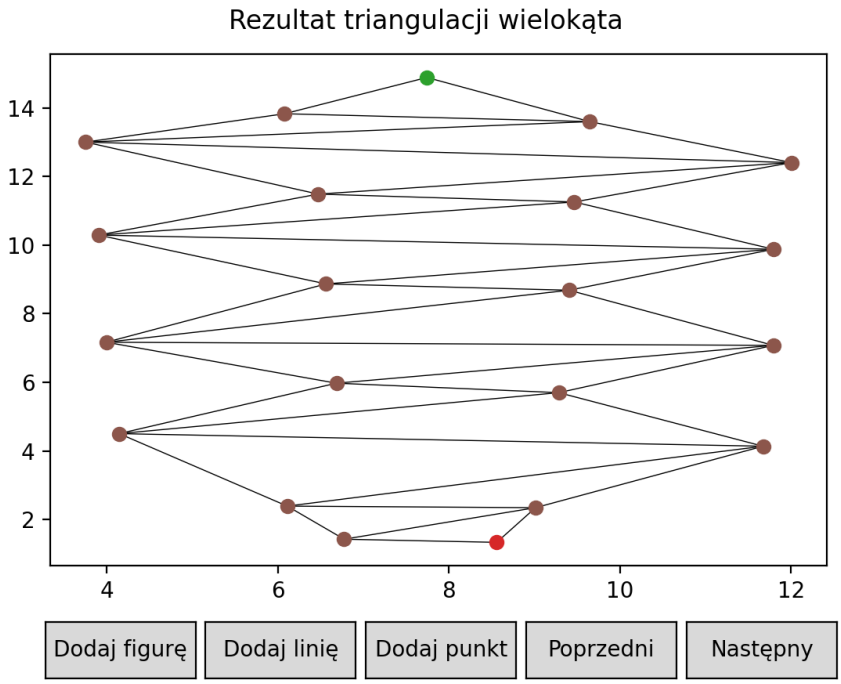
5 Triangulacja figur y-monotonicznych

Wszystkie triangulacje, wnioskując z poniżej przedstawionych wyników, zostały wykonane prawidłowo. Dla figur numer 11 oraz 12 nie została przeprowadzona triangulacja, ponieważ te figury nie są y-monotoniczne.

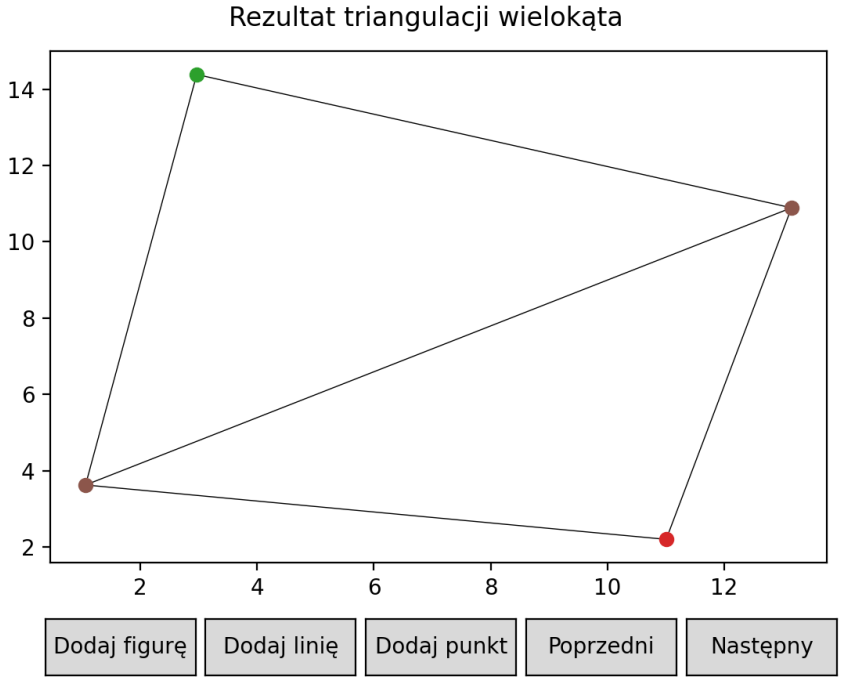
5.1 Triangulacje wielokątów



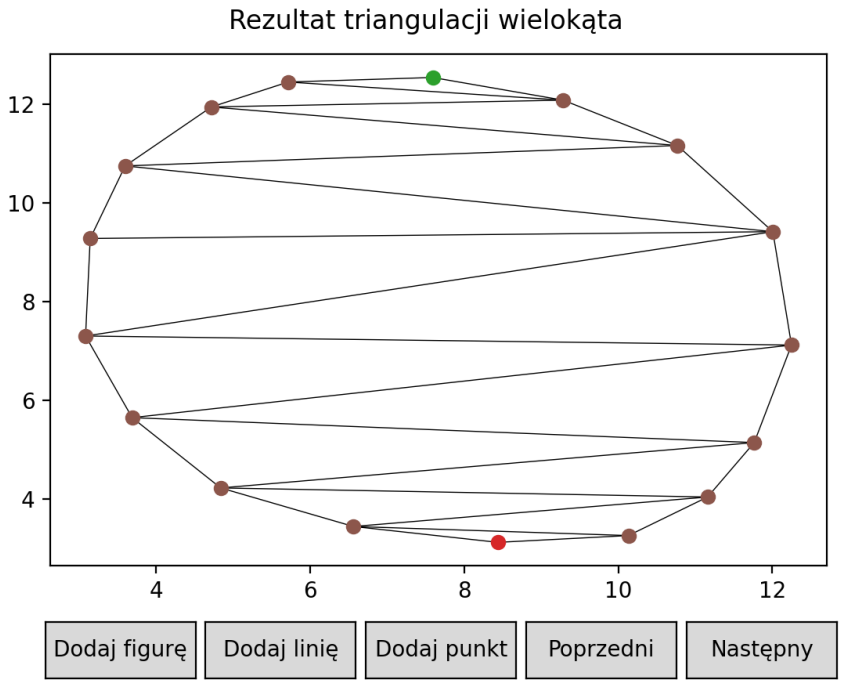
Rysunek 29: Triangulacja dla figury nr 1



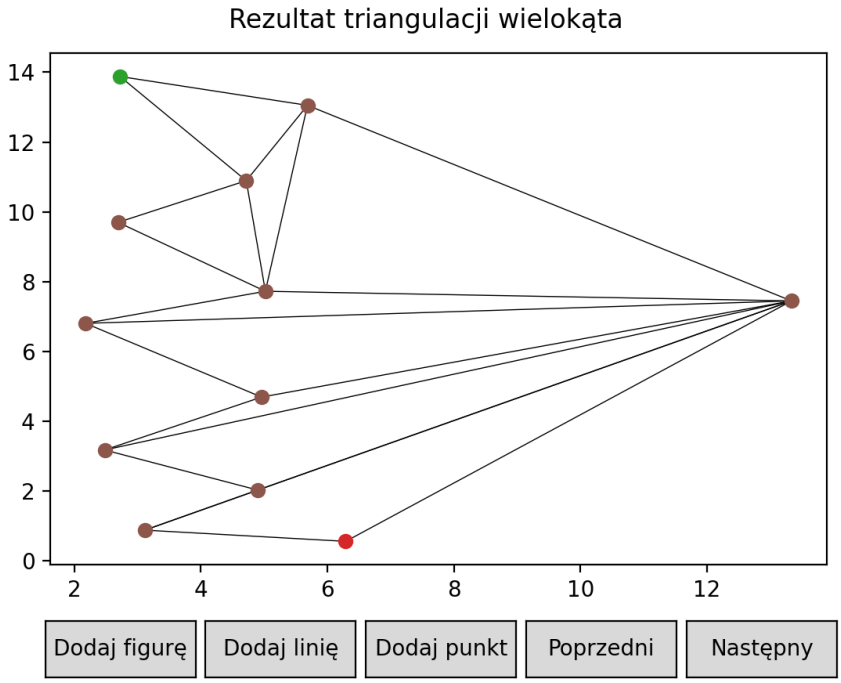
Rysunek 30: Triangulacja dla figury nr 2



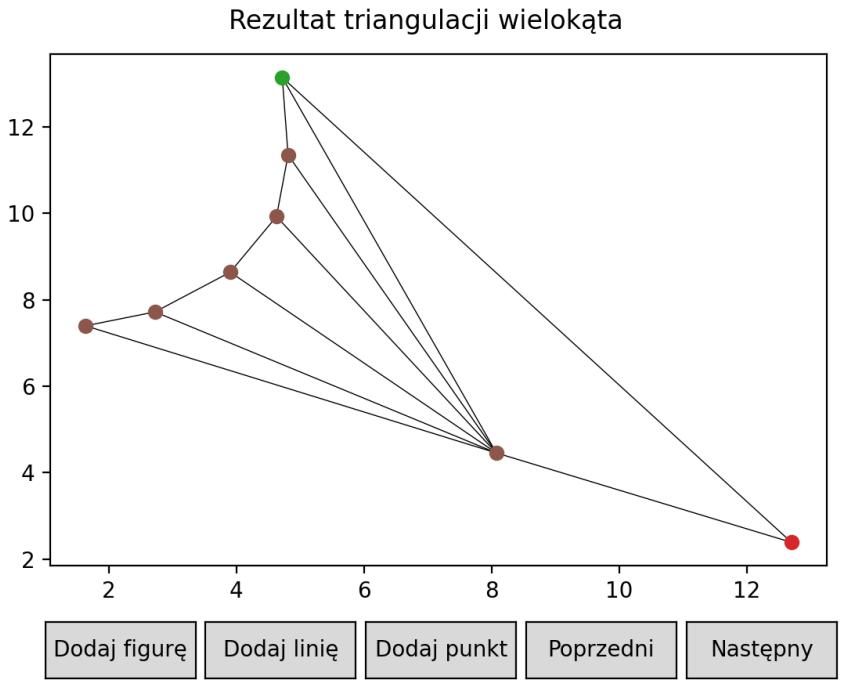
Rysunek 31: Triangulacja dla figury nr 3



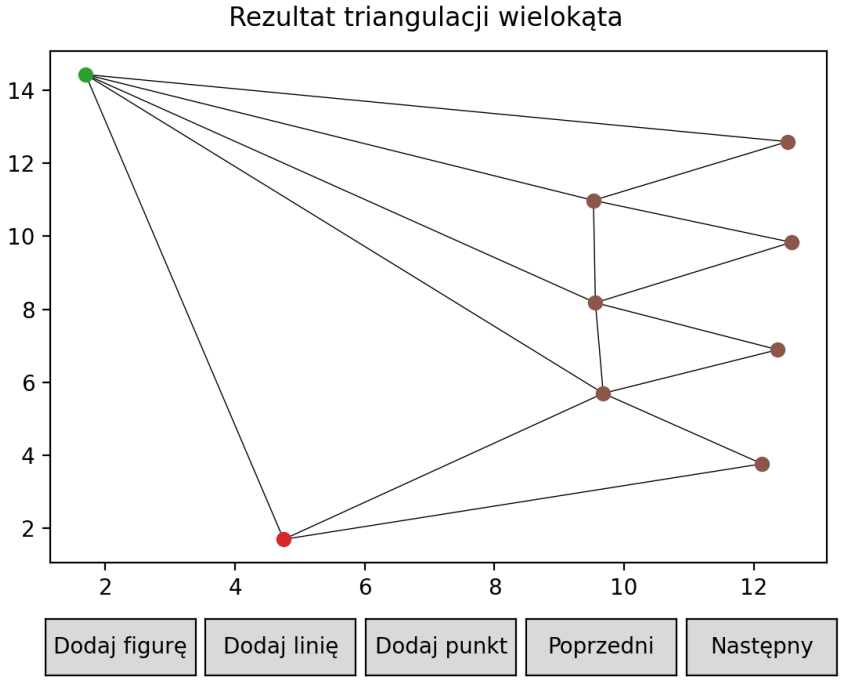
Rysunek 32: Triangulacja dla figury nr 4



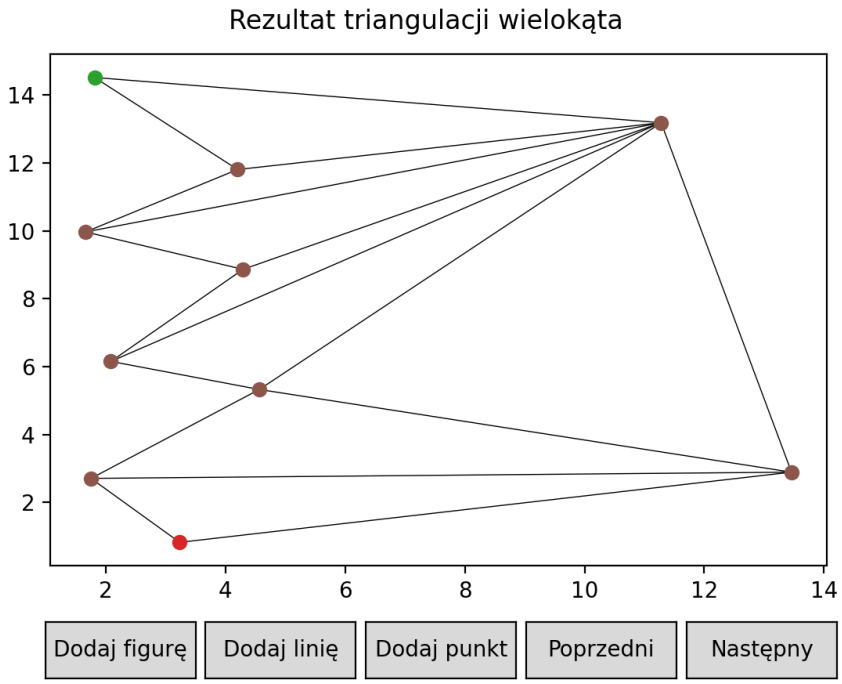
Rysunek 33: Triangulacja dla figury nr 5



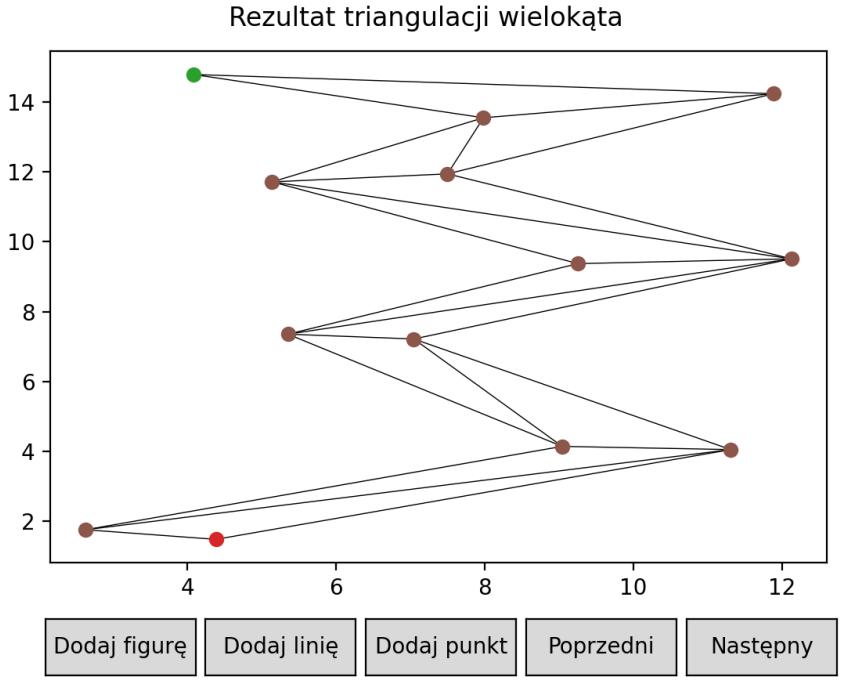
Rysunek 34: Triangulacja dla figury nr 6



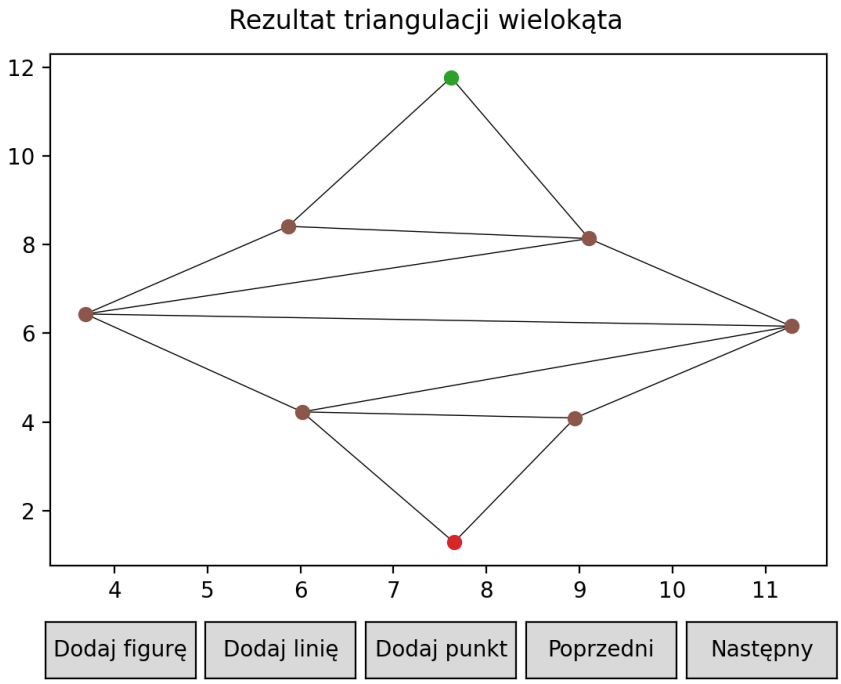
Rysunek 35: Triangulacja dla figury nr 7



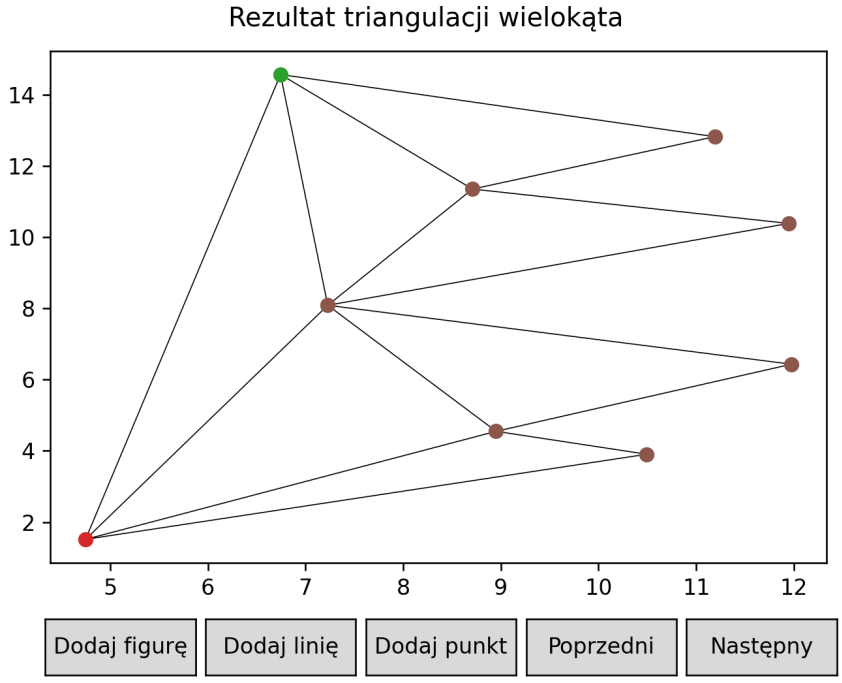
Rysunek 36: Triangulacja dla figury nr 8



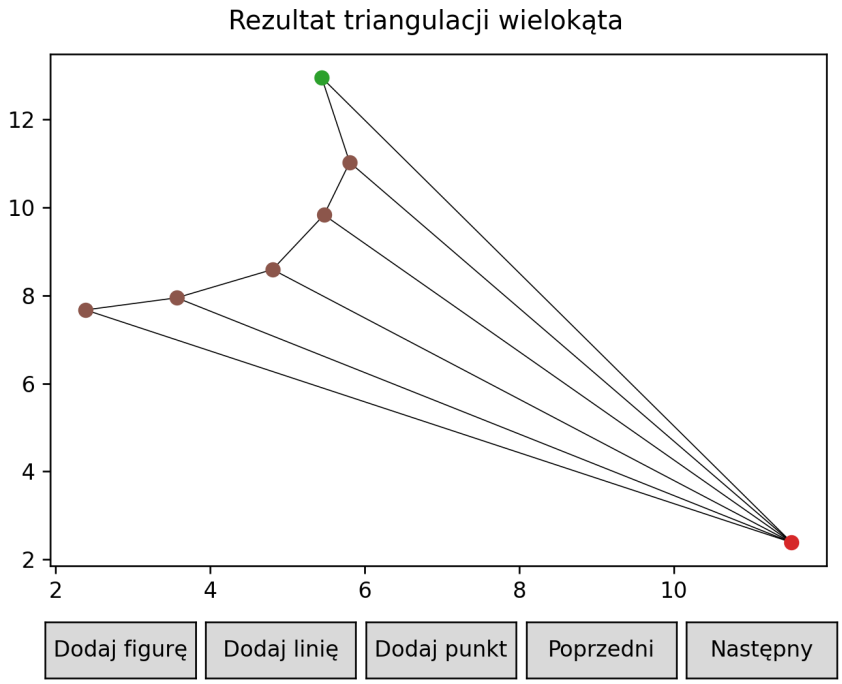
Rysunek 37: Triangulacja dla figury nr 9



Rysunek 38: Triangulacja dla figury nr 10



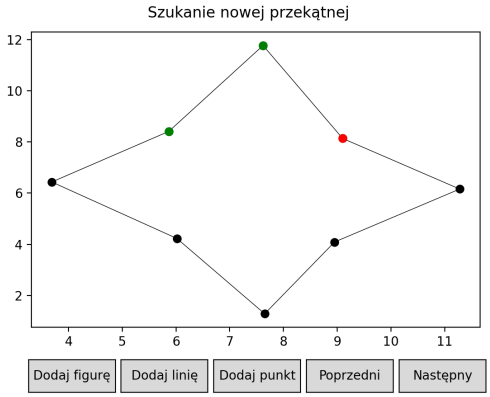
Rysunek 39: Triangulacja dla figury nr 13



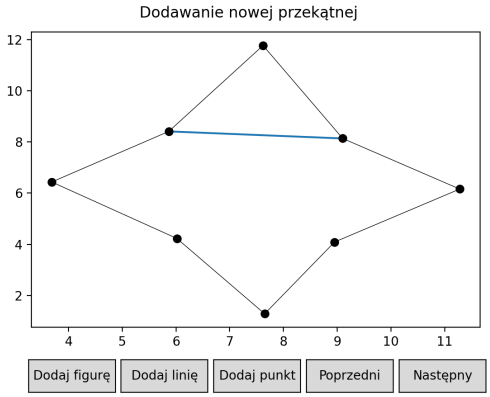
Rysunek 40: Triangulacja dla figury nr 14

5.2 Przykładowe działanie algorytmu y-monotoniczności dla wielokąta nr 10

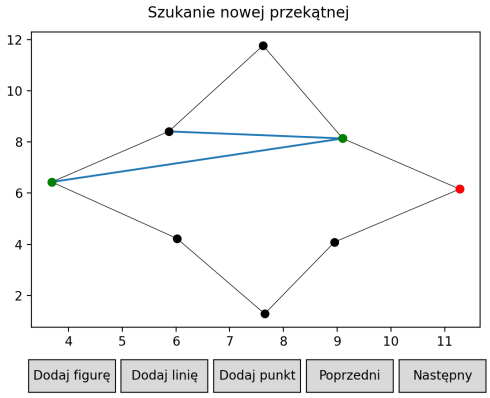
W celu ilustracji działania algorytmu każdy wykres zapisuje przebieg dodawanych przekątnych jak i aktualnie badanych punktów. Kolor czerwony to kolor wierzchołka aktualnie badanego, kolor zielony wierzchołka to punkt ze stosu. Nowo dodane przekątne są pokolorowane na niebiesko.



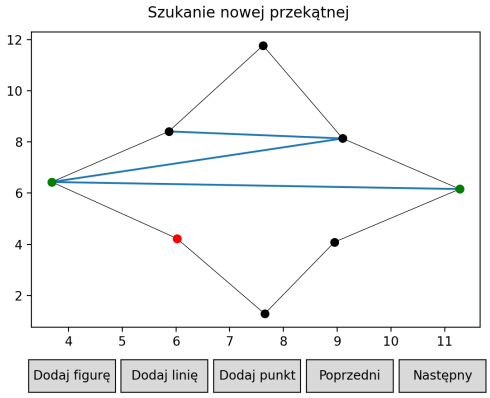
(a) Krok 1



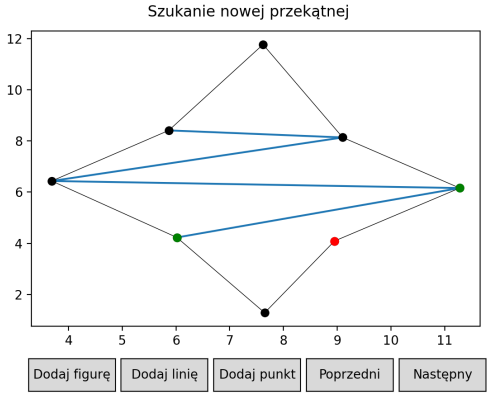
(b) Krok 2



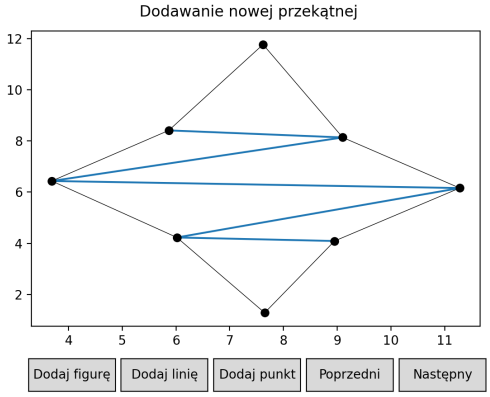
(c) Krok 3



(d) Krok 4



(e) Krok 5



(f) Krok 6

Rysunek 41: Poszczególne kroki algorytmu triangulacji dla figury nr 10

6 Wnioski

Wszystkie algorytmy działają prawidłowo, a użycie struktur danych takich jak `set`, czy `dict`, pozwoliło na zminimalizowanie czasu triangulacji wielokąta.