Algorytmy Geometryczne - Laboratorium 2

Otoczka wypukła

Imię i nazwisko: Wermiński Gabriel **Data wykonania:** 05.11.2025

Grupa: 6

1. Środowisko obliczeniowe

Eksperymenty zostały przeprowadzone w następującej konfiguracji:

- **Procesor:** CPU 12th Gen Intel(R) Core(TM) i5-1235U, 1.30 GHz
- **Pamięć RAM:** 16.0 GB, 3200 MT/s
- **System operacyjny:** Microsoft Windows 11 Home
- **Środowisko programistyczne:** Jupyter Notebook, Pycharm
- **Język programowania:** Python 3.13.5
- **Biblioteki:** numpy, pandas, matplotlib

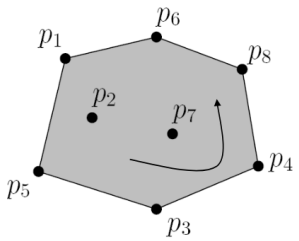
2. Cel ćwiczenia

Głównym zadaniem przeprowadzonego ćwiczenia było opracowanie dwóch algorytmów służących do wyznaczania otoczki wypukłej: Grahama oraz Jarvisa. W ramach projektu porównano ich efektywność obliczeniową, analizując zależność czasu wykonania od liczby punktów w zbiorze danych. Dodatkowo, dla każdego przypadku przeprowadzono wizualizację działania obu algorytmów oraz zestawiono uzyskane wyniki w celu oceny różnic w złożoności czasowej i jakości wyznaczonych otoczek.

3. Wstęp teoretyczny

3.1. Definicja otoczki wypukłej

Otoczka wypukła zbioru punktów S, oznaczana jako *CH(S)*, to najmniejszy zbiór wypukły zawierający wszystkie punkty należące do S. Innymi słowy, jest to figura, której każdy punkt znajduje się na zewnątrz lub wewnątrz względem wszystkich elementów zbioru, a żaden punkt spoza niej nie należy do otoczki.



Rysunek 1: Ilustracja przykładowej otoczki wypukłej $\mathcal{L} = (p_3, p_4, p_8, p_6, p_1, p_5)$ utworzonej ze zbioru punktów $\{p_1, p_2, ..., p_8\}$.

W przypadku przestrzeni dwuwymiarowej, otoczka ma postać wielokąta wypukłego o minimalnym obwodzie obejmującego dany układ punktów. Wierzchołki tego wielokąta zapisuje się kolejno w kierunku przeciwnym do ruchu wskazówek zegara, co odpowiada konwencji układu prawoskrętnego.

W sytuacji, gdy część punktów zbioru leży na jednej prostej, do otoczki włączane są jedynie te, które stanowią punkty skrajne na danym odcinku. Pozostałe punkty współliniowe nie wpływają na kształt otoczki i są pomijane w końcowym zbiorze wierzchołków.

3.2. Algorytmy znajdowania otoczki wypukłej

3.2.1. Algorytm Grahama

Działanie algorytmu Grahama opiera się na systematycznym eliminowaniu wierzchołków tworzących wklęsłości. Mechanizm ten wykorzystuje stos przechowujący kandydatów do otoczki oraz przegląd punktów uporządkowanych kątowo.

1. Ze zbioru Q wyznaczamy punkt startowy p_0 o najmniejszej wartości współrzędnej y. Gdy istnieje wiele punktów o tej samej współrzędnej y, spośród nich wybieramy punkt o minimalnej współrzędnej x.

2. Punkty pozostałe porządkujemy względem kąta, jaki tworzy wektor (p_0, p) z dodatnim kierunkiem osi OX. Do wyznaczenia kąta stosujemy funkcję `arctan2(vy, vx)`, gdzie `vx` oraz `vy` reprezentują składowe wektora łączącego p_0 z punktem p. W sytuacji, gdy różne punkty wyznaczają ten sam kąt (z tolerancją `eps`), w ciągu pozostawiamy wyłącznie punkt najdalszy od p_0, eliminując pozostałe, co daje uporządkowaną sekwencję $\{p_1, p_2, ..., p_m\}$.

3. Inicjalizujemy stos S poprzez umieszczenie na nim punktów p_0 oraz p_1. Indeks i ustawiamy na wartość 2.

4. Iterujemy, póki $i < m$, badając położenie punktu p_i względem linii prostej wyznaczonej przez dwa wierzchołki ze szczytu stosu. Sprawdzenie wykonujemy poprzez obliczenie wyznacznika macierzy 3×3 (funkcja `mat_det_3x3`). Gdy wyznacznik przekracza `eps`, punkt znajduje się po lewej stronie – wówczas umieszczamy p_i na stosie i inkrementujemy i. W przeciwnym wypadku zdejmujemy element ze stosu i kontynuujemy weryfikację dla niezmienionego p_i.

Złożoność obliczeniowa algorytmu Grahama wynika z operacji wyszukiwania ekstremum, uporządkowania punktów, eliminacji współkątowych oraz konstrukcji otoczki stosując stos. Ponieważ każdy punkt podlega analizie maksymalnie jednokrotnie w fazie budowy stosu, kluczowym czynnikiem determinującym złożoność jest etap sortowania. Rezultatem jest złożoność rzędu $O(n \log n)$, gdzie n oznacza liczbę elementów zbioru wejściowego.

3.2.2. Algorytm Jarvisa

Mechanizm algorytmu Jarvisa polega na sekwencyjnym podejmowaniu optymalnych decyzji lokalnych poprzez wskazywanie punktu wyznaczającego najmniejszy kąt z bieżącą krawędzią, ignorując konsekwencje dla kolejnych kroków. Takie podejście prowadzi ostatecznie do uzyskania globalnie optymalnego wyniku w postaci prawidłowej otoczki wypukłej.

1. Wyznaczamy w zbiorze Q punkt p_0 charakteryzujący się najmniejszą wartością współrzędnej y, a w przypadku równości – najmniejszą współrzędną x. Punkt ten włączamy do otoczki H i traktujemy jako punkt bieżący.

2. Dla kolejnych elementów zbioru Q poszukujemy punktu następnego, który zajmuje najbardziej lewostronną pozycję w stosunku do kierunku wyznaczonego przez punkt bieżący i obecnego kandydata. Weryfikację przeprowadzamy wykorzystując wyznacznik macierzy 3×3 (funkcja `mat_det_3x3`):
   - Gdy wyznacznik przyjmuje wartość ujemną (poniżej `-eps`), punkt kandydujący lokalizuje się bardziej na lewo niż punkt następny, więc aktualizujemy punkt następny na punkt kandydujący.
   - Gdy wyznacznik oscyluje wokół zera (w zakresie $[-\text{eps}, \text{eps}]$), punkty leżą na jednej prostej. Wówczas preferencję otrzymuje punkt bardziej oddalony od punktu bieżącego, co ustalamy poprzez porównanie kwadratów odległości.

3. Po analizie wszystkich kandydatów, wybrany punkt następny staje się kolejnym elementem otoczki i zostaje dołączony do H. Punkt bieżący przyjmuje wartość punktu następnego.

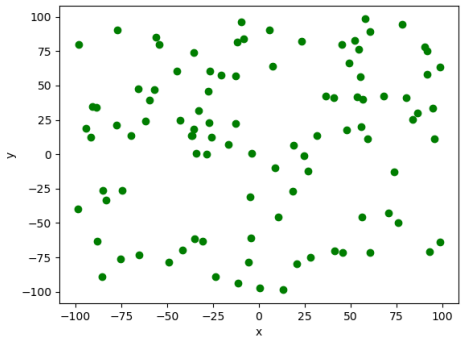
4. Cykl kontynuujemy aż do osiągnięcia punktu p_0 (warunek: punkt bieżący jest identyczny z punktem startowym).

Algorytm Jarvisa charakteryzuje się złożonością $O(nh)$, gdzie h oznacza liczbę wierzchołków otoczki, natomiast n reprezentuje moc zbioru wejściowego. Dla każdego z h elementów otoczki wykonujemy przegląd wszystkich n punktów początkowych. W szczególnych konfiguracjach danych, jak punkty

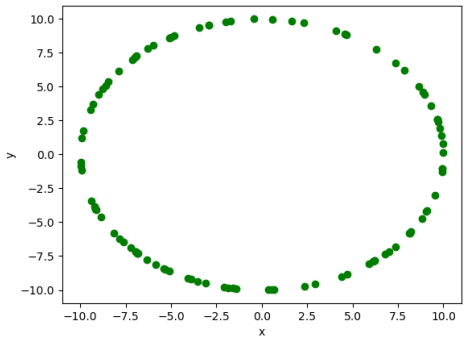
rozmieszczone wzdłuż obwodu prostokąta, wartość h utrzymuje się na poziomie $O(1)$, co skutkuje liniową zależnością czasową algorytmu od liczebności zbioru wejściowego.

4. Realizacja ćwiczenia

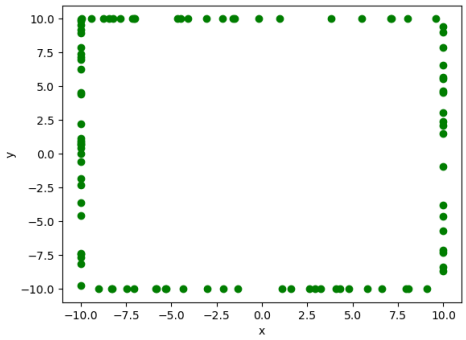
Punkt wyjścia stanowiła generacja czterech różnorodnych zbiorów punktów (reprezentowanych typem double) w przestrzeni euklidesowej $\mathbb{R}^2$. Proces tworzenia danych oparto na funkcjach biblioteki *numpy* (`numpy.random.uniform()`).



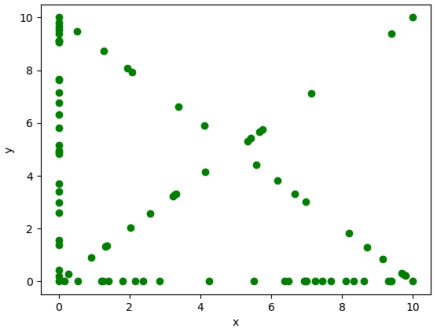
Rysunek 2: Zbiór A: 100 losowo wygenerowanych punktów o współrzędnych z przedziału $[-100, 100]$



Rysunek 3: Zbiór B: 100 losowo wygenerowanych punktów leżących na okręgu o środku $(0, 0)$ i promieniu $R = 10$



Rysunek 4: Zbiór C: 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach $(-10, 10), (-10, -10), (10, -10), (10, 10)$



Rysunek 5: Zbiór D: wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu

Kolejnym krokiem było opracowanie implementacji algorytmów z wykorzystaniem autorskiej funkcji obliczającej wyznacznik macierzy 3×3 (`mat_det_3x3`), która pozwala określić orientację trójki punktów w płaszczyźnie.

W zastosowanej implementacji algorytmu Grahama proces rozpoczyna się od identyfikacji punktu bazowego o minimalnej współrzędnej y (przy równości wybierany jest punkt o najmniejszej współrzędnej x). Następnie pozostałe punkty sortowane są według dwóch kryteriów: kąta obliczanego funkcją `arctan2()` oraz odległości od punktu bazowego. Sortowanie realizowane jest poprzez utworzenie słownika `Q_angles`, który dla każdego punktu przechowuje parę (kąt, odległość kwadratowa). Punkty są najpierw sortowane według tej pary wartości przy użyciu funkcji `sorted()` z odpowiednim kluczem. W kolejnym etapie przeprowadzana jest filtracja - iteracyjnie przeglądane są punkty i usuwane te, które mają identyczny kąt (z dokładnością do `eps`) jak poprzedni punkt, zachowując jedynie najbardziej odległy. Proces budowy otoczki wykorzystuje stos S inicjalizowany dwoma pierwszymi

punktami z posortowanej listy. Dla każdego kolejnego punktu sprawdzana jest jego pozycja względem linii tworzonej przez dwa ostatnie punkty na stosie - jeśli wyznacznik jest niedodatni (punkt nie leży na lewo), element ze stosu jest usuwany.

Realizacja algorytmu Jarvisa rozpoczyna się od wyznaczenia tego samego punktu startowego co w algorytmie Grahama. Algorytm utrzymuje listę H punktów otoczki oraz zmienną `current` wskazującą aktualnie rozpatrywany punkt. W pętli głównej dla każdego kandydata ze zbioru wejściowego obliczany jest wyznacznik macierzy 3×3 dla trójki: punkt aktualny, punkt następny (dotychczas najlepszy kandydat), punkt kandydujący. Gdy wyznacznik jest ujemny (mniejszy od `-eps`), oznacza to, że kandydat leży bardziej na lewo od aktualnie rozpatrywanego punktu następnego - w takiej sytuacji następuje aktualizacja. Dla przypadku współliniowości (wyznacznik w przedziale $[-\text{eps}, \text{eps}]$) porównywane są kwadraty odległości od punktu aktualnego, wybierany jest punkt bardziej oddalony. Pętla kończy się po powrocie do punktu startowego.

Parametr tolerancji numerycznej został ustalony na $\varepsilon = \mathbf{10^{-24}}$. Eksperymentalnie stwierdzono, że wartości większe powodowały niepoprawne traktowanie punktów współliniowych.

Przeprowadzono kompletną walidację obu algorytmów na wszystkich czterech zbiorach testowych, dokumentując graficznie zarówno proces iteracyjny, jak i końcowe rezultaty. Kolejnym etapem była modyfikacja zbiorów w celu systematycznych testów wydajnościowych. Zgromadzone dane empiryczne poddano analizie i wizualizacji wykresowej.

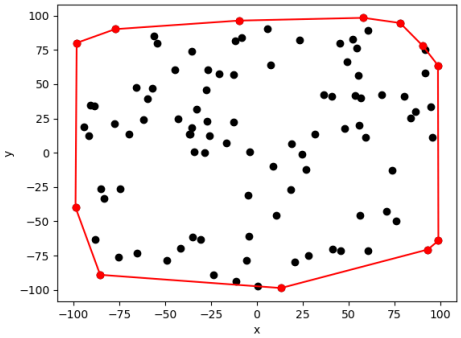
5. Analiza wyników

5.1. Zbiory oryginalne

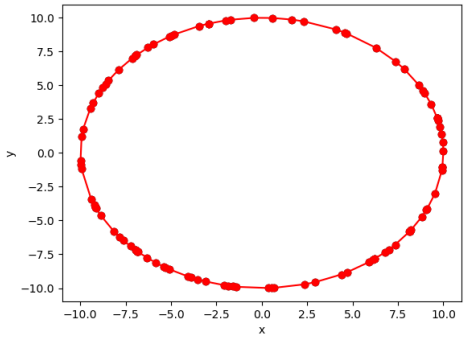
W prezentowanych wizualizacjach zastosowano następującą konwencję kolorystyczną:

- elementy zbioru wejściowego oznaczono **kolorem czerwonym**,
- wierzchołki oraz krawędzie otoczki wypukłej zaznaczono **kolorem czarnym**.

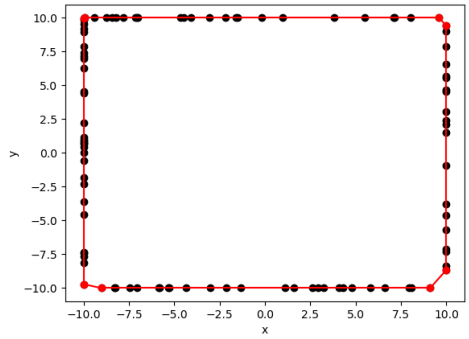
Kompletne sekwencje wierzchołków wyznaczonych otoczek wraz ze szczegółową dokumentacją iteracyjną obu algorytmów znajdują się w notebooku *Jupyter* zawierającym również kod źródłowy implementacji.



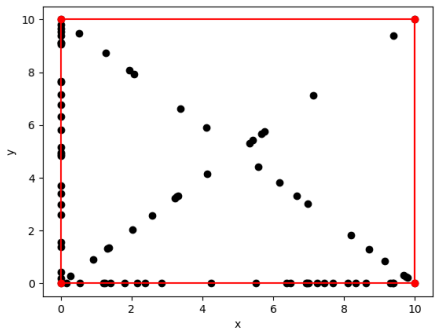
Rysunek 6: Wizualizacja otoczki wypukłej dla zbioru A



Rysunek 7: Wizualizacja otoczki wypukłej dla zbioru B



Rysunek 8: Wizualizacja otoczki wypukłej dla zbioru C



Rysunek 9: Wizualizacja otoczki wypukłej dla zbioru D

Zbiór	**Algorytm Grahama**	**Algorytm Jarvisa**
A	12	12
B	100	100
C	8	8
D	4	4

Tabela 1: Liczba punktów należąca do otoczki wypukłej dla oryginalnych zbiorów

Dane zawarte w Tabeli 1 wraz z pełnymi ciągami punktów dostępnymi w notebooku *Jupyter* jednoznacznie potwierdzają zgodność wyników obu algorytmów. Identyczność zarówno liczności, jak i składu wierzchołkowego otoczek stanowi weryfikację poprawności implementacji dla wszystkich konfiguracji testowych.

Analiza zbioru A na podstawie Rysunku 6 potwierdza geometryczną poprawność otoczki – wszystkie punkty wejściowe znajdują się wewnątrz lub na brzegu wielokąta wypukłego. Rezultat dla zbioru B (Rysunku 7) wykazuje włączenie absolutnie wszystkich punktów do otoczki, co stanowi konsekwencję ich rozmieszczenia na okręgu. Zbiory C (Rysunku 8) oraz D (Rysunku 9) stanowiły potencjalnie problematyczne przypadki testowe ze względu na obecność struktur współliniowych, jednak zastosowane implementacje poprawnie obsłużyły te scenariusze poprzez właściwą strategię eliminacji punktów wewnętrznych. Szczególnie interesujący jest przypadek zbioru C, którego otoczka liczy 8 wierzchołków (Tabeli 1) — jest to wartość maksymalna dla losowej konfiguracji na prostokącie. Jak uwidacznia Rysunku 8, generacja nie wytworzyła żadnego punktu pokrywającego się z wierzchołkami prostokąta (prawdopodobieństwo takiego zdarzenia jest minimalne), co implikuje konieczność włączenia do otoczki dwóch punktów ekstremalnych z każdego boku. Zbiór D prezentuje odmienny scenariusz — jak demonstruje Rysunku 9, obecność czterech wierzchołków kwadratu determinuje kształt otoczki wypukłej, która redukuje się do samej figury bazowej.

5.2. Zbiory zmodyfikowane

5.2.1. Modyfikacja zbiorów testowych

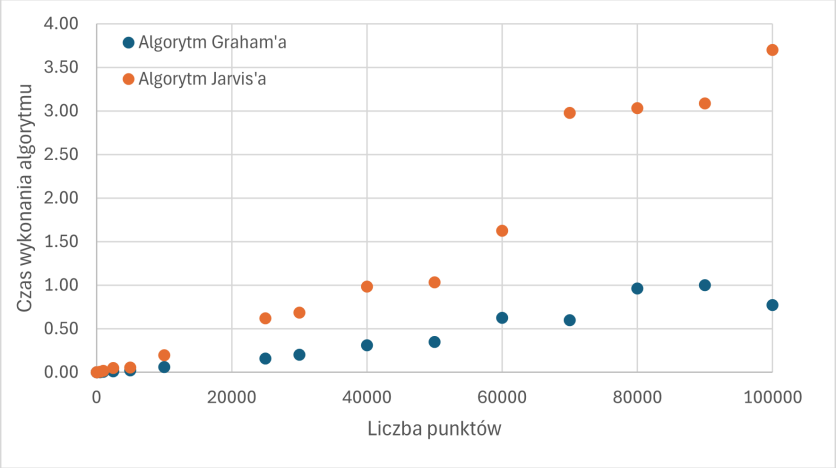
Dla potrzeb eksperymentu opracowano rozszerzone wersje bazowych zbiorów punktów o różnej liczebności.

Podczas pomiarów nie generowano dodatkowych plików z punktami otoczek, aby uniknąć wpływu operacji dyskowych na czas wykonania. Weryfikacja wyników obejmowała również porównanie liczby punktów w otoczce pomiędzy algorytmami, co pozwalało potwierdzić poprawność implementacji.

5.2.2. Wyniki testów dla zmodyfikowanego zbioru A

Wszystkie punkty	Otoczka Graham	Algorytm Grahama	Otoczka Jarvis	Algorytm Jarvisa
50	13	0.001	13	0.001
100	12	0.001	12	0.001
500	17	0.002	17	0.004
1000	20	0.003	20	0.017
2500	22	0.011	22	0.047
5000	20	0.023	20	0.053
10000	27	0.061	27	0.197
25000	31	0.157	31	0.622
30000	28	0.200	28	0.686
40000	28	0.313	28	0.984
50000	21	0.348	21	1.032
60000	27	0.624	27	1.625
70000	34	0.601	34	2.977
80000	28	0.962	28	3.031
90000	34	0.998	34	3.088
100000	34	0.775	34	3.698

Tabela 2: Wyniki pomiaru czasu wykonania dla zmodyfikowanego zbioru A przy różnych licznościach



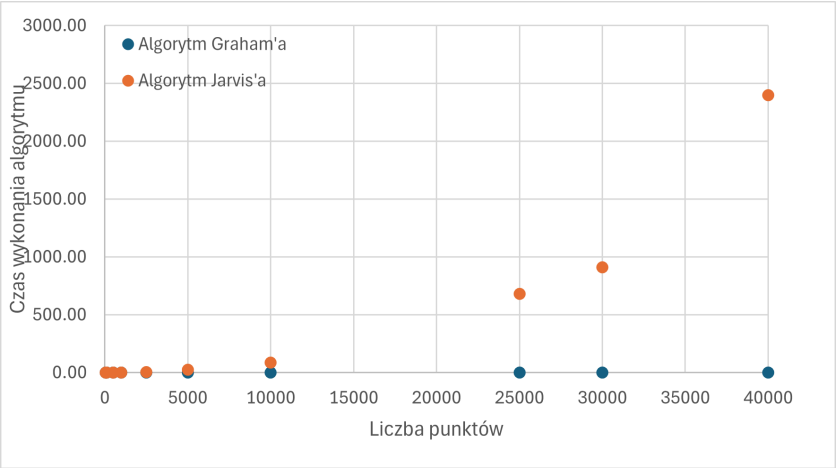
Rysunek 10: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru A

Dane przedstawione w Tabeli 2 i Rysunku 10 wskazują, że dla losowego rozkładu punktów algorytm Grahama utrzymuje stabilny i stosunkowo niski czas działania. Dla niewielkich zbiorów różnice są minimalne, jednak wraz ze wzrostem liczności Jarvis staje się wyraźnie wolniejszy. Przy liczbie punktów powyżej 60 000 obserwuje się wzrost tempa przyrostu czasu obliczeń dla obu metod, co można przypisać rosnącemu wpływowi pamięci podręcznej i zarządzania strukturami danych.

5.2.3. Wyniki testów dla zmodyfikowanego zbioru B

Wszystkie punkty	Otoczka Graham	Algorytm Grahama	Otoczka Jarvis	Algorytm Jarvisa
50	50	0.001	50	0.009
100	100	0.001	100	0.013
500	500	0.005	500	0.971
1000	1000	0.011	1000	1.731
2500	2500	0.036	2500	5.336
5000	5000	0.022	5000	23.788
10000	10000	0.047	10000	85.968
25000	25000	0.156	25000	681.256
30000	30000	0.171	30000	912.791
40000	40000	0.223	40000	2400.529

Tabela 3: Wyniki pomiaru czasu wykonania dla zmodyfikowanego zbioru B przy różnych licznościach



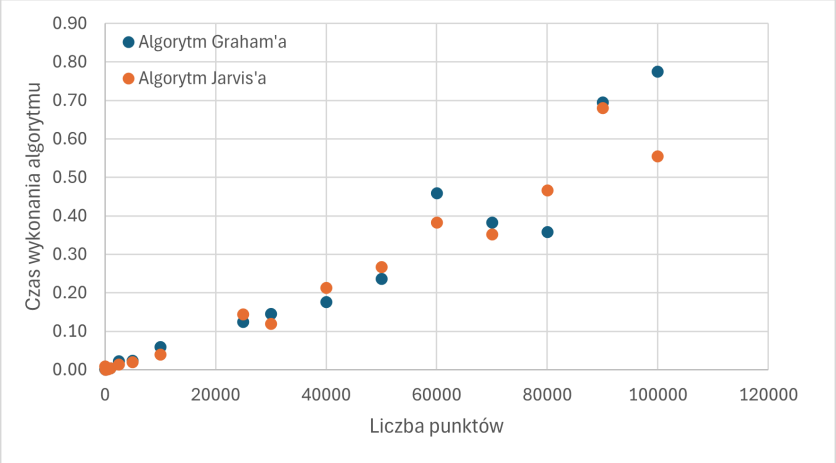
Rysunek 11: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru B

W przypadku zbioru B, którego punkty leżą na obwodzie figury, liczba wierzchołków otoczki jest równa liczbie wszystkich punktów. Oznacza to, że dla algorytmu Jarvisa złożoność czasowa praktycznie osiąga poziom kwadratowy. Na Rysunku 11 zauważalny jest gwałtowny wzrost czasu wykonania — już przy 10 000 punktach przekracza on kilkadziesiąt sekund, a dla 40 000 punktów osiąga wartości rzędu kilku tysięcy. W tym kontekście algorytm Grahama zachowuje przewidywalną, niską złożoność, co czyni go znacznie bardziej efektywnym.

5.2.4. Wyniki testów dla zmodyfikowanego zbioru C

Wszystkie punkty	Otoczka Graham	Algorytm Grahama	Otoczka Jarvis	Algorytm Jarvisa
50	8	0.001	8	0.009
100	8	0.000	8	0.000
500	8	0.002	8	0.002
1000	8	0.004	8	0.003
2500	8	0.022	8	0.014
5000	8	0.024	8	0.019
10000	8	0.059	8	0.040
25000	8	0.125	8	0.144
30000	8	0.145	8	0.119
40000	8	0.176	8	0.213
50000	8	0.236	8	0.267
60000	8	0.459	8	0.383
70000	8	0.382	8	0.352
80000	8	0.358	8	0.466
90000	8	0.694	8	0.680
100000	8	0.775	8	0.555

Tabela 4: Wyniki pomiaru czasu wykonania dla zmodyfikowanego zbioru C przy różnych licznościach



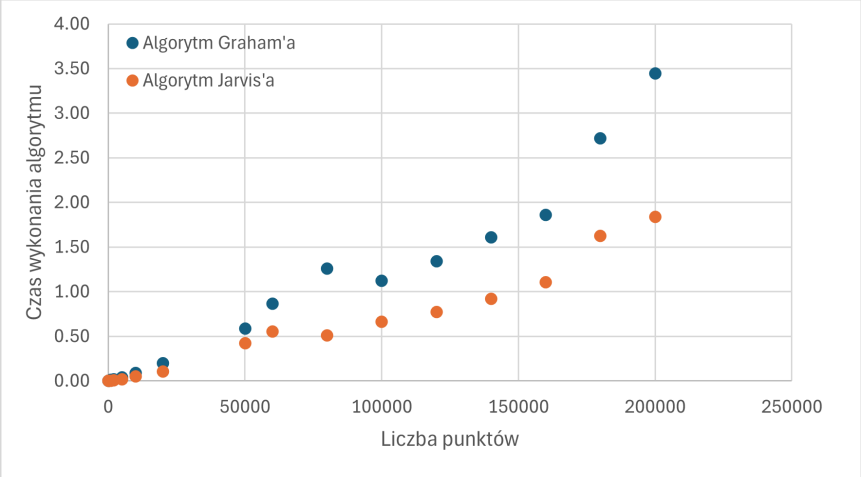
Rysunek 12: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru C

Analiza Tabeli 4 ujawnia, że oba algorytmy zachowują bardzo podobny charakter wzrostu czasu. Liczba wierzchołków otoczki pozostaje stała, co ogranicza różnice w zachowaniu między metodami. Dla większych wartości n obie implementacje wykazują niemal liniowy wzrost czasu obliczeń. Można więc stwierdzić, że w przypadku zbiorów o silnej regularności przestrzennej efektywność Grahama i Jarvisa jest porównywalna.

5.2.5. Wyniki testów dla zmodyfikowanego zbioru D

Punkty (krawędzie)	Punkty (przekątne)	Otoczka Graham	Algorytm Grahama	Otoczka Jarvis	Algorytm Jarvisa
50	50	4	0.001	4	0.000
100	100	4	0.003	4	0.001
500	500	4	0.014	4	0.005
1000	1000	4	0.019	4	0.009
2500	2500	4	0.039	4	0.021
5000	5000	4	0.092	4	0.050
10000	10000	4	0.197	4	0.106
25000	25000	4	0.586	4	0.422
30000	30000	4	0.868	4	0.557
40000	40000	4	1.257	4	0.509
50000	50000	4	1.122	4	0.663
60000	60000	4	1.341	4	0.772
70000	70000	4	1.611	4	0.922
80000	80000	4	1.860	4	1.109
90000	90000	4	2.719	4	1.628
100000	100000	4	3.447	4	1.842

Tabela 5: Wyniki pomiaru czasu wykonania dla zmodyfikowanego zbioru D przy różnych licznościach



Rysunek 13: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru D

Zbiór D składa się z punktów rozmieszczonych w sposób regularny – na krawędziach i przekątnych kwadratu. Jak wynika z Tabeli 5, liczba punktów otoczki pozostaje stała ($k = 4$), natomiast czas obliczeń obu algorytmów rośnie proporcjonalnie do liczby punktów wejściowych. W przeciwieństwie do pozostałych zestawów danych, w tym przypadku **algorytm Jarvisa** osiąga nieco lepsze wyniki czasowe. Różnica jest szczególnie widoczna dla większych zbiorów – przy 100 000 punktach Graham potrzebuje około 3.4 s, podczas gdy Jarvis ok. 1.8 s. Na Rysunku 13 widoczny jest zbliżony rozkład punktów, co sugeruje podobną złożoność obliczeniową, jednak implementacja Jarvisa okazuje się wydajniejsza w kontekście tego typu uporządkowanych danych.

6. Wnioski

Podsumowując przeprowadzone doświadczenia oraz analizę wyników, można stwierdzić, że oba badane algorytmy – Grahama i Jarvisa – prawidłowo realizują zadanie wyznaczania otoczki wypukłej, niezależnie od struktury danych wejściowych. Ich zgodność wyników dla wszystkich testowych zbiorów stanowi potwierdzenie poprawności implementacji oraz stabilności działania.

W kontekście wydajności można zauważyć wyraźne różnice między metodami. Algorytm Grahama, którego kluczowym etapem jest sortowanie punktów według kąta, zachowuje wysoką efektywność w przypadku danych o losowym rozkładzie. Jego złożoność obliczeniowa rzędu $O(n \log n)$ sprawia, że nawet przy dużej liczbie punktów czas przetwarzania rośnie w sposób umiarkowany. W praktyce oznacza to, że jest to metoda preferowana w sytuacjach, gdy nie znamy struktury geometrycznej danych lub gdy liczba punktów jest znaczna.

Z kolei algorytm Jarvisa, oparty na stopniowym „owijaniu" zbioru, okazał się bardziej korzystny przy danych o uporządkowanej lub powtarzalnej strukturze. Dla przypadków, w których liczba punktów należących do otoczki jest niewielka (np. w zbiorach C i D), jego czas działania jest niemal liniowy względem rozmiaru wejścia, co stanowi dużą zaletę w kontekście takich danych. Warto jednak zauważyć, że w sytuacji, gdy większość punktów tworzy otoczkę (jak w zbiorze B), jego efektywność gwałtownie spada – złożoność wzrasta do $O(n^2)$, co przekłada się na bardzo długi czas wykonania.

Przeprowadzone testy wskazują, że:
- algorytm Grahama lepiej sprawdza się w warunkach losowych i dla dużych zbiorów punktów,
- algorytm Jarvisa zyskuje przewagę w przypadkach, gdy liczba punktów otoczki k jest ograniczona lub dane mają regularny układ geometryczny.

Dodatkowo zauważono, że wzrost liczby punktów wejściowych wpływa na obie metody w sposób zbliżony — czasy obliczeń rosną proporcjonalnie do rozmiaru danych, przy czym różnice w tempie wzrostu zależą od konkretnej konfiguracji przestrzennej punktów. Efektywność algorytmów była również częściowo zależna od czynników implementacyjnych, takich jak zarządzanie pamięcią czy dokładność obliczeń zmiennoprzecinkowych.

Ostatecznie można stwierdzić, że wybór algorytmu powinien być uzależniony od natury problemu:
- dla zbiorów o dużej losowości lub nieznanym układzie – optymalny będzie **algorytm Grahama**,
- dla zbiorów o silnej regularności lub niewielkiej liczbie punktów brzegowych – bardziej efektywny okaże się **algorytm Jarvisa**.

Wnioskiem końcowym jest więc to, że oba podejścia stanowią wartościowe narzędzia analizy geometrycznej, a ich praktyczne zastosowanie powinno być dobierane kontekstowo – w zależności od charakteru danych, wymagań czasowych oraz oczekiwanej precyzji obliczeń.