Algorytmy Geometryczne, Ćwiczenie 2 – Otoczka Wypukła

Maciej Kmąk

Listopad 2024

1 Cel ćwiczenia

Celem ćwiczenia było zapoznanie się z algorytmami wyznaczania otoczki wypukłej oraz porównanie ich efektywności. Ćwiczenie skupiało się na implementacji dwóch metod wyznaczania otoczki wypukłej: algorytmu Grahama oraz algorytmu Jarvisa, a także na przeprowadzeniu eksperymentów z różnymi zestawami punktów na płaszczyźnie. Analiza działania obu algorytmów na różnych zbiorach danych pozwala lepiej zrozumieć ich charakterystykę, wydajność czasową oraz skuteczność w radzeniu sobie z różnorodnymi układami punktów.

2 Dane techniczne

Program (w pliku „kmak_kod_2.ipynb") jest napisany w języku Python w środowisku Jupyter Notebook. Wykresy punktów i wizualizacje w formacie GIF powstały dzięki narzędziu przygotowanemu przez koło naukowe Bit, które korzysta z biblioteki `matplotlib`. Do generowania punktów została użyta funkcja z biblioteki `NumPy - random.uniform()`. Pomocniczo do sortowania punktów użyto funkcji `cmp_to_key()` z modułu `functools`. Przedstawiony program został wyegzekwowany na komputerze z procesorem Intel® Core™ i5-1235U z systemem operacyjnym Windows 11 Pro.

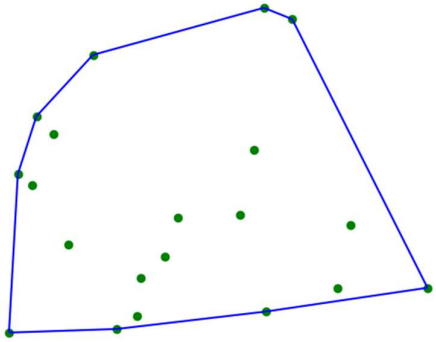
3 Wstęp teoretyczny

3.1 Definicja otoczki wypukłej

Otoczką wypukłą $\boldsymbol{CH(S)}$ (niepustego zbioru punktów $\boldsymbol{S}$) nazwiemy najmniejszy zbiór wypukły zawierający wszystkie punkty zbioru S.

Otoczka wypukła punktów na płaszczyźnie jest najmniejszym wielokątem wypukłym zawierającym zbiór S.

Według konwencji listę punktów otoczki z S podaje się w porządku przeciwnym do ruchu wskazówek zegara. W przypadku punktów współliniowych, do otoczki należy jedynie najbardziej oddalony od poprzedniego (zgodnie z konwencją) punktu otoczki.



Rysunek 3.1. Przykładowa otoczka wypukła

3.2 Algorytmy znajdywania otoczki wypukłej

3.2.1 Algorytm Grahama

Algorytm Grahama działa na zasadzie usuwania wierzchołków wklęsłych, przy jednoczesnym utrzymywaniu stosu punktów, które mogą stać się częścią otoczki. Opis kroków algorytmu wygląda następująco:

1. W zbiorze S wybieramy punkt p_0 o najmniejszej współrzędnej y. Jeżeli jest kilka takich punktów, to wybieramy ten z nich, który ma najmniejszą współrzędną x.
2. Sortujemy pozostałe punkty ze względu na kąt, jaki tworzy wektor (p_0,p) z dodatnim kierunkiem osi x. Jeśli kilka punktów tworzy ten sam kąt, usuwamy wszystkie z wyjątkiem najbardziej oddalonego od p_0 . Niech uzyskanym ciągiem będzie $p_1, p_2,...p_m$.
3. Do początkowo pustego stosu s wkładamy punkty p_0, p_1 , p_2. t – indeks stosu; i ← 3
4. while i < m do

   if p_i leży na lewo od prostej (p_{t-1} , p_t))

   then push(p_i), i ← i+1

   else pop(s)

Cały proces przebiega w kilku krokach, które umożliwiają skuteczne wyznaczenie punktów otoczki wypukłej. Złożoność algorytmu Grahama zależy od głównych operacji:

- Znajdowanie punktu początkowego $O(n)$
- Sortowanie punktów według kąta względem punktu początkowego $O(n \cdot \log(n))$
- Iteracja po punktach i operacje na stosie są realizowane w czasie $O(n)$

Dominującą operacją jest sortowanie punktów, dlatego całkowita złożoność algorytmu Grahama wynosi $O(n \cdot \log(n))$ gdzie n to liczba punktów w zbiorze S.

3.2.2 Algorytm Jarvisa

Algorytm Jarvisa, znany również jako algorytm owijania prezentu, działa na zasadzie podejmowania lokalnie optymalnych decyzji, wybierając w każdym kroku punkt, który tworzy najmniejszy kąt z aktualną krawędzią otoczki, bez uwzględniania przyszłych wyborów. Proces ten prowadzi do znalezienia globalnie optymalnego rozwiązania, czyli otoczki wypukłej. Opis algorytmu wygląda następująco:

znajdź punkt i_0 z S o najmniejszej współrzędnej y; i ← i_0
repeat
for j ≠ i do
znajdź punkt, dla którego kąt liczony przeciwnie do ruchu wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy
niech k będzie indeksem punktu z najmniejszym kątem
zwróć (p_i, p_k) jako krawędź otoczki
i ← k
until i = i_0

Złożoność algorytmu Jarvisa zależy od głównych operacji:

- Wybór punktu początkowego $O(n)$
- Dla każdego punktu na otoczce należy znaleźć punkt, którego kąt względem poprzednich punktów na otoczce jest najmniejszy. Zatem dla k punktów na otoczce złożoność tej operacji wynosi $O(n)$ w każdej k iteracji co daje nam złożoność $O(k \cdot n)$

Dominującą operacją jest znajdywanie kolejnego punktu na otoczce, dlatego całkowita złożoność algorytmu Jarvisa wynosi $O(k \cdot n)$ gdzie n to liczba punktów w zbiorze S, a k to liczba punktów w otoczce $CH(S)$. Należy zauważyć, że dla zbiorów gdzie $k = O(n)$ tzn. liczba punktów otoczki jest proporcjonalna do liczby punktów zbioru S, złożoność algorytmu Jarvisa wynosi $O(n^2)$.

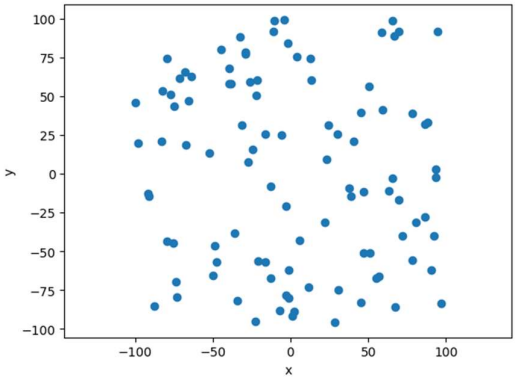
4 Zbiory i przebieg doświadczenia

4.1 Generowane zbiory

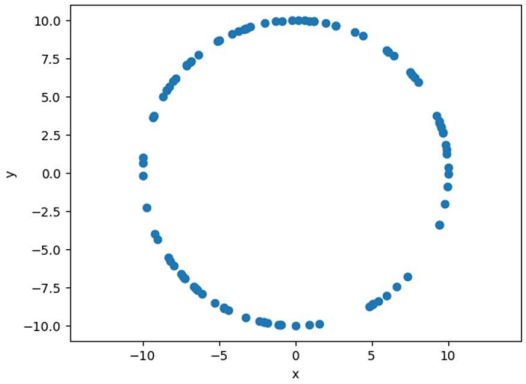
A. 100 losowo wygenerowanych punktów o współrzędnych z przedziału [−100, 100] dalej zwany zbiorem A
B. 100 losowo wygenerowanych punktów leżących na okręgu o środku (0, 0) i promieniu R = 10 dalej zwany zbiorem B
C. 100 losowo wygenerowanych punktów na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10) dalej zwany zbiorem C
D. zbiór zawierający wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo w sposób następujący: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu dalej zwany zbiorem D

W dalszej części (6. Modyfikacja zbiorów) zmodyfikowano te zbiory w celu przeprowadzenia testów wydajnościowych i porównania efektywności algorytmów.

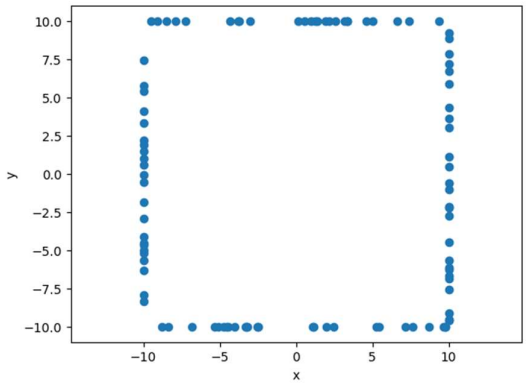
4.2 Wizualizacja zbiorów



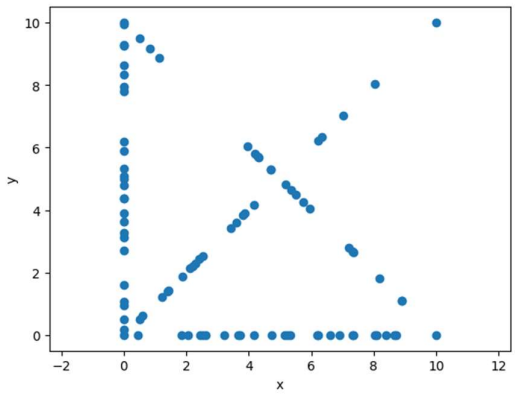
Wykres 4.2.1 Zbiór A



Wykres 4.2.1 Zbiór B



Wykres 4.2.3 Zbiór C



Wykres 4.2.4 Zbiór D

4.3 Przebieg doświadczenia

Po wygenerowaniu zbiorów zaimplementowano algorytmy: Grahama i Jarvisa.

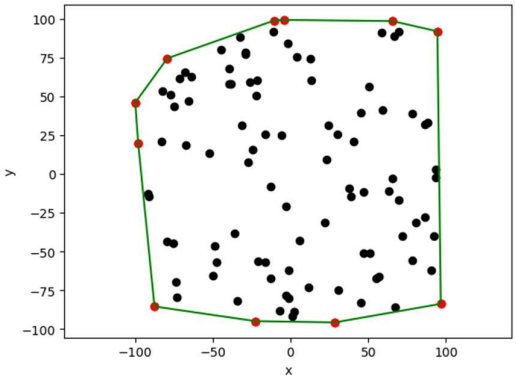
Do implementacji algorytmu Grahama, w sortowaniu punktów wykorzystano funkcję `cmp_to_key`() oraz funkcję `orient()`, obliczającą wyznacznik macierzy (przez co nie musieliśmy korzystać z funkcji trygonometrycznych). Następnie usunięto wszystkie „duplikaty kątowe" pozostawiając jedynie punkty najbardziej oddalone od punktu startowego. Gdy punkty są współliniowe, możemy użyć metryki Manhattan, ponieważ dla takich punktów jednoznacznie porządkuje je względem odległości od punktu bazowego (`head`), mierząc sumę różnic współrzędnych, co jest odpowiednie w przypadku punktów na tej samej linii. Zastosowanie metryki Manhattan okazało się szybsze od metryki euklidesowej. Dla testów przygotowanych przez koło naukowe Bit była ona około 32% szybsza od standardowego liczenia odległości (0.391 s vs. 0.578 s).

Do implementacji algorytmu Jarvisa także użyto funkcji `orient`, co umożliwiło optymalne wyszukiwanie kolejnych punktów otoczki. Szukaliśmy przy jej pomocy punktu, dla którego nie istnieje już żaden punkt znajdujący się po prawej stronie względem krawędzi otoczki utworzonej przez ten punkt. Podobnie jak wcześniej, w przypadku punktów współliniowych wybieraliśmy najbardziej odległy.

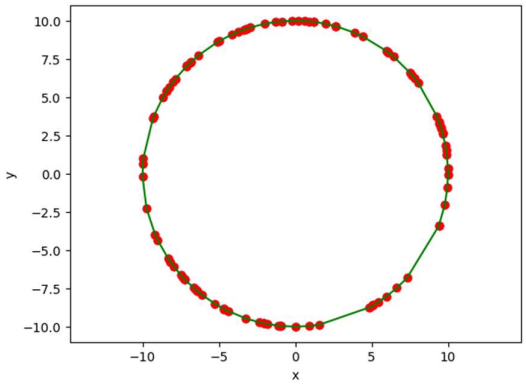
Następnie dla prezentowanych zbiorów (4.2 Wizualizacja zbiorów) wyznaczono otoczki wypukłe, wygenerowano także pliki .gif, które wizualizowały proces powstawania otoczki. Nie znalazły one się w sprawozdaniu z powodu ograniczeń formatu .pdf (są dostępne w folderze kmak_gifs.zip wysłanym razem ze sprawozdaniem).

5 Analiza wyników doświadczenia

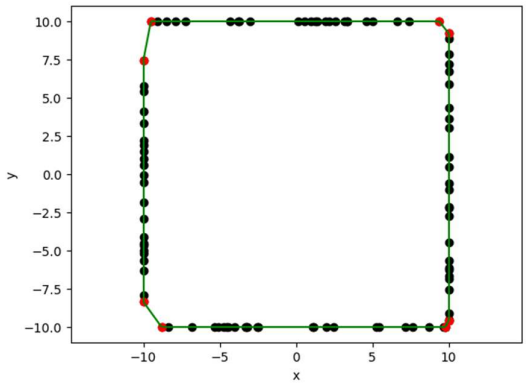
Przedstawione wykresy zostały pokolorowane według konwencji: na czerwono zaznaczono punkty otoczki $CH(S)$, na zielono zaznaczono odcinki łączące kolejne punkty otoczki $CH(S)$, na **czarno** zaznaczono punkty zbioru S nienależące do otoczki $CH(S)$.



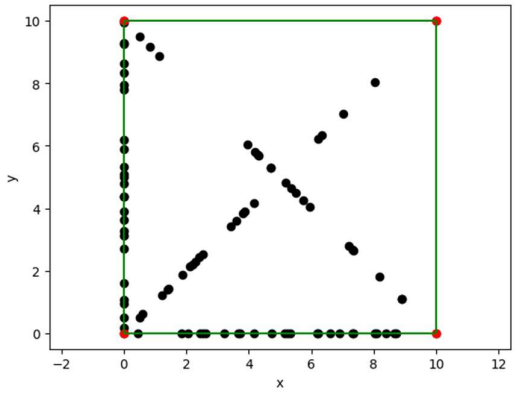
Wykres 5.1 Otoczka wypukła zbioru A



Wykres 5.2 Otoczka wypukła zbioru B



Wykres 5.3 Otoczka wypukła zbioru C



Wykres 5.4 Otoczka wypukła zbioru D

Tabela 5.1 Wyniki działania algorytmów dla oryginalnych zbiorów

Zbiór	Liczba punktów otoczki	
	Algorytm Grahama	Algorytm Jarvisa
A	13	13
B	100	100
C	8	8
D	4	4

Wyniki działania algorytmów przedstawione zostały w tabeli (Tabela 5.1). Możemy zauważyć, że dla każdego z algorytmów liczba punktów otoczek jest taka sama (takie same są też zwracane punkty). Oznacza to, że obydwa algorytmy działają poprawnie.

Możemy zauważyć, że dla zbioru A otoczka jest poprawna, zawierając wszystkie punkty wewnątrz (Wykres 5.1). W przypadku zbioru B (Wykres 5.2) każdy punkt okręgu należy do otoczki. Z kolei dla zbiorów C i D (Rysunki Wykres 5.3 i 5.4) użyte implementacje skutecznie poradziły sobie z punktami współliniowymi: otoczka zbioru C obejmuje osiem punktów na bokach prostokąta, natomiast dla zbioru D otoczka składa się z czterech wierzchołków kwadratu.

6 Modyfikacja zbiorów

W celu sprawdzenia efektywności algorytmów (przeprowadzenia testów wydajnościowych) zmodyfikowano zbiory (4.1 Generowane zbiory):

Przy pomocy tych samych funkcji generujących wygenerowano zbiory o liczności kolejno:

- [1000,2000,5000,10000,20000,50000,75000,100000,150000,250000,500000] dla zbioru A,
- [10,100,500,1000,2000,5000,7500,10000,15000,25000,50000] dla zbioru B,
- [1000,2000,5000,10000,20000,50000,75000,100000,200000,500000] dla zbiorów C i D.

Zmodyfikowano także obszar występowania punktów:

- Zbiór A zawiera losowo wygenerowane punkty o współrzędnych z przedziału [-5000, 5000],
- Zbiór B zawiera losowo wygenerowane punkty leżące na okręgu o środku (200, 200) i promieniu R = 2500,
- Zbiór C zawiera losowo wygenerowane punkty na bokach prostokąta o wierzchołkach (-500, -300), (100, -300), (100, 100), (-500, 100),
- Zbiór D to zbiór zawierający wierzchołki kwadratu (0,0), (500,0), (500,500), (0,500) oraz punkty wygenerowane losowo w sposób następujący: po $(0{,}3 \cdot n - 1)$ punktów na dwóch bokach kwadratu leżących na osiach i po $(0{,}2 \cdot n - 1)$ punktów na przekątnych kwadratu, gdzie n to liczba punktów w zbiorze.

7 Testy wydajnościowe

W ramach testów wydajnościowych dokonano modyfikacji zbiorów danych w celu dokładnego zbadania efektywności algorytmów przy różnych rozmiarach i układach punktów. Skorzystano z wcześniejszych funkcji generujących.

Podczas testów, dla każdego z tych zbiorów, mierzono czas działania algorytmu, aby zidentyfikować jego wydajność przy rosnącej liczbie punktów. Równocześnie sprawdzano, czy liczba punktów w otoczkach zwracanych przez oba algorytmy jest zgodna, co umożliwiło weryfikację poprawności wyników. Wyniki testów przedstawiono w formie tabeli, gdzie zapisano czasy obliczeń dla poszczególnych zbiorów i liczności, a także liczbę punktów otoczki dla każdego algorytmu.

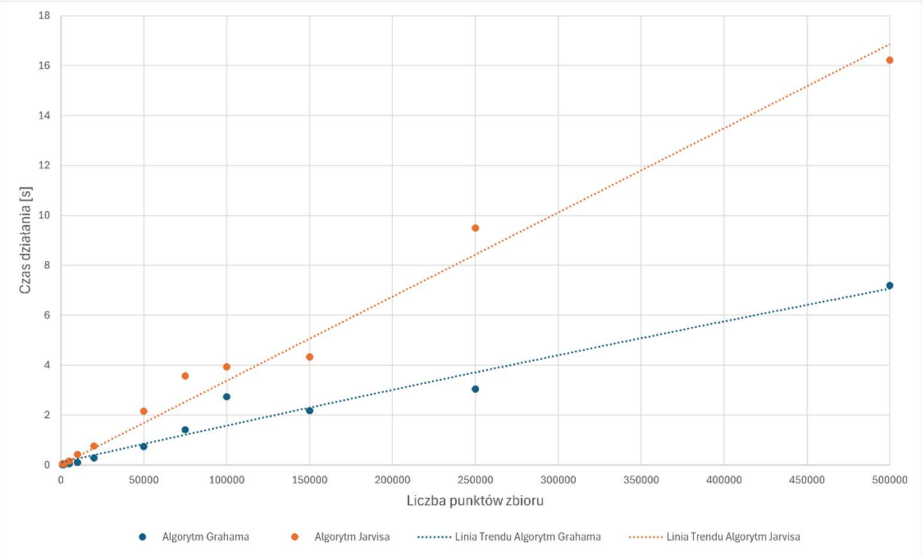
Na podstawie zebranych danych wykonano wykresy z naniesionymi liniami trendu dla obu algorytmów, co pozwoliło na analizę ich wydajności w zależności od typu danych – zbiór typu A, B, C lub D. Dzięki tym wykresom możliwe było zidentyfikowanie różnic w działaniu algorytmów i ocenienie ich efektywności w zależności od charakteru oraz rozkładu punktów w każdym zbiorze.

Na następnych stronach przedstawione zostały wyniki oraz analizy testów wydajnościowych dla poszczególnych typów zbiorów punktów. Każdy z typów zbiorów został dokładnie przetestowany pod kątem czasu działania algorytmów Grahama i Jarvisa, co pozwoliło na ocenę efektywności obu podejść w różnych scenariuszach. Wnioski z przeprowadzonych testów są szczegółowo omówione, a wyniki porównawcze ukazują, jak struktura zbioru punktów wpływa na wybór optymalnego algorytmu do obliczenia otoczki wypukłej.

7.1 Wyniki testów wydajnościowych dla zmodyfikowanych zbiorów typu A.

Tabela 7.1.1 Wyniki pomiaru czasów wykonywania dla zbiorów typu A przy różnych licznościach

Dane zbioru		Czas działania [s]	
Liczba punktów zbioru	Liczba punktów w otoczce	Algorytm Grahama	Algorytm Jarvisa
1000	21	0.0125	0.0420
2000	16	0.0180	0.0555
5000	22	0.0586	0.1561
10000	32	0.1005	0.4242
20000	30	0.2855	0.7588
50000	29	0.7405	2.1487
75000	29	1.4163	3.5679
100000	32	2.7319	3.9278
150000	28	2.1720	4.3276
250000	36	3.0393	9.4911
500000	34	7.1935	16.2224



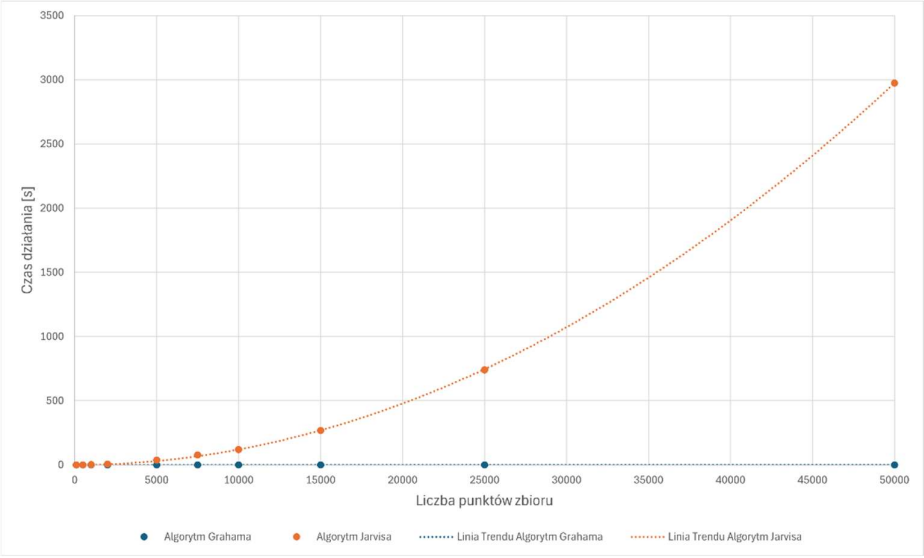
Wykres 7.1.1 Porównanie czasu wyznaczania otoczki wypukłej przez algorytmy dla zbiorów typu A z naniesioną linią trendu

Dzięki wynikom widocznym w tabeli (Tabela 7.1.1) możemy zauważyć, że w przypadku zbiorów typu A algorytm Grahama jest efektywniejszy od algorytmu Jarvisa. Na wykresie (Wykres 7.1.1) z zaznaczonymi liniami trendu możemy zauważyć, że linia trendu algorytmu Grahama choć przyjmuje kształt liniowo-logarytmiczny, jest bardzo spłaszczona i zbliżona do linii prostej. Oznacza to, że przy rozważanych rozmiarach zbiorów punktów, wpływ składnika logarytmicznego na czas obliczeń jest stosunkowo mały. Dla zbiorów typu A algorytm Grahama jest średnio o około 61% szybszy. W tabeli (Tabela 7.1.1) wyraźnie widać, że w przypadku wzrostu liczby punktów algorytm Grahama nad algorytmem Jarvisa utrzymuje przewagę czasową, a różnica w czasie działania algorytmów staje się coraz bardziej wyraźna.

7.2 Wyniki testów wydajnościowych dla zmodyfikowanych zbiorów typu B.

Tabela 7.2.1 Wyniki pomiaru czasów wykonywania dla zbiorów typu B przy różnych licznościach

Dane zbioru		Czas działania [s]	
Liczba punktów zbioru	Liczba punktów w otoczce	Algorytm Grahama	Algorytm Jarvisa
100	100	0.0010	0.0171
500	500	0.0040	0.4241
1000	1000	0.0111	2.0235
2000	2000	0.0235	6.3473
5000	5000	0.0466	38.0503
7500	7500	0.0772	77.8100
10000	10000	0.0817	119.6934
15000	15000	0.1125	267.2951
25000	25000	0.2154	740.1650
50000	50000	0.4684	2974.4182



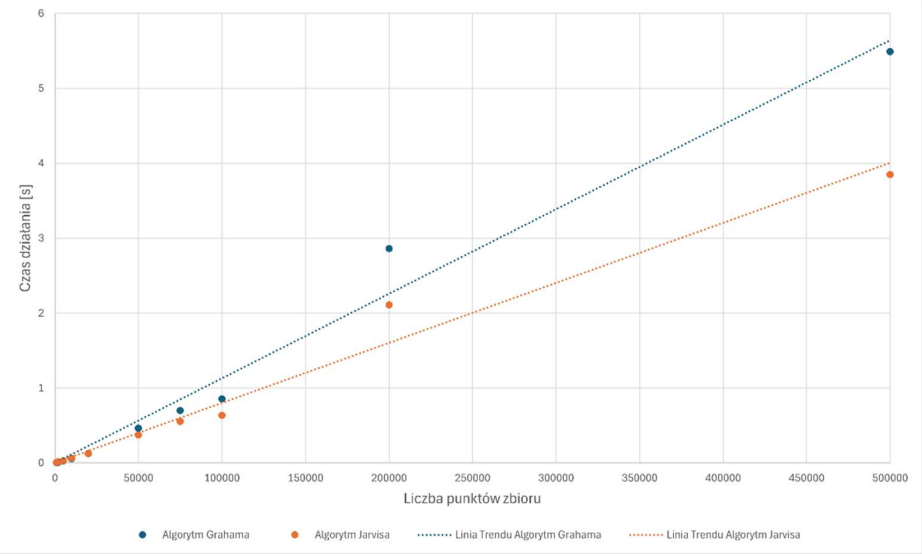
Wykres 7.2.1 Porównanie czasu wyznaczania otoczki wypukłej przez algorytmy dla zbiorów typu B z naniesioną linią trendu

Wyniki widoczne w tabeli (Tabela 7.2.1) pokazują, że w przypadku zbiorów typu B algorytm Grahama również wykazuje większą efektywność w porównaniu do algorytmu Jarvisa. Dla zbiorów typu B algorytm Grahama jest średnio ponad 99 % szybszy. Na wykresie (Wykres 7.2.1) z zaznaczonymi liniami trendu możemy zauważyć, że dla algorytmu Jarvisa linia trendu przyjmuje postać funkcji kwadratowej, co wskazuje na znaczną zależność czasu działania od liczby punktów w zbiorze. Dla $k = O(n)$ tzn. liczba punktów otoczki jest proporcjonalna do liczby punktów zbioru S, teoretyczna złożoność algorytmu Jarvisa wynosi $O(n^2)$, co znajduje swoje potwierdzenie w przeprowadzonych testach. Oznacza to, że w przypadku zbiorów typu B (gdzie wszystkie punkty zbioru należą do otoczki), lub podobnych do B (w sensie liczności punktów otoczki, czyli gdzie k jest rzędu $O(n)$) czas potrzebny do obliczenia otoczki wypukłej rośnie proporcjonalnie do kwadratu liczby punktów w zbiorze.

7.3 Wyniki testów wydajnościowych dla zmodyfikowanych zbiorów typu C.

Tabela 7.3.1 Wyniki pomiaru czasów wykonywania dla zbiorów typu C przy różnych licznościach

Dane zbioru		Czas działania [s]	
Liczba punktów zbioru	Liczba punktów w otoczce	Algorytm Grahama	Algorytm Jarvisa
1000	8	0.0051	0.0077
2000	8	0.0080	0.0149
5000	8	0.0270	0.0321
10000	8	0.0566	0.0666
20000	8	0.1277	0.1267
50000	8	0.4630	0.3711
75000	8	0.7002	0.5534
100000	8	0.8560	0.6341
200000	8	2.8603	2.1077
500000	8	5.4914	3.8498



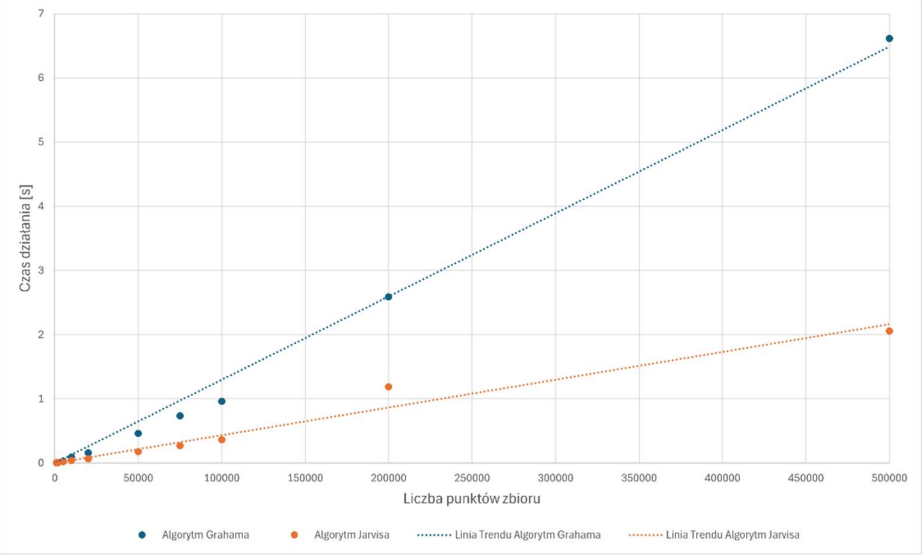
Wykres 7.3.1 Porównanie czasu wyznaczania otoczki wypukłej przez algorytmy dla zbiorów typu C z naniesioną linią trendu

Wyniki widoczne w tabeli (Tabela 7.3.1) pokazują, że w przypadku zbiorów typu C, algorytm Jarvisa wykazuje lepszą efektywność niż algorytm Grahama. Wyniki testów wskazują, że dla zbiorów typu C algorytm Jarvisa, dla większych liczności danych (tj. dla $n > 50000$) jest średnio około 25% szybszy od algorytmu Grahama. Na wykresie (Wykres 7.3.1) z zaznaczonymi liniami trendu widać, że czas działania algorytmu Jarvisa rośnie w sposób liniowy, co wskazuje na stabilność jego wydajności, gdy liczba punktów w zbiorze rośnie, ale liczba punktów otoczki ($k = O(1)$, w naszym przypadku $k = 8$) pozostaje stała. Dzięki temu złożoność algorytmu Jarvisa wynosi $O(n)$. Przez to algorytm Jarvisa w porównaniu do algorytmu Grahama, o złożoności $O(n \cdot \log(n))$, okazuje się być lepszym wyborem w przypadku zbiorów o ograniczonej liczbie punktów w otoczce $k = O(1)$.

7.4 Wyniki testów wydajnościowych dla zmodyfikowanych zbiorów typu D.

Tabela 7.4.1 Wyniki pomiaru czasów wykonywania dla zbiorów typu D przy różnych licznościach

Dane zbioru		Czas działania [s]	
Liczba punktów zbioru	Liczba punktów w otoczce	Algorytm Grahama	Algorytm Jarvisa
1000	4	0.0053	0.0029
2000	4	0.0100	0.0070
5000	4	0.0319	0.0237
10000	4	0.0934	0.0376
20000	4	0.1596	0.0643
50000	4	0.4609	0.1771
75000	4	0.7339	0.2716
100000	4	0.9610	0.3646
200000	4	2.5877	1.1874
500000	4	6.6147	2.0583



Wykres 7.4.1 Porównanie czasu wyznaczania otoczki wypukłej przez algorytmy dla zbiorów typu D z naniesioną linią trendu

Wyniki widoczne w tabeli (Tabela 7.4.1) pokazują, że w przypadku zbiorów typu D różnice w wydajności między algorytmem Grahama a algorytmem Jarvisa są jeszcze bardziej wyraźne. Zbiory typu D charakteryzują się tym, że liczba punktów w otoczce ($k = 4$) jest stosunkowo mała, co sprzyja efektywności algorytmu Jarvisa. Testy wykazały, że dla zbiorów typu D, algorytm Jarvisa, przy większych licznościach punktów (tj. $n > 10000$) jest średnio nawet o około 61% szybszy od algorytmu Grahama. Różnice w czasie wykonania pomiędzy algorytmami stają się coraz bardziej wyraźne w miarę wzrostu rozmiaru zbioru, co dodatkowo potwierdza teoretyczną przewagę złożoności algorytmu Jarvisa. Na wykresie (Wykres 7.4.1) z zaznaczonymi liniami trendu widać, że czas działania algorytmu Jarvisa rośnie w sposób liniowy, co wskazuje na jego stabilność w miarę wzrostu liczby punktów w zbiorze, przy stałej liczbie punktów w otoczce ($k = 4$). Nawet biorąc pod uwagę, że algorytm Grahama również eliminuje „duplikaty kątowe" na bokach oraz wzdłuż jednej z przekątnych, algorytm Jarvisa wykazuje lepszą efektywność w zbiorach, w których liczba punktów w otoczce jest niewielka. Zatem, dla zbiorów gdzie $k = O(1)$, algorytm Jarvisa jest zdecydowanie lepszym wyborem.

8 Wnioski

Na podstawie przeprowadzonych testów i analizy działania algorytmów można sformułować następujące wnioski:

- Zarówno algorytm Grahama, jak i algorytm Jarvisa prawidłowo wyznaczyły otoczki wypukłe we wszystkich analizowanych zbiorach punktów, co potwierdza ich poprawność i efektywność w różnych scenariuszach testowych.
- Algorytm Grahama wykazał się większą wydajnością dla losowych zbiorów punktów, takich jak zbiór typu A, zwłaszcza przy dużych rozmiarach danych. Dzięki złożoności czasowej $O(n \cdot \log(n))$ jest bardziej efektywny niż algorytm Jarvisa w przypadku zbiorów, gdzie liczba punktów otoczki k jest proporcjonalna do liczby punktów n. Testy potwierdziły tę przewagę, szczególnie przy dużych zbiorach.
- Wyniki testów wykazały, że liczba punktów tworzących otoczkę k ma istotny wpływ na czas działania algorytmu Jarvisa. Dla zbiorów, gdzie wszystkie punkty należą do otoczki (jak w zbiorze typu B), złożoność algorytmu Jarvisa wzrasta do $O(n^2)$, co skutkuje znacznym wydłużeniem czasu obliczeń. Testy potwierdziły tę teoretyczną złożoność, co było widoczne w wyższych czasach działania dla takiego typu danych. Potwierdził to także wykres linii trendu (Wykres 7.2.1), który został dopasowany przez program Microsoft Excel jako wykres funkcji wielomianowej stopnia = 2, a więc faktycznie funkcji kwadratowej $f(x) = x^2$.
- Algorytm Jarvisa okazał się bardziej wydajny w zbiorach o ograniczonej liczbie punktów otoczki k, takich jak zbiory C i D, zawierających stosunkowo dużo punktów współliniowych. Przy stałej liczbie punktów otoczki k rzędu $O(1)$, jego złożoność wynosząca $O(k \cdot n)$ staje się złożonością liniową $O(n)$, co zostało potwierdzone w testach. Przewaga algorytmu Jarvisa nad algorytmem Grahama szczególnie widoczna jest w przypadku zbiorów o małej liczbie punktów otoczki i relatywnie dużej liczbie punktów współliniowych.

9 Podsumowanie zagadnienia

Podsumowując wyniki przeprowadzonych testów, potwierdzono teoretyczne złożoności czasowe obu algorytmów, co jednoznacznie wskazuje na ich różnice w wydajności w zależności od charakterystyki zbioru punktów.

Zgodnie z przeprowadzonymi analizami, wybór odpowiedniego algorytmu do obliczenia otoczki wypukłej powinien być uzależniony od struktury zbioru punktów oraz jego specyfiki:

- Algorytm Grahama jest bardziej uniwersalny i efektywny dla zbiorów o losowym rozkładzie punktów, ponieważ jego złożoność czasowa wynosząca $O(n \cdot \log(n))$ pozwala na szybkie obliczenia w przypadkach, gdy punkty są rozproszone w przestrzeni. Dzięki tej złożoności, algorytm Grahama radzi sobie dobrze w różnych scenariuszach, niezależnie od tego, czy punkty są współliniowe, czy tworzą bardziej skomplikowane struktury.
- Algorytm Jarvisa lepiej sprawdza się, gdy większość punktów jest współliniowa lub liczba punktów otoczki jest ograniczona, co sprawia, że jego złożoność czasowa $O(k \cdot n)$ staje się praktycznie liniowa przy małej liczbie punktów w otoczce. W takich przypadkach, kiedy punkty leżą wzdłuż jednej prostej, algorytm Jarvisa szybko znajduje otoczkę wypukłą. Dzięki temu jest szczególnie efektywny, gdy struktura zbioru punktów sprzyja minimalizacji liczby operacji.