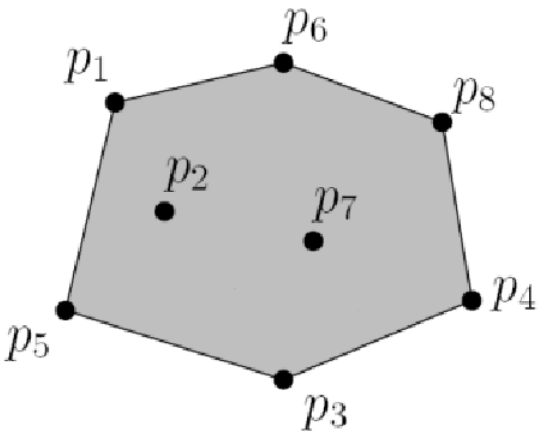
Mateusz Bartnicki, grupa nr 6 czwartek 13:00-14:30

Algorytmy geometryczne, sprawozdanie – ćwiczenie 2 - 19.10.2023 r.

1. Opis ćwiczenia

Głównym celem ćwiczenia było zaimplementowanie algorytmów Grahama i Jarvisa, które służą do wyznaczania otoczki wypukłej zbioru punktów, a następnie porównanie ich działania.

Rysunek 1 – Otoczka wypukła na zbiorze punktów p1, p2, …, p8



2. Dane techniczne

Ćwiczenie zostało wykonane na systemie operacyjnym Windows na procesorze Intel Core i5-11400H 2.70GHz. Do jego wykonania posłużyłem się Jupyter Notebook. Program został napisany przy użyciu języka Python, wykorzystując biblioteki takie jak numpy, pandas, time oraz functools.

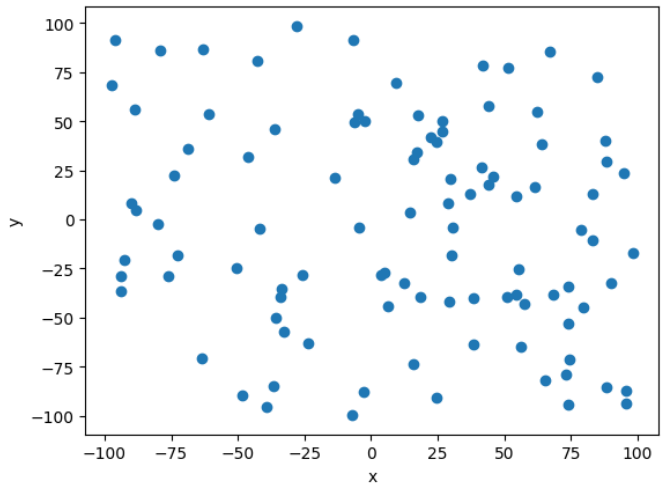
3. Przebieg ćwiczenia

1) a) Stworzenie zbioru 100 punktów w przestrzeni 2D z przedziału $x, y \in [-100, 100]^2$,
   b) Stworzenie zbioru 100 punktów leżących na okręgu o środku w punkcie 0 = (0,0) i promieniu R = 10,
   c) Stworzenie zbioru 100 punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10),
   d) Stworzenie zbioru punktów zawartych na wierzchołkach kwadratu o bokach (-10, 10), (-10,-10), (10,-10), (10,10), oraz punktów generowanych w następujący sposób: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu.
2) Przygotowanie zmodyfikowanych zestawów danych (w moim przypadku zastosowane zostały takie same parametry, zmieniona została tylko ilość punktów) w celu porównania czasu wykonywania algorytmów.
3) Napisanie algorytmów Grahama i Jarvisa oraz zwizualizowanie przebiegu ich działania.

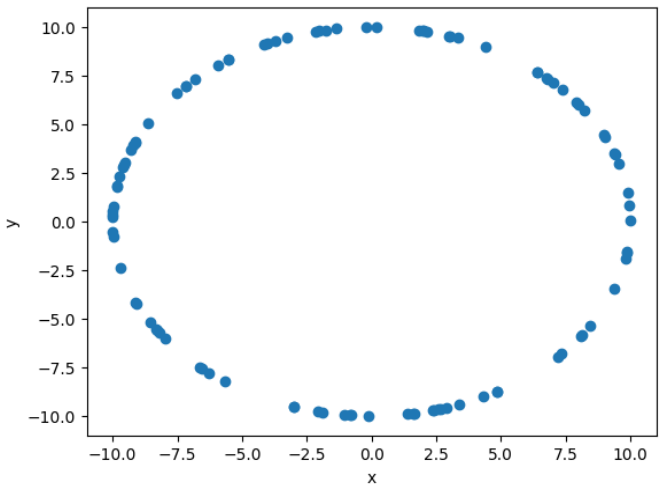
Do tworzenia zbiorów punktów wykorzystana została funkcja random.uniform z biblioteki numpy.

Wizualizacja zestawów danych

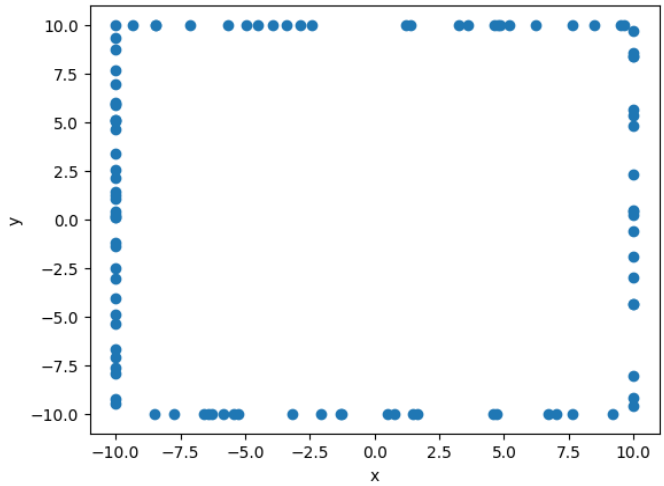
Wykres 1 - Zestaw danych 1)



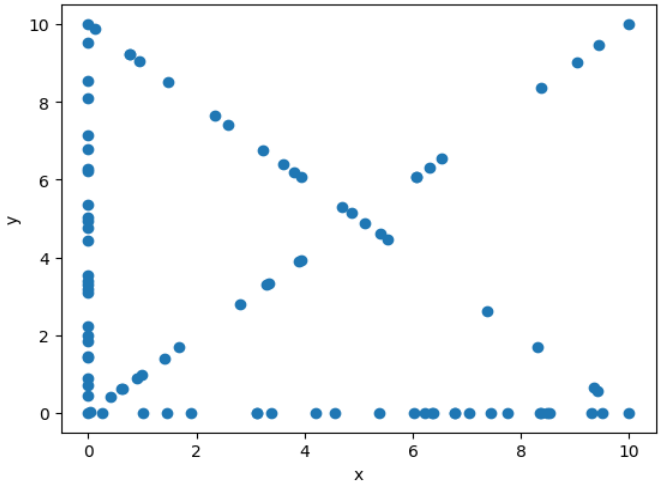
Wykres 2 - Zestaw danych 2)



Wykres 3 - Zestaw danych 3)



Wykres 4 - Zestaw danych 4)



4. Algorytm Grahama

Pseudokod algorytmu Grahama:

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 p0. 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 (pi leży na lewo od prostej (p_{t-1}, p_t))
   then
   - push(p_i),
   - i ← i+1

   else pop(s)

Koszt algorytmu:
Operacje dominujące to porównywanie współrzędnych lub badanie położenia punktu względem prostej.

$$O(n) + O(n \log n) + O(1) + O(n\text{-}3) = O(n \log n)$$

szukanie minimum	sortowanie	inicjalizacja stosu	krok4

We własnym kodzie do posortowania punktów ze względu na kąt posłużyłem się funkcją cmp_to_key z biblioteki functools.

5. Algorytm Jarvisa

Psedokod algorytmu Jarvisa:

1. znajdź punkt i_0 z S o najmniejszej współrzędnej y
   $i \leftarrow i_0$
2. repeat
   for $j \neq i$ do
   - znajdź punkt, dla którego kąt liczony przeciwnie do 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
   - $i \leftarrow k$

   until $i = i_0$

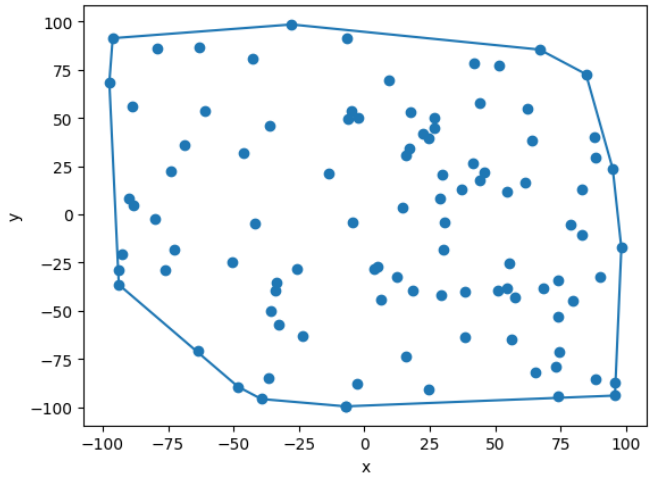
Koszt algorytmu:
Złożoność algorytmu jest rzędu O(n^2). Gdy liczba wierzchołków otoczki jest ograniczona przez stałą k, to jego złożoność jest rzędu O(kn).

We własnym kodzie w celu znalezienia punktu, którego kąt liczony przeciwnie do wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy, posłużyłem się funkcją reduce z biblioteki functools.

6. Wyniki działania algorytmów

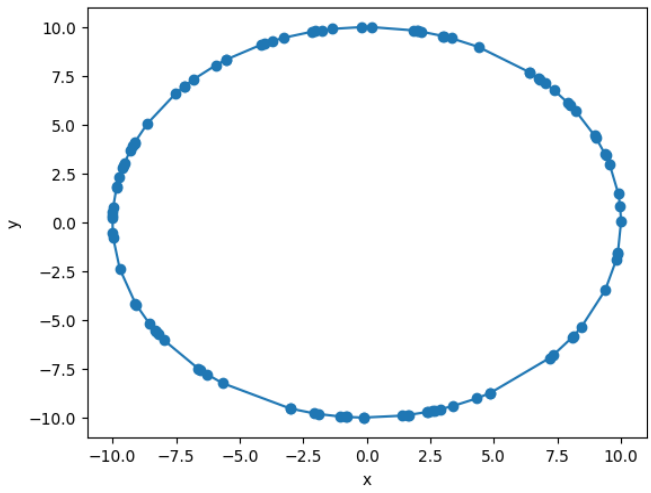
W każdym przypadku otoczki zostały wyznaczone poprawnie dla obydwóch algorytmów, więc wyniki działań tych algorytmów są takie same. Jedyną różnicą jest różnica w czasie działania, których porównanie znajduje się w dalszej części sprawozdania.

Wykres 5



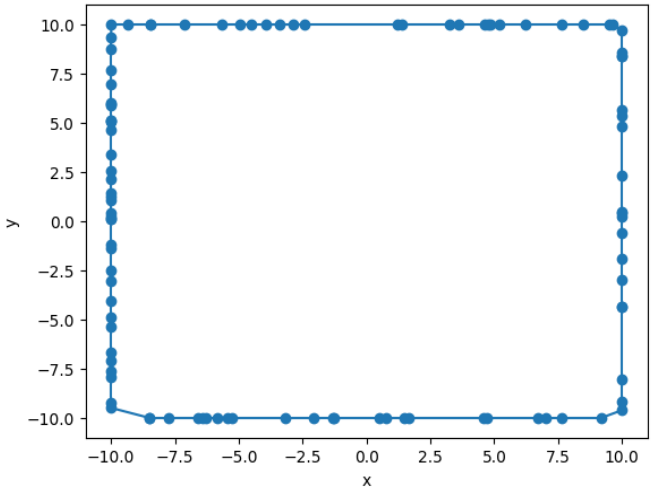
Otoczka wypukła dla zestawu danych 1)

Wykres 6



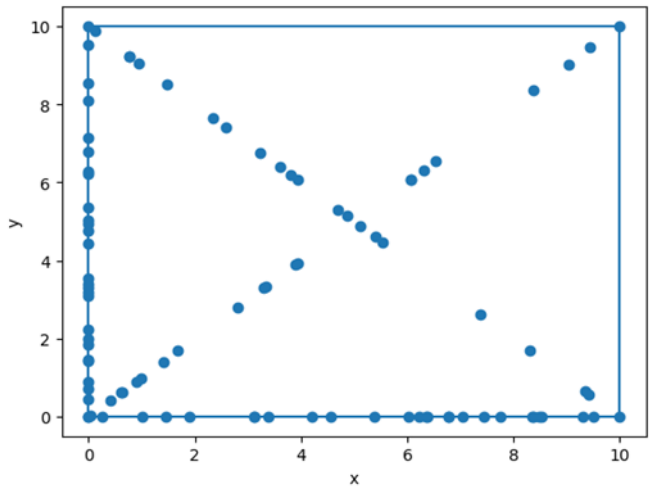
Otoczka wypukła dla zestawu danych 2)

Wykres 7



Otoczka wypukła dla zestawu danych 3)

Wykres 8



Otoczka wypukła dla zestawu danych 4)

7. Porównanie czasów działania algorytmów

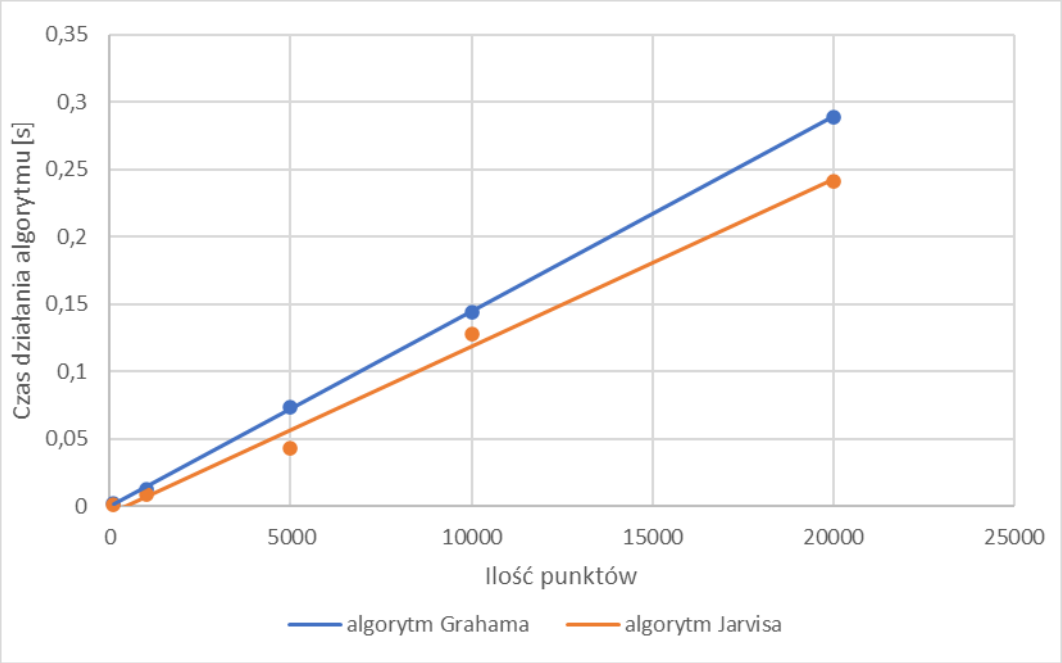
Do porównania czasów działania algorytmów posługuję się funkcją time z biblioteki time. Sprawdzam czas przed wykonaniem algorytmu oraz po wykonaniu, a następnie wyświetlam ich różnicę, dzięki czemu dostaję czas, w jakim został wykonany dany algorytm.

Tabela 1 - Tabela porównawcza czasów działania algorytmów

Zmodyfikowany zestaw danych	Ilość punktów	Czas działania [s]	
		Algorytm Grahama	**Algorytm Jarvisa**
Zestaw danych 1)	100	0,0024	0,0011
	1000	0,0119	0,0081
	5000	0,0737	0,0431
	10000	0,1442	0,1273
	20000	0,2886	0,2414
Zestaw danych 2)	100	0,0135	0,0075
	1000	0,0145	0,668
	2500	0,0737	4,0039
	5000	0,1019	15,4401
	10000	0,1965	59,7326
Zestaw danych 3)	100	0,0009	0,001
	1000	0,0176	0,0126
	5000	0,0988	0,0329
	10000	0,2327	0,0610
	20000	0,4300	0,1252
Zestaw danych 4)	25 na boku, 20 na przekątnej	0,0015	0,001
	250 na boku, 200 na przekątnej	0,0226	0,003
	500 na boku, 400 na przekątnej	0,0564	0,0086
	1000 na boku, 800 na przekątnej	0,1136	0,0166
	2000 na boku, 1600 na przekątnej	0,2222	0,0257

a) Opracowanie zestawu danych 1)

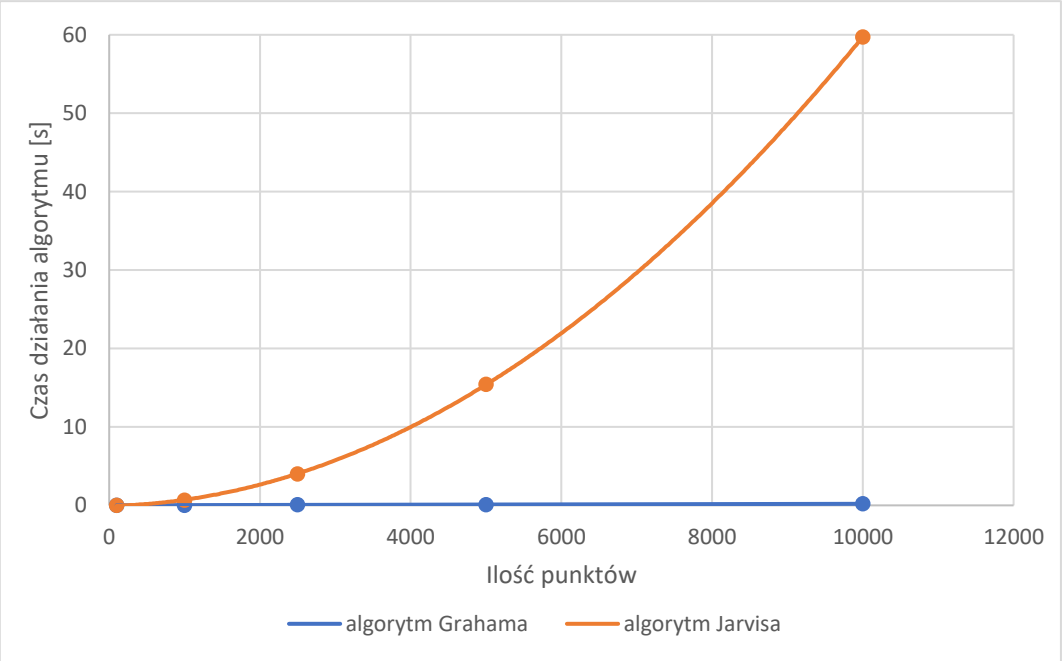
Wykres 9 - Porównanie czasu działania algorytmów dla zestawu 1



Na powyższym wykresie nr 9 widać, że algorytm Jarvisa działa nieznacznie szybciej dla każdego zbioru punktów z pierwszego zestawu danych. Jest on tylko ~1,2 razy szybszy, a więc takie różnice w czasie wykonania nie mają dużego znaczenia dla tak przygotowanych zbiorów.

b) Opracowanie zestawu danych 2)

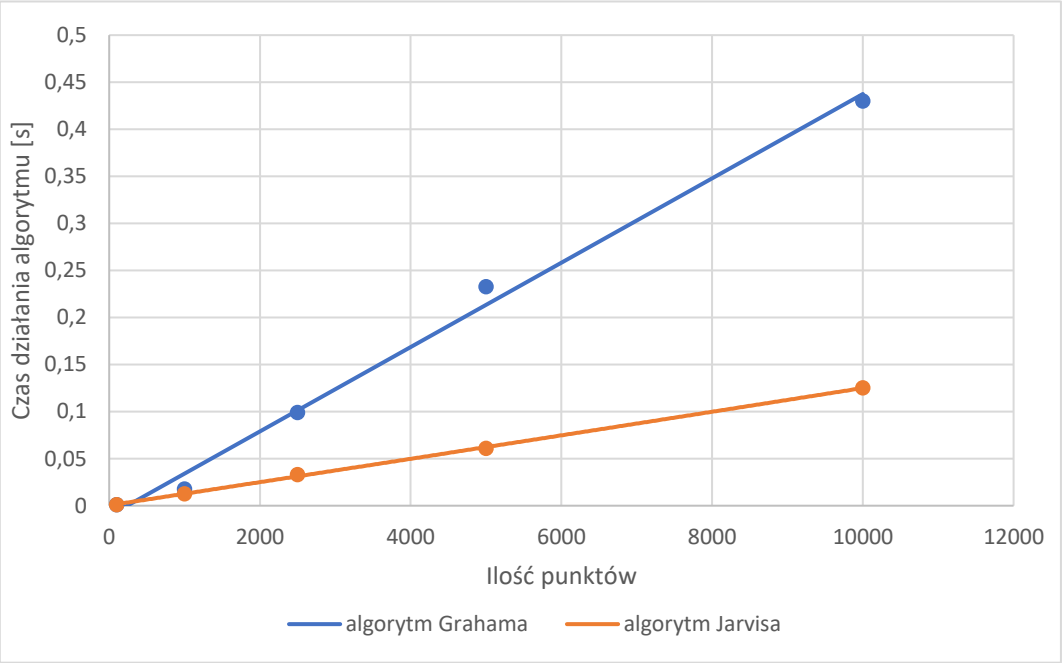
Wykres 10 - Porównanie czasu działania algorytmów dla zestawu 2



W tym zestawie danych każdy punkt znajdował się w otoczce wypukłej figury, a więc złożoność algorytmu Jarvisa jest $O(n^2)$, co doskonale widać na wykresie 10. Czas wykonania algorytmu Jarvisa dla najliczniejszego zbioru jest ponad 30 razy większy, a więc w tym przypadku algorytm Grahama ma znaczną przewagę.

c) Opracowanie zestawu danych 3)

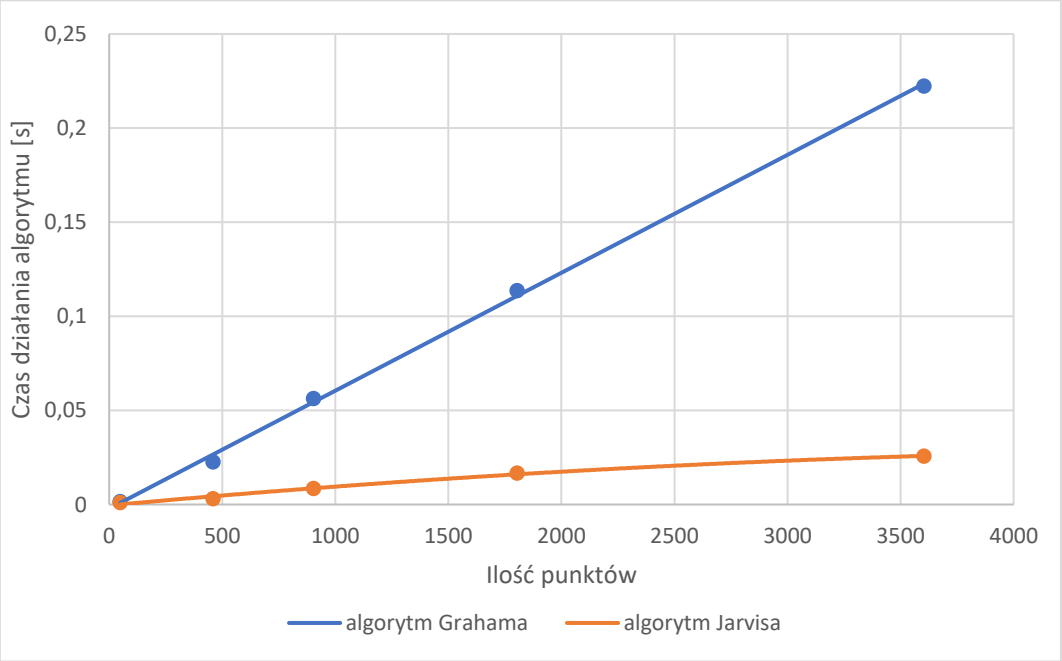
Wykres 11 - Porównanie czasu działania algorytmów dla zestawu 3



Dla tego typu zbiorów bardzo mało punktów wchodziło w skład otoczki, dlatego algorytm Jarvisa zbliżał się już w stronę złożoności liniowej, przez co okazał się dużo szybszy na tle algorytmu Grahama (około 3,5 razy szybszy względem najliczniejszego zbioru punktów).

d) Opracowanie zestawu danych 4)

Wykres 12 - Porównanie czasu działania algorytmów dla zestawu 4



W tym zestawie najbardziej zaznaczona jest przewaga algorytmu Jarvisa, która jest praktycznie liniowa. Przyczynia się do tego fakt, iż tylko 4 skrajne punkty, które są wierzchołkami kwadratu, wchodzą w skład otoczki. Algorytm Jarvisa znajduje od razu te 4 punkty, podczas gdy algorytm Grahama rozważa wszystkie punkty zbioru.

8. Wnioski

Obydwa algorytmy poprawnie wyznaczają otoczkę wypukłą zbioru punktów, co widać na wykresach 5, 6, 7 i 8. Różnice między tymi algorytmami, oprócz samego ich wykonania, są jednak dostrzegalne dopiero, gdy zwrócimy uwagę na ich czas obliczania. Algorytm Grahama, dla tak samo licznych, lecz inaczej skonstruowanych zbiorów wykonuje się w niemalże identycznym czasie. Algorytm Jarvisa natomiast ma już bardzo znaczące różnice czasowe, co można dostrzec przy porównaniu wykresu 10 i 12. Dzieje się tak, ponieważ przy zestawie danych 2), wszystkie punkty wchodzą w skład otoczki, co sprawia, że czas wynosi $O(n^2)$, natomiast przy zestawie danych 4) w skład otoczki wchodzą tylko 4 punkty, które algorytm Jarvisa wyłapuje na samym początku, więc czas wynosi już tylko $O(n)$.
Algorytm Jarvisa może być lepszym wyborem, jeśli wiemy, z jak skonstruowanym zestawem danych mamy do czynienia, jednak gdy nie posiadamy takiej informacji, bezpieczną opcją jest skorzystanie z algorytmu Grahama, który ma praktycznie zawsze taki sam przewidywalny czas wykonania dla tak samo licznych zbiorów.