Algorytmy geometryczne
Sprawozdanie z laboratoriów 2

grupa nr.4 Czw_12.20_B
Paweł Surdyka

Dane techniczne urządzenia na którym wykonano ćwiczenie:
Komputer z systemem Windows 10 x64
Procesor: Intel Core I5 9300HF CPU @2.4Ghz
Pamięć RAM: 8GB
Środowisko: Jupyter notebook

Język: Python 3

Opis ćwiczenia

Ćwiczenie polegało na implementacji i testach algorytmów wyznaczających otoczkę wypukłą tj. algorytmu Grahama oraz algorytmu Jarvisa.

1.Generacja punktów

W celu wykonania ćwiczenia wygenerowane zostały 4 zbiory punktów(2D, typu double), które zostały umieszczone w osobnych tablicach:

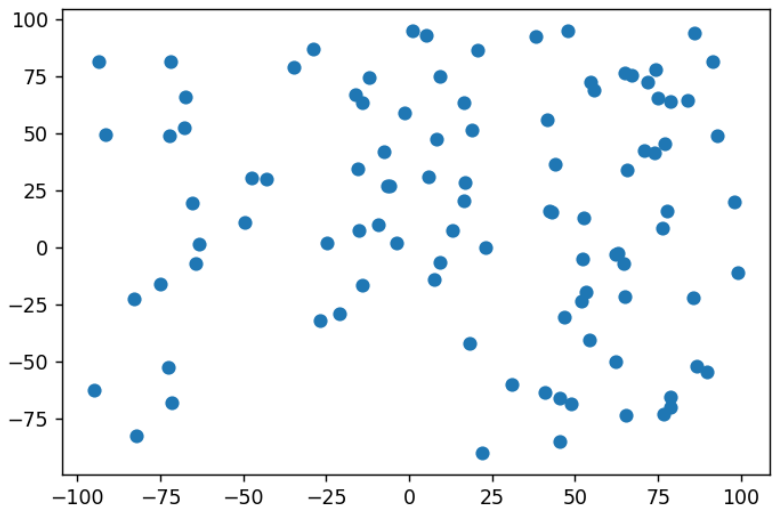
a) zawierający 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100],
b) zawierający 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
c) zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10),
d) 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

Zadaniem które należy wykonać na laboratorium 2 jest wyznaczenie otoczki wypukłej dla danego zbioru punktów, czyli najmniejszy zbiór wypukły zawierający podzbiór wszystkich punktów na płaszczyźnie. Do wyznaczenia otoczki zostały użyte dwa algorytmy: algorytm Grahama oraz algorytm Jarvisa.

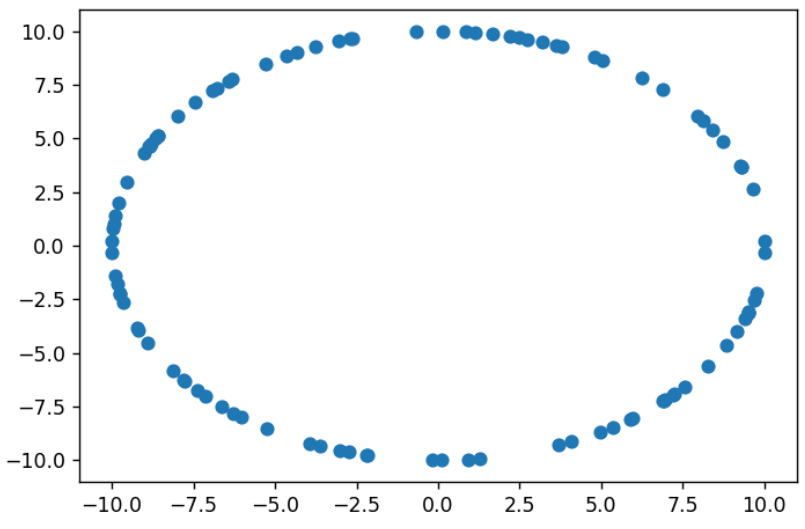
Losowanie położenia punktów odbyło się za pomocą metody random.uniform z biblioteki random. Funkcja ta generuje liczby typu double dla zadanego zakresu obustronnie domkniętego.
Punkty z zestawu 2. zostały wygenerowane z wykorzystaniem funkcji trygonometrycznych z biblioteki numpy.
Dla punktów z zestawu 3 wylosowano bok na którym mają zostać ułożone, a następnie konkretne współrzędne na tym boku.
Do wygenerowania punktów z zestawu 4. użyto równania prostej w celu ułożenia punktów ich na przekątnych kwadratu

Wszystkie zestawy punktów zostały zwizualizowane za pomocą dostarczonego narzędzia graficznego opartego o bibliotekę matplotlib.

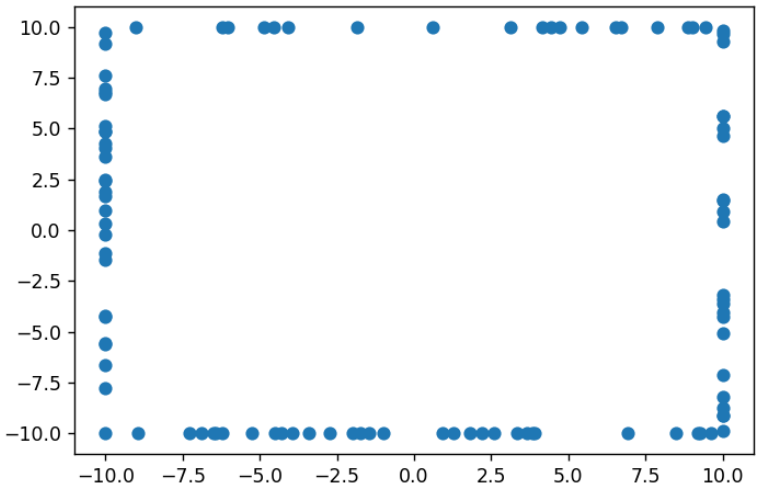
Wykres 1.1 Zestaw danych 1.



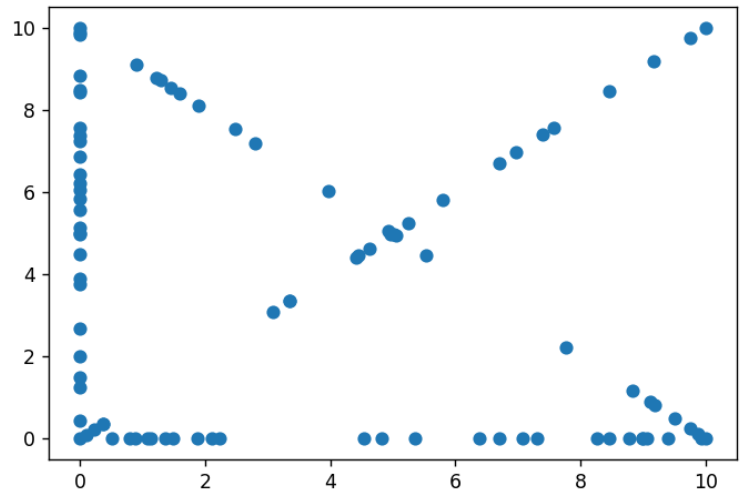
Wykres 1.2 Zestaw danych 2.



Wykres 1.3 Zestaw danych 3.



Wykres 1.4 Zestaw danych 4.



2. Metoda obliczania wyznacznika oraz tolerancja dla zera

Do tego ćwiczenia wykorzystano wyznacznik 2x2 własnej implementacji. Jako tolerancję dla zera przy określaniu orientacji punktu względem odcinka przyjęto 10^{-12}.

3. Implementacja algorytmu Grahama

Algorytm przebiega następująco:

1. Wybierz punkt (ozn. O) o najniższej wartości współrzędnej *y*.
2. Przesuń wszystkie punkty tak, by punkt O pokrył się z początkiem układu współrzędnych.
3. Posortuj punkty leksykograficznie względem:
    - kąta pomiędzy wektorem a dodatnią osią układu współrzędnych,
    - odległości punktu od początku układu współrzędnych.
4. Wybierz punkt (ozn. S) o najmniejszej współrzędnej y; jeśli kilka punktów ma tę samą współrzędną y, wybierz spośród nich ten o najmniejszej współrzędnej x.
5. Przeglądaj listę posortowanych punktów poczynając od punktu S:
    - Od bieżącej pozycji weź trzy kolejne punkty (ozn. A, B, C).
    - Jeśli punkt B leży na zewnątrz AOC, to może należeć do otoczki wypukłej. Przejdź do następnego punktu na liście.
    - Jeśli punkt B leży wewnątrz trójkąta AOC, to znaczy, że nie należy do otoczki. Usuń punkt B z listy i cofnij się o jedną pozycję (o ile bieżąca pozycja jest różna od początkowej).

4. Implementacja algorytmu Jarvisa

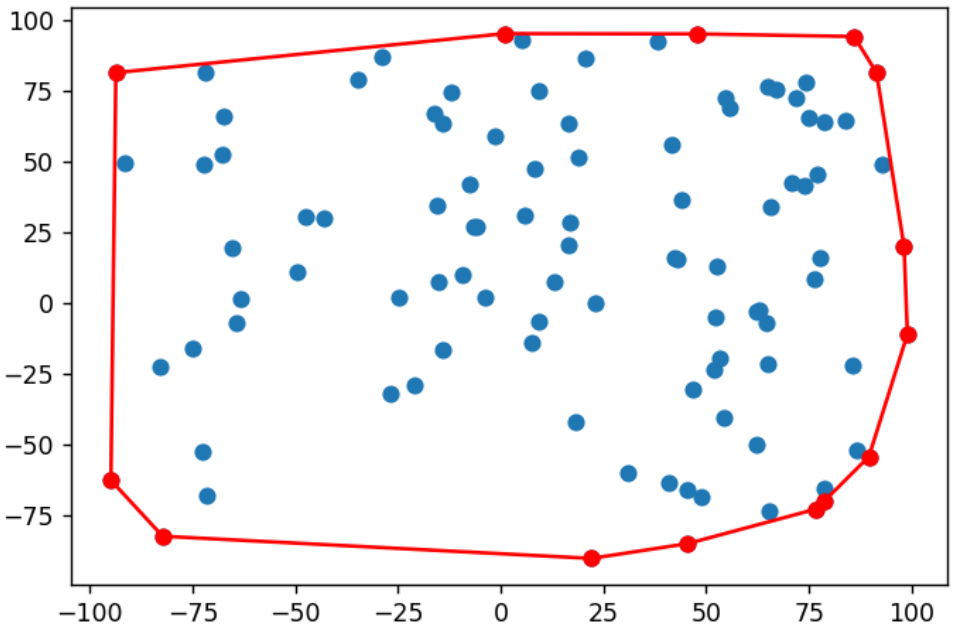
Algorytm przebiega następująco:

- P1– punkt na otoczce wypukłej o najmniejszej współrzędnej y (jeśli jest więcej niż jeden, wybierany jest ten o najmniejszej współrzędnej x),
- Q1– punkt na otoczce wypukłej o największej współrzędnej y (jeśli jest więcej niż jeden, wybierany jest ten o największej współrzędnej x),
- wyznaczanie prawego łańcucha otoczki:
    1. i := 1
    2. powtarzaj:
        - S := P_i
        - P_{i+1}– punkt N dla którego kąt między wektorem SN a wektorem [1,0] jest najmniejszy; N należy do otoczki,
        - jeśli N = Q_1koniec iterowania,
        - i := i+1
- wyznaczanie lewego łańcucha otoczki :
    1. i := 1
    2. powtarzaj:
        - S = Qi
        - Q_{i+1}– punkt N dla którego kąt między wektorem SN a wektorem [1,0]jest najmniejszy; N należy do otoczki,
        - jeśli N = P_1koniec iterowania,
        - i := i+1
- ostatecznie otoczkę wypukłą określają punkty P i Q(z pominięciem tych powtarzających się na granicach łańcuchów)

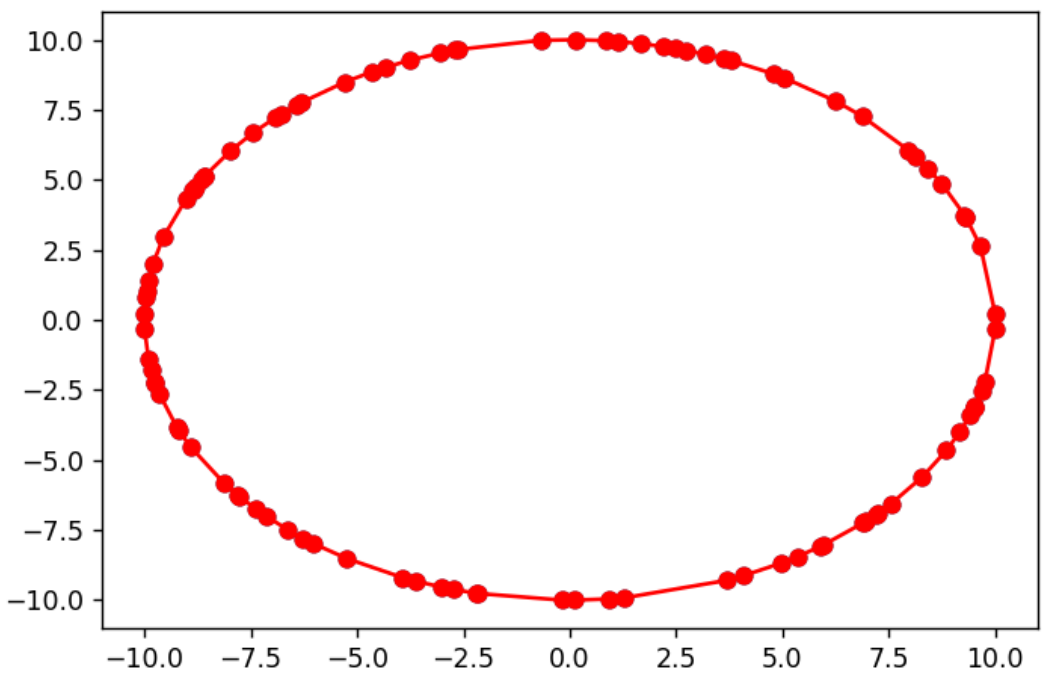
5. Wyniki działania algorytmów Grahama i Jarvisa

Oba algorytmy przetestowano na tych samych zbiorach danych. W przypadku obu algorytmów uzyskane zbiory punktów otoczki wypukłej były poprawne, dla każdego zbioru takie same dla obydwu algorytmów. Wykres 1.3 Zestaw danych 3. Wykres 1.4 Zestaw danych 4.

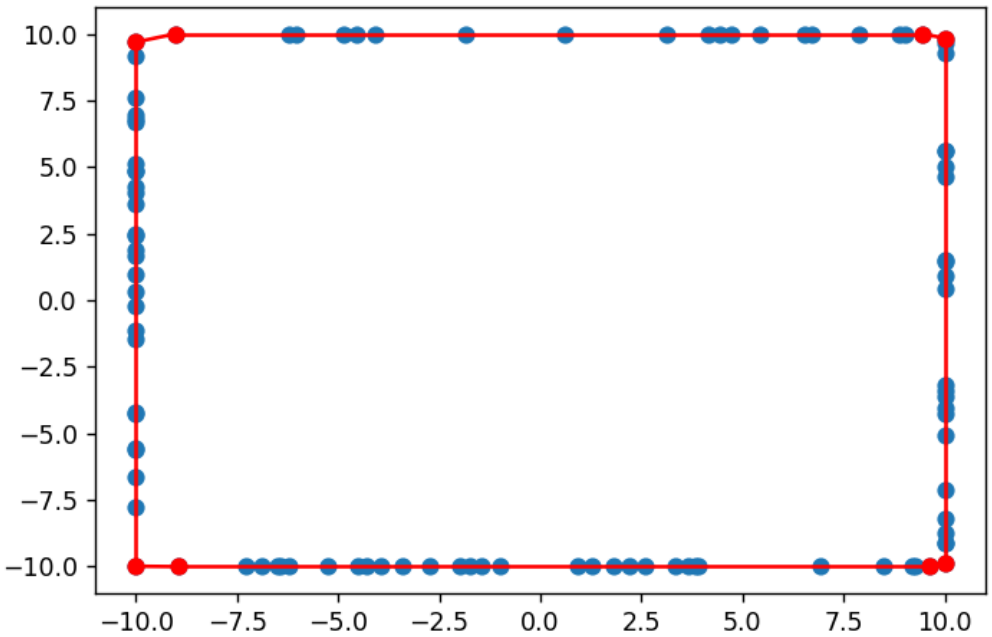
Wykres 2.1 Otoczka wypukła 1. zestawu danych wygenerowana przez algorytmy Grahama i Jarvisa



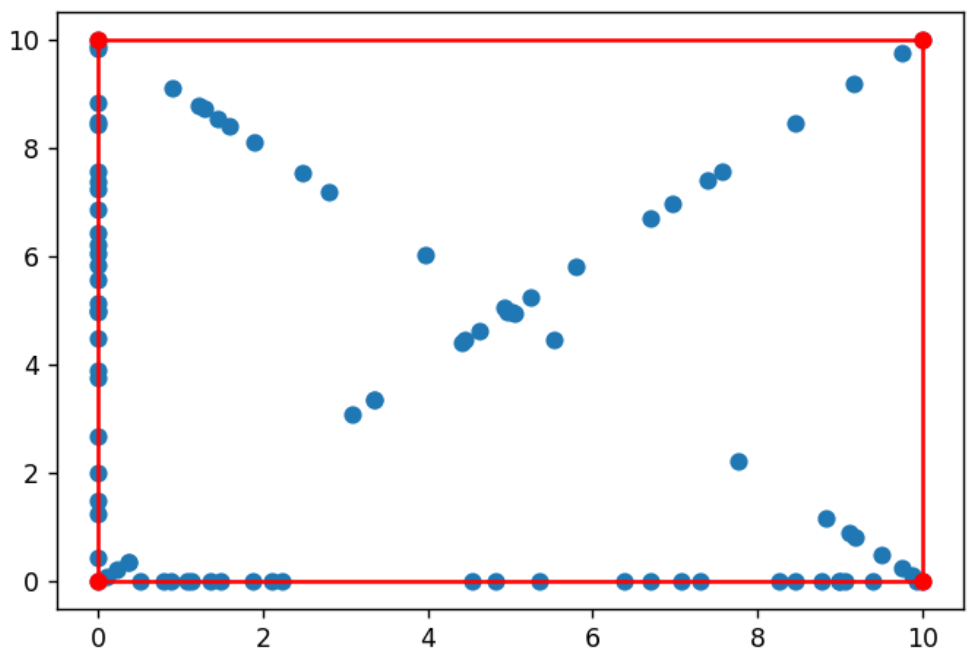
Wykres 2.2 Otoczka wypukła 2. zestawu danych wygenerowana przez algorytmy Grahama i Jarvisa



Wykres 2.3 Otoczka wypukła 3. zestawu danych wygenerowana przez algorytmy Grahama i Jarvisa



Wykres 2.4 Otoczka wypukła 4. zestawu danych wygenerowana przez algorytmy Grahama i Jarvisa



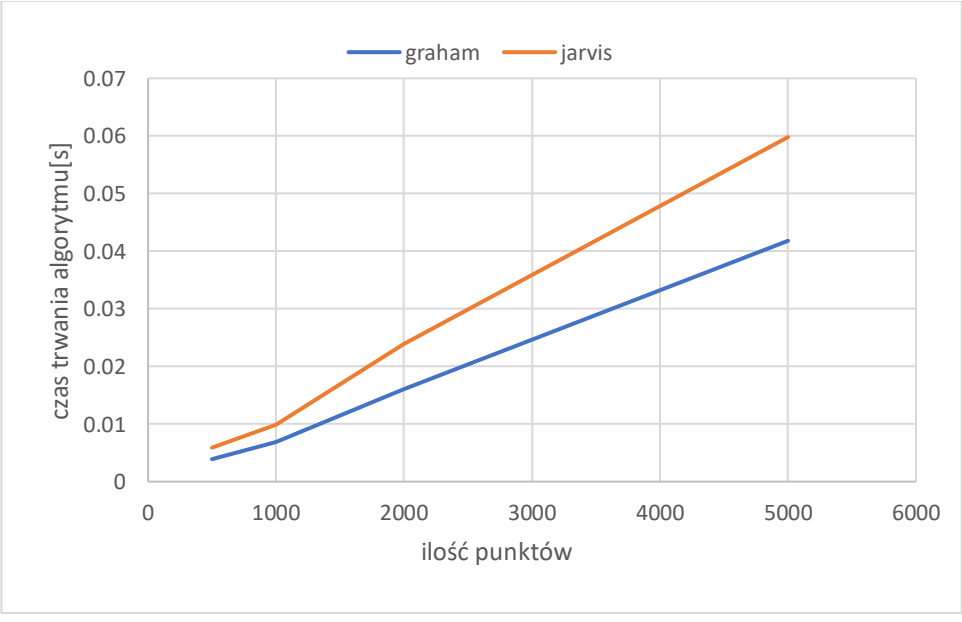
6. Porównanie czasu wykonania algorytmów Grahama i Jarvisa

W tej części ćwiczenia algorytmy uruchomiono ponownie (tylko teraz bez wizualizacji) dla tych samych zestawów, w celu porównania ich czasu wykonania. Testy wydajności przeprowadzono dla trzech następujących ilości punktów z zestawach:
500, 1000, 2000, 5000
Dla pierwszego zbioru algorytm Grahama okazał się szybszy od Jarvisa, około dwukrotnie.
Większa różnica wystąpiła dla zestawu drugiego. Już dla najmniejszej liczby punktów algorytm Grahama okazał się szybszy o aż 46x od algorytmu Jarvisa, co zostało spowodowane dużą liczbą punktów należących do otoczki. Bardzo dobrze widać tutaj kwadratową (złożoność algorytmu Jarvisa. W związku z naturą tej złożoności nie udało się przeprowadzić pomiaru dla algorytmu Jarvisa w zestawie 2. dla większej liczby punktów niż 5000 ze względów praktycznych, ponieważ algorytm wykonywałby się kilkanaście minut lub nawet kilkadziesiąt. Zestaw 3. Dla większej liczby punktów wypadł na korzyść algorytmu Jarvisa. Spowodowane to było małym rozmiarem otoczki, dzięki czemu algorytm Jarvisa musiał znaleźć niedużą liczbę punktów aby zakończyć swoje działanie. Taka sama sytuacja miała w ostatnim zestawie, aczkolwiek w jego przypadku liczba punktów otoczki była tak mała, że algorytm Jarvisa działał wielokrotnie szybciej niż algorytm Grahama. Jak napisano wcześniej, wynika to z ograniczenia złożoności algorytmu Jarvisa przez liczbę wierzchołków w otoczce, co dla bardzo małej ilości wierzchołków (jak tylko 4 punkty w zestawie 4.) daje nam złożoność praktycznie liniową, co widzimy w tabeli 1. Warto zwrócić uwagę na czas wykonania algorytmu Grahama pomiędzy zestawami - jest on bardzo zbliżony.

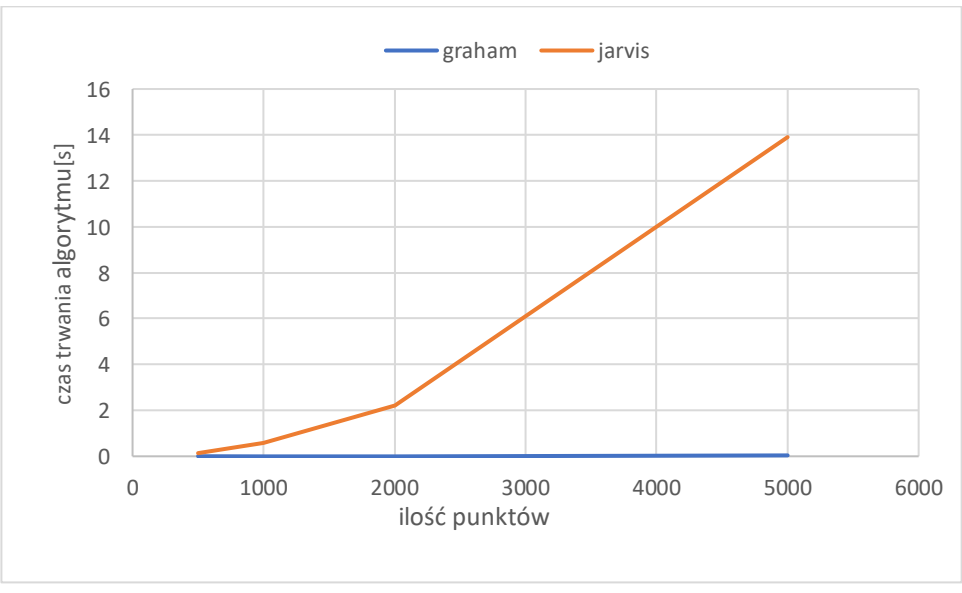
Wynika to ze złożoności tego algorytmu - O(nlogn), a więc niezależnej od rodzaju/położenia punktów na płaszczyźnie, a jedynie od ich liczby.

Zestaw	Algorytm	500 punktów	1000 punktów	2000 punktów	5000 punktów
Zestaw 1	**Graham**	0.0039s	0.0069s	0.0160s	0.0418s
	Jarvis	0.0059s	0.0099s	0.0239s	0.0598s
Zestaw 2	**Graham**	0.0029s	0.0069s	0.0149s	0.0397s
	Jarvis	0.1381s	0.5864s	2.2276s	13.9040s
Zestaw 3	**Graham**	0.0039s	0.0090s	0.0209s	0.0551s
	Jarvis	0.0039s	0.0059s	0.0159s	0.0320s
Zestaw 4	**Graham**	0.0299s	0.0737s	0.1476s	0.4115s
	Jarvis	0.0069s	0.0139s	0.0259s	0.0662s

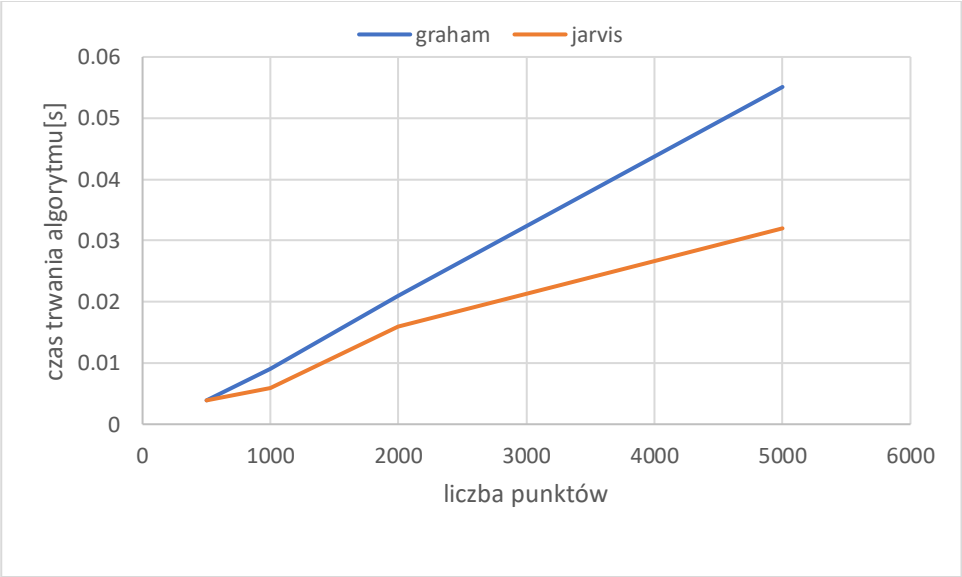
Wykres 3.1 Czas wykonania algorytmów Grahama i Jarvisa dla 1. zestawu danych w zależności od ilości punktów w zestawie



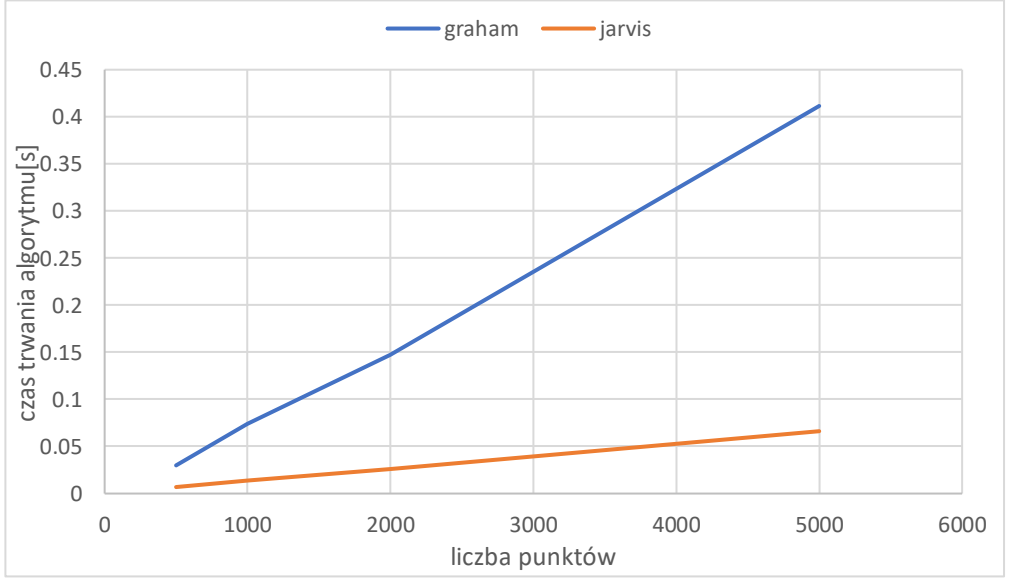
Wykres 3.2 Czas wykonania algorytmów Grahama i Jarvisa dla 2. zestawu danych w zależności od ilości punktów w zestawie



Wykres 3.3 Czas wykonania algorytmów Grahama i Jarvisa dla 3. zestawu danych w zależności od ilości punktów w zestawie



Wykres 3.4 Czas wykonania algorytmów Grahama i Jarvisa dla 4. zestawu danych w zależności od ilości punktów w zestawie



Wnioski

Jak widać na podstawie powyższych danych oba algorytmy działają prawidłowo i potrafią wyznaczyć otoczkę wypukłą dla dowolnych zbiorów danych, nawet takich, które z teoretycznego punktu widzenia mogłyby sprawiać trudności. Jak widzimy czas wykonania algorytmu Grahama dla różnych zbiorów o zbliżonej liczbie punktów jest bardzo podobny. W przypadku algorytmu Jarvisa różnice są kolosalne - jest to tłumaczone złożonością mającą związek z rozmiarem otoczki - rzędu O(kn) - gdzie k to ilość wierzchołków otoczki.

Wyjaśnia to dobre czasy dla dwóch ostatnich zestawów danych i fatalnego czasu dla zestawu drugiego, w którym jako że otoczka zawiera wszystkie punkty mamy do czynienia z złożonością O(n^2), efekt ten jest zwłaszcza widoczny dla większej ilości punktów. Już dla 1000 w przypadku zestawu drugiego algorytm Jarvisa wykonywał się ok. 0,5 sekund, podczas gdy algorytm Grahama zrobił to w ułamku sekundy.

Algorytm Jarvisa może znaleźć zastosowanie w przypadku wyznaczania dużej ilości otoczek dla zbiorów, których otoczki będą posiadały mało wierzchołków. W innych przypadkach algorytm Grahama jest lepszym rozwiązaniem. Nie jest on bardziej skomplikowany ani trudniejszy w implementacji od algorytmu Jarvisa, wymaga jednak posiadania funkcji efektywnie sortującej punkty, daje on jednak znacznie lepszą złożoność, która pozwala wyznaczać otoczkę wypukłą dla dużych, skomplikowanych zbiorów punktów, a czas jego wykonania jest łatwo przewidywalny.