Maciej Michoń, Informatyka II rok, gr. 2 Śr_13.00_A

Algorytmy geometryczne, sprawozdanie z ćwiczenia 2

1. Środowisko, biblioteki oraz dane techniczne urządzenia

Ćwiczenie zostało zrealizowane w Jupyter Notebooku i napisane w języku Python, z wykorzystaniem bibliotek numpy (umożliwiająca zaawansowane operacje numeryczne), pandas (do przedstawiania wyników poszczególnych obliczeń w DataFrame). Dodatkowo do rysowania wykresów (Visualizer) i sprawdzenia poprawności niektórych funkcji wykorzystałem projekt opracowany przez studentów Koła Naukowego BIT. Wszystko to było wykonane na laptopie z systemem operacyjnym Windows 11, procesorem AMD Ryzen 5 5600H 3.30 GHz oraz pamięcią RAM 32 GB.

2. Opis realizacji ćwiczenia:

Celem ćwiczenia była implementacja oraz testy algorytmów Grahama oraz Jarvisa, wyznaczających otoczkę wypukłą zbioru punktów.

2.1. Generacja zbiorów punktów

Żeby wykonać ćwiczenie na początku wygenerowane zostały 4 zbiory punktów (2D, współrzędne typu double):

- Zestaw 1

  100 losowych punktów o współrzędnych z przedziału [-100, 100]

- Zestaw 2

  100 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=10

- Zestaw 3

  100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10),

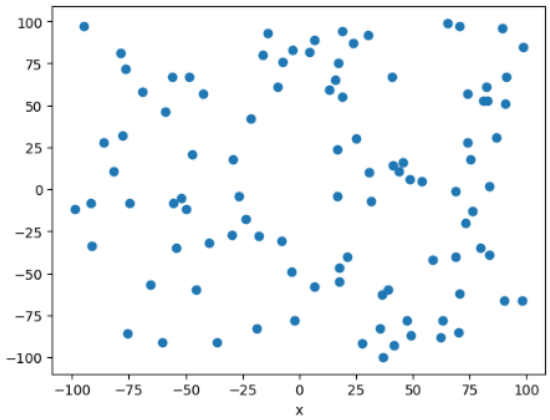
- Zestaw 4

  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.

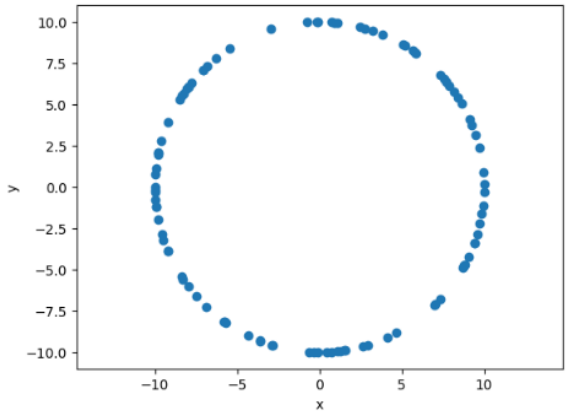
Współrzędne punktów zostały wygenerowane za pomocą metody random.uniform z biblioteki numpy. Funkcja ta generuje losowe punkty typu double w podanym przedziale domkniętym.
Punkty z zestawu 2 zostały wygenerowane za pomocą funkcji trygonometrycznych z biblioteki numpy.
Punkty z zestawu 3 zostały wygenerowane tak, że pierwsze losujemy bok, a później na tym boku generujemy konkretne współrzędne.

Dla punktów z zestawu 4 użyto równania prostej aby ułożyć na przekątnych kwadratu punkty.

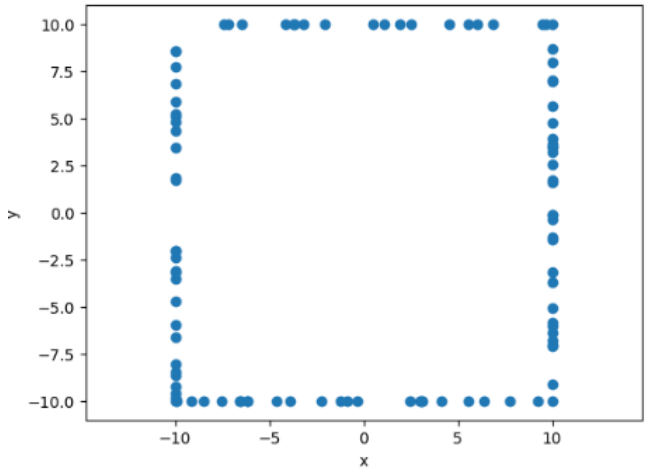
Wykres 1.1 Zestaw danych 1



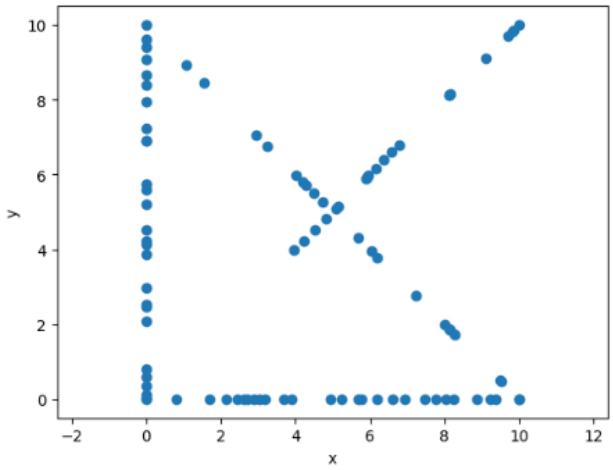
Wykres 1.2 Zestaw danych 2



Wykres 1.3 Zestaw danych 3



Wykres 1.4 Zestaw danych 4



2.2. Metoda obliczania wyznacznika oraz tolerancja dla zera

Do tego ćwiczenia został wykorzystany wyznacznik 3x3 własnej implementacji. W wyborze tolerancji dla zera przy określaniu punktu względem odcinka przyjąłem 0. Co ciekawe dla innych tolerancji dla zera występowały błędy wykorzystując testy dostarczonych z projektu Koła Naukowego BIT. Może być to spowodowane tym, że dla wyższych tolerancji istnieją punkty, które na przykład dla epsilon równego 10^{-14}, mogą klasyfikować dwa punkty jako współliniowe, a tak w rzeczywistości nie jest.

2.3. Implementacja algorytmu Grahama

Algorytm działa w następujący sposób: Na początku znajduje punkt o najmniejszej współrzędnej y. Jeżeli jest kilka takich punktów, to wybiera ten z nich, który ma najmniejszą współrzędną x. Jest to punkt startowy. Następnie wykonuje sortowanie z wbudowaną funkcją sorted() z dołączonym własnej implementacji kluczem, który powoduje że punkty są sortowane wg. kąta jaki tworzy wektor punktu startowego z rozważanym względem dodatniego kierunku osi. Kiedy wystąpi przypadek, że punkty tworzą taki sam kąt, usuwamy wszystkie z wyjątkiem najbardziej oddalonego od punktu startowego. W pętli głównej sprawdzane jest orientacja punktu względem ostatniej krawędzi i dokładamy punkty do otoczki lub cofamy się.

2.4. Implementacja algorytmu Jarvisa

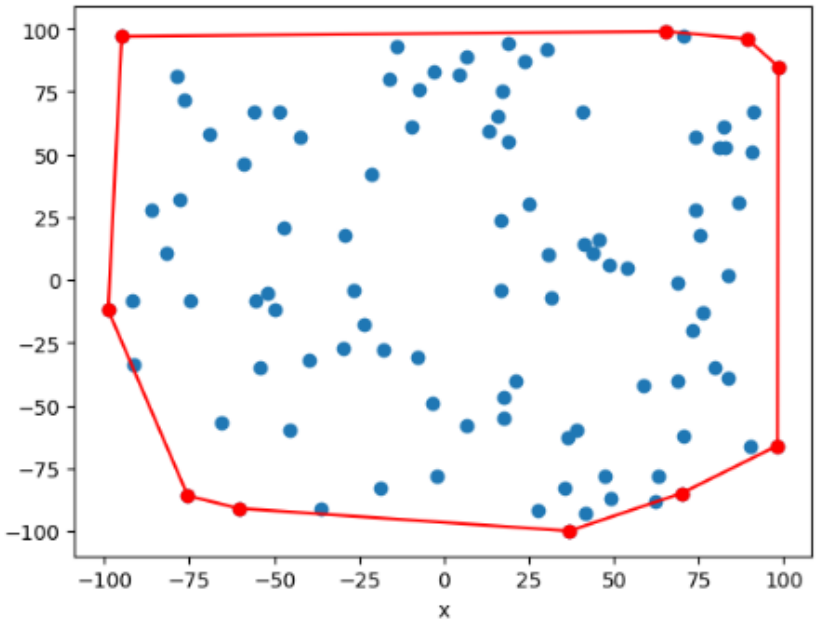
Podobnie jak w przypadku algorytmu Grahama znajdujemy punkt początkowy. W wewnętrznej pętli algorytmu poszukiwany jest punkt, dla którego wszystkie pozostałe leżą po lewej stronie odcinka wyznaczonego przez poprzedni punkt otoczki i tego punktu. Jeśli jest kilka takich punktów to wybieramy ten najbardziej oddalony od ostatniego punktu otoczki czyli dążymy do stworzenia jak najdłuższej krawędzi. Powtarzamy ten proces do momentu powrotu do punktu startowego.

2.5. Wyniki działania algorytmów Grahama i Jarvisa

Algorytmy zostały przetestowane przez przygotowane cztery zbiory danych. W przypadku obu tych algorytmów uzyskane zbiory punktów otoczki wypukłej były poprawne i wyniki były takie same. Dodatkowo poprawność działania algorytmów został sprawdzony przez zaimportowane testy z projektu Koła Naukowego BIT.

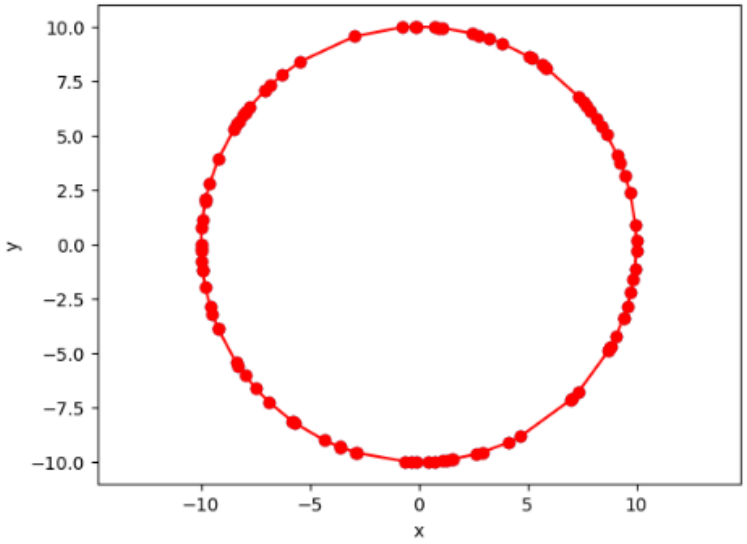
Zestaw 1 nie sprawiał trudności obu algorytmom. Wystąpiła w nim bardzo mała ilość punktów współliniowych leżących na krawędzi otoczki, które poprawnie zostały nie zawarte.

Wykres 2.1. Otoczka wypukła Zestawu danych 1, wygenerowana przez algorytm Grahama i Jarvisa



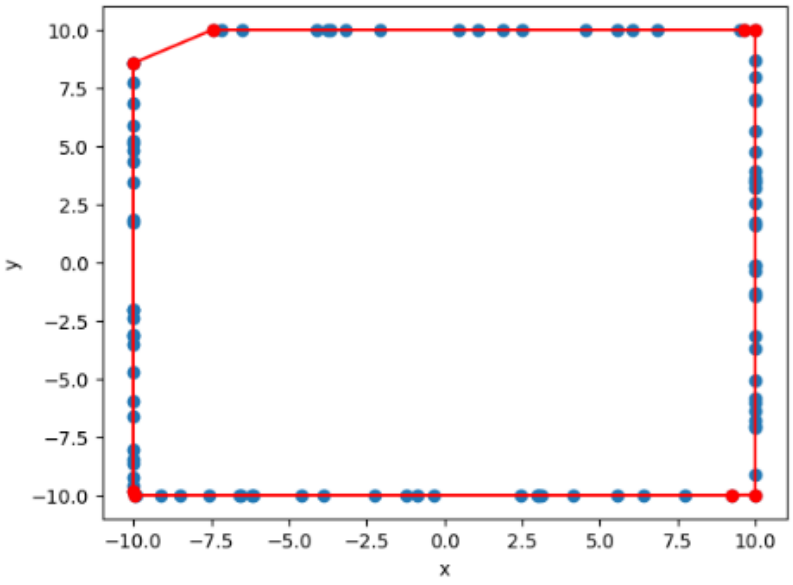
W Zestawie 2 można zaobserwować dominację algorytmu Grahama nad algorytmem Jarvisa. Oba algorytmy wyznaczyły poprawnie otoczkę, lecz algorytm Jarvisa zrobił to odczuwalnie wolniej. Dużą wadą algorytmu Jarvisa jest, że musi sprawdzić wiele razy dużą ilość punktów w celu wyznaczenia kolejnego punktu otoczki. Porządek tego zestawu sprawia, że algorytm Grahama bardzo szybko odnajduje otoczkę, ponieważ każdy kolejny analizowany punkt będzie należał do otoczki.

Wykres 2.2. Otoczka wypukła Zestawu danych 2, wygenerowana przez algorytm Grahama i Jarvisa



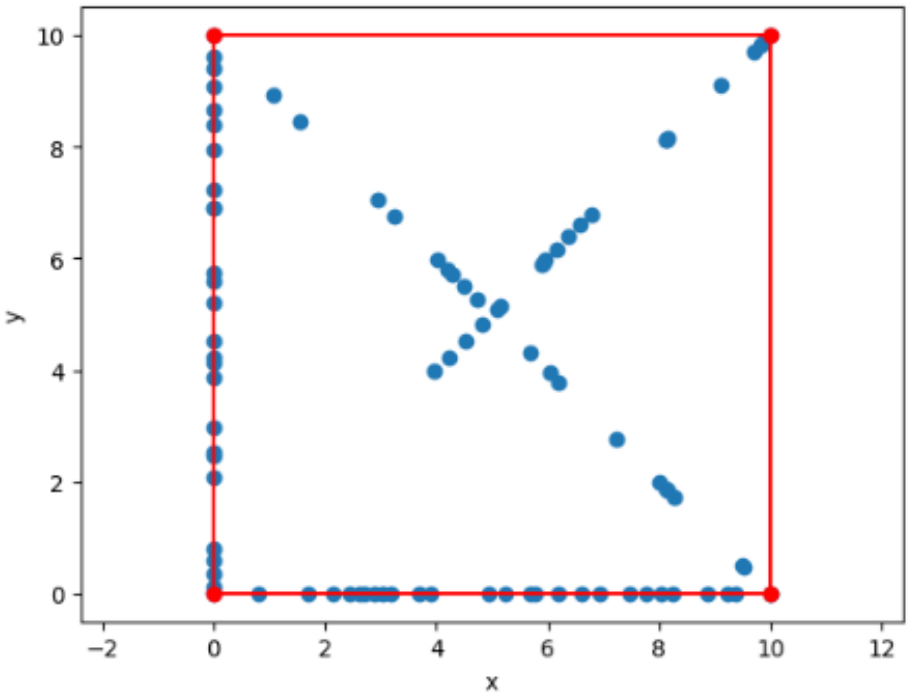
Zestaw 3 rozpatruje problem współliniowości w otoczce. Oba algorytmy szybko wyznaczyły poprawne otoczki. Powstała otoczka ze skrajnych punktów leżących na boku prostokąta. W tym przypadku algorytm Jarvisa lepiej sobie radzi niż algorytm Grahama. Algorytm Jarvisa musi znaleźć od 4 do 8 punktów żeby zakończyć działanie, bo właśnie dla tego zbioru punktów tyle wierzchołków otoczki może wystąpić, a algorytm Grahama musi sprawdzić większość punktów.

Wykres 2.3. Otoczka wypukła Zestawu danych 3, wygenerowana przez algorytm Grahama i Jarvisa



Zestaw 3 również rozpatruje problem współliniowości w otoczce z istniejącymi dokładnie 4 wierzchołkami otoczki. Oba algorytmy jak zwykle poprawnie wyznaczyły otoczkę i tak jak w przypadku powyżej lepiej sobie poradził algorytm Jarvisa. Jest to specjalny przypadek w którym algorytm Grahama musi wykonać duża ilość cofnięć. W obu przypadkach widzimy, że jeśli jest wiadome, że mamy do czynienia mało skomplikowanymi otoczkami to podejście losowe jest lepsze czyli w tych przypadkach algorytm Jarvisa będzie szybszy.

Wykres 2.4. Otoczka wypukła Zestawu danych 3, wygenerowana przez algorytm Grahama i Jarvisa



2.6. Porównanie czasów działania algorytmu Grahama i Jarvisa

Ostatnią częścią ćwiczenia było porównanie czasów działania. W tym celu wykorzystałem funkcje nie posiadające żadnych operacji do wizualizacji danych.

Testy wydajności przetestowałem na 4 zestawach o następującej ilości punktów każda: 100,1000,2500,5000,7500,10000,20000.

Dla Zestawu 1, w pełni losowego algorytm Grahama był szybszy od Jarvisa, około dwukrotnie

Największe różnice można było zaobserwować w Zestawie 2. Dla tego zbioru zdecydowanie szybszy był algorytm Grahama, który już dla najmniejszej liczby punktów czas wykonania był 11,6x szybszy niż algorytm Jarvisa. Tutaj dobrze widać kwadratową złożoność algorytmu Jarwisa. Dla zbioru 10x większego wykonuje się ok. 100x dłużej oraz dla zbioru 100x większego wykonuje się ok. 10000x dłużej. Związku z tym nie udało mi się wykonać testu dla 20000 punktów, ponieważ algorytm wykonywałby się za długo nawet do kilku minut, a przy o wiele większych zbiorach nawet do kilkunastu minut.

W Zestawie 3 i 4 dla większej ilości punktów lepiej wypadł algorytm Jarwisa. Powodem takiego wyniku jest mała ilość wierzchołków otoczki do znalezienia. Dodatkowo w Zestawie 4

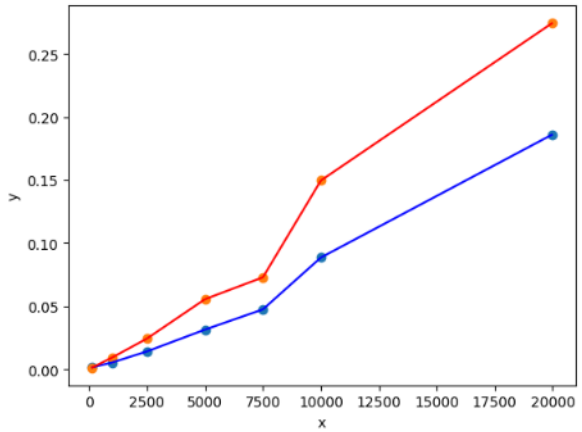
algorytm Jarvisa jest kilka razy szybszy niż algorytm Grahama. Złożoność algorytmu Jarwisa jest ograniczona przez liczbę wierzchołków w otoczce, czyli dla małej ilości punktów (od 4 do 8 w Zestawie 3, dokładnie 4 w Zestawie 4) złożoność wychodzi w przybliżeniu liniowa, co widać w Tabeli 1.

Tabela 1. Czas wykonania algorytmów Grahama i Jarvisa dla każdego z zestawów w zależności od liczby punktów w zestawie (w sekundach)

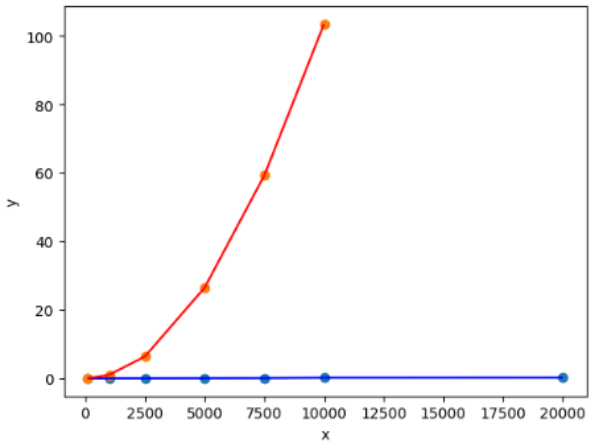
Zestaw	Algorytm	100	1000	2500	5000	7500	10000	20000
Zestaw 1	Grahama	0,0014	0,0054	0,0142	0,0314	0,0475	0,0886	0,1863
	Jarvisa	0,0010	0,0093	0,0244	0,0556	0,0729	0,1497	0,2748
Zestaw 2	Grahama	0,0012	0,0113	0,0192	0,0408	0,0737	0,1800	0,1970
	Jarvisa	0,0120	1,0297	6,3914	26,492	59,263	103,47	-
Zestaw 3	Grahama	0,0010	0,0088	0,0231	0,0539	0,0803	0,1091	0,2357
	Jarvisa	0,0015	0,0070	0,0175	0,0350	0,0525	0,0721	0,1368
Zestaw 4	Grahama	0,0010	0,0166	0,0433	0,0958	0,1514	0,2128	0,5552
	Jarvisa	0,0004	0,0045	0,0090	0,0200	0,0299	0,0402	0,0790

Jarwis – kolor czerwony **Graham – kolor niebieski**

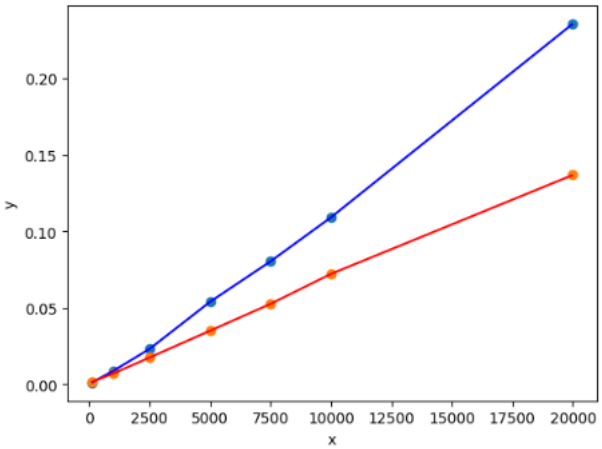
Wykres 3.1 Czas wykonania algorytmów Grahama i Jarvisa dla Zestawu 1 w zależności od liczby punktów w zestawie



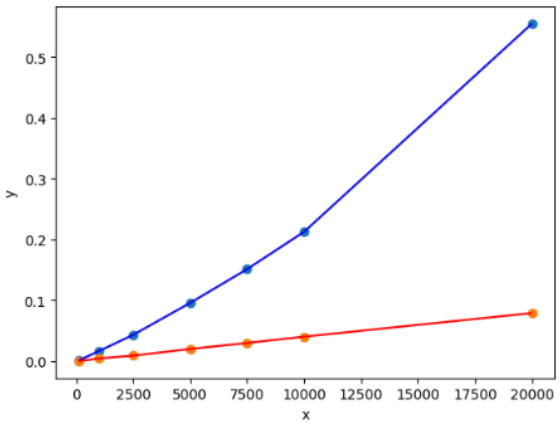
Wykres 3.2 Czas wykonania algorytmów Grahama i Jarvisa dla Zestawu 2 w zależności od liczby punktów w zestawie



Wykres 3.3 Czas wykonania algorytmów Grahama i Jarvisa dla Zestawu 3 w zależności od liczby punktów w zestawie



Wykres 3.4 Czas wykonania algorytmów Grahama i Jarvisa dla Zestawu 1 w zależności od liczby punktów w zestawie



3. Wnioski

Jak można zauważyć na oba algorytmy potrafiły prawidłowo wyznaczyć otoczkę wypukłą dla dowolnych zbiorów, nawet takich, które teoretycznie powinny sprawiać trudność. Czas wykonania algorytmu Grahama dla wszystkich przypadków był podobny ze względu na stałą złożoność O(nlogn), gdzie n – ilość punktów w zbiorze. W przypadku algorytmu Jarwisa można zauważyć zróżnicowane wyniki w czasie wykonania, który jest spowodowany złożonością zależną od liczby istniejących wierzchołków otoczki w zestawach danych – rzędu O(kn) – gdzie k to ilość wierzchołków otoczki. Skutkiem takiej złożoności są bardzo dobre czasy Zestawów 3 i 4 oraz tragicznego czasu dla Zestawu 3 gdzie złożoność wynosi O(n^2) wynikająca z tego, że liczba wierzchołków otoczki jest równa ilości wszystkich punktów w zbiorze.

Jak można wywnioskować algorytm Jarvisa jest dobry w przypadkach kiedy mamy duże zbiory punktów i wiemy że ilość wierzchołków otoczki jest niewielka. W większej ilości przypadków algorytm Grahama będzie lepszym wyborem, bo wpływ ilości wierzchołków otoczki nie wpłynie na czas jego działania i będzie relatywnie stała. Implementacja obu algorytmów nie jest bardzo skomplikowana. Jedynie algorytm Grahama wymaga posiadania efektywnego sortowania w celu osiągnięcia dobrej złożoności.

Podsumowując algorytm Graham jest na ogół lepszym wyborem ze względu na łatwe wyznaczenie przewidywanego czasu działania oraz możliwości pracy na bardzo skomplikowanych zestawach danych.