Wojciech Michaluk grupa nr 6 godz. 13:00-14:30 czw. A

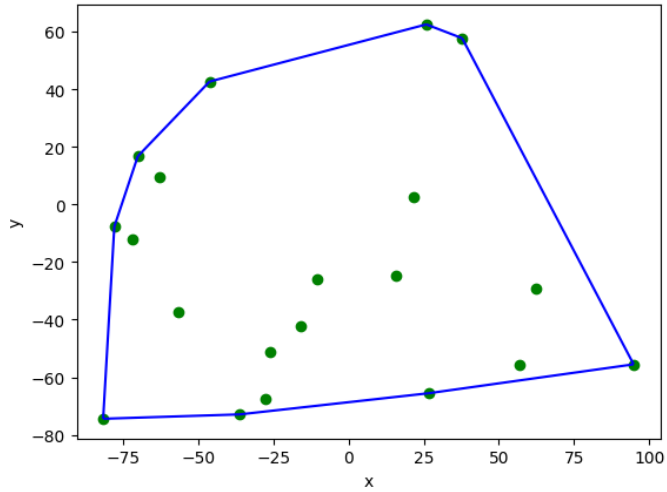
Sprawozdanie z laboratorium nr 2: Otoczka wypukła

Data wykonania: 19.10.2023r. Data oddania: 18.11.2023r.

1. Cel ćwiczenia

Celem ćwiczenia jest implementacja algorytmów Grahama i Jarvisa znajdowania otoczki wypukłej i porównanie ich złożoności czasowej oraz zastosowanie ich do znalezienia otoczek wypukłych dla wybranych zbiorów punktów, a także wizualizacja i opracowanie wyników.

2. Wstęp teoretyczny



Rys. 2.1 Przykładowy zbiór punktów wraz z ich otoczką

Podzbiór płaszczyzny Q nazywamy wypukłym ⟺ dla dowolnej pary punktów $p, q \in Q$ odcinek pq jest całkowicie zawarty w Q.

Otoczka wypukła $CH(Q)$ zbioru Q jest najmniejszym wypukłym zbiorem zawierającym Q. Podczas tego laboratorium wyznaczę otoczki wypukłe, używając algorytmów Grahama i Jarvisa. Ich krótkie opisy przedstawiam poniżej.

Algorytm Jarvisa

Algorytm Jarvisa oblicza otoczkę wypukłą dla zbioru punktów Q poprzez technikę zwaną owijaniem paczki/prezentu (package/gift wrapping). Algorytm ten buduje sekwencję $H = \langle p_0 p_1 \dots p_m \rangle$ będącą wierzchołkami $CH(Q)$, uporządkowanymi przeciwnie do ruchu wskazówek zegara. Zasada działania:

1) Znajdź punkt p_0 o najmniejszej współrzędnej y (najmniejszej współrzędnej x dla punktów o tej samej współrzędnej y)
2) Każdy następny punkt p_i znajdź jako punkt, który tworzy najmniejszy kąt w odniesieniu do poprzedniego punktu (dla kilku punktów tworzących ten sam kąt – weź najdalszy)
3) Zakończ działanie algorytmu, gdy następny znaleziony punkt jest punktem p_0. Zbudowana sekwencja H zawiera wierzchołki otoczki wypukłej

Algorytm Grahama

Algorytm Grahama tworzy otoczkę wypukłą poprzez utrzymywanie stosu S, w którym znajdują się punkty, które mogą, ale nie muszą tworzyć otoczki wypukłej. Za każdym razem jest wstawiany na stos jeden punkt ze zbioru punktów Q i jest on usuwany ze stosu, jeżeli nie jest punktem $CH(Q)$. Kiedy algorytm kończy się, stos S zawiera tylko punkty otoczki wypukłej $CH(Q)$, uporządkowane w kierunku przeciwnym do ruchu wskazówek zegara. Zasada działania:

1) Znajdź punkt p_0 (analogicznie jak w algorytmie Jarvisa)
2) Pozostały zbiór punktów posortuj wokół punktu p_0 przeciwnie do ruchu wskazówek zegara, otrzymując ciąg $\langle p_1 p_2 \dots p_m \rangle$ (jeżeli jakieś punkty tworzą ten sam kąt – weź ten najdalszy)
3) Stwórz pusty stos S, wstaw na niego punkty p_0, p_1, p_2
4) Dla pozostałych punktów $\langle p_3 \dots p_m \rangle$: dopóki punkty $S_{-2}S_{-1}p_i$ (punkt poniżej wierzchołka stosu, wierzchołek stosu, aktualnie badany punkt) tworzą tzw. lewostronny skręt, czyli trójkąt $S_{-2}S_{-1}p_i$ jest zorientowany zgodnie z ruchem wskazówek zegara, zdejmij wierzchołek stosu; następnie wstaw na stos punkt p_i
5) zwróć stos S - zawiera on punkty tworzące otoczkę

Szerszy opis dotyczący otoczki, jej zastosowań oraz algorytmów znajdują się w pliku z implementacją.

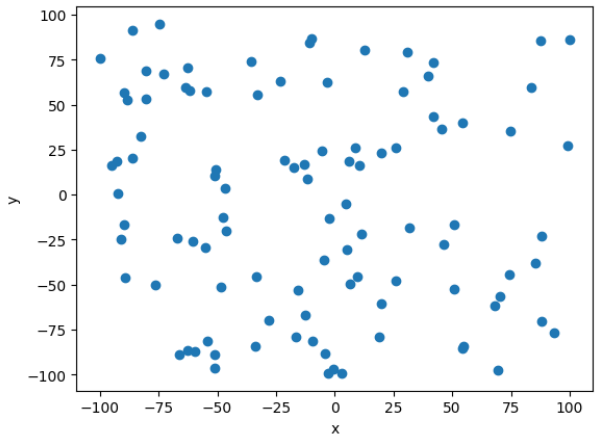
3. Metodologia, specyfikacja narzędzi i sprzętu

Zbiory punktów potrzebne do wykonania ćwiczenia zostały wygenerowane losowo z użyciem funkcji random.uniform() oraz random.random() z biblioteki numpy. Do sprawdzania przynależności punktu do otoczki wypukłej w algorytmie Grahama jest wykorzystywany wyznacznik macierzy 3x3, przedstawiony na poprzednim laboratorium. Przyjmuję tolerancję dla zera $\varepsilon = 10^{-24}$ oraz precyzję obliczania kąta wyznaczanego przez badane punkty i odległości między nimi jako 12 cyfr po przecinku. Wykresy przedstawiające wygenerowane zbiory punktów wraz z ich otoczką wypukłą, w tym wizualizacja kroków działania algorytmu, powstały przy użyciu biblioteki matplotlib oraz dzięki narzędziu przygotowanemu przez koło naukowe Bit. Program (w pliku „michaluk_kod_2.ipynb”) jest napisany w języku Python w środowisku Jupyter Notebook. Przedstawione wyniki pochodzą z uruchomienia programu na komputerze z systemem Windows 11 i procesorem Intel Core i5-8300H 2.30 GHz.

4. Program ćwiczenia

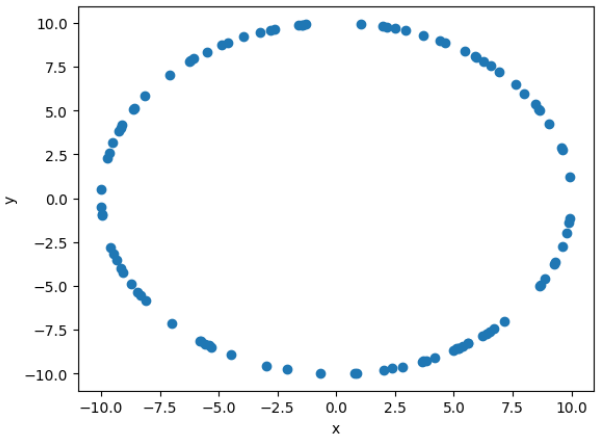
Na początku przygotowałem następujące zbiory punktów:

a) 100 losowych punktów o współrzędnych z przedziału [-100, 100] (zbiór A)



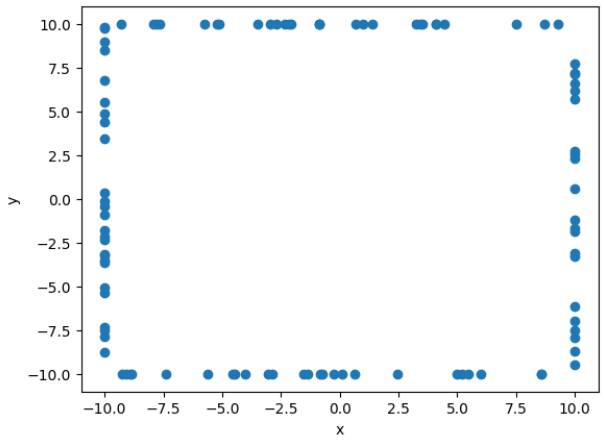
Rys. 4.1 Zbiór A

b) 100 losowych punktów leżących na okręgu o środku w punkcie (0,0) i promieniu R=10 (zbiór B)



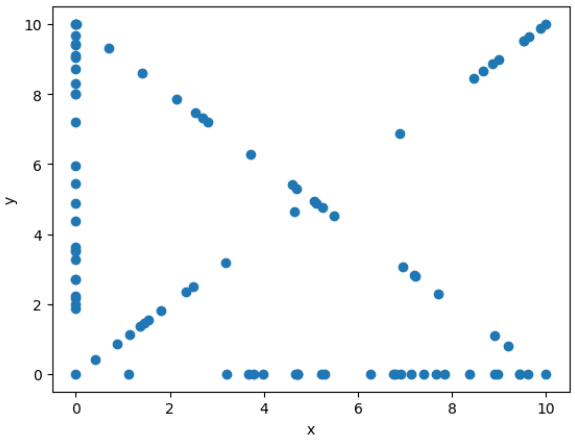
Rys. 4.2 Zbiór B

c) 100 losowych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10) (zbiór C)



Rys. 4.3 Zbiór C

d) zawierający wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz po 25 punktów na bokach leżących na osiach i po 20 punktów na przekątnych (zbiór D)



Rys 4.4 Zbiór D

Teraz przedstawię dla każdego zbioru jego otoczkę wypukłą. Rysunek będzie zawierał wierzchołki otoczki wyróżnione kolorem czerwonym i połączone czerwonymi krawędziami, a pozostałe wierzchołki będą zaznaczone kolorem niebieskim. W animacji dla algorytmu Grahama dodatkowo na brązowo zaznaczone są punkty już przetworzone, które **nie** należą do otoczki, a dla algorytmu Jarvisa na żółto zaznaczone są na bieżąco lokalnie znalezione „najlepsze" punkty. Animacja przedstawiająca poszczególne kroki algorytmu znajduje się w pliku z implementacją.

5. Otrzymane wyniki oraz ich analiza

W poniższej tabeli przedstawiam liczbę wierzchołków otoczki wypukłej obliczonej przez każdy z dwóch algorytmów dla każdego ze zbiorów.

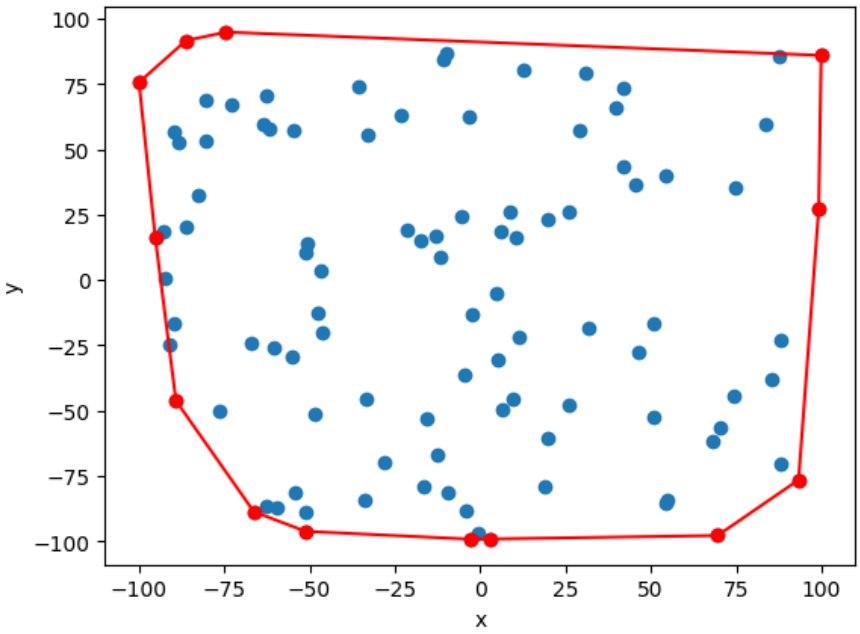
Zbiór punktów	Liczba wierzchołków otoczki – algorytm Grahama	Liczba wierzchołków otoczki – algorytm Jarvisa
Zbiór A	13	13
Zbiór B	100	100
Zbiór C	8	8
Zbiór D	4	4

Tabela 5.1 Liczba wierzchołków otoczki w obu algorytmach dla danych zbiorów punktów

Jak widać w tabeli 5.1, oba algorytmy wskazują taką samą liczbę wierzchołków otoczki dla każdego z przygotowanych zbiorów punktów. Okazuje się, że istotnie są to te same otoczki, dlatego tutaj umieszczę po jednym rysunku dla każdego zbioru, natomiast np. na potrzeby porównania, wszystkie wyznaczone otoczki dla obu algorytmów znajdują się w pliku z implementacją.

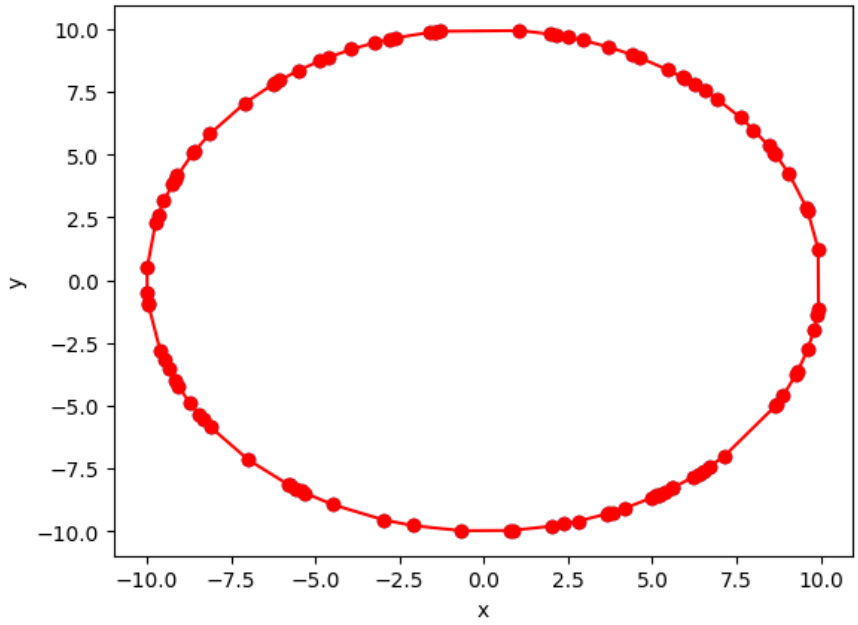
Poniżej oraz na następnych stronach znajdują się graficzne przedstawienia otoczki wypukłej.

Zbiór A



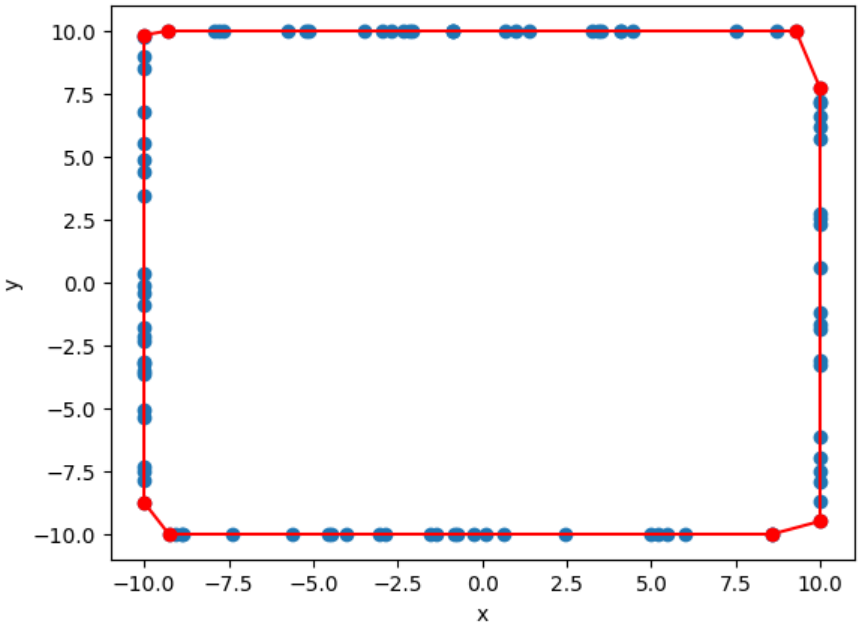
Rys. 5.2 Otoczka wypukła zbioru A wyznaczona przez algorytmy Grahama i Jarvisa

Zbiór B



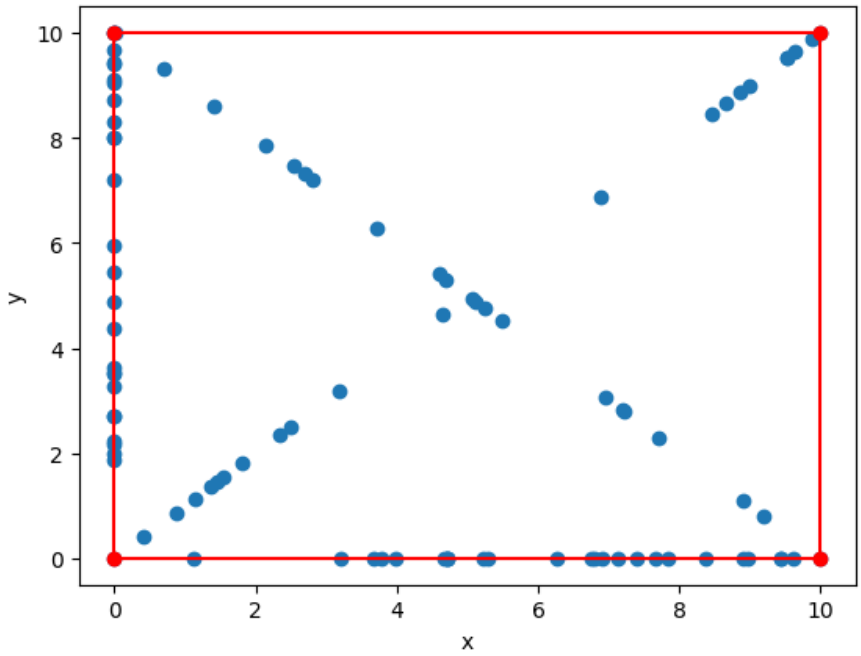
Rys. 5.3 Otoczka wypukła zbioru B obliczona przez algorytmy Grahama i Jarvisa

Zbiór C



Rys. 5.4 Otoczka wypukła zbioru C obliczona algorytmami Grahama i Jarvisa

Zbiór D



Rys. 5.5 Otoczka wypukła zbioru D wyznaczona algorytmami Grahama i Jarvisa

Porównanie złożoności czasowej algorytmów

Teoretyczna złożoność czasowa algorytmu Grahama to $O(nlogn)$, natomiast algorytmu Jarvisa $O(kn)$, gdzie n oznacza liczbę wierzchołków zbioru, dla którego wyznaczamy otoczkę, a k to liczba wierzchołków otoczki. Pesymistycznie zatem algorytm Jarvisa ma złożoność $\Theta(n^2)$.

Aby porównać czasy obliczania otoczki, przygotuję **zmodyfikowane** zbiory punktów:

- Nowy zbiór A: n losowych punktów o współrzędnych z przedziału [-1000, 1000]
- Nowy zbiór B: m losowych punktów leżących na okręgu o środku w punkcie (100,100) i promieniu R=500
- Nowy zbiór C: n losowych punktów leżących na bokach prostokąta o wierzchołkach (-200,50), (-200,-150), (100,-150), (100,50)
- Nowy zbiór D: zawierający wierzchołki kwadratu (0,0), (100,0), (100,100), (0,100) oraz po n_1 punktów leżących na osiach i po n_2 punktów na przekątnych (łącznie n),

gdzie $n \in \{100,1000,5000,10000,50000\}$, $m \in \{10,100,500,1000,5000\}$ oraz n_1 i n_2 są równe odpowiednio $0{,}3 \cdot n - 1$ oraz $0{,}2 \cdot n - 1$.

Zmiany dotyczą głównie liczby punktów, natomiast specyfika zbiorów w kontekście obliczania otoczki wypukłej pozostaje podobna. Czemu dla zbioru B ma przyjmuję mniejszą liczbę punktów? Okaże się już za chwilę.

Poniżej znajduje się przedstawienie czasów działania algorytmów dla nowych zbiorów.

Zbiory punktów	Liczba punktów n				
	100	1000	5000	10000	50000
Nowy zbiór A	0,0010 s	0,0060 s	0,0319 s	0,0618 s	0,3441 s
Nowy zbiór C	0,0020 s	0,0060 s	0,0279 s	0,0618 s	0,3121 s
Nowy zbiór D	0,0000 s	0,0050 s	0,0280 s	0,0479 s	0,2952 s

Tabela 5.6 Czasy wykonania algorytmu Grahama dla różnych zbiorów

Zbiory punktów	Liczba punktów m				
	10	100	500	1000	5000
Nowy zbiór B	0,0010 s	0,0010 s	0,0030 s	0,0079 s	0,0369 s

Tabela 5.7 Czasy wykonania algorytmu Grahama dla zbiorów „podobnych do B" o różnej liczbie punktów

Zbiory punktów	Liczba punktów n				
	100	1000	5000	10000	50000
Nowy zbiór A	0,0050 s	0,0738 s	0,4378 s	1,1867 s	4,5718 s
Nowy zbiór C	0,0030 s	0,0309 s	0,1536 s	0,3381 s	1,5758 s
Nowy zbiór D	0,0020 s	0,0160 s	0,1097 s	0,1576 s	0,8058 s

Tabela 5.8 Czasy wykonania algorytmu Jarvisa dla różnych zbiorów

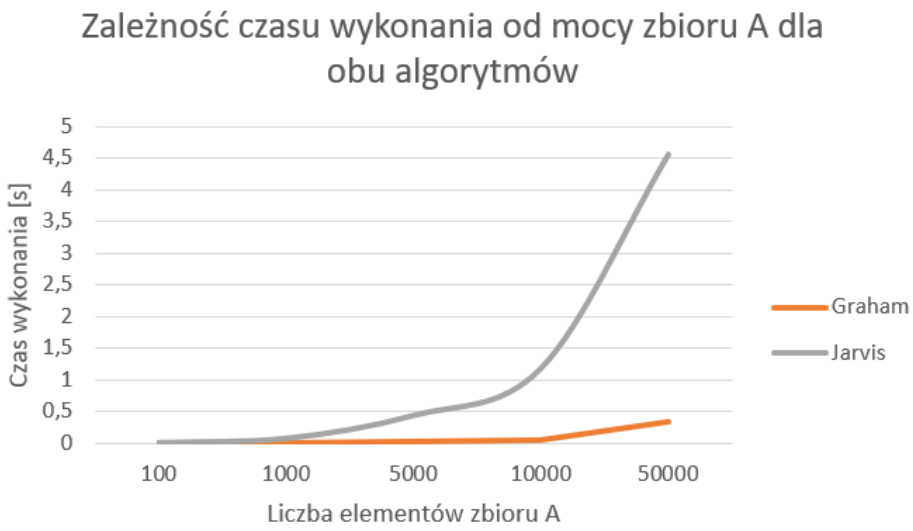
Zbiory punktów	Liczba punktów m				
	10	100	500	1000	5000
Nowy zbiór B	0,0000 s	0,0519 s	1,3325 s	6,1270 s	138,6681 s

Tabela 5.9 Czasy wykonania algorytmu Jarvisa dla zbiorów „podobnych do B" o różnej liczbie punktów

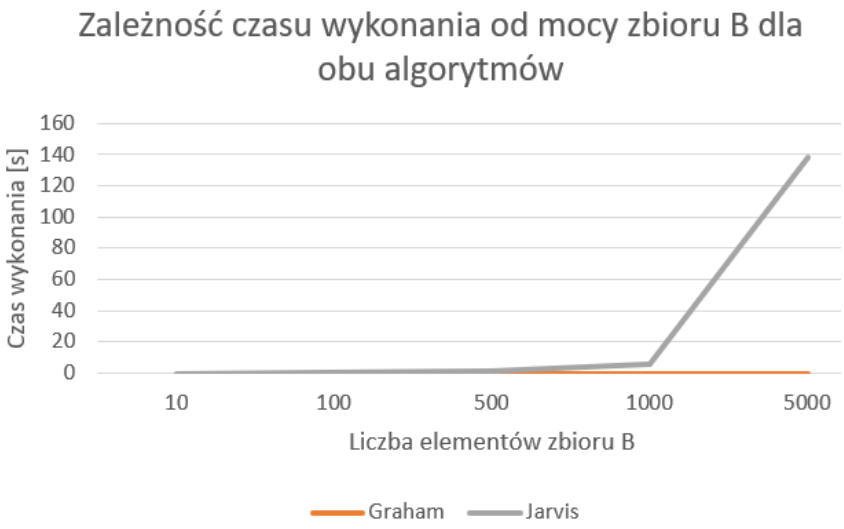
Uwaga: Czas 0,0000 s oznacza, że czas wykonania był mniejszy niż $0{,}5 \cdot 10^{-5}$ s, ale nie zerowy.

Analizując wyniki z tabel 5.6 – 5.9, dochodzę do wniosku, że w przypadku nowych zbiorów C oraz D algorytm Jarvisa radzi sobie **nie aż tak źle** w porównaniu do algorytmu Grahama. Jednakże, te zbiory mają charakter „przyjazny" temu algorytmowi – do otoczki należy mało punktów w porównaniu do mocy zbioru. Dla „neutralnego" zbioru A różnica już jest duża, a brutalną prawdę obnaża nowy zbiór B (i wszystkie o podobnej specyfice) – różnica jest kolosalna. Dla analogicznego zbioru 50000 punktów na rezultat należałoby czekać… szacując, jakieś 2 godziny. Oczywiście jest to czas osiągalny za życia człowieka, ale pozwoliłem sobie na przyjęcie zbiorów o nieco mniejszej mocy dla badania złożoności zbiorów „typu B" – tendencja i tak jest widoczna.

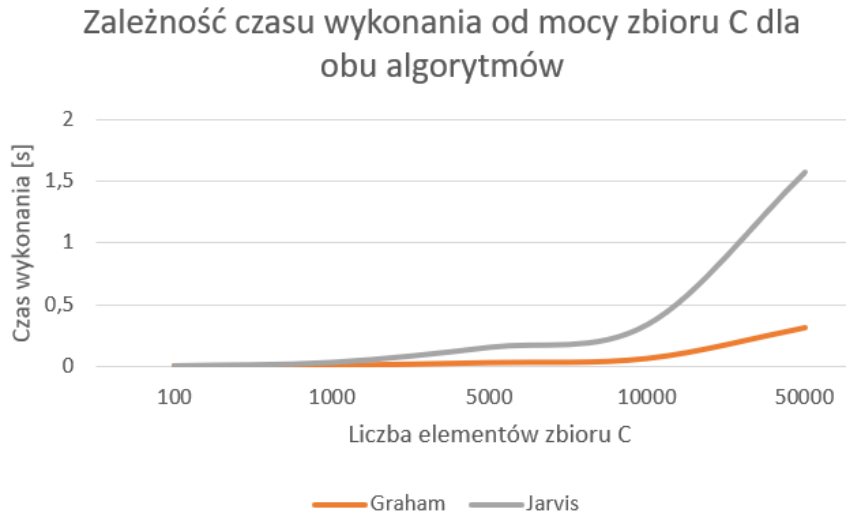
Na następnych stronach umieszczam wykresy przedstawiające zależność czasu wykonania od liczby punktów dla obu algorytmów.



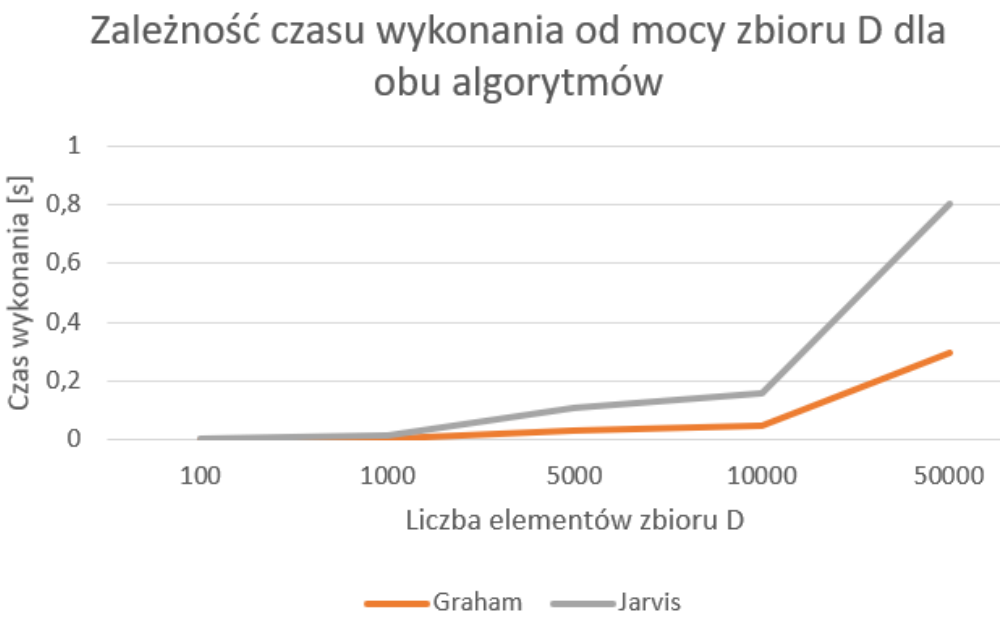
Rys. 5.10
Wykres porównawczy czasów obliczania otoczki dla zbioru typu A dla obu algorytmów



Rys. 5.11
Wykres porównawczy czasów obliczania otoczki dla zbioru typu B dla obu algorytmów



Rys. 5.12
Wykres porównawczy czasów obliczania otoczki dla zbioru typu C dla obu algorytmów



Rys. 5.13 Wykres porównawczy czasów obliczania otoczki dla zbioru typu D dla obu algorytmów

W teorii dla zbiorów typu C oraz typu D dla większej liczby punktów algorytm Grahama powinien obliczać otoczkę dłużej niż algorytm Jarvisa. W przypadku mojego programu, prawdopodobnie ze względu na dość duży współczynnik stałej w algorytmie Jarvisa, nie doszło do tego. Niemniej jednak możliwe, że dla większego n nastąpiłoby „przecięcie” i później już algorytm Jarvisa radziłby sobie lepiej niż algorytm Grahama.

Warto też zwrócić uwagę na skalę osi poziomej – nie jest ona liniowa, więc większe nachylenie odcinka niekoniecznie świadczy o wyższym rzędzie / stałej w złożoności. Uznałem, że połączę punkty wykresu ze sobą, aby zwiększyć czytelność wykresu.

6. Podsumowanie i wnioski

Na podstawie przeprowadzonych testów i wyników można wywnioskować, że dla zaproponowanych „podstawowych” zbiorów danych algorytmy radziły w sobie miarę dobrze. Myślę, że ewentualne niedociągnięcia mogły mieć związek z precyzją przechowywania liczb rzeczywistych w pamięci komputera (np. niektóre współliniowe punkty tworzące ten sam kąt nie były wykluczane, bo wartość wyznacznika nie mieściła się w zadanej tolerancji). Przyjęta przeze mnie tolerancja dla zera została „wyznaczona” eksperymentalnie, metodą prób i błędów. Kierowałem się głównie tym, aby wyniki dla wszystkich testów zaproponowanych przez AGH Bit były poprawne, mając nadzieję, że są one rzetelnie przygotowane.

Uważam, że zaproponowano zbiory punktów o takiej specyfice, gdyż mogły być one problematyczne. Tak jak zbiór A jest w miarę „neutralny”, tak zbiór B jest wymagający, bo wszystkie punkty należą do otoczki wypukłej. Z kolei zbiory C i D zawierają wiele współliniowych punktów, co testuje eliminowanie wszystkich punktów z wyjątkiem tego najdalej położonego przy wyznaczaniu otoczki wypukłej. Wydaje mi się, że mój program podołał rzuconemu wyzwaniu.