Dariusz Marecik gr. 4, Pon. godz. 15:00 A

Data wykonania: 21.10.2024 Data oddania: 6.11.2024

Algorytmy Geometryczne - laboratorium 1

Otoczka wypukła

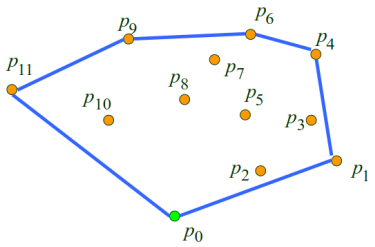
1. Cel ćwiczenia

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

2.1. Pojęcie otoczki wypukłej

Otoczką wypukłą $CH(S)$ dowolnego niepustego zbioru punktów S nazywamy najmniejszy zbiór wypukły zawierający S. Dla zbioru punktów na płaszczyźnie definicja uszczególnia się do najmniejszego wielokąta wypukłego zawierającego punkty ze zbioru S. Rysunek 1 pokazuje wizualizacje otoczki wypukłej dla zbioru punktów na płaszczyźnie



Rysunek 1: Otoczka wypukła (punkty $p_0, p_1, p_4, p_6, p_9, p_{11}$) dla zbioru na płaszczyźnie

Do otoczki należą jedynie punkty „ekstremalne”, bez punktów współliniowych „wewnętrznych”.

2.2. Metody znajdowania otoczki wypukłej

2.2.1. Algorytm Grahama

Algorytm Grahama polega na systematycznym usuwaniu wierzchołków wklęsłych ze stosu iterując po posortowanej tablicy punktów względem kąta.
Schemat działania algorytmu:

- Znajdź punkt o najmniejszej współrzędnej y. W przypadku remisu wybierz punkt o mniejszej współrzędnej x. Ten punkt staje się punktem początkowym p_0 otoczki wypukłej, a zarazem punktem odniesienia dla pozostałych punktów.
- Posortuj punkty stosując kryteriu kąta nachylenia odcinka p_0p_i względem osi x. Wśród punktów współliniowych wybierany jest najdalszy od punktu p_o.
- Utwórz stos, na który włóż punkt p_0 oraz dwa kolejne punkty z posortowanego zbioru. Dla każdego kolejnego punktu p_i sprawdzaj, czy tworzy on obrót w prawo czy w lewo względem poprzednich dwóch punktów na stosie: Jeśli punkt p_i tworzy obrót w lewo, dodaj go na stos. Jeśli tworzy obrót w prawo, usuń poprzedni punkt ze stosu, aż otrzymasz obrót w lewo.

- Kontynuuj dodawanie punktów na stos. Po przetworzeniu wszystkich punktów, punkty pozostałe na stosie tworzą wierzchołki otoczki wypukłej w kolejności przeciwnej do ruchu wskazówek zegara.

2.2.2. Algorytm Jarvisa

Algorytm Jarvisa znajduje otoczkę wypukłą dzięki technice zwanej owijaniem paczki/prezentu (package/gift wrapping). Algorytm ten buduje sekwencje $H =< p_0, p_1, ..., p_n >$ będącą wierzchołkami $CH(Q)$, za pomocą zachłannego wyboru kolejnego punktu będącego najbardziej „na prawo" względem ostatnio dodanego punktu zbioru H. Schemat działania algorytmu:

- Wybierz punkt p_0 o najmniejszej współrzędnej y. W razie remisu wybieramy punkt o najmniejszej współrzędnej x. Ten punkt należy do otoczki wypukłej, więc dodaj go do zbioru H.
- Dla bieżącego punktu w otoczce, wyszukaj punkt, który tworzy najmniejszy dodatni kąt względem odcinka prowadzącego do aktualnego punktu w H. W razie remisu preferowany jest punkt bardziej oddalony względem ostatnio dodanego punktu do zbioru H.
- Dodaj znaleziony punkt do zbioru H.
- Ustaw nowy punkt jako bieżący i powtarzaj kroki 2 i 3, aż wybranym punktem stanie się punkt początkowy p_0. Oznacza to zamknięcie otoczki, a zbiór H stanowi pełną otoczkę wypukłą.

3. Dane techniczne

3.1. Metodologia

Zbiory danych analizowane w niniejszym opracowaniu bazują na punktach generowanych losowo za pomocą funkcji `random.uniform()` z biblioteki *numpy*. W pierwszym etapie algorytm sortuje punkty, przy czym wyznacznik stanowi kryterium porównawcze. Dla punktów współliniowych porządek ustalany jest na podstawie odległości od punktu początkowego p_0. Następnie algorytm eliminuje współliniowe punkty względem p_0 zgodnie z ustalonym porządkiem sortowania, po czym przechodzi do głównej pętli obliczeniowej.

Do sortowania wykorzystano funkcję biblioteczną `sort()`, do której zaimplementowałem własną funkcję porównującą `compare_for_Graham()`. Funkcja ta jest dołączana do `sort()` za pomocą `cmp_to_key()` z biblioteki *functools*.

Algorytm Jarvisa, w poszukiwaniu wierzchołka najbardziej najbardziej optymalnego, wybiera punkt, dla którego nie istnieje inny punkt z zestawu położony na prawo od krawędzi utworzonej przez ostatnie punkty otoczki. W przypadku współliniowości wybiera punkt najbardziej oddalony od ostatniego punktu otoczki.

Funkcje implementujące algorytmy zwracają listę punktów należących do otoczki. Obliczenia wyznacznika w obu funkcjach opierają się na macierzy $3x3$. Tolerancję błędu dla wartości zerowej ustalono eksperymentalnie na poziomie $\varepsilon = 10^{-24}$, ponieważ większe wartości prowadziły do błędnej klasyfikacji punktów leżących na okręgu jako współliniowych.

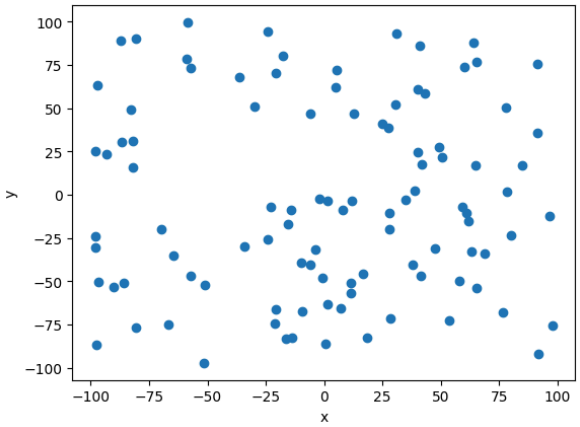
3.2. Specyfikacja narzędzi i sprzętu

Do generowania obrazów przedstawiających znalezioną otoczkę, tabel i gif'ów przedstawiających krok po kroku działanie poszczególnych algorytmów została wykorzystana biblioteka *matplotlib*, *pandas* i *functools* oraz narzędzie przygotowane przez *koło naukowe Bit*. Program został napisany w języku Python 3.10 w środowisku *Jupiter Notebook*. Wykorzystany procesor do zebrania danych to Intel Core i5-12500H 4.5 GHz pracujący w systemie Linux Mint 21.4.

4. Wykonanie ćwiczenia

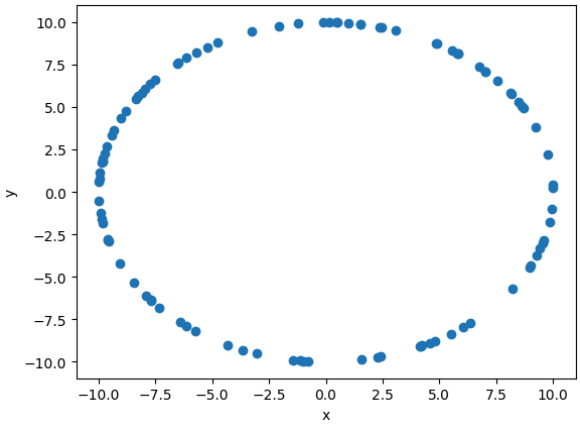
Na początku przygotowałem 4 zbiory danych do analizy:

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



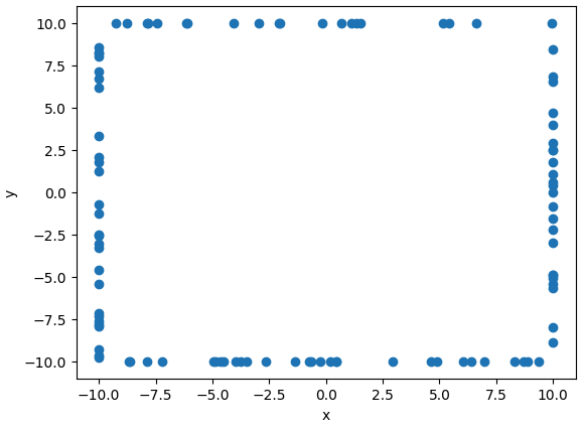
Rysunek 2: Zbiór A

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



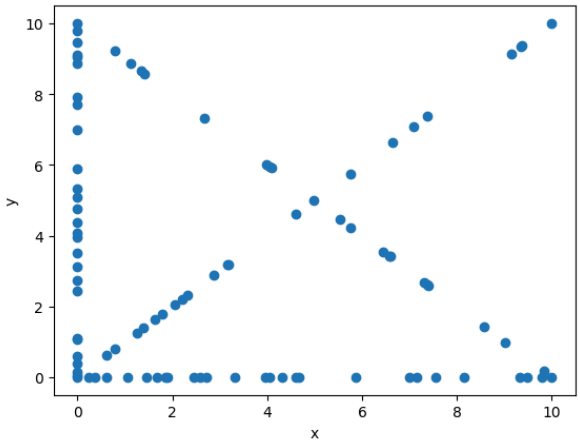
Rysunek 3: Zbiór B

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



Rysunek 4: Zbiór C

d) 25 punktów leżących na bokach przyległych do osi kwadratu o wierzchołkach (0,0), (10,0), (10,10), (0,10) wraz z samymi wierzchołkami oraz 20 punktami na każdej przekątnej (zbiór D)



Rysunek 5: Zbiór D

Następnie dla każdego zbioru punktów uruchomiłem zarówno algorytm Grahama jak i Jarvisa oraz dokonałem wizualizacji działania każdego z nich. Następnie przeprowadziłem testy wydajnościowe obu algorytmów na dużo większych zbiorach testowych. Zostały one stosowanie sprecyzowane w punkcie (5.2.2). Otrzymane wyniki zebrałem w odpowiednich tabelach i zwizualizowałem je na wykresach. Gify pokazujące działania obu algorytmów zostały zapisane w archiwum *marecik_2_gify_algorytmow.zip*. Są również wyrenderowane w kodzie w pliku *marecik_kod_2.ipynb*.

5. Opracowanie danych

5.1. Zbiory niezmodyfikowane

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

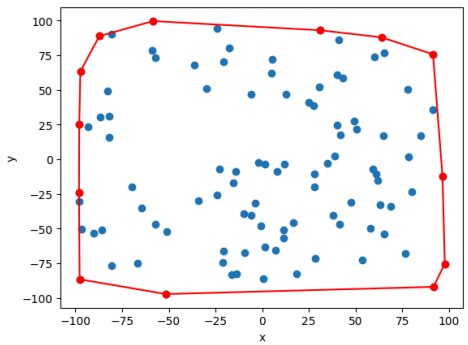
Tabela 1: Rezultat działania algorytmów znajdowania otoczki wypukłej dla niezmodyfikowanych zbiorów

Tabela 1 przedstawia liczbę punktów otoczki znalezionych przez algorytm Grahama oraz Jarvisa. Możemy z niej wyczytać, że znalezione przez oba algorytmy otoczki mają tyle samo elementów dla poszczególnych zbiorów. Okazuje się, że w istocie obie metody znalazły takie same, poprawne otoczki, co zostało sprawdzone porównując listy zwróconych punktów przez oba algorytmy w programie *Jupiter Notebook*.
Zbiór B jest okręgiem, więc nie istnieje taka krawędź pomiędzy punktami, żeby nie zawierała się we wnętrzu okręgu. Oznacza to, że wszystkie punkty tego zbioru muszą znajdować się w otoczce wypukłej, co obserwujemy na Rysunek 7.
Zbiór C zawiera punkty na bokach prostokąta. Znikoma jest szansa, że wylosowany punkt jest wierzchołkiem kwadratu, więc oczekujemy 8 punktów będących skrajnymi punktami każdego boku. Nasze przypuszczenia znajdują odzwierciedlenie na Rysunek 8.
Zbiór D zawiera wierzchołki kwadratu oraz punkty na jego bokach i przekątnych. Otoczka powinna zawierać tylko 4 punkty będące wierzchołkami naszego kwadratu i ten zbiór jest znajdowany w obu algorytmach co widzimy w Tabela 1 i Rysunek 9.

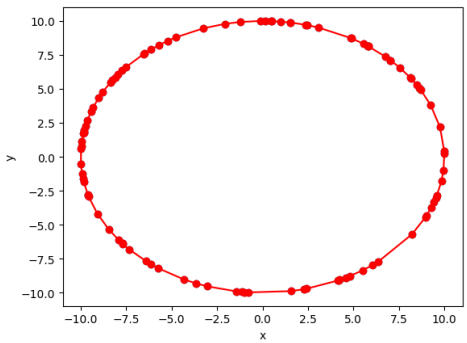
Głównym problemem zbiorów C i D są punkty współliniowe z którymi analizowane algorytmu musiały sobie poradzić.

Rysunek 6, Rysunek 7, Rysunek 8 i Rysunek 9 przedstawiają wizualizacje znalezionych otoczek wypukłych przez oba algorytmy.
Legenda wykresów:

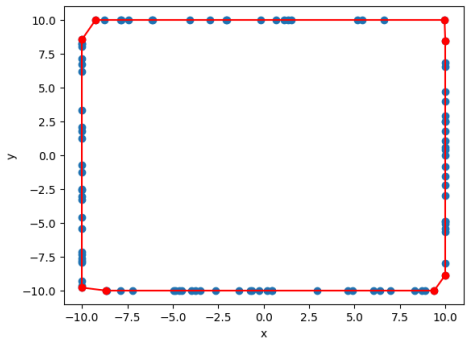
- *kolor niebieski* - punkty należące do zbioru
- *kolor czerwony* - punkty należące do otoczki oraz krawędzie je łączące



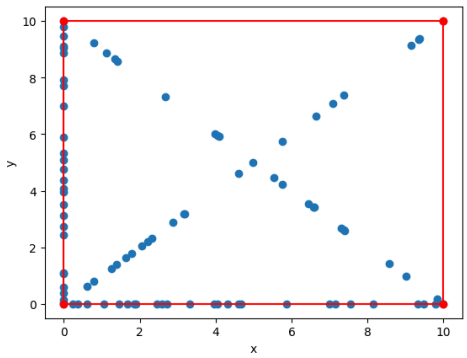
Rysunek 6: Wizualizacja otoczki wypukłej dla zbioru A



Rysunek 7: Wizualizacja otoczki wypukłej dla zbioru B



Rysunek 8: Wizualizacja otoczki wypukłej dla zbioru C



Rysunek 9: Wizualizacja otoczki wypukłej dla zbioru D

5.2. Analiza złożoności czasowej algorytmów

Złożoność czasowa algorytmu Grahama i algorytmu Jarvisa dla obliczania otoczki wypukłej różni się ze względu na ich podejście oraz zależność od liczby punktów w otoczce.

Złożoność czasowa algorytmu Grahama jest determinowana przez operację sortowania punktów względem kąta, co zwykle wymaga $O(n \log n)$ czasu, gdzie n to liczba punktów wejściowych. Po sortowaniu każdy punkt jest analizowany tylko raz w procesie konstruowania otoczki, co dodaje liniową składową $O(n)$. Jednak ze względu na dominującą złożoność sortowania, ostateczna złożoność algorytmu wynosi $O(n \log n)$. Algorytm Grahama jest zatem stabilny względem danych wejściowych – zarówno w przypadku korzystnego, jak i niekorzystnego rozmieszczenia punktów.

Algorytm Jarvisa: Złożoność czasowa algorytmu Jarvisa wynosi $O(nh)$, gdzie h to liczba punktów na otoczce wypukłej. Gdy liczba punktów na otoczce wypukłej h jest stała (np. kilka punktów w centralnym położeniu tworzy otoczkę), złożoność wynosi $O(n)$. W takiej sytuacji algorytm przechodzi przez każdy punkt wejściowy tylko raz, bez dodatkowych przeskoków między punktami tworzącymi otoczkę. Gdy $h = O(n)$, czyli niemal wszystkie punkty leżą na otoczce wypukłej, złożoność staje się kwadratowa, czyli $O(n^2)$. W takim przypadku algorytm musi przeanalizować każdą parę punktów, aby wyznaczyć kolejny punkt na otoczce, co znacząco zwiększa liczbę operacji.

5.2.1. Zbiory testujące

Aby rzetelnie przetestować złożoność czasową algorytmów potrzebowałem większą liczbę zbiorów danych, które obejmowały szerszy zakres przyjmowanych zmiennych oraz większą liczbę punktów. Zbiory testujące zostały wygenerowane na bazie powyższych zbiorów z poniższymi modyfikacjami:

- Współrzędne przedziałów ze **zbioru A** zostały powiększone do [−1000,1000]
- Środek okręgu ze **zbioru B** został przeniesiony do punktu (100,100), a promień powiększony do 2000
- Współrzędne wierzchołków prostokąta ze **zbioru C** zostały zamienione na (-500, −200), (100, −200), (100, 300) i (-500, 300)
- Współrzędne wierzchołków kwadratu ze **zbioru D** zostały zamienione na (0, 0), (400, 0), (400, 400) i (0, 400), a stosunek ilości punktów na przekątnych do punktów na bokach wynosił $\frac{2}{3}$.

Dla tak zmodyfikowanych zbiorów wygenerowałem punkty o łącznej liczbie kolejno:

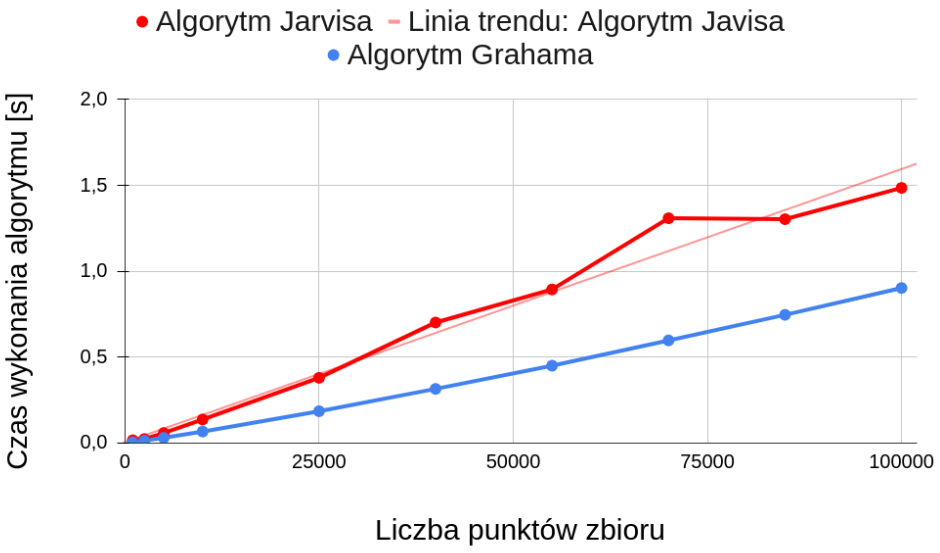
- [1000, 2500, 5000, 10000, 25000, 40000, 55000, 70000, 85000 i 100000] - dla zmodyfikowanych zbiorów A, C, D
- [100, 500, 1000, 2000, 3000, 5000, 7000, 9000, 11000, 13000] - dla zmodyfikowanego zbioru B

Podczas testów weryfikowano zgodność liczby punktów otoczek zwracanych przez oba algorytmy.

5.2.2. Wyniki testów na bazie zbioru A

Liczność zbiorów		Czas wykonania [s]	
Zbiór testujący	Otoczka wypukła	Algorytm Grahama	Algorytm Jarvisa
1000	18	0,0048	0,0164
2500	20	0,0137	0,0235
5000	24	0,0306	0,0586
10000	28	0,0673	0,1372
25000	29	0,1855	0,3798
40000	34	0,3155	0,7016
55000	31	0,4505	0,8943
70000	35	0,5976	1,3088
85000	29	0,7466	1,3033
100000	28	0,9026	1,4852

Tabela 2: Czasy wykonania algorytmów Grahama i Jarvisa dla zbioru A



Rysunek 10: Wykres porównawczy czasu wyznaczenia otoczki wypukłej dla obu algorytmów dla zbioru A

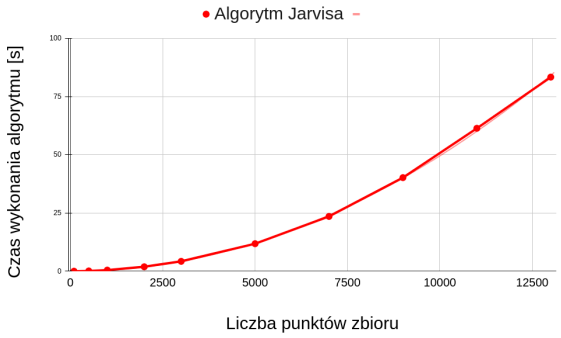
Z danych przedstawionych w Tabela 2 oraz na Rysunek 10 wynika, że algorytm Grahama wykazuje znaczną przewagę wydajności nad algorytmem Jarvisa w przypadku losowego zbioru punktów. Wykres działania algorytmu Grahama układa się w linię bliską funkcji liniowej, co jest zgodne z teoretycznymi oczekiwaniami. Wynika to z faktu, że na wąskim zakresie danych różnica między funkcją liniową a funkcją o złożoności liniowo-logarytmicznej jest trudna do zauważenia.

Wykres przedstawiający działanie algorytmu Jarvisa wykazuje odchylenie od idealnie liniowego przebiegu. Zjawisko to wynika z liczby punktów w zbiorze, które uczestniczą w tworzeniu otoczki wypukłej. W przypadkach, gdy liczba punktów generujących otoczkę wypukłą pozostaje względnie stała, wykres przyjmuje charakter zbliżony do linii prostej. Największe odstępstwo od tej regularności obserwuje się dla zbioru zawierającego 70 000 punktów. Powodem tego zjawiska jest wysoka liczba punktów wchodzących w skład otoczki wypukłej – około 35, w porównaniu do średniej wynoszącej około 30 dla zbiorów sąsiednich. Stanowi to wzrost o niespełna 20%, co w konsekwencji wydłuża czas działania algorytmu na tyle, że w tej sekcji wykresu nie można zaobserwować przebiegu liniowego.

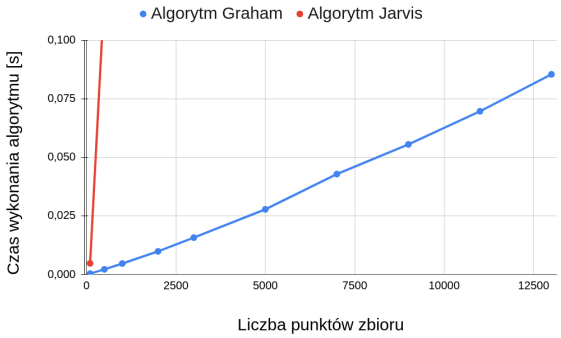
5.2.3. Wyniki testów na bazie zbioru B

Liczność zbiorów		Czas wykonania [s]	
Zbiór testujący	Otoczka wypukła	Algorytm Grahama	Algorytm Jarvisa
100	100	0,0003	0,0047
500	500	0,0022	0,1207
1000	1000	0,0046	0,4700
2000	2000	0,0098	1,9011
3000	3000	0,0157	4,2426
5000	5000	0,0278	11,8262
7000	7000	0,0428	23,5554
9000	9000	0,0555	40,2299
11000	11000	0,0696	61,4292
13000	13000	0,0854	83,4285

Tabela 3: Czasy wykonania algorytmów Grahama i Jarvisa dla zbioru B



Rysunek 11: Wykres algorytmu Jarvisa dla zbioru B



Rysunek 12: Wykres porównawczy czasu wyznaczenia otoczki wypukłej dla obu algorytmów dla zbioru B

Zbiór B stanowi układ punktów tworzących okrąg, co oznacza, że wszystkie punkty należą do otoczki wypukłej. Charakterystyka tego zbioru nie stanowi problemu dla algorytmu Grahama, o czym świadczą wyniki przedstawione na Rysunek 12 oraz Tabela 3. Wykres czasu działania algorytmu Grahama w tym przypadku zachowuje strukturę bliską liniowej, podobnie jak dla zbioru A. Natomiast dla algorytmu Jarvisa, widocznego na Rysunek 11, czas wykonania przybiera charakter kwadratowy. Wynika to z zależności algorytmu Jarvisa od liczby punktów tworzących otoczkę wypukłą. W przypadku okręgu, gdy $h = n$, złożoność czasowa algorytmu wzrasta do $O(n^2)$.

Pod względem czasu wykonania algorytm Grahama znacząco przewyższa algorytm Jarvisa. Dla zbioru zawierającego 13 000 punktów, algorytm Jarvisa jest ponad 976 razy wolniejszy od algorytmu Grahama, a różnica ta wzrasta w sposób kwadratowy wraz ze wzrostem liczby punktów w zbiorze. Nawet dla zbioru o małej liczności algorytm Jarvisa nie jest w stanie nawiązać równej walki co widać na Rysunek 12

5.2.4. Wyniki testów na bazie zbioru C

Liczność zbiorów		Czas wykonania [s]	
Zbiór testujący	Otoczka wypukła	Algorytm Grahama	Algorytm Jarvisa
1000	8	0,0041	0,0042
2500	8	0,0120	0,0082
5000	8	0,0260	0,0158
10000	8	0,0552	0,0317
25000	8	0,1583	0,0798
40000	8	0,2793	0,1274
55000	8	0,3808	0,1742
70000	8	0,5012	0,2235
85000	8	0,6285	0,2708
100000	8	0,7652	0,3200

Tabela 4: Czasy wykonania algorytmów Grahama i Jarvisa dla zbioru C

Rysunek 13: Wykres porównawczy czasu wyznaczenia otoczki wypukłej dla obu algorytmów dla zbioru C

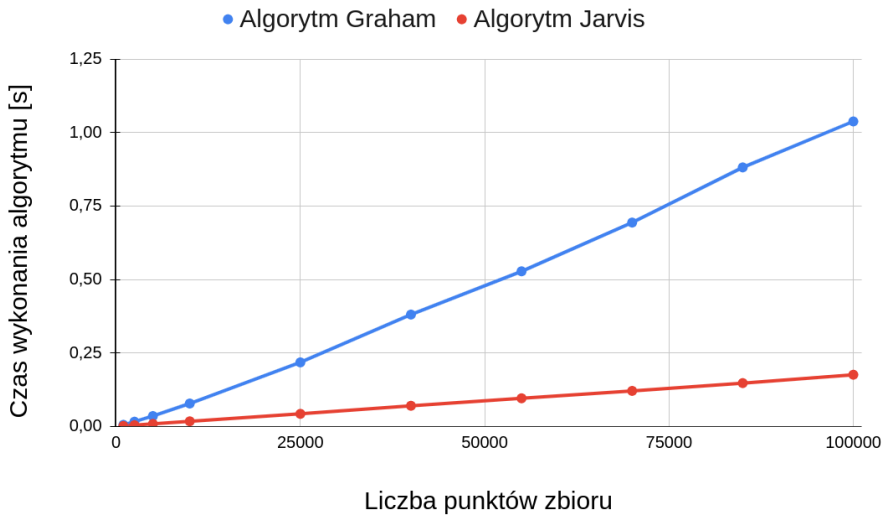
Wyniki przedstawione w tabeli Tabela 4 oraz na wykresie Rysunek 13 wskazują na ponad dwukrotną przewagę algorytmu Jarvisa nad algorytmem Grahama. Wykres dla algorytmu Jarvisa ma charakter liniowy, co jest zgodne z teoretycznymi przewidywaniami, ponieważ liczba punktów tworzących otoczkę jest ograniczona. Z kolei wykres algorytmu Grahama wykazuje bardziej liniowo-logarytmiczny kształt, co można zauważyć w rosnącym stosunku czasu wykonania algorytmu Grahama do czasu wykonania algorytmu Jarvisa dla coraz liczniejszych zbiorów.

Przewaga algorytmu Jarvisa wynika ze specyfiki badanego zbioru punktów. W przypadku prostokąta, liczba punktów należących do otoczki wypukłej jest stała i wynosi 8, niezależnie od liczby pozostałych punktów w zbiorze. W takiej konfiguracji złożoność czasowa algorytmu Jarvisa wynosi $O(n)$, natomiast algorytm Grahama osiąga złożoność $O(n \log n)$. Ta różnica staje się zauważalna nawet przy ograniczonym zakresie przypadków testowych, co uwidacznia przewagę algorytmu Jarvisa dla tego konkretnego zbioru.

5.2.5. Wyniki testów na bazie zbioru D

Liczność zbiorów		Czas wykonania [s]	
Zbiór testujący	Otoczka wypukła	Algorytm Grahama	Algorytm Jarvisa
1000	4	0,0055	0,0018
2500	4	0,0161	0,0043
5000	4	0,0353	0,0088
10000	4	0,0780	0,0174
25000	4	0,2179	0,0428
40000	4	0,3806	0,0703
55000	4	0,5278	0,0957
70000	4	0,6936	0,1209
85000	4	0,8813	0,1475
100000	4	1,0377	0,1760

Tabela 5: Czasy wykonania algorytmów Grahama i Jarvisa dla zbioru B



Rysunek 14: Wykres porównawczy czasu wyznaczenia otoczki wypukłej dla obu algorytmów dla zbioru D

Zbiór D składa się z punktów leżących na przekątnych kwadratu, na jego bokach przyległych do obu osi oraz z wierzchołków kwadratu. Obecność tylko czterech wierzchołków jako punktów tworzących otoczkę wypukłą znacząco przyspiesza działanie algorytmu Jarvisa. W tabeli Tabela 5 i na wykresie Rysunek 14 zauważamy podobną sytuację jak w przypadku zbioru C – zarówno algorytm Jarvisa wykazuje strukturę liniową zaś algorytm Grahama - liniowo-logarytmiczną.

Algorytm Jarvisa, z uwagi na ograniczoną liczność punktów tworzących otoczkę wypukłą, ma w tym przypadku złożoność $O(n)$, podczas gdy algorytm Grahama osiąga złożoność $O(n \log n)$. Warto zaznaczyć, że przewaga algorytmu Jarvisa jest tutaj wyraźniejsza niż dla zbioru C. Dla 100 000 punktów w zbiorze C, algorytm Jarvisa był jedynie 2,4 raza szybszy od Grahama, natomiast dla zbioru D jego przewaga wzrasta do 5,9 razy.

Różnica ta wynika z liczby punktów tworzących otoczkę wypukłą – dla zbioru D jest ona dwa razy mniejsza niż dla zbioru C, co istotnie zmniejsza stałą w złożoności czasowej algorytmu Jarvisa, czyniąc go jeszcze bardziej efektywnym w tym kontekście.

6. Podsumowanie

Na podstawie przeprowadzonych eksperymentów dotyczących algorytmów Grahama i Jarvisa, można sformułować następujące wnioski:

- Wydajność algorytmów zależna od typu zbioru – Algorytm Grahama jest bardziej wydajny przy dużych zbiorach punktów o przypadkowym rozmieszczeniu, co wynika z jego złożoności $O(n \log n)$. Algorytm Jarvisa okazuje się natomiast efektywniejszy przy mniejszych zbiorach punktów brzegowych (małej liczbie punktów tworzących otoczkę), ponieważ jego złożoność wynosi $O(nh)$, gdzie h to liczba punktów w otoczce.
- Optymalizacja dla regularnych kształtów – Algorytm Jarvisa jest korzystniejszy dla zbiorów o regularnym kształcie, takich jak prostokąty czy kwadraty, gdzie liczba punktów otoczki jest niewielka. W takich przypadkach wyprzedza algorytm Grahama, dzięki niższej stałej złożoności.
- Przewaga Grahama przy dużej liczbie punktów otoczki – Dla zbiorów o punktach tworzących pełny okrąg, jak w przypadku zbioru B, algorytm Jarvisa staje się wolniejszy od Grahama, ze złożonością $O(n^2)$. Sytuacja ta występuje, gdy $h \approx n$, co znacząco zwiększa czas działania algorytmu Jarvisa w porównaniu do Grahama.
- Uniwersalność algorytmu Grahama – Algorytm Grahama wykazuje większą wszechstronność i stabilność przy różnych typach zbiorów, podczas gdy Jarvis jest lepiej dopasowany do specyficznych przypadków z małą liczbą punktów otoczki.

Podsumowując, wybór algorytmu zależy od charakterystyki badanego zbioru punktów – algorytm Grahama sprawdza się lepiej przy zbiorach o losowym rozkładzie punktów, a algorytm Jarvisa w zbiorach o regularnym kształcie i małej liczbie punktów na otoczce.