

Sprawozdanie z laboratorium 2. z algorytmów geometrycznych - Otoczka wypukła

Hubert Miklas

Grupa 5, Poniedziałek 15:00-16:30

Data wykonania ćwiczenia:

28-10-2024

Data oddania sprawozdania:

16-11-2024

1 Wstęp do ćwiczenia

1.1 Specyfikacja środowiska

System: Windows 10 Home

Procesor: Intel(R) Core(TM) i5-7400 CPU 3.00 GHz

Pamięć RAM: 24 GB

Karta Graficzna: NVIDIA GeForce GTX 1060

Środowisko: Jupyter Notebook, Python 3.12.7

1.2 Opis i cel ćwiczenia

Celem ćwiczenia jest wygenerowanie różnych zbiorów punktów na płaszczyźnie oraz wyznaczenie ich otoczki wypukłej, czyli najmniejszego wypukłego wielokąta, który zawiera wszystkie punkty ze zbioru. W tym celu zastosowano dwa algorytmy: Algorytm Jarvisa i Algorytm Grahama. Do obliczeń orientacji punktów względem prostych wyznaczanych przez pary punktów, zarówno dla sortowania kąтового, jak i "owijania prezentu," używany jest wyznacznik macierzy 2x2 równanie (1). Dla każdego algorytmu dokonano pomiaru czasu wykonania, aby porównać ich efektywność.

$$\det(\mathbf{a}, \mathbf{b}, \mathbf{c}) = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix} \quad (1)$$

2 Wstęp teoretyczny

Definicja otoczki wypukłej: Podzbiór Q płaszczyzny nazywamy wypukłym wtedy i tylko wtedy, gdy dla dowolnych dwóch punktów $p, q \in Q$, odcinek łączący p i q jest całkowicie zawarty w Q .

Otoczka wypukła zbioru Q , oznaczana jako $CH(Q)$, to najmniejszy wypukły zbiór zawierający Q . Wyznaczanie otoczki wypukłej ma szerokie zastosowanie, m. in. w grafice komputerowej, robotyce oraz systemach GIS.

W ćwiczeniu opisano i zaimplementowano dwa algorytmy wyznaczania otoczki wypukłej: Algorytm Grahama i Algorytm Jarvisa.

2.1 Algorytm Grahama

Algorytm Grahama wyznacza otoczkę wypukłą w złożoności $O(n \log_2 n)$, sortując punkty według kątowno względem punktu początkowego i iteracyjnie budując otoczkę. Kroki algorytmu są następujące:

1. **Inicjalizacja:** Tworzy pustą listę **hull**.
2. **Wybór punktu bazowego:** Znajduje punkt o najmniejszej współrzędnej y , a w przypadku remisu – o najmniejszej współrzędnej x . Punkt ten staje się punktem startowym i zamienia się miejscami z pierwszym elementem listy.
3. **Sortowanie kątowne:** Punkty są sortowane względem kąta biegunowego za pomocą funkcji `sorted()` oraz komparatora `cmp()`, który porównuje orientację punktów i uwzględnia współliniowość.
4. **Budowanie otoczki:** Algorytm iteruje po posortowanej liście punktów. Dla każdego punktu sprawdza, czy dodanie go do otoczki zachowuje wypukłość:
 - Jeśli kąt między dwoma ostatnimi punktami otoczki a obecnym punktem jest niewypukły, ostatni punkt jest usuwany z otoczki.
 - Proces ten powtarza się, dopóki ostatni punkt nie tworzy kąta wypukłego z obecnym punktem.
5. **Zakończenie:** Po iteracji zwracana jest lista punktów **hull**, które tworzą otoczkę wypukłą, oraz czas wykonania algorytmu.

Algorytm realizuje warunek orientacji za pomocą funkcji `det()`, czyli wyznacznika (1). Współliniowość rozstrzyga, porównując kwadrat odległości punktów tzn. im dalej znajduje się punkt, tym później będzie rozpatrywany

2.2 Algorytm Jarvisa

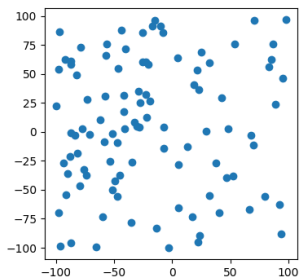
Algorytm Jarvisa, znany również jako algorytm "zawijania prezentu", działa w złożoności $O(nk)$, gdzie k to liczba punktów na otoczce wypukłej. Kroki implementacji są następujące:

1. **Inicjalizacja:** Tworzy pustą listę **hull** i wybiera punkt startowy (o najmniejszej współrzędnej y , a w przypadku remisu – o najmniejszej współrzędnej x).
2. **Iteracyjne wybieranie punktów:**
 - Dla każdego punktu w zbiorze wybiera ten, który jest najbardziej wysunięty w kierunku przeciwnym do ruchu wskazówek zegara względem obecnego punktu. Wybór realizowany jest przez funkcję `choose_next()` wykorzystującą `follows()`, która sprawdza orientację punktów.
 - Nowo wybrany punkt jest dodawany do listy **hull**.
3. **Zakończenie:** Algorytm kończy działanie, gdy ponownie osiągnięty zostaje punkt startowy. Zwraca listę punktów otoczki oraz czas wykonania algorytmu.

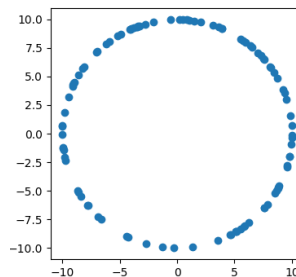
Algorytm Jarvisa iteracyjnie buduje otoczkę wypukłą, wybierając najbardziej zewnętrzne punkty w porządku geometrycznym, co czyni go szczególnie efektywnym dla zbiorów punktów o "rzadkiej" otoczce. Po obliczeniu otoczki wypukłej punkty zapisywane są w pliku tekstowym. Ostatni punkt z pierwszym punktem zamyka otoczkę.

3 Generowanie danych

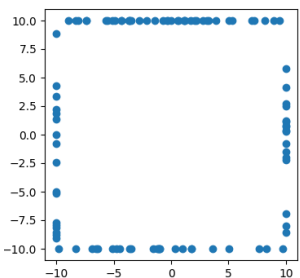
Poniżej znajdują się wykresy i opisy metod wykorzystanych do generowania zbiorów punktów użytych do obliczeń.



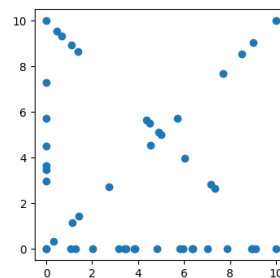
(a) Wygenerowano 100 losowych punktów w przedziale $[-100, 100]$ przy użyciu funkcji `uniform(-100, 100)` z biblioteki `random`. Wynikowa chmura punktów została zwizualizowana na Wykresie powyżej.



(b) W kolejnym zbiorze punkty są równomiernie rozmieszczone na okręgu o promieniu $R = 10$ i o środku w $(0, 0)$. Wizualizacja tych punktów znajduje się na wykresie powyżej.



(c) Wygenerowano punkty leżące na bokach kwadratu o wierzchołkach $(-10, 10)$, $(-10, -10)$, $(10, -10)$, $(10, 10)$. Każdy bok prostokąta zawiera proporcjonalnie rozłożone punkty, a wizualizacja tego zbioru jest przedstawiona na wykresie powyżej.



(d) Punkty w tej sekcji rozmieszczono wokół kwadratu o wierzchołkach $(0, 0)$, $(10, 0)$, $(10, 10)$, $(0, 10)$. Na osiach OX i OY umieszczono po 25 punktów, natomiast na przekątnych $y = x$ oraz $y = 10 - x$ po 20 punktów. Wizualizacja punktów na bokach i przekątnych znajduje się na wykresie powyżej.

4 Przebieg obliczeń

Dla każdego zbioru danych wyznaczono otoczkę wypukłą za pomocą obu opisanych algorytmów: Grahama i Jarvisa. W obliczeniach zastosowano precyzję typu `float64` raz parametr $\epsilon = 10^{-20}$, dobrany eksperymentalnie w celu uniknięcia błędów zaokrągleń.

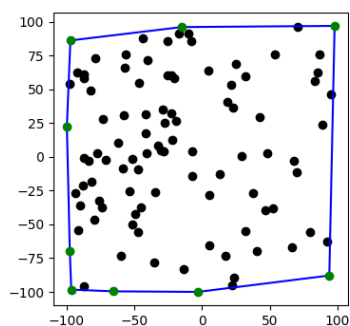
5 Prezentacja wyników

W tabelach zestawiono czasy wykonania poszczególnych algorytmów dla różnych zbiorów danych. Wyniki przedstawiono na wykresach, gdzie oznaczenia są następujące:

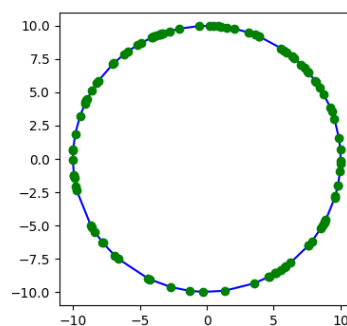
- Punkty należące do otoczki są zaznaczone na zielono.
- Punkty wewnętrzne, które nie należą do otoczki, są zaznaczone na czarno.
- Krawędzie otoczki wypukłej są zaznaczone na niebiesko.

5.1 Obliczenia dla przykładowych zbiorów

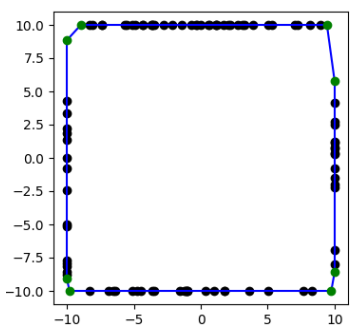
Dla zilustrowania działania algorytmów i ich dokładności, przeprowadzono obliczenia na kilku przykładowych zbiorach punktów. Przedstawiono zarówno czas wykonania obliczeń, jak i uzyskaną otoczkę wypukłą dla każdego przypadku, aby umożliwić porównanie efektywności obu metod.



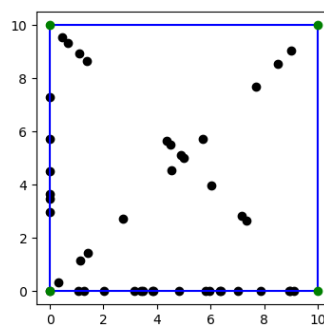
(a) Wizualizacja otoczki wypukłej dla zbioru A



(b) Wizualizacja otoczki wypukłej dla zbioru B



(c) Wizualizacja otoczki wypukłej dla zbioru C



(d) Wizualizacja otoczki wypukłej dla zbioru D

Tabela poniżej przedstawia porównanie czasów wykonania dla algorytmu Grahama i algorytmu Jarvisa w zaokrągleniu do 4 miejsc po przecinku dla milisekund.

Tabela 1: Tabela różnica czasów obliczeń dla podstawowych zbiorów A, B, C i D

| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa | Punkty na otoczce wypukłej |
|----------------|------------------|------------------|----------------------------|
| 100 | 0,0000 μs | 999,4507 μs | 9 |
| 100 | 595,0928 μs | 6,6411 ms | 100 |
| 100 | 1,0006 ms | 0,0000 μs | 8 |
| 45 | 0,0000 μs | 0,0000 μs | 4 |

6 Zbiory zmodyfikowane

6.1 Modyfikacja zbiorów testowych

W celu przeprowadzenia testów wydajnościowych algorytmów zmodyfikowano poprzednie zbiory:

- **Zbiór A:** Zawiera losowo wygenerowane punkty o współrzędnych z przedziału $[-1000, 1000]$.
- **Zbiór B:** Zawiera losowo wygenerowane punkty leżące na okręgu o środku $(100, 100)$ i promieniu $R = 3000$.
- **Zbiór C:** Zawiera losowo wygenerowane punkty leżące na bokach prostokąta o wierzchołkach $(-900, 1000)$, $(900, 1000)$, $(900, -1000)$, $(-900, -1000)$.
- **Zbiór D:** Zawiera wierzchołki kwadratu $(0, 0)$, $(1000, 0)$, $(1000, 1000)$, $(0, 1000)$ oraz punkty wygenerowane losowo na przekątnych kwadratu o liczności n i dwóch bokach kwadratu znajdujących się na osiach o liczności m . Przyjąłem, że $\frac{n}{m} = \frac{4}{5}$

Zbiory te wygenerowano dla zadanych o licznosciach:

- $n \in \{10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000, 50000, 55000, 60000\}$ dla zbiorów A i C,
- $n \in \{100, 200, 300, 500, 1000, 1500, 2000, 2500, 3000, 3500, 6000, 10000\}$ dla zbioru B,
- $m + n \in \{22500, 33750, 45000, 56250, 67500, 78750, 90000, 101250, 112500, 123750, 135000\}$ dla zbioru D.

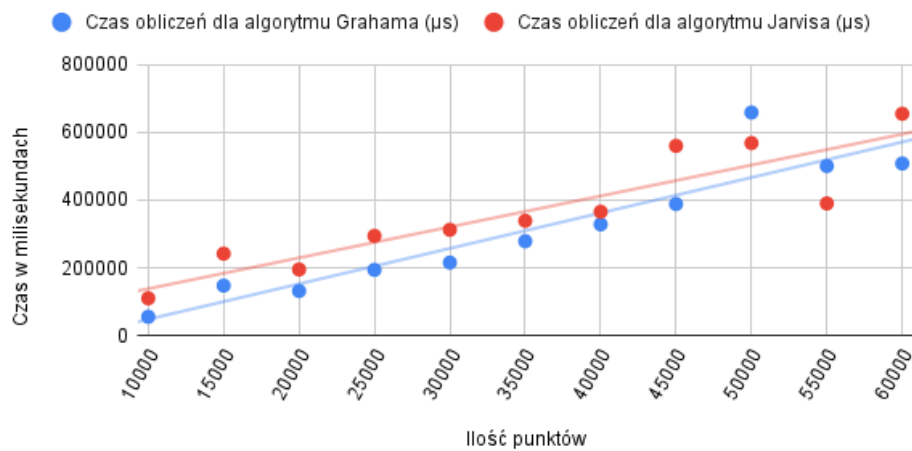
Podczas testów weryfikowano zgodność liczby punktów otoczki wynikowych zwracanych przez oba algorytmy, w celu weryfikacji implementacji, zwłaszcza dla przypadku zbioru B.

6.2 Zbiór A

Tabela 2: Tabela różnic czasów obliczeń dla algorytmu Grahama i Jarvisa dla zmodyfikowanego zbioru A

| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa | Punkty na otoczce wypukłej |
|----------------|------------------|------------------|----------------------------|
| 10000 | 55,5928 ms | 110,0578 ms | 24 |
| 15000 | 147,5341 ms | 241,8611 ms | 27 |
| 20000 | 131,6946 ms | 195,2589 ms | 25 |
| 25000 | 194,0067 ms | 294,481 ms | 28 |
| 30000 | 215,5235 ms | 312,4542 ms | 26 |
| 35000 | 278,6458 ms | 338,9909 ms | 25 |
| 40000 | 328,4674 ms | 365,5345 ms | 22 |
| 45000 | 388,4759 ms | 560,2539 ms | 29 |
| 50000 | 658,7453 ms | 568,4695 ms | 29 |
| 55000 | 500,7644 ms | 390,1613 ms | 19 |
| 60000 | 508,148 ms | 654,9058 ms | 29 |

Wykres zależności czasu (μs) obliczeń dla algorytmu Grahama i algorytmu Jarvisa od ilości punktów w zbiorze A



Wykres 3: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru A

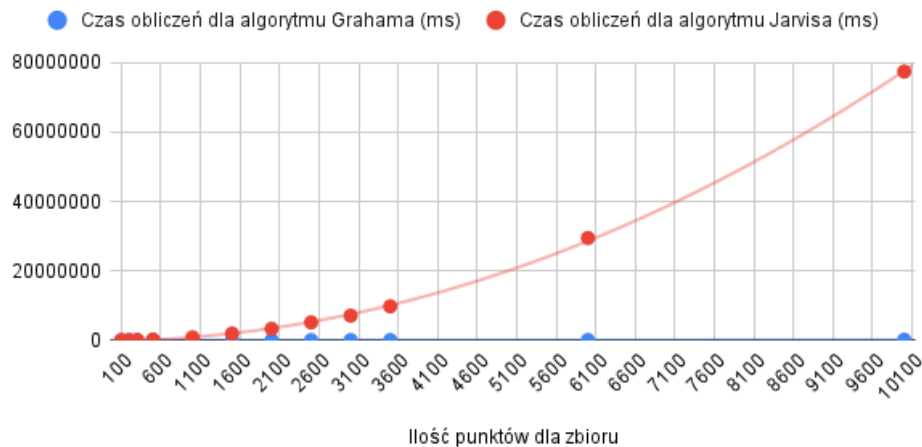
W Tabeli 2 i na Wykresie 3, algorytm Grahama wykazuje się lepszą efektywnością w stosunku do algorytmu Jarvisa dla zbioru losowych punktów należących do zadanego prostokątnego zbioru. Linia trendu dla algorytmu Grahama jest zbliżona do liniowej, ponieważ dla różnic liczności punktów rzędu $\Delta n = 10^4$, składnik $\log_2 n \approx 13$, w najgorszym przypadku $\log_2 60000 \approx 16$. Algorytm Jarvisa jest w tym przypadku wolniejszy, jednak różnica jest niewielka, ponieważ największa liczba punktów znajdujących się na otoczce to $k = 29$ i algorytm Grahama może mieć w mojej implementacji dużą stałą. Dla rozkładów powyżej wartość można zrobić oszacowanie $k \approx 1.3 \cdot \log_2 n$.

6.3 Zbiór B

Tabela 3: Tabela różnic czasów obliczeń dla algorytmu Grahama i Jarvisa dla zmodyfikowanego zbioru B

| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa | Punkty na otoczce wypukłej |
|----------------|------------------|------------------|----------------------------|
| 100 | 0,0000 μs | 9,0001 ms | 100 |
| 200 | 999,9275 μs | 32,0237 ms | 200 |
| 300 | 2,0235 ms | 69,0019 ms | 300 |
| 500 | 3,0246 ms | 198,199 ms | 500 |
| 1000 | 7,9994 ms | 820,411 ms | 1000 |
| 1500 | 12,9988 ms | 1,938 s | 1500 |
| 2000 | 16,9988 ms | 3,2977 s | 2000 |
| 2500 | 28,9989 ms | 5,1429 s | 2500 |
| 3000 | 30,9975 ms | 7,0856 s | 3000 |
| 3500 | 37,029 ms | 9,7628 s | 3500 |
| 6000 | 60,998 ms | 29,4608 s | 6000 |
| 10000 | 127,9974 ms | 77,4403 s | 10000 |

Wykres zależności czasu (μm) obliczeń dla algorytmu Grahama i algorytmu Jarvisa od ilości punktów w zbiorze B



Wykres 4: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru B

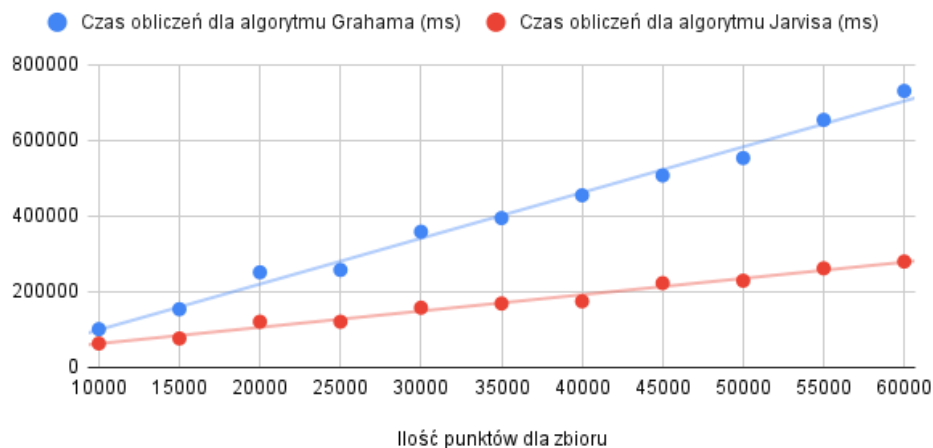
Dla każdej kolejnej liczebności zmodyfikowanego generatora zbiorów B, liczba punktów otoczki jest równa liczbie wszystkich punktów, co było oczekiwane i dowodzi poprawności implementacji obu algorytmów. Jak można zauważyć na **Wykresie 4**, krzywa trendu dla tego algorytmu ma postać kwadratową. Związane jest to z faktem, że liczba punktów na otoczce $k = n$, co skutkuje osiągnięciem przez ten algorytm złożoności $O(n^2)$. Jak widać w **Tabeli 3**, złożoność algorytmu Grahama sprawia, że jest on zdecydowanie lepszym wyborem dla tego zbioru. W najgorszym przypadku 10000 punktów w zbiorze stosunek czasów wykonania algorytmów to $\frac{t_J}{t_G} \approx 682$.

6.4 Zbiór C

Tabela 4: Tabela różnic czasów obliczeń dla algorytmu Grahama i Jarvisa dla zmodyfikowanego zbioru C

| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa | Punkty na otoczce wypukłej |
|----------------|------------------|------------------|----------------------------|
| 10000 | 101,1715 ms | 63,3497 ms | 8 |
| 15000 | 153,9557 ms | 76,6666 ms | 8 |
| 20000 | 251,3793 ms | 120,8305 ms | 8 |
| 25000 | 257,6144 ms | 120,8324 ms | 8 |
| 30000 | 358,6156 ms | 157,8729 ms | 8 |
| 35000 | 394,6457 ms | 168,9529 ms | 8 |
| 40000 | 454,8154 ms | 174,7561 ms | 8 |
| 45000 | 507,2525 ms | 222,8427 ms | 8 |
| 50000 | 553,3626 ms | 229,1861 ms | 8 |
| 55000 | 654,083 ms | 261,8713 ms | 8 |
| 60000 | 730,5443 ms | 279,7983 ms | 8 |

Wykres zależności czasu (μ m) obliczeń dla algorytmu Grahama i algorytmu Jarvisa od ilości punktów w zbiorze C



Wykres 5: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru C

W przypadku zmodyfikowanym C, jak widać na **Tabeli 4** oraz na **Wykresie 5**, występuje inna różnica czasów obliczeń. W tym przypadku liczba punktów na otoczce $k = 8$, przez co złożoność algorytmu Jarvisa jest rzędu $O(n)$, a dla algorytmu Grahama złożoność to dalej $O(n \log_2 n)$. Wynika z tego, że dla zbiorów o m.in. dużej zawartości punktów współliniowych (a w efekcie ograniczonym k), algorytm Jarvisa jest ma krótszy czas obliczeń.

6.5 Zbiór D

Tabela 5: Tabela różnic czasów obliczeń dla algorytmu Grahama i Jarvisa dla zmodyfikowanego zbioru D

| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa | Punkty na otoczce wypukłej |
|----------------|------------------|------------------|----------------------------|
| 22500 | 356,2996 ms | 51,9669 ms | 4 |
| 33750 | 550,7028 ms | 82,9678 ms | 4 |
| 45000 | 771,1387 ms | 100,9698 ms | 4 |
| 56250 | 1,0179 s | 141,001 ms | 4 |
| 67500 | 1,1629 s | 149,0333 ms | 4 |
| 78750 | 1,3947 s | 172,9193 ms | 4 |
| 90000 | 1,5884 s | 200,0017 ms | 4 |
| 101250 | 1,9623 s | 240,0343 ms | 4 |
| 112500 | 2,2371 s | 260,53 ms | 4 |
| 123750 | 2,4383 s | 296,999 ms | 4 |
| 135000 | 2,6013 s | 299,9678 ms | 4 |

Wykres zależności czasu (μ m) obliczeń dla algorytmu Grahama i algorytmu Jarvisa od ilości punktów w zbiorze D



Wykres 6: Porównanie graficzne czasu wyznaczania otoczki wypukłej przez algorytmy dla zmodyfikowanego zbioru D

W tym przypadku również algorytm Jarvisa dużo szybciej oblicza otoczę wypukłą, jak widać na **Tabeli 5** oraz na **Wykresie 6**. Otoczka wypukła zbioru D zawiera wyłącznie wierzchołki kwadratu i liczba punktów na otoczce to oczywiście $k = 4$. Liczba ta jest dwukrotnie mniejsza niż w ostatnim przypadku, dlatego też algorytm Jarvisa znacznie lepiej sobie radzi (obliczenia są dwukrotnie szybsze niż w C), niż algorytm Grahama.

7 Wnioski

Na podstawie przeprowadzonego ćwiczenia można wyciągnąć następujące wnioski:

- Po wykonaniu ćwiczenia można zauważyć, dlaczego zostały zaproponowane zbiory A, B, C i D:
 - Zbiór A - miał za zadanie uwzględnić przypadek jednorodnego losowego rozkładu punktów, w zmodyfikowanej formie była to dobry zbiór dla wizualizacji algorytmu Grahama.
 - Zbiór B - weryfikował, czy cały okrąg zostanie oznaczony jako otoczka wypukła. Był to dobry test dla tolerancji dla zera ϵ , dzięki któremu dla większych danych założyłem ostatecznie $\epsilon = 10^{-18}$. Był to też test pokazujący złożoność $O(n^2)$ dla algorytmu Jarvisa w najgorszym przypadku ($n = k$).
 - Zbiór C - sprawdzał działanie wykluczania współliniowości punktów, otoczka ma co najmniej 4 punkty i co najwyżej 8 punktów.
 - Zbiór D - miał tylko jedną poprawną odpowiedź, były to 4 punkty na krawędziach kwadratu. Sprawdzał poprawność i wykluczanie punktów współliniowych z otoczki.
- Algorytm Grahama wykazał przewagę czasową nad algorytmem Jarvisa dla dużych zbiorów punktów o losowym rozkładzie (np. zbiór A, C), co wynika z jego złożoności $O(n \log_2 n)$ w porównaniu do $O(nk)$ Jarvisa, dla których $\log_2 n < k$. Dzięki temu jest bardziej wydajny w przypadku zbiorów o dużej liczbie punktów należących do otoczki wypukłej.
- Algorytm Jarvisa okazał się bardziej efektywny w przypadku zbiorów, gdzie liczba punktów tworzących otoczkę wypukłą, k , jest stosunkowo mała (np. zbiory z punktami ułożonymi w siatkę, jak w zbiorze C i D). W takich sytuacjach algorytm osiąga złożoność bliską liniowej względem liczby punktów w otoczce, co przekłada się na krótszy czas wykonania w porównaniu do algorytmu Grahama.
- Testy wskazują, że liczba punktów tworzących otoczkę wypukłą (k) znacząco wpływa na czas działania algorytmu Jarvisa. W przypadku zbioru, w którym wszystkie punkty leżą na otoczce (np. zbiór B), złożoność algorytmu Jarvisa rośnie do $O(n^2)$, co sprawia, że jego czas wykonania jest znacznie dłuższy niż czas działania algorytmu Grahama.
- Wybór odpowiedniego algorytmu zależy od charakterystyki zbioru punktów:
 - Algorytm Grahama jest bardziej uniwersalny i lepiej sprawdza się w przypadku zbiorów o rozkładzie losowym oraz dla dużych zbiorów punktów.
 - Algorytm Jarvisa wykazuje przewagę przy zbiorach, gdzie większość punktów znajduje się wewnątrz otoczki lub rozkład punktów jest taki, że tylko nieliczne tworzą jej wierzchołek.

Podsumowując, oba algorytmy efektywnie rozwiązują problem wyznaczania otoczki wypukłej, jednak wybór algorytmu powinien opierać się na porównaniu między k i $\log_2 n$, jeżeli jesteśmy w stanie oszacować te wartości. W praktyce może to oznaczać że dla łatwego zaznaczania punktów na mapie dla danego obszaru ograniczonego lepszy będzie algorytm Grahama, ze względu na losowe rozmieszczenie punktów na obszarze. Zaś dla danych o określonej strukturze (wykluczając krzywe i okręgi) lepszy byłby algorytm Jarvisa.