Akademia Górniczo-Hutnicza
im. Stanisława Staszica
w Krakowie

Algorytmy Geometryczne

Laboratorium nr 2

Tomasz Smyda

10 listopada 2023

Spis treści

1 Wstęp

1.1 Opis ćwiczenia

Ćwiczenie polegało na wyznaczeniu otoczki wypukłej dla danego zbioru punktów. Otoczką wypukłą nazywamy najmniejszy zbiór wypukły zawierający podzbiór wszystkich punktów na płaszczyźnie. Do wykonania ćwiczenia zostały wykorzystane dwa algorytmy: algorytm Grahama oraz algorytm Jarvisa.

1.2 Algorytm Grahama

1. Niech p_0 będzie punktem w zbiorze Q z najmniejszą współrzędną y, oraz najmniejszą współrzędną x w przypadku, gdy wiele punktów ma tą samą współrzędną x
2. niech $\langle \mathtt{p_1}, \mathtt{p_2}, \ldots, \mathtt{p_m} \rangle$ będzie pozostałym zbiorem punktów w Q posortowanym zgodnie z przeciwnym ruchem wskazówek zegara wokół punktu p_0 (jeżeli więcej niż jeden punkt ma ten sam kąt to usuwamy wszystkie punkty z wyjątkiem tego najbardziej oddalonego od p_0)
3. stwórz pusty stos S
4. $\mathtt{PUSH(p_0, S)}$
5. $\mathtt{PUSH(p_1, S)}$
6. $\mathtt{PUSH(p_2, S)}$
7. **for** $i = 3$ **to** m
   - 7.1 **while** kąt utworzony przez $\mathtt{NEXT - TO - TOP(S)}$, $\mathtt{TOP(S)}$ oraz p_i tworzy lewostronny skręt
     - 7.1.1 $\mathtt{POP(S)}$
   - 7.2 $\mathtt{PUSH(p_i, S)}$
8. **return** S

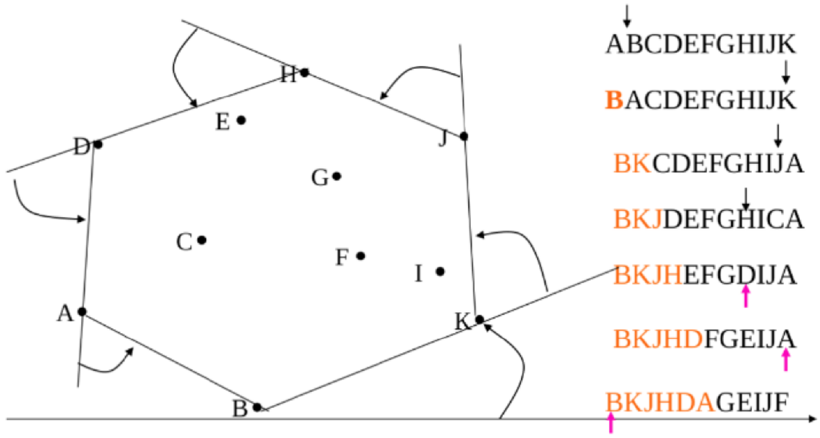
Koszt algorytmu:

- Szukanie minimum - $O(n)$
- Sortowanie - $O(n \log n)$
- Inicjalizacja stosu - $O(1)$
- Pętla w punkcie nr 7 - $O(n-3)$

$$O(n) + O(n \log n) + O(1) + O(n-3) = O(n \log n)$$

1.3 Algorytm Jarvisa

1. Niech p_0 będzie punktem w zbiorze Q z najmniejszą współrzędną y, oraz najmniejszą współrzędną x w przypadku, gdy wiele punktów ma tą samą współrzędną x
2. **do**
   - 2.1 **for** $j \neq i$ **do**
     - 2.1.1 znajdź punkt, dla którego kąt liczony przeciwnie do wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy
   - 2.2 niech k będzie indeksem punktu z najmniejszym kątem zwróć (p_i, p_k) jako krawędź otoczki
   - 2.3 $i \leftarrow k$
3. **while** $i \neq i_0$



Rysunek 1: Wizualizacja działania algorytmu Jarvisa

Koszt algorytmu:

Złożoność rzędu $O(n^2)$ Gdy liczba wierzchołków otoczki jest ograniczona przez stałą k, jego złożoność jest rzędu $O(k \cdot n)$.

1.4 Biblioteki

Ćwiczenie zostało wykonane przy użyciu narzędzia Jupyter Notebook z wykorzystaniem języka Python w wersji 3.9.18. Do obliczeń, generowania liczb pseudolosowych, rysowania wykresów i generowania tabelek zostały wykorzystane biblioteki: Numpy, Random, Pandas, Matplotlib oraz Seaborn.

1.5 Specyfikacja

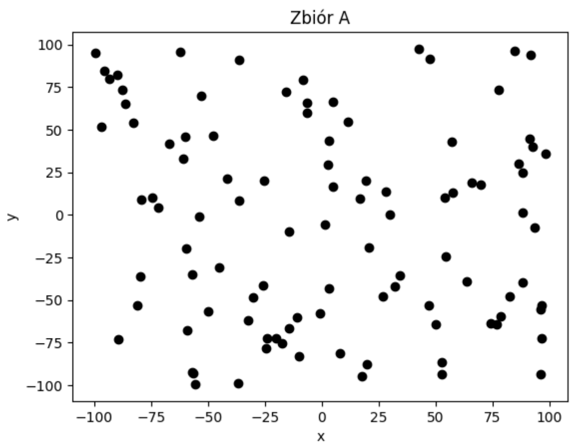
Wszystkie obliczenia były prowadzone na komputerze Lenovo Legion 5 Pro z systemem operacyjnym Windows 11 Home w wersji 22H2, procesorem Intel Core i7-11800H.

2 Realizacja ćwiczenia

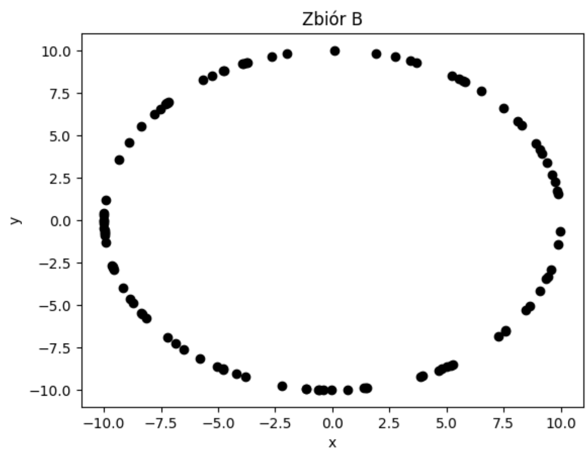
2.1 Wygenerowanie zbiorów punktów

Do wylosowania punktów wykorzystano funkcję random. Wygenerowane zbiory:

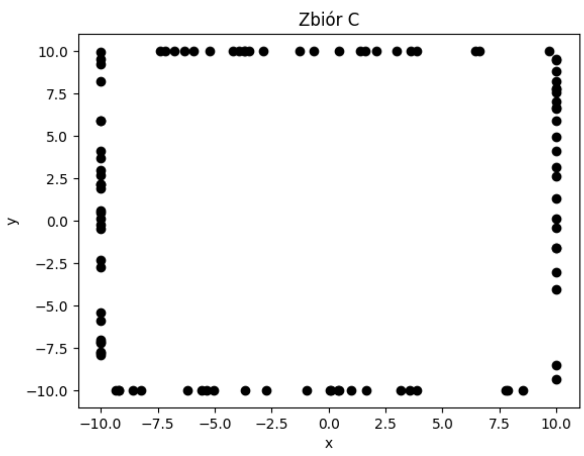
- **Zbiór A**: 100 losowych punktów (x, y), takich że $x, y \in [-100; 100]^2$
- **Zbiór B**: 100 losowych punktów (x, y) leżących na okręgu o środku w punkcie $O = (0, 0)$ i promieniu równym $R = 10$
- **Zbiór C**: 100 losowych punktów (x, y) leżących na bokach prostokąta o wierzchołkach $(-10, -10)$, $(10, -10)$, $(10, 10)$, $(-10, 10)$
- **Zbiór D**: Zawierający wierzchołki kwadratu $(0, 0)$, $(10, 0)$, $(10, 10)$, $(0, 10)$ oraz po 25 punktów wygenerowanych losowo na dwóch bokach kwadratu leżących na osiach i po 20 punktów na każdej z przekątnych



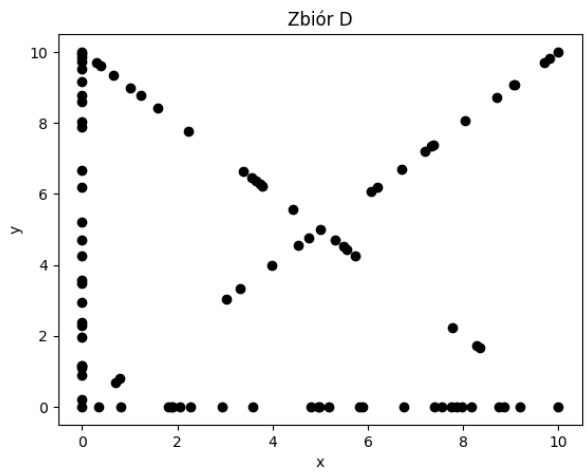
Rysunek 2: Zbiór A



Rysunek 3: Zbiór B



Rysunek 4: Zbiór C



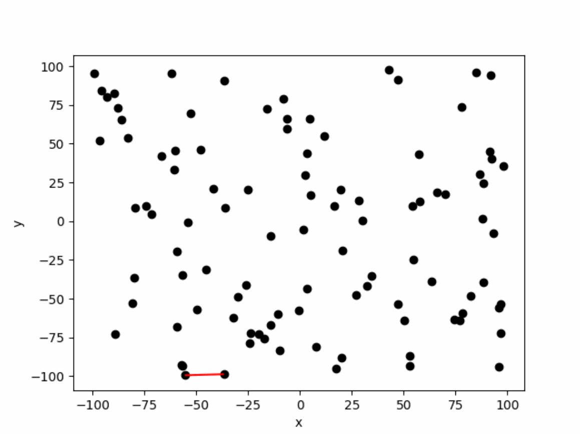
Rysunek 5: Zbiór D

2.2 Wyznaczenie otoczek dla wygenerowanych zbiorów

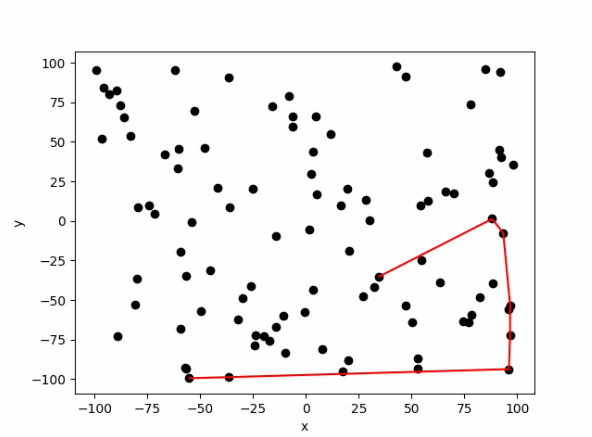
Do implementacji obu algorytmów użyłem pomocniczych funkcji m.in. do wyznaczenia położenia punktu względem prostej - wykorzystałem własną funkcję obliczającą wyznacznik 2x2. Do sortowania punktów użyłem własnoręcznie zaimplemenotwanego algorytmu QuickSort. Poniżej zaprezentowane są po 4 klatki z wizualizacji dla każdego zbioru i danego algorytmu.

2.2.1 Zbiór A

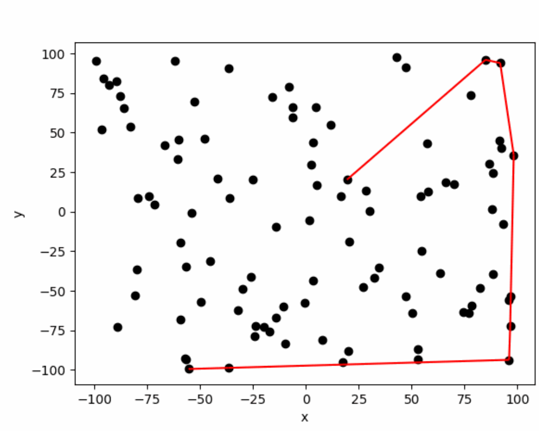
Algorytm Grahama



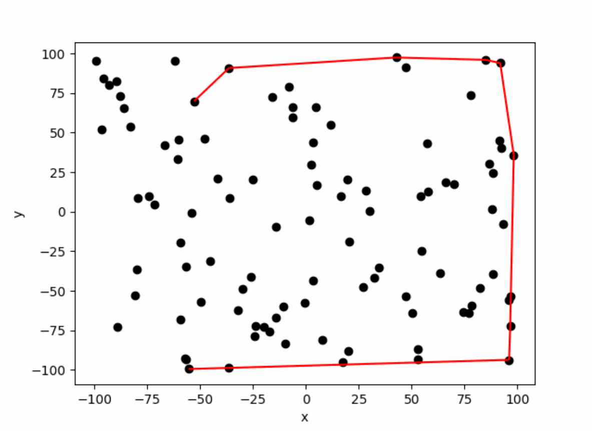
Rysunek 6



Rysunek 7

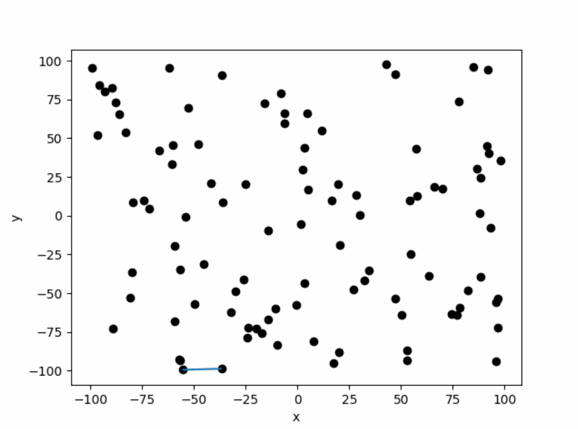


Rysunek 8

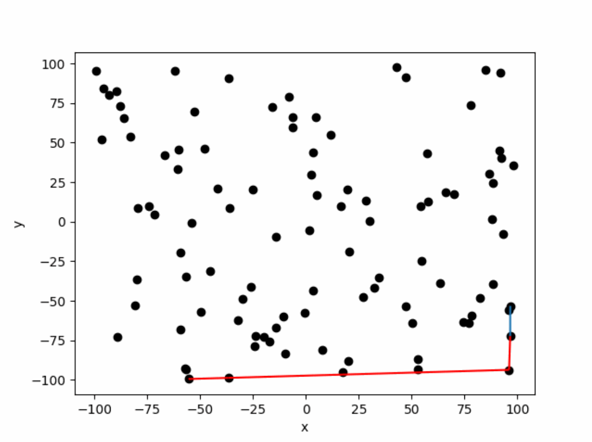


Rysunek 9

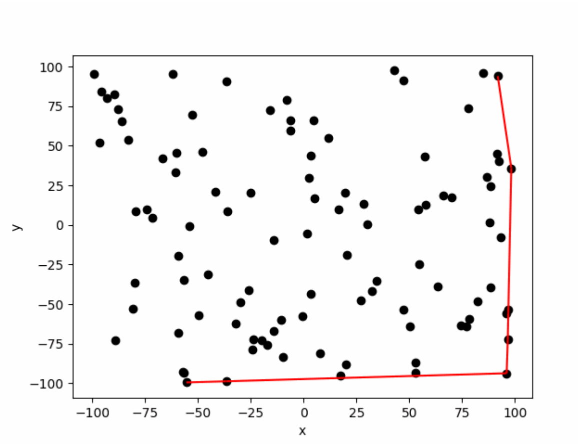
Algorytm Jarvisa



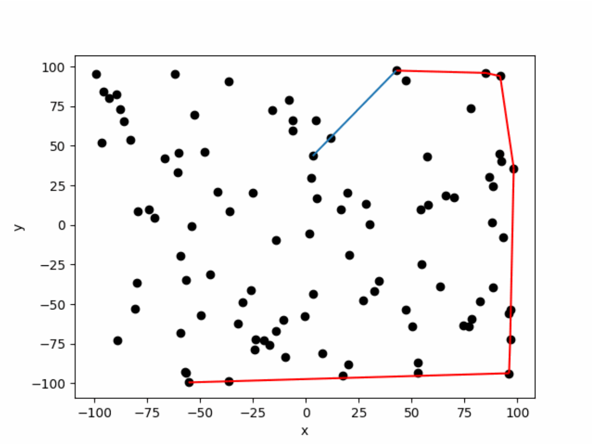
Rysunek 10



Rysunek 11



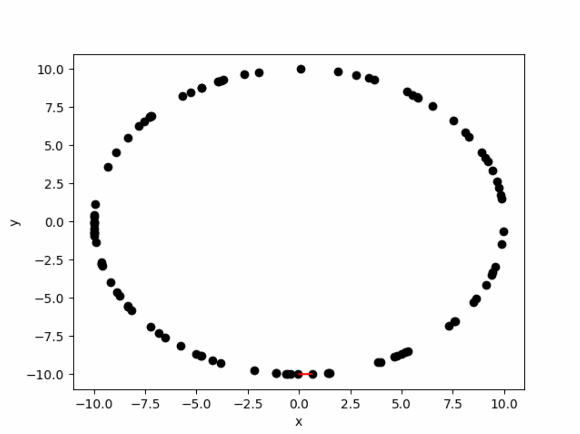
Rysunek 12



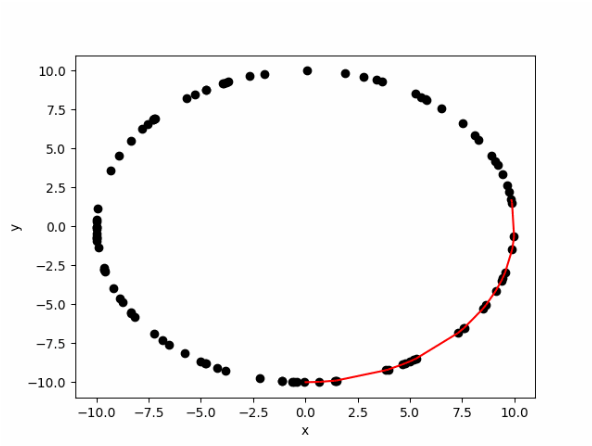
Rysunek 13

2.2.2 Zbiór B

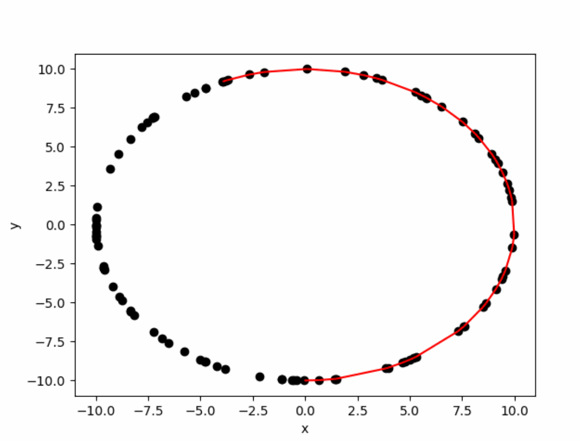
Algorytm Grahama



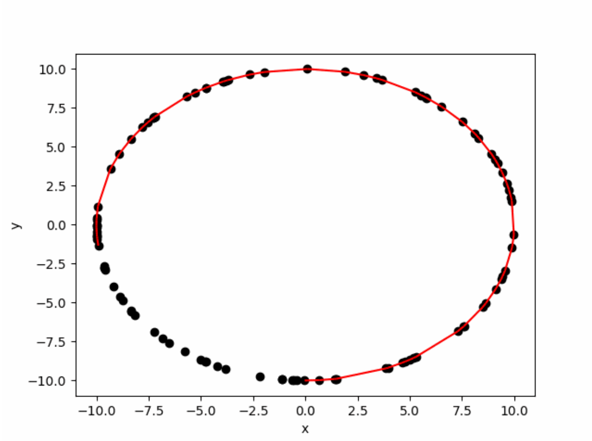
Rysunek 14



Rysunek 15

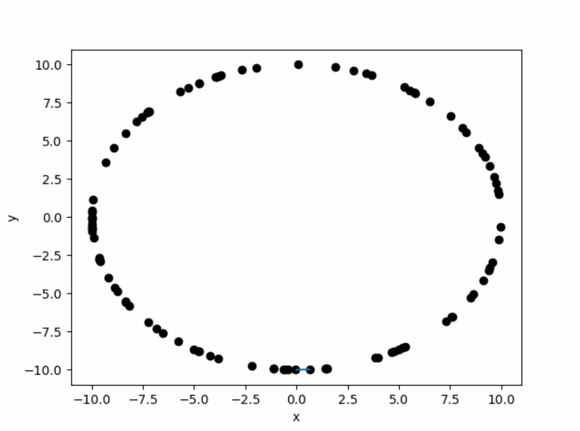


Rysunek 16

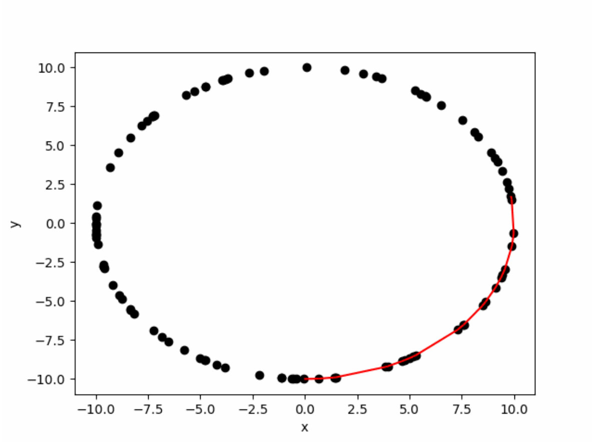


Rysunek 17

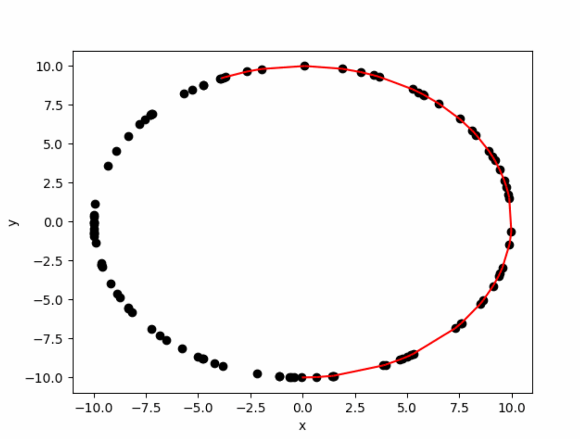
Algorytm Jarvisa



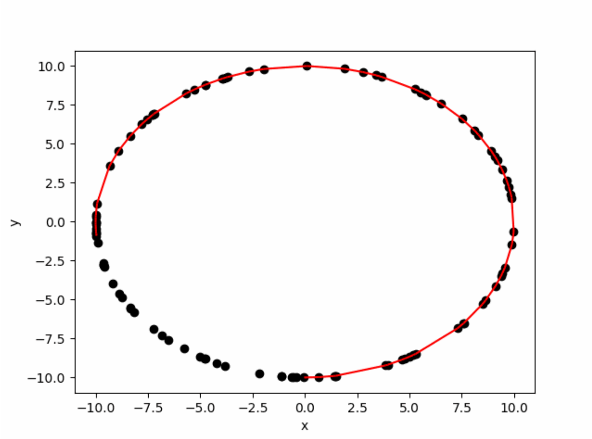
Rysunek 18



Rysunek 19



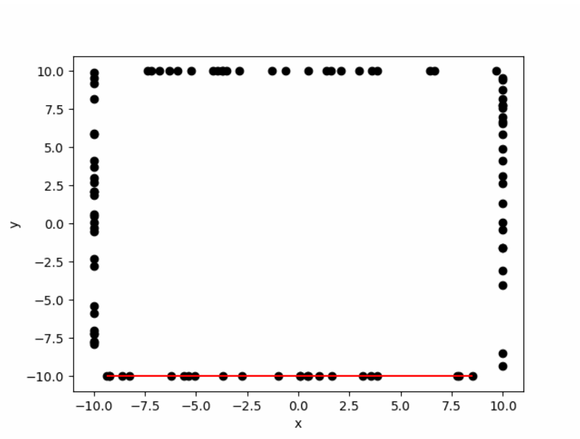
Rysunek 20



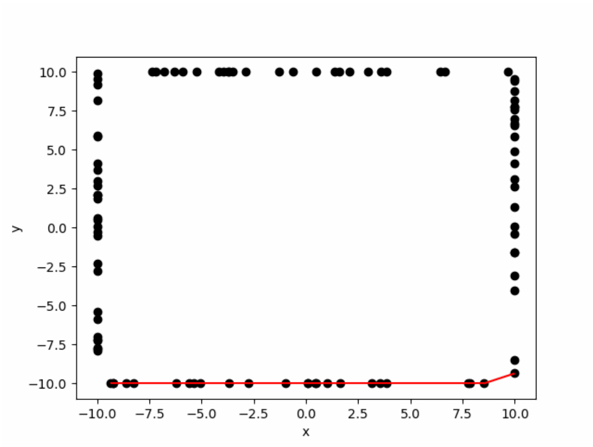
Rysunek 21

2.2.3 Zbiór C

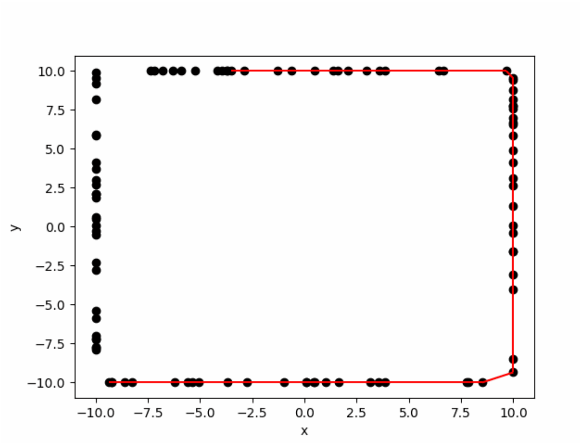
Algorytm Grahama



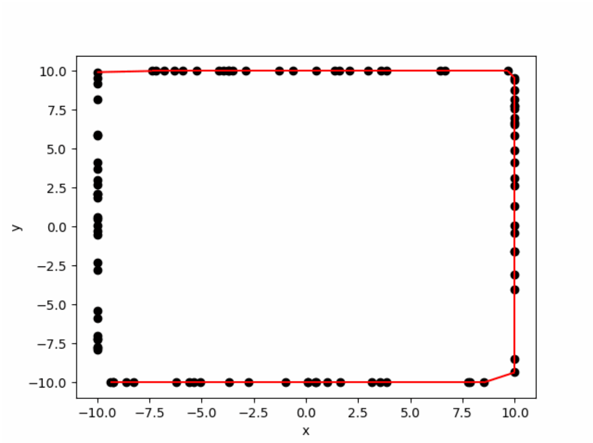
Rysunek 22



Rysunek 23

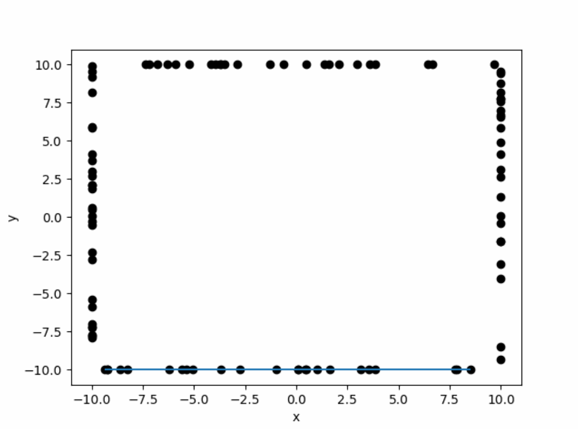


Rysunek 24

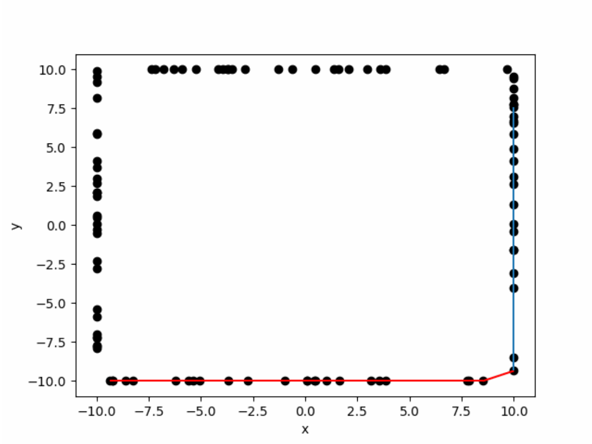


Rysunek 25

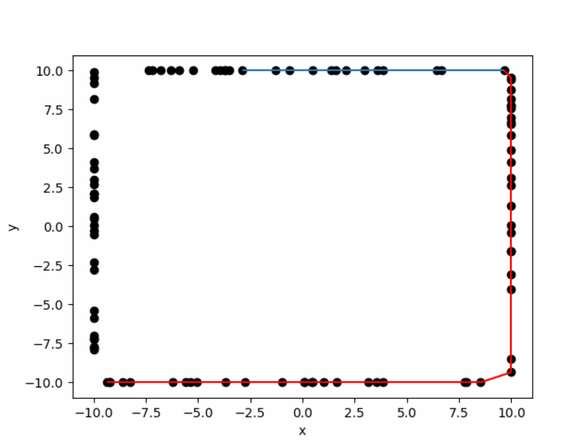
Algorytm Jarvisa



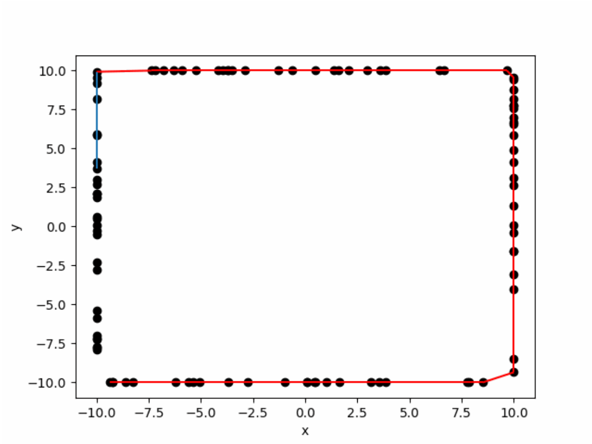
Rysunek 26



Rysunek 27



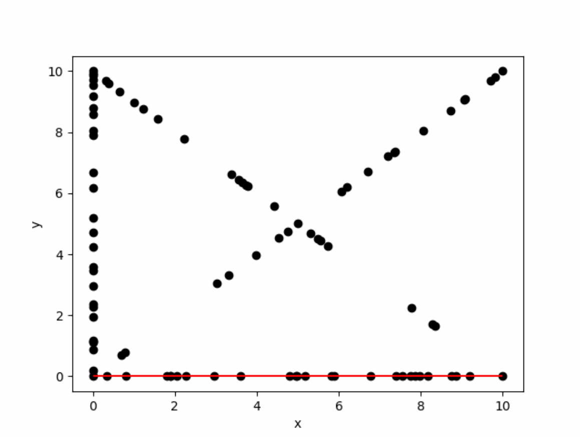
Rysunek 28



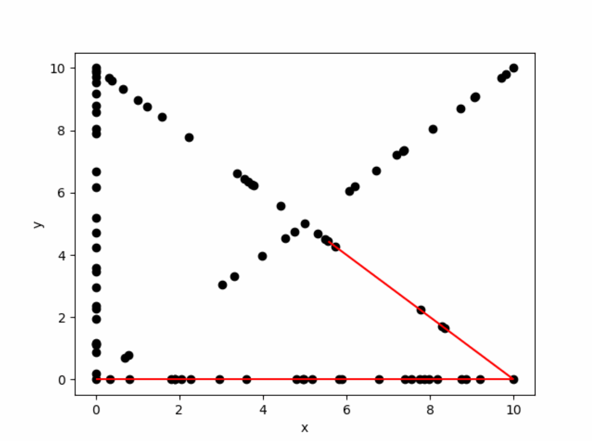
Rysunek 29

2.2.4 Zbiór D

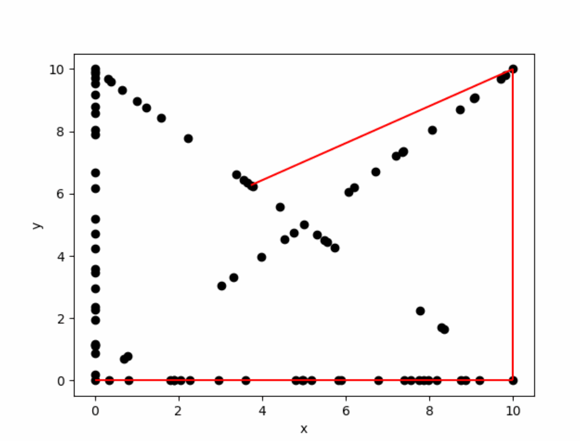
Algorytm Grahama



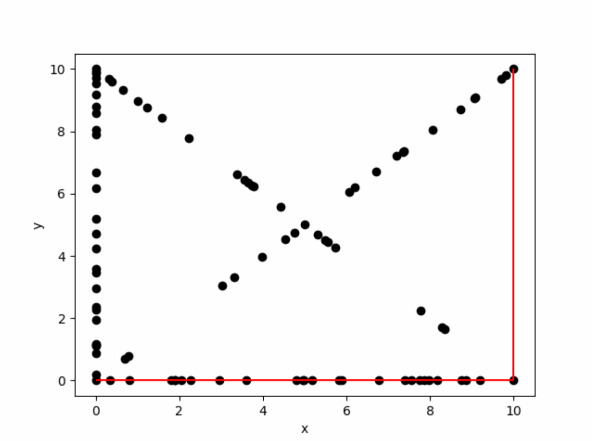
Rysunek 30



Rysunek 31

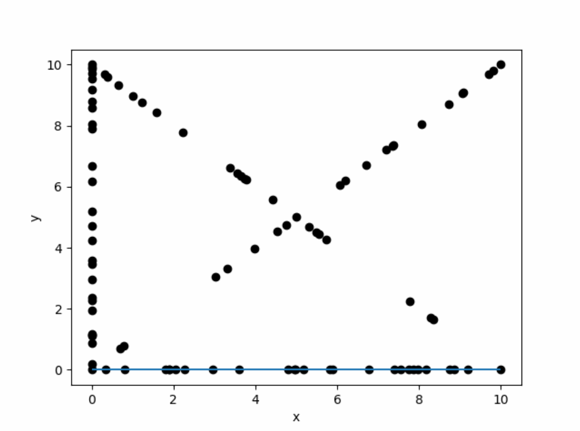


Rysunek 32

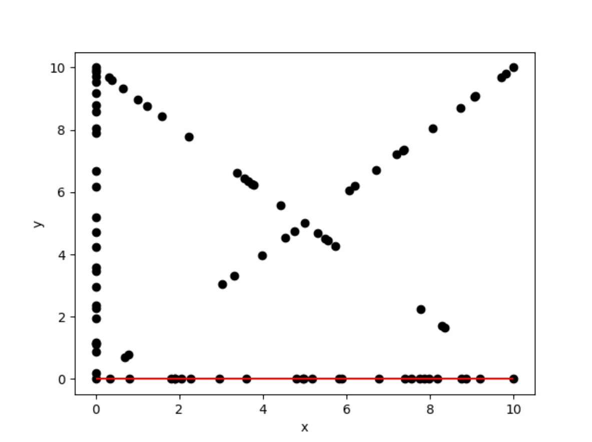


Rysunek 33

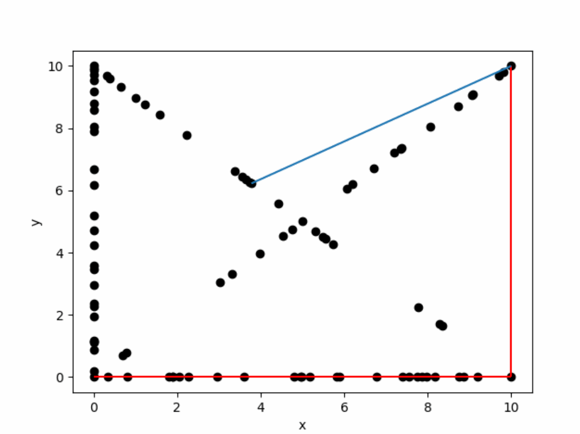
Algorytm Jarvisa



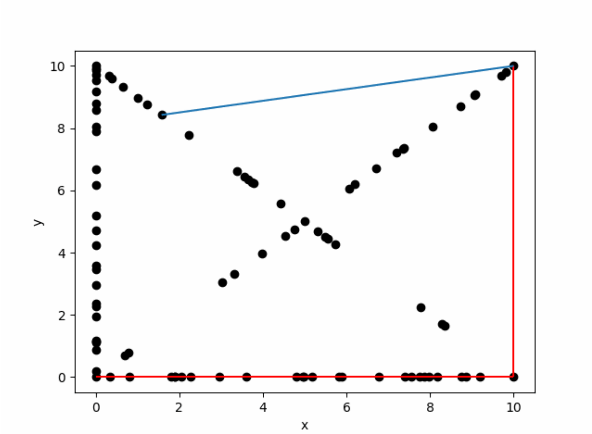
Rysunek 34



Rysunek 35



Rysunek 36



Rysunek 37

2.3 Porównanie czasu działania funkcji

Wykorzystano funkcję, która dla danego zestawu danych mierzy czas, w którym dana funckja wyznaczyła otoczkę. Na rysunku poniżej możemy porównać wyniki czasowe dla wyznaczonych zbiorów.

```
Czas działania funkcji graham_algorithm dla zbioru points_a: 0.002804279327392578 s
Czas działania funkcji jarvis_algorithm dla zbioru points_a: 0.0 s
Czas działania funkcji graham_algorithm dla zbioru points_b: 0.004306316375732422 s
Czas działania funkcji jarvis_algorithm dla zbioru points_b: 0.01215982437133789 s
Czas działania funkcji graham_algorithm dla zbioru points_c: 0.0019998550415039062 s
Czas działania funkcji jarvis_algorithm dla zbioru points_c: 0.0010013580322265625 s
Czas działania funkcji graham_algorithm dla zbioru points_d: 0.0036287307739257812 s
Czas działania funkcji jarvis_algorithm dla zbioru points_d: 0.0009963512420654297 s
```

Rysunek 38: Czasy wyznaczenia otoczek dla podanych zbiorów oraz algorytmów

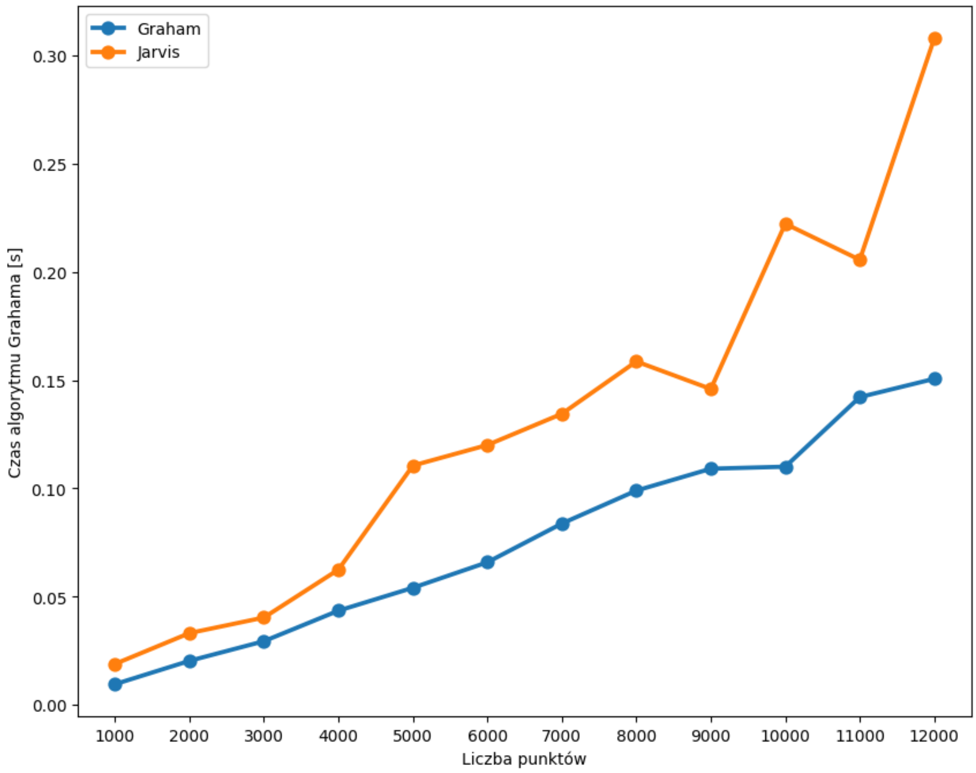
Widzimy, że czasy działania algorytmów dla tak wyznaczonch zbiorów są bardzo małe (szczególnie algorytm Jarvisa dla zbiorów C oraz D). Zatem dla lepszego zobrazowania różnic czasowych posłużymy się dodatkowo wygenerowanymi zbiorami.

2.4 Wyniki czasowe dla bardziej miarodajnych zbiorów

2.4.1 Zbiory typu A

	Liczba punktów	Początek zakresu	Koniec zakresu	Czas algorytmu Grahama [s]	Czas algorytmu Jarvisa [s]	Szybszy algorytm	Róznica czasu [s]
0	1000	-250	250	0.009562	0.018916	Algorytm Grahama	0.009355
1	2000	-250	250	0.020348	0.033171	Algorytm Grahama	0.012823
2	3000	-250	250	0.029431	0.040361	Algorytm Grahama	0.010930
3	4000	-250	250	0.043528	0.062497	Algorytm Grahama	0.018970
4	5000	-250	250	0.054104	0.110569	Algorytm Grahama	0.056465
5	6000	-250	250	0.065891	0.120061	Algorytm Grahama	0.054169
6	7000	-250	250	0.083786	0.134477	Algorytm Grahama	0.050691
7	8000	-250	250	0.099006	0.158655	Algorytm Grahama	0.059649
8	9000	-250	250	0.109109	0.145952	Algorytm Grahama	0.036843
9	10000	-250	250	0.110049	0.222275	Algorytm Grahama	0.112225
10	11000	-250	250	0.142200	0.205637	Algorytm Grahama	0.063436
11	12000	-250	250	0.150641	0.307900	Algorytm Grahama	0.157259

Tabela 1: Wyniki czasowe dla zbiorów typu A

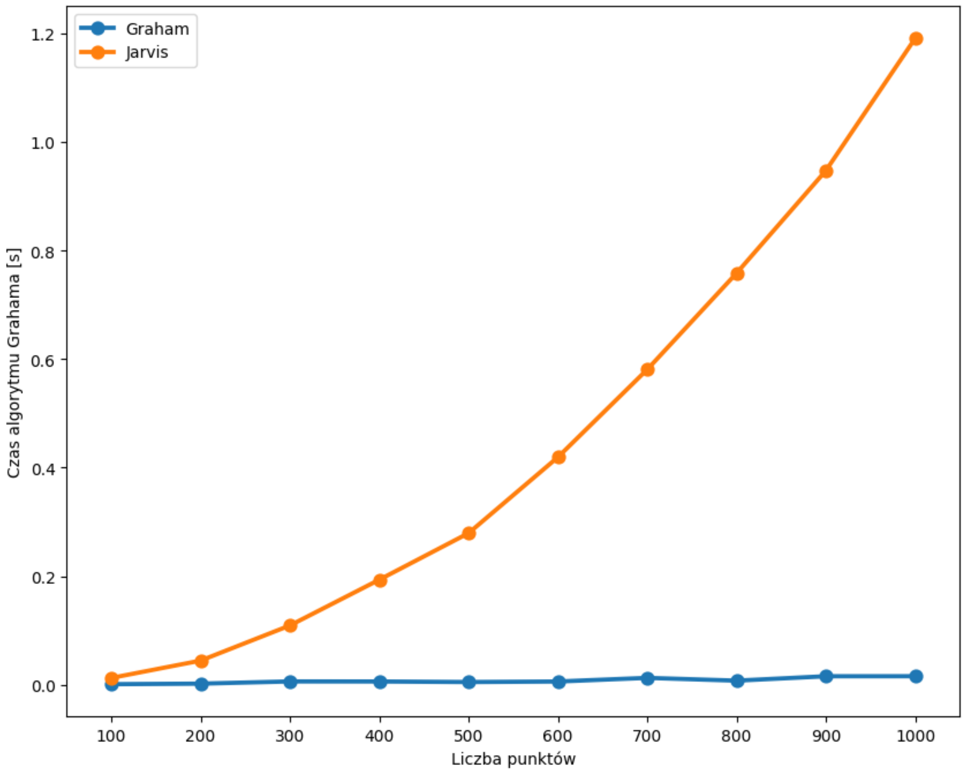


Rysunek 39: Wykres przedstawiający porównanie czasu działania obu algorytmów dla zbiorów typu A

2.4.2 Zbiory typu B

	Liczba punktów	**Środek okręgu**	**Promień okręgu**	**Czas algorytmu Grahama [s]**	**Czas algorytmu Jarvisa [s]**	**Szybszy algorytm**	**Różnica czasu [s]**
0	100	(9, -9)	12	0.000995	0.012507	Algorytm Grahama	0.011513
1	200	(-5, -19)	19	0.002000	0.044467	Algorytm Grahama	0.042467
2	300	(22, -1)	16	0.006088	0.109219	Algorytm Grahama	0.103132
3	400	(-26, -29)	31	0.005984	0.193497	Algorytm Grahama	0.187513
4	500	(28, 5)	13	0.004981	0.279367	Algorytm Grahama	0.274386
5	600	(54, -15)	56	0.005943	0.419329	Algorytm Grahama	0.413386
6	700	(70, -43)	15	0.012641	0.580909	Algorytm Grahama	0.568268
7	800	(-35, -12)	155	0.007512	0.758419	Algorytm Grahama	0.750907
8	900	(-73, 17)	105	0.015625	0.947443	Algorytm Grahama	0.931818
9	1000	(-65, -38)	104	0.015690	1.191180	Algorytm Grahama	1.175490

Tabela 2: Wyniki czasowe dla zbiorów typu B

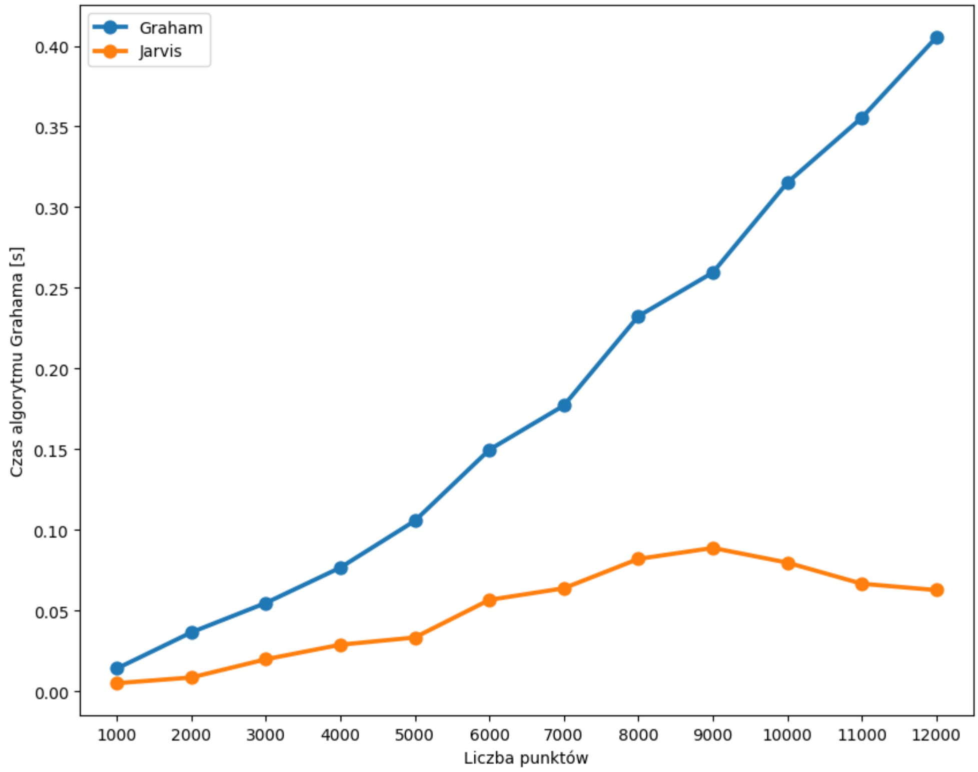


Rysunek 40: Wykres przedstawiający porównanie czasu działania obu algorytmów dla zbiorów typu B

2.4.3 Zbiory typu C

	Liczba punktów	Dwa przeciwległe wierzchołki prostokąta	Czas algorytmu Grahama [s]	Czas algorytmu Jarvisa [s]	Szybszy algorytm	Róznica czasu [s]
0	1000	(-56, 2) i (-48, 53)	0.014210	0.005073	Algorytm Jarvisa	0.009137
1	2000	(-13, 2) i (-11, 29)	0.036611	0.008627	Algorytm Jarvisa	0.027985
2	3000	(-42, 29) i (55, 47)	0.054829	0.019918	Algorytm Jarvisa	0.034911
3	4000	(-40, 22) i (67, 58)	0.076729	0.028915	Algorytm Jarvisa	0.047814
4	5000	(-58, -51) i (60, -41)	0.105712	0.033412	Algorytm Jarvisa	0.072300
5	6000	(23, -59) i (30, -45)	0.149750	0.056716	Algorytm Jarvisa	0.093035
6	7000	(11, -42) i (37, 13)	0.177126	0.063896	Algorytm Jarvisa	0.113230
7	8000	(-19, 30) i (17, 59)	0.232481	0.082082	Algorytm Jarvisa	0.150399
8	9000	(17, -34) i (31, -17)	0.259520	0.088874	Algorytm Jarvisa	0.170646
9	10000	(5, -58) i (10, 14)	0.315222	0.079784	Algorytm Jarvisa	0.235437
10	11000	(0, 44) i (36, 62)	0.355481	0.066711	Algorytm Jarvisa	0.288770
11	12000	(-44, -35) i (65, 16)	0.405203	0.062760	Algorytm Jarvisa	0.342443

Tabela 3: Wyniki czasowe dla zbiorów typu C

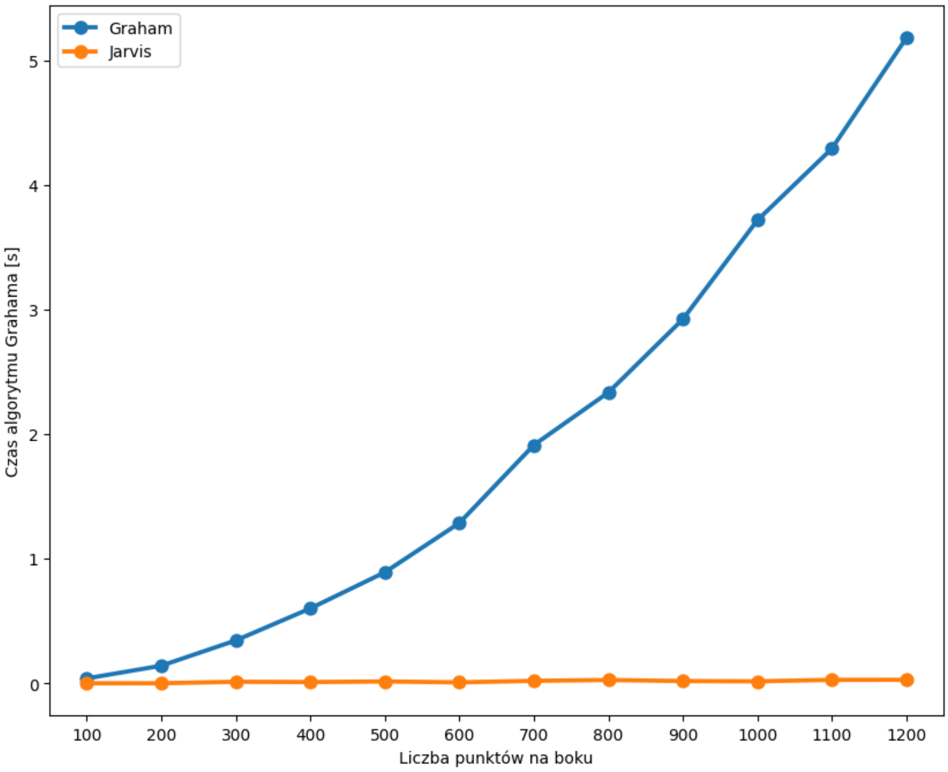


Rysunek 41: Wykres przedstawiający porównanie czasu działania obu algorytmów dla zbiorów typu C

2.4.4 Zbiory typu D

	Liczba punktów na boku	Liczba punktów na przekątnej	Dwa przeciwległe wierzchołki kwadratu	Czas algorytmu Grahama [s]	Czas algorytmu Jarvisa [s]	Szybszy algorytm	Różnica czasu [s]
0	100	50	(28, -29) i (57, 0)	0.039030	0.000000	Algorytm Jarvisa	0.039030
1	200	100	(-37, 15) i (5, 57)	0.141014	0.000000	Algorytm Jarvisa	0.141014
2	300	150	(-49, -17) i (37, 69)	0.344214	0.012004	Algorytm Jarvisa	0.332210
3	400	200	(12, -17) i (28, -1)	0.601213	0.009520	Algorytm Jarvisa	0.591693
4	500	250	(32, -1) i (108, 75)	0.891496	0.014505	Algorytm Jarvisa	0.876990
5	600	300	(-15, -18) i (73, 70)	1.287122	0.007074	Algorytm Jarvisa	1.280047
6	700	350	(12, -3) i (47, 32)	1.913570	0.019640	Algorytm Jarvisa	1.893930
7	800	400	(-1, -21) i (86, 66)	2.335879	0.026200	Algorytm Jarvisa	2.309679
8	900	450	(-35, -41) i (52, 46)	2.924479	0.017846	Algorytm Jarvisa	2.906633
9	1000	500	(10, 25) i (55, 70)	3.719440	0.015605	Algorytm Jarvisa	3.703835
10	1100	550	(25, 4) i (41, 20)	4.295208	0.027505	Algorytm Jarvisa	4.267703
11	1200	600	(39, 36) i (60, 57)	5.184240	0.027686	Algorytm Jarvisa	5.156554

Tabela 4: Wyniki czasowe dla zbiorów typu D



Rysunek 42: Wykres przedstawiający porównanie czasu działania obu algorytmów dla zbiorów typu D

3 Wnioski

Na wykresach z poprzedniej sekcji możemy zauważyć, że algorytm Grahama okazał się szybszy dla zbiorów A oraz B, natomiast dla zbiorów C i D role się odwróciły. Zapewne wynika to z faktu, że algorytm Grahama jest szybszy dla zbiorów, w których otoczka zawiera dużą liczbę punktów wejściowych, natomiast algorytm Jarvisa działa lepiej, gdy otoczka zbioru jest ograniczona przez pewną stałą k - wtedy złożoność algorytmu spada z $O(n^2)$ do $O(k \cdot n)$, co jest lepsze niż złożoność algorytmu Grahama - $O(n \log n)$. Zatem, na podstawie zaprezentowanych tabelek oraz wykresów pokazanych na rysunkach nr 39, 40, 41 oraz 42 możemy stwierdzić że działanie tych algorytmów zgadza się z przewidywaną złożonością.