Algorytmy geometryczne

Sprawozdanie z ćwiczeń nr. 2

Wojciech Łoboda
grupa nr. 1 (środa_13_P_1)

Środowisko oraz narzędzia:

- System operacyjny: Windows 10 x64
- Procesor: Intel Core i7-7700HQ 2.80GHz
- Pamięć RAM: 8 GB
- Środowisko: Jupyter Notebook

Ćwiczenie wykonano przy pomocy języka python3 z wykorzystaniem następujących bibliotek:

- Numpy
- Matplotlib
- Random
- Pandas

Opis realizacji ćwiczenia:

Ćwiczenie polegało na wygenerowaniu odpowiednich zbiorów punktów, zwizualizowaniu ich graficznie oraz na modyfikacji funkcji generującej aby przyjmowała odpowiednie parametry dla zbiorów generowanych. Następnie należało zaimplementować algorytmy Grahama oraz Jarvisa wyznaczające otoczkę wypukłą dla zbioru punktów. Należało również sprawdzić poprawność działania implementacji algorytmów dla zadanych danych, sprawdzić czas działania obu algorytmów oraz zwizualizować kolejne kroki wyznaczania otoczki wypukłej.

Wygenerowane punkty:

Na potrzeby wykonania ćwiczenia zostały wygenerowane 4 zbiory punktów:

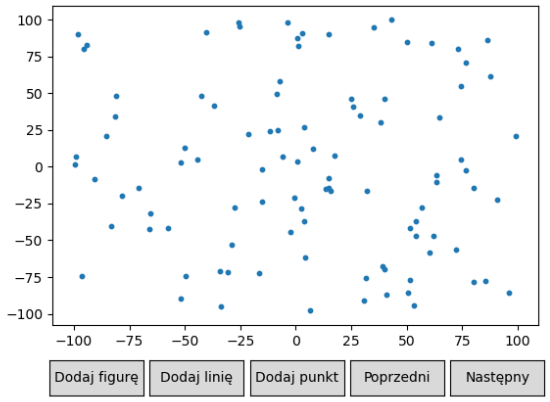
- Zestaw A: 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100].
- Zestaw B: 100 losowo wygenerowanych punktów leżących na okręgu o środku w punkcie (0, 0) i promieniu R = 10.
- Zestaw C: 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10, -10), (10, -10), (10, 10).
- Zestaw D: wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10), po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów leżących na przekątnych kwadratu.

Punkty generowane były przy pomocy funkcji *uniform()* z biblioteki *random*. Funkcja ta zwraca liczbę typu *double* z zadanego przedziału domkniętego. W zestawie B wykorzystana została następująca parametryzacja okręgu:

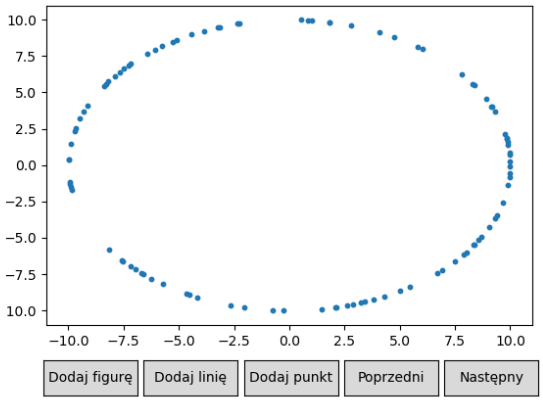
$$B(t) = (\cos(\frac{\pi}{2}t), \sin(\frac{\pi}{2}t)), t \in [0, 4).$$

Wykorzystano funkcje trygonometryczne z biblioteki *numpy*. Generacja punktów z zestawu C i D polegała na stworzeniu wektorów odpowiadających bokom prostokątów, następnie wybraniu losowo jednego z odcinków i wygenerowaniu losowego $t \epsilon [0, 1]$. Losowa zmienna t wyznacza miejsce punktu na odcinku tj. dla t = 0 lub t = 1 punkt będzie znajdował się na końcu odcinka.

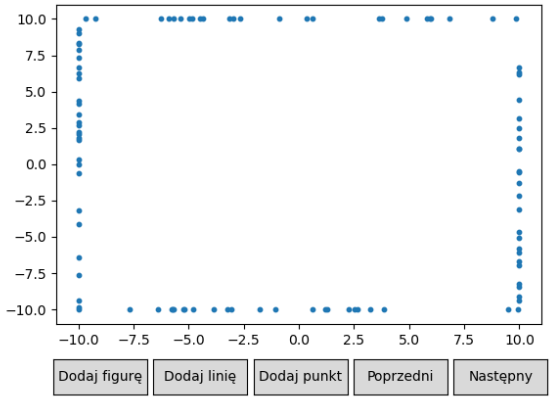
Wykres 1.1 Zestaw A



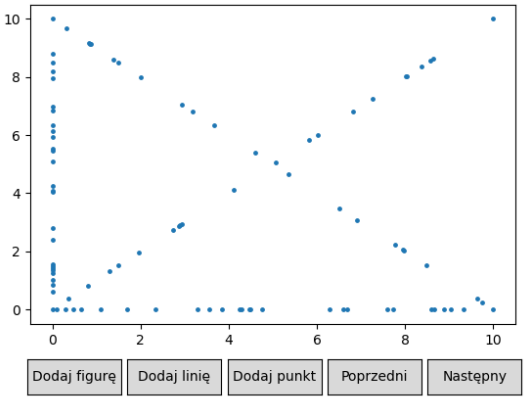
Wykres 1.2 Zestaw B



Wykres 1.3 Zestaw C



Wykres 1.4 Zestaw D



Funkcje generujące punkty należało tak zmodyfikować aby mogły przyjmować odpowiednie parametry dla generowanych zbiorów. Poniżej możliwe parametry dla każdej z funkcji:

- *generate_points_a(no_points, begin, end), no_points* – liczba punktów, *begin, end* – początek i koniec przedziału do którego należeć będą współrzędne.
- *generate_points_b(no_points, x, y, r), no_points* – liczba punktów, *(x, y)* – współrzędne środka okręgu, *r* – długość promienia.
- *generate_points_c(no_points, a, b, c, d), no_points* – liczba punktów, *(a, b, c, d)* – wierchołki prostokąta
- *generate_points_d(a, b, c, d, no_diagonal, no_edges), (a, b, c, d)* – wierchołki kwadratu, *no_diagonal* – liczba punktów na przekątnych, *no_edges* – liczba punktów na krawędziach.

Wykorzystane algorytmy:

Algorytmy wyznaczające otoczkę wypukłą zbioru S:

Algorytm Grahama:

1. W zbiorze S wybieramy punkt p_0 o najmniejszej współrzędnej y. Jeżeli jest kilka takich punktów, to wybieramy ten z nich, który ma najmniejszą współrzędną x.

2. Sortujemy pozostałe punkty ze względu na kąt, jaki tworzy wektor (p_0,p) z dodatnim kierunkiem osi x. Jeśli kilka punktów tworzy ten sam kąt, usuwamy wszystkie z wyjątkiem najbardziej oddalonego od p_0. Niech uzyskanym ciągiem będzie $p_1, p_2,...p_m$.

3. Do początkowo pustego stosu s wkładamy punkty p_0, p_1, p_2.
t – indeks stosu; i ← 3

4**. while** i < m **do**

if pi leży na lewo od prostej (p_{t-1}, p_t)

then push(p_i), i ← i+1

else pop(s)

Punkty znajdujące się na koniec na stosie tworzą otoczkę wypukłą.

Złożoność: O(n) + O(nlogn) + O(1) + O(n - 3) = O(nlogn), n = $|S|$

Algorytm Jarvisa:

1. znajdź punkt i_0 z S o najmniejszej współrzędnej y; i ← i_0

2. **repeat**
   for j ≠ i do
   znajdź punkt, dla którego kąt liczony przeciwnie do wskazówek zegara w odniesieniu do ostatniej krawędzi otoczki jest najmniejszy niech k będzie indeksem punktu z najmniejszym kątem zwróć (pi, pk) jako krawędź otoczki i ← k

until i = i_0

Złożoność: Złożonośc obliczeniowa algorytmu jarvisa wynosi $O(n^2)$, dla liczby wierzchołków otoczki wypukłej k wynoki O(nk). Dla n = $|S|$.

Do liczenia orientacji punktów wykorzystany został wyznacznik:

$$\det(a,b,c) = \begin{vmatrix} a_x - c_x & a_y - c_y \\ b_x - c_x & b_y - c_y \end{vmatrix} \qquad \begin{cases} < 0 \\ > 0 \\ = 0 \end{cases}$$

Dla det(a, b, c) > 0, punkty zorientowane są przeciwnie do ruchu wskazówek zegara.

Dla det(a, b, c) < 0, punkty zorientowane są zgodnie do ruchu wskazówek zegara.

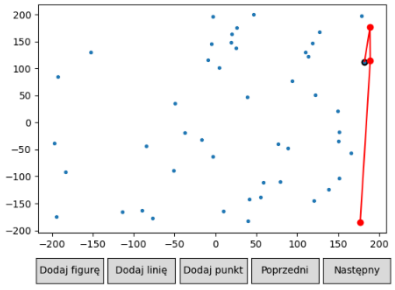
Dla det(a, b, c) = 0, punkty są współliniowe.

Wizualizacja działania algorytmów:

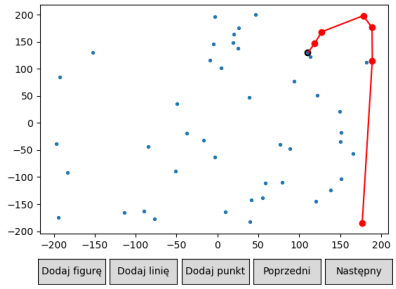
Poniżej znajduje się wizualizacja graficzna wybranych kroków działania obu algorytmów dla zbiorów z odpowiednich podpunktów ze zmodyfikowanymi parametrami. Przyjęta tolerancja dla zera wynosi 10^{-14}. Wizualizacja każdego kroku dostępna jest w jupyter notebooku. Punkt oznaczony kolorem czarnym to punkt aktualnie rozpatrywany, punkty oznaczone na czerwono są kandydatami do otoczki wypukłej (krawędzie między nimi są również czerwone), Zielone punkty należą do otoczki (zielone krawędzie wyznaczają otoczke)

1. Algorytm Grahama:

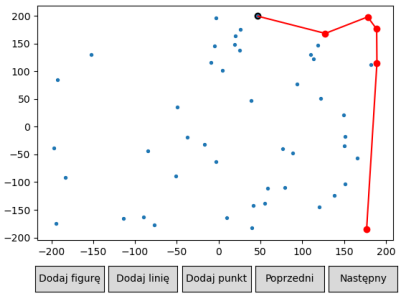
Zbiór A, parametry: liczba punktów: 50, przedział: [-200, 200]



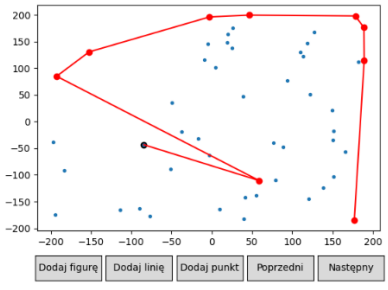
Krok 0/86



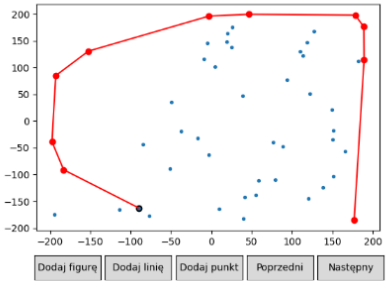
Krok 10/86



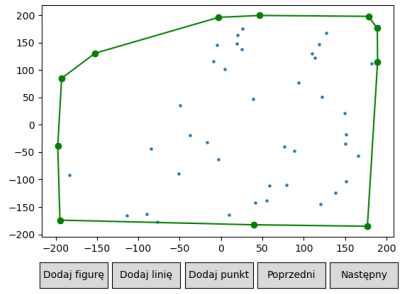
Krok 29/86



Krok 64/86

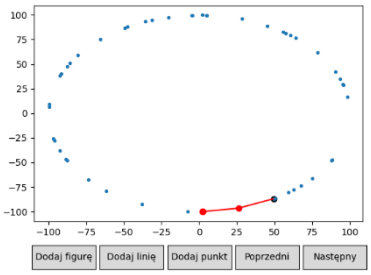


Krok 75/86

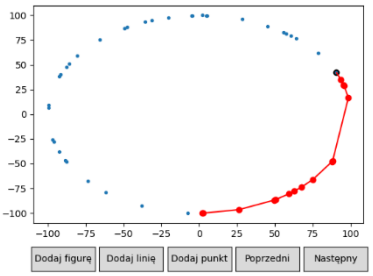


Krok 86/86

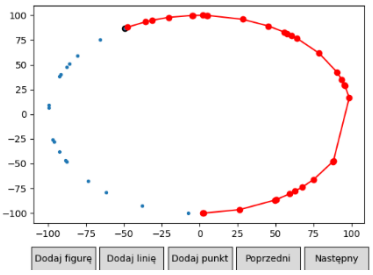
Zbiór B, parametry: liczba punktów: 50, środek = (0, 0), R = 100.



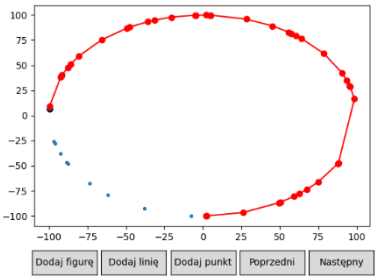
Krok 0/47



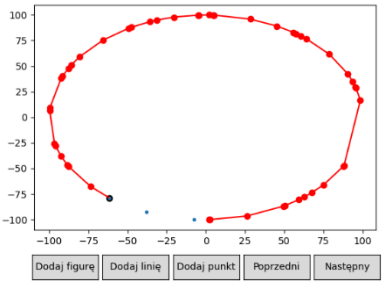
Krok 14/47



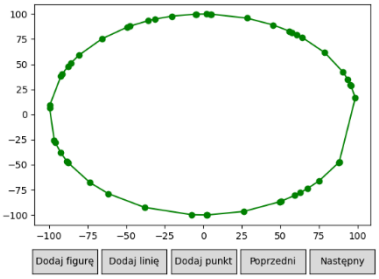
Krok 27/47



Krok 36/47

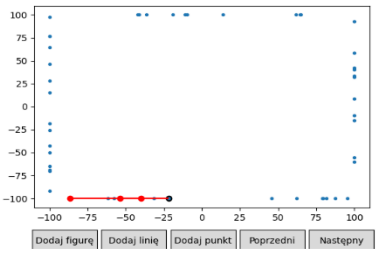


Krok 45/47

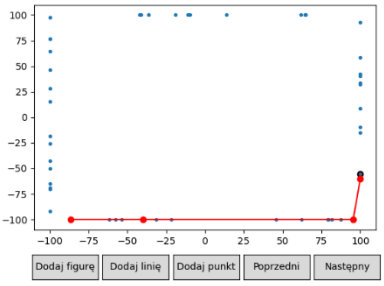


Krok 47/47

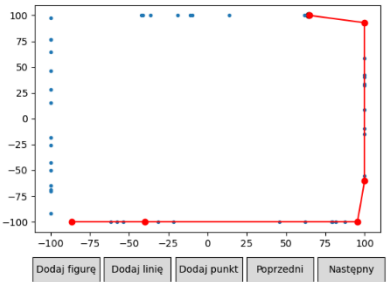
Zbiór C, parametry: liczba punktów: 50, wierzchołki: (-100, -100), (100, -100), (100, 100), (-100, 100).



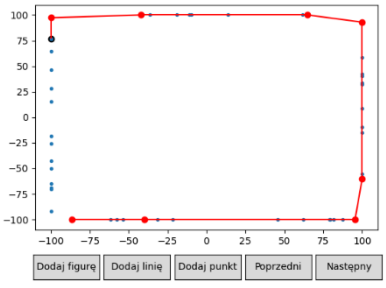
Krok 0/47



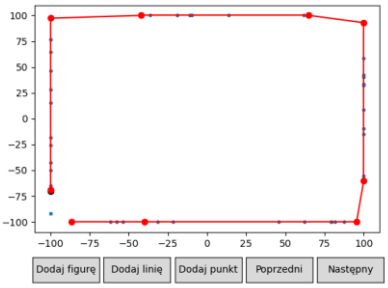
Krok 11/47



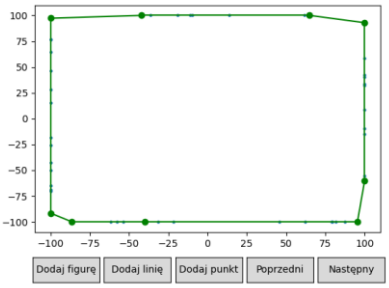
Krok 24/47



Krok 35/47

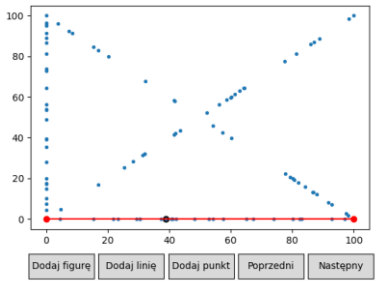


Krok 41/47

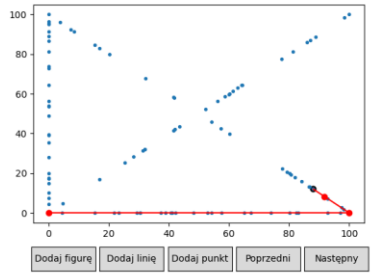


Krok 47/47

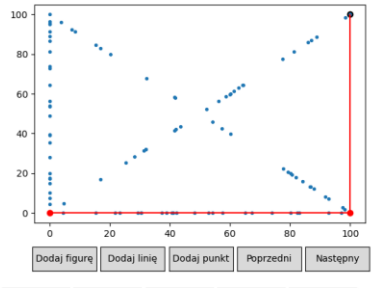
Zbiór D, parametry: liczba punktów na bokach: 25, liczba punktów na przekątnych 25, wierzchołki: (0, 100), (0, 0), (100, 0), (100, 100). Przy tolerancji na zero 10^{-12}.



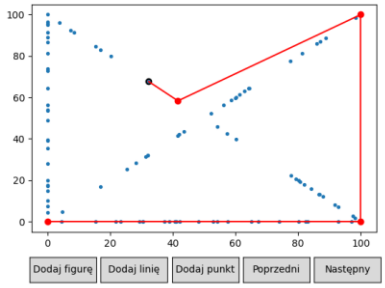
Krok 0/118



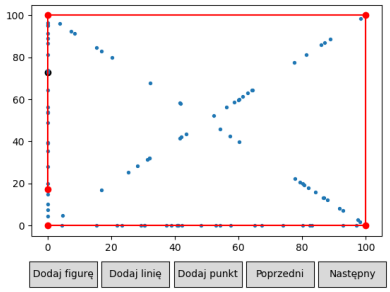
Krok 24/118



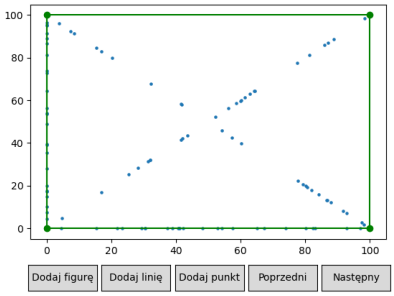
Krok 39/118



Krok 68/118



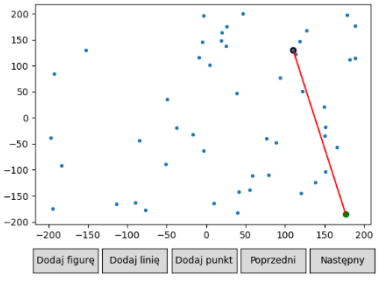
Krok 112/118



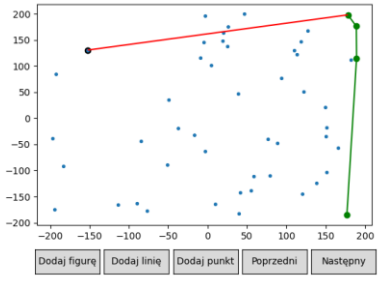
Krok 118/118

2. Algorytm Jarvisa:

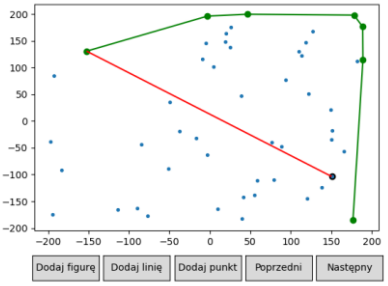
Zbiór A, parametry: liczba punktów: 50, przedział: [-200, 200].



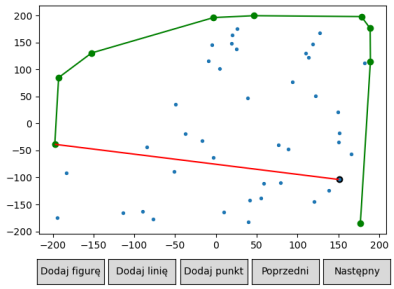
Krok 0/51



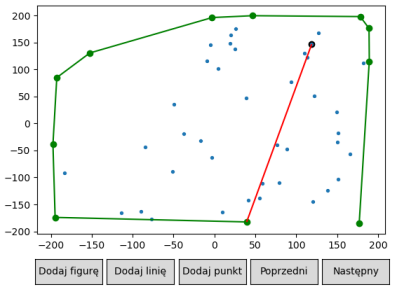
Krok 15/51



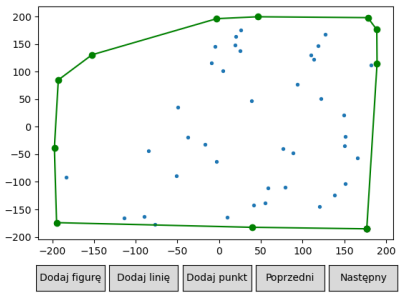
Krok 33/51



Krok 36/51

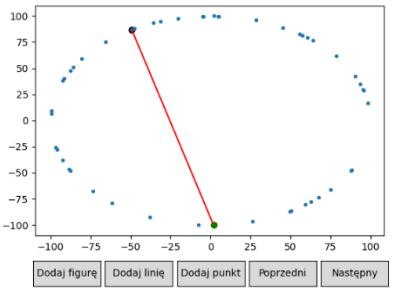


Krok 47/51

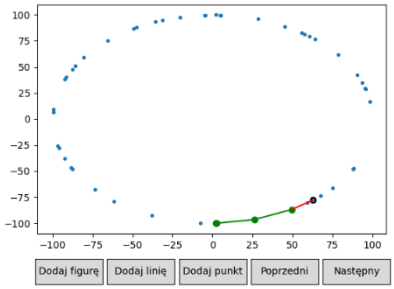


Krok 51/51

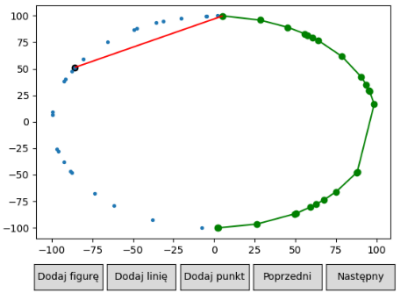
Zbiór B, parametry: liczba punktów: 50, środek = (0, 0), R = 100.



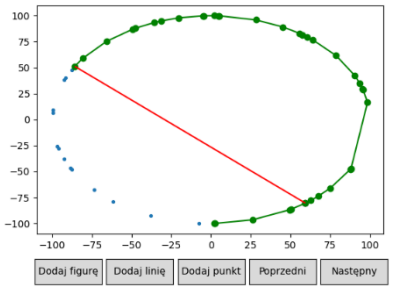
Krok 0/209



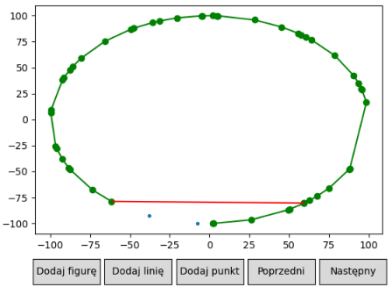
Krok 23/209



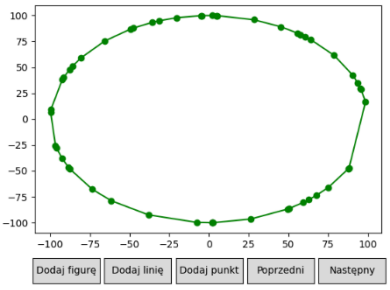
Krok 118/209



Krok 131/209

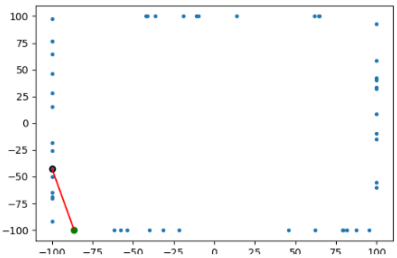


Krok 197/209

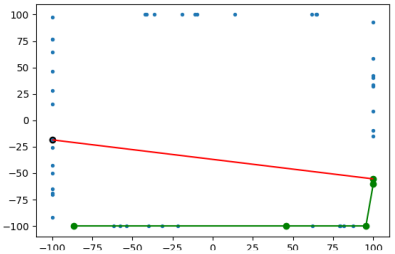


Krok 209/209

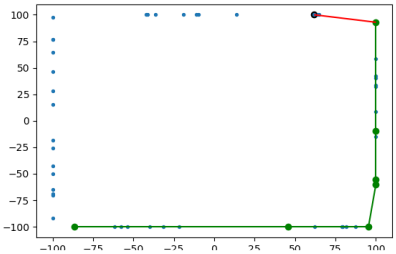
Zbiór C, parametry: liczba punktów: 50, wierzchołki: (-100, -100), (100, -100), (100, 100), (-100, 100).



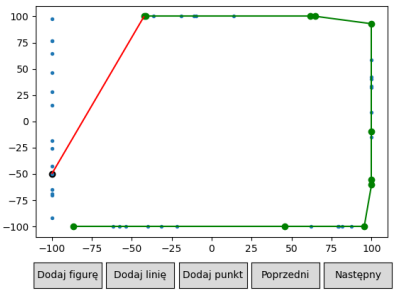
Krok 0/56



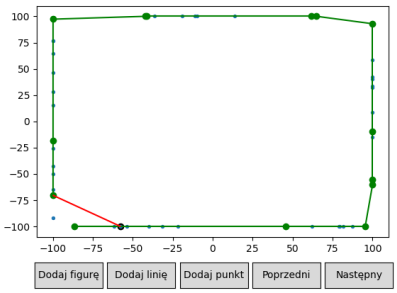
Krok 12/56



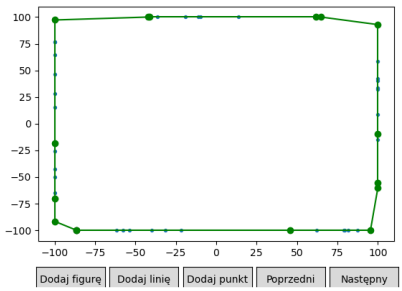
Krok 25/56



Krok 36/56

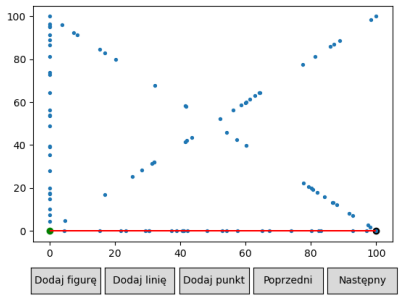


Krok 53/56

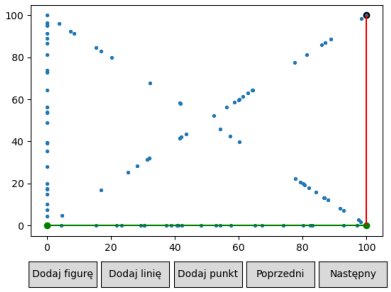


Krok 56/56

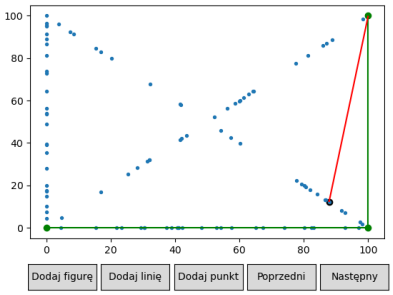
Zbiór D, parametry: liczba punktów na bokach: 25, liczba punktów na przekątnych 25, wierzchołki: (0, 100), (0, 0), (100, 0), (100, 100).



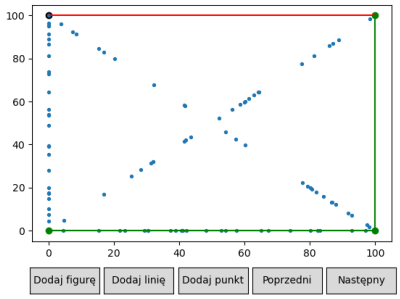
Krok 0/5



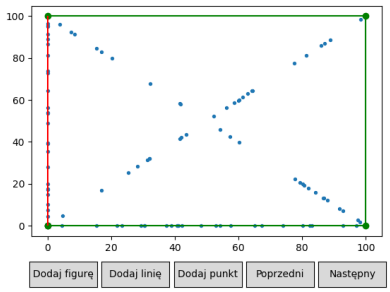
Krok 1/5



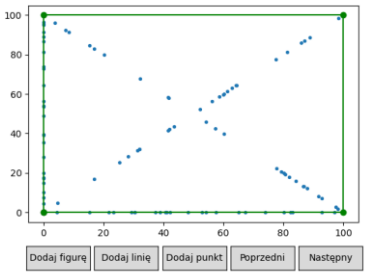
Krok 2/5



Krok 3/5



Krok 4/5



Krok 5/5

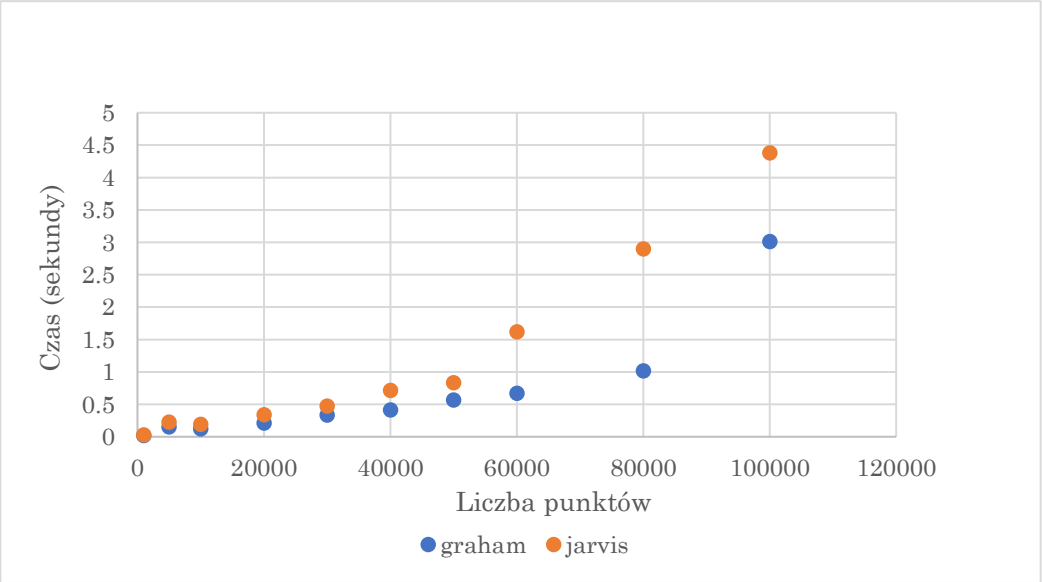
Porównanie czasu działania algorytmów:

Poniżej znajdują się tabele i wykresy w których zamieszczono czasy działania algorytmów Grahama i Jarvisa dla różnych ilości punktów w zbiorach przy ustalonych parametrach podanych nad wykresem i tabelą

Zestaw A. Przedział: [-1000, 1000].

liczba punktów	czas algorytmu Grahama (sekundy)	czas algorytmu Jarvisa (sekundy)	Szybszy algorytm	Różnica czasu (sekundy)
1000	0.01411	0.02408	Graham	0.00997
5000	0.14893	0.22549	Graham	0.07656
10000	0.12122	0.18742	Graham	0.06620
20000	0.20798	0.33858	Graham	0.13060
30000	0.33334	0.47187	Graham	0.13853
40000	0.41215	0.71404	Graham	0.30189
50000	0.56555	0.82948	Graham	0.26393
60000	0.66849	1.61891	Graham	0.95042
80000	1.01492	2.89532	Graham	1.88040
100000	3.00968	4.37848	Graham	1.36880

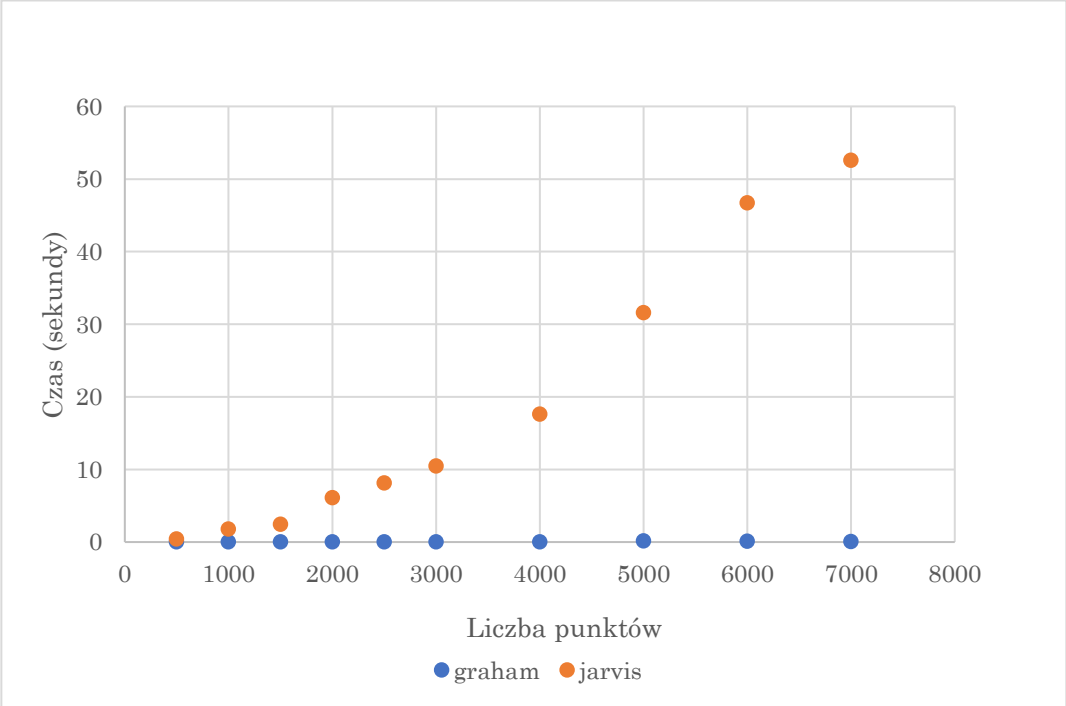
Wykres 3.1 Zestaw A, porównanie czasu algorytmu Grahama i Jarvisa.



Zestaw B. Środek okręgu: (0, 0), Promień: 10000.

liczba punktów	czas algorytmu Grahama (sekundy)	czas algorytmu Jarvisa (sekundy)	Szybszy algorytm	Różnica czasu (sekundy)
500	0.01134	0.41020	Graham	0.39886
1000	0.03416	1.79012	Graham	1.75596
1500	0.01432	2.46025	Graham	2.44593
2000	0.04020	6.09502	Graham	6.05482
2500	0.02052	8.14219	Graham	8.12167
3000	0.03910	10.47388	Graham	10.43478
4000	0.04042	17.61821	Graham	17.57779
5000	0.16296	31.57636	Graham	31.41340
6000	0.10977	46.71080	Graham	46.60103
7000	0.07444	52.59662	Graham	52.52218

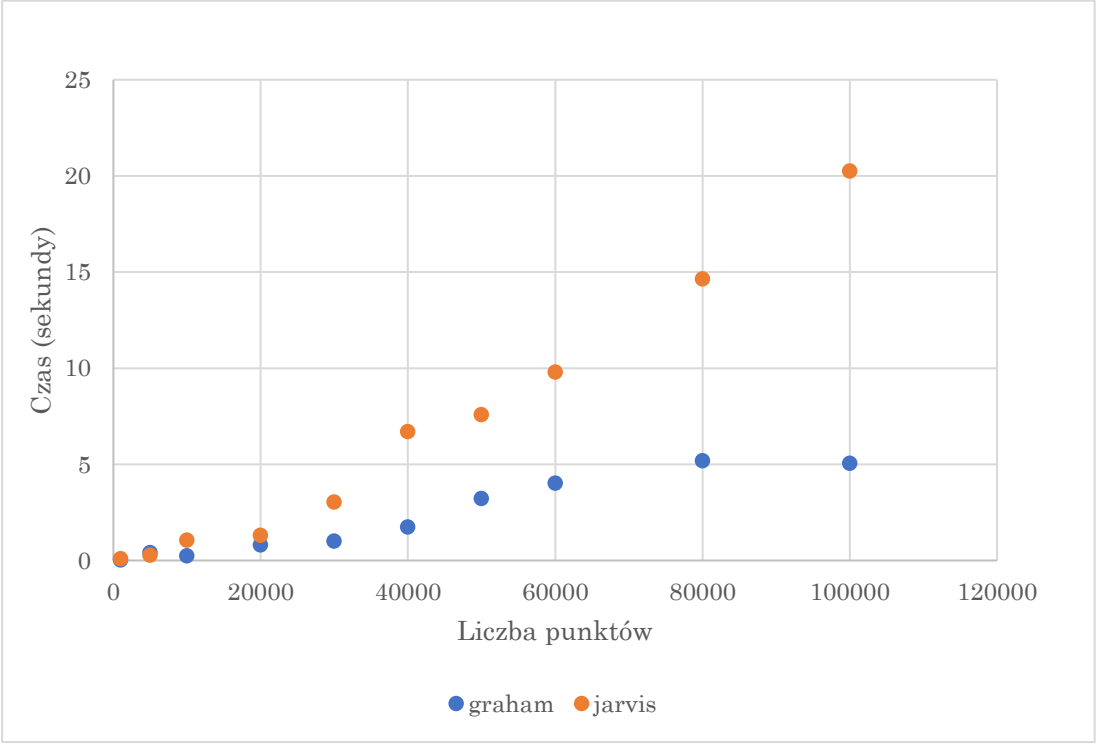
Wykres 3.2 Zestaw B, porównanie czasu algorytmu Grahama i Jarvisa.



Zestaw C. Wierzchołki: (-1000, 1000), (1000, -1000), (1000, 1000), (-1000, 1000).

liczba punktów	czas algorytmu Grahama (sekundy)	czas algorytmu Jarvisa (sekundy)	Szybszy algorytm	Różnica czasu (sekundy)
1000	0.03748	0.09758	Graham	0.06010
5000	0.42887	0.29335	Jarvis	0.13552
10000	0.26178	1.06838	Graham	0.80660
20000	0.81785	1.31762	Graham	0.49977
30000	1.01343	3.05513	Graham	2.04170
40000	1.75627	6.72133	Graham	4.96506
50000	3.23371	7.59809	Graham	4.36438
60000	4.03228	9.80747	Graham	5.77519
80000	5.19758	14.65241	Graham	9.45483
100000	5.06454	20.27151	Graham	15.20697

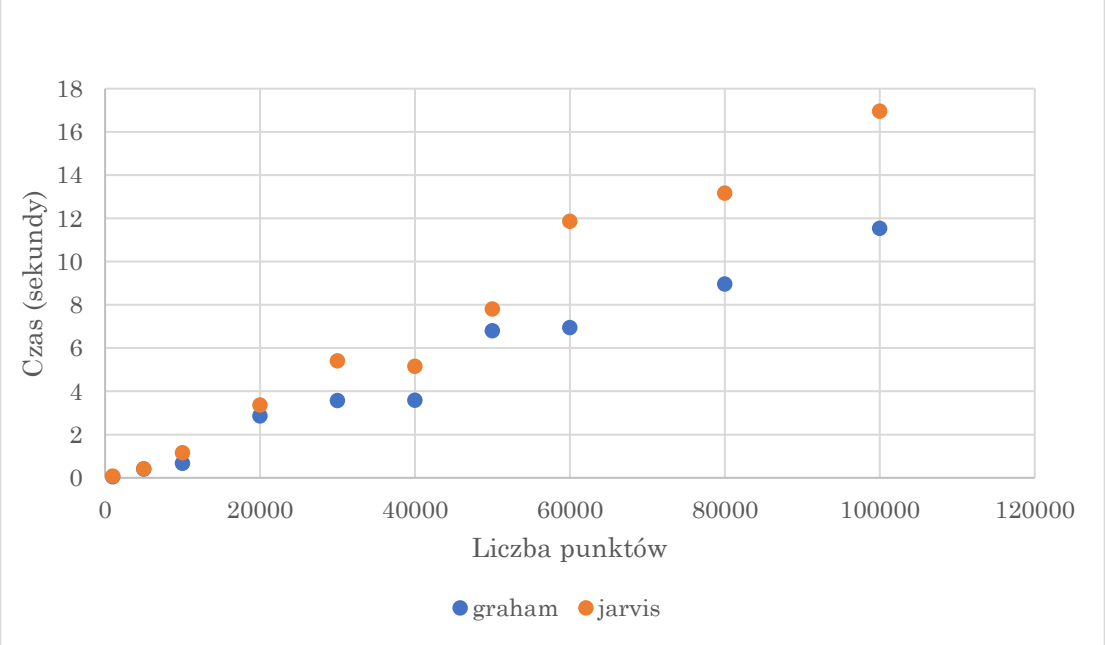
Wykres 3.3 Zestaw C, porównanie czasu algorytmu Grahama i Jarvisa.



Zestaw D. Wierzchołki: (-1000, 1000), (-1000, -1000), (1000, -1000), (1000, 1000). 75% punktów na osiach, 25% punktów na przekątnych:

liczba punktów	czas algorytmu Grahama (sekundy)	czas algorytmu Jarvisa (sekundy)	Szybszy algorytm	Różnica czasu (sekundy)
1000	0.05842	0.07818	Graham	0.01976
5000	0.40339	0.43143	Graham	0.02804
10000	0.67589	1.16787	Graham	0.49198
20000	2.87356	3.37424	Graham	0.50068
30000	3.58332	5.40657	Graham	1.82325
40000	3.59978	5.16303	Graham	1.56325
50000	6.7996	7.81502	Graham	1.01542
60000	6.95363	11.87514	Graham	4.92151
80000	8.96529	13.17275	Graham	4.20746
100000	11.54876	16.96755	Graham	5.41879

Wykres 3.4 Zestaw D, porównanie czasu algorytmu Grahama i Jarvisa.



Wnioski:

Algorytmy Grahama oraz Jarvisa zaimplementowane na potrzeby ćwiczenia poprawnie wyznaczyły otoczkę wypukłą dla zbiorów A i B z zadanymi parametrami oraz ze zmodyfikowanymi parametrami. W zbiorze C i D otoczka obliczona przez algorytmy zawiera niewielką ilość punktów które teoretycznie powinny leżeć na bokach prostokąta i nie należeć do otoczki, jednak dzięki małej tolerancji na zero wynoszącej 10^{-14} punkty te nie zostały odpowiednio zakwalifikowane. Zbiory zostały zaproponowane aby przetestować poprawność działania oraz szybkość wykonania algorytmu w skrajnie różnych przypadkach. Algorytm Grahama okazał się być szybszy dla każdego zbioru danych co prawdopodobnie wynika ze sposobu implementacji obu algorytmów, największą różnice widać w zbiorze B w którym do otoczki wypukłej powinien należeć każdy punk zbioru. W tym przypadku złożoność obliczeniowa algorytmu Jarvisa wynosi $O(n^2)$, gdzie n to liczba punktów w zbiorze i znacznie przewyższa złożoność obliczeniową algorytmu Grahama wynoszącą n* log(n).

Zgodnie z oczekiwaniami po zwiększeniu tolerancji na zero na 10^{-12} Algorytmy poprawnie wyznaczają otoczkę również dla zbiorów C i D.