**Sprawozdanie z laboratorium 2**

**Temat: Otoczka wypukła**

**Data: 17.11.2023**

**Algorytmy geometryczne**

Wojciech Kaźmierczak

Nr albumu: 416692

Gr. 4, czwartek 11:20, tydzień A

Informatyka, Wydział Informatyki

AGH UST

Procesor: AMD Ryzen 7 5700U 8 rdzeni

RAM: 16GB

System: Windows 11 Home

Użyty język: Python, Biblioteki: numpy, pandas, matplotlib

Środowisko: Jupyter Notebook

1. **Cel ćwiczenia**

Ćwiczenie polegało na wyznaczeniu otoczki wypukłej zbioru punktów przy użyciu algorytmu Graham’a oraz Jarvis’a.

1. **Generowanie punktów zbiorów**

Wygenerowane zostaną 4 zbiory punktów na płaszczyźnie (współrzędne rzeczywiste typu double):

1. Zbiór A zawierający 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100], funkcja generate\_uniform\_points(left=-100, right=100, n=100)

b) Zbiór B zawierający 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10, funkcja generate\_circle\_points(O, R, n=100)

c) Zbiór C zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10), funkcja generate\_rectangle\_points(a=(-10,-10), b=(10,-10), c=(10,10), d=(-10,10), n=100)

d) Zbiór D zawierający wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo w sposób następujący: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu, funkcja generate\_square\_points(a=(0,0), b=(10,0), c=(10,10), d=(0,10), axis\_n=25, diag\_n=20)

Na końcu wygenerowane zostaną ponadto zbiory o liczebnościach 25, 50, 75, 100, 200, 300, 400, 500, 1000 dla (zbioru D liczebności nieparzyste zastąpione pobliskimi parzystymi z uwagi na założenia rozłożenia punktów w zbiorze). Ponadto dla zbioru B dobierano losowe środki okręgu z współrzędnymi o wartościach z przedziału [-100, 100) oraz długość promienia z przedziału [1, 30). Dla zbioru D wartość z tablicy: 25, 50, 75, 100, 200, 300, 400, 500, 1000 podzielona przez dwa całkowitoliczbowo jest górną granicą dodatniej wartości losowej, którą jest liczba punktów na boku, a liczba punktów na przekątnej wynosi wartość z wyżej wymienionej tablicy odjąć dwukrotność liczby punktów na boku i to całe podzielone przez dwa całkowitoliczbowo.

Cztery pierwsze wygenerowane zbiory:

Wykres 1.1 Zbiór A Wykres 1.2 Zbiór B

Obraz zawierający zrzut ekranu, diagram

Opis wygenerowany automatycznie Obraz zawierający zrzut ekranu, krąg, diagram, linia

Opis wygenerowany automatycznie

Wykres 1.3 Zbiór C Wykres 1.4 Zbiór D

Obraz zawierający zrzut ekranu, tekst, Prostokąt, linia

Opis wygenerowany automatycznie Obraz zawierający zrzut ekranu, linia, diagram, Wykres

Opis wygenerowany automatycznie

1. Implementacja algorytmów

Wyznacznik obliczano zaimplementowaną funkcją obliczającą wyznacznik 2x2, przyjęty epsilon reprezentujący tolerancję błędu ustawiony na 0.

* 1. **Algorytm Graham’a**

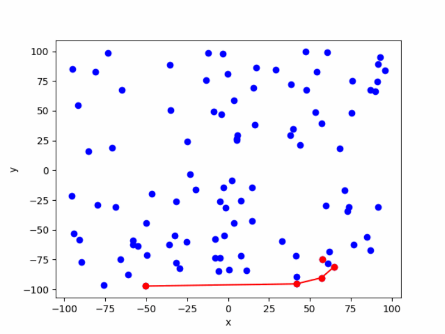
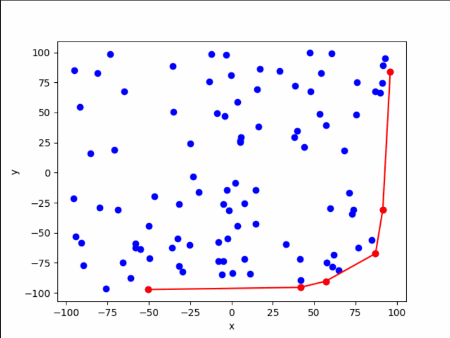
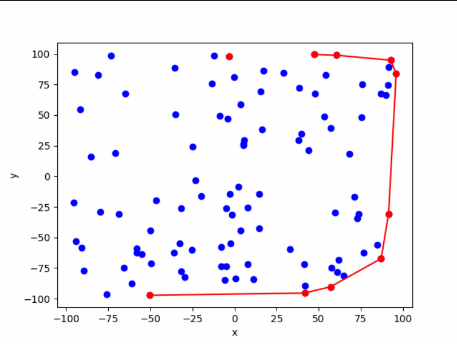
Działanie algorytmu rozpoczyna się od znalezienia punktu p0 z najmniejszą współrzędną y, a jeśli jest ich więcej niż jeden to także z najmniejszą wartością współrzędnej x. Po czym rozpoczyna się sortowanie punktów względem kąta między wektorem utworzonym przez punkty p0 oraz aktualnie rozważany punkt, a dodatnią częścią osi OX. Z punktów o tym samym kącie pozostawia tylko ten, który jest najbardziej odległy od p0. Dodajemy trzy pierwsze wierzchołki z posortowanej listy do otoczki. Następnie przetwarzając kolejne posortowane wierzchołki, jeśli dany tworzy lewoskręt względem wektora tworzonego przez ostatnie dwa wierzchołki dodane do otoczki to i on jest dodawany do otoczki jednak, jeśli nie ostatni punkt dodany do otoczki jest z niej usuwany, a aktualnie przetwarzany trafia na jego miejsce. Wykonujemy te operacje dla wszystkich punktów. Złożoność czasowa to O(nlogn) gdzie n to liczba punktów zbioru.

* 1. **Algorytm Jarvis’a**

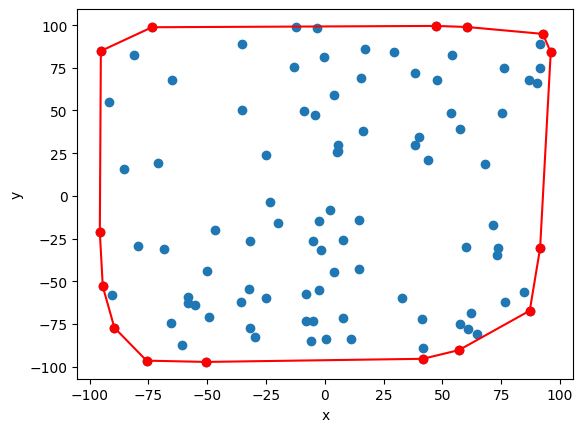
Działanie algorytmu rozpoczyna się od znalezienia punktu p0 z najmniejszą współrzędną y, a jeśli jest ich więcej niż jeden to także z najmniejszą wartością współrzędnej x. następnie wybieramy dowolny punkt z tego zestawu i oznaczamy go jako next potem iterujemy po każdym punkcie w zbiorze i patrzymy czy next nie ma żadnego prawostronnego sąsiada, jeśli ma to ten sąsiad staje się nowym next. I tak iterujemy aż nie dotrzemy do p0. Złożoność czasowa to O(nk), gdzie n to liczba punktów zbioru, a k to liczba punktów na otoczce, w pesymistycznym przypadku złożoność może osiągać O(n^2).

1. **Wyznaczanie otoczek**
   1. **Algorytm Grahama**
      1. **Zbiór A**

**Wykres 2.1 Wykres 2.2 Wykres 2.3**

**  **

**Wykres 2.4 otoczka wypukła zbioru A**

****

* + 1. **Zbiór B**

**Wykres 3.1 Wykres 3.2 Wykres 3.3**

Obraz zawierający zrzut ekranu, krąg, diagram, linia

Opis wygenerowany automatycznieObraz zawierający zrzut ekranu, krąg, diagram, Wykres

Opis wygenerowany automatycznie Obraz zawierający zrzut ekranu, krąg, diagram, linia

Opis wygenerowany automatycznie

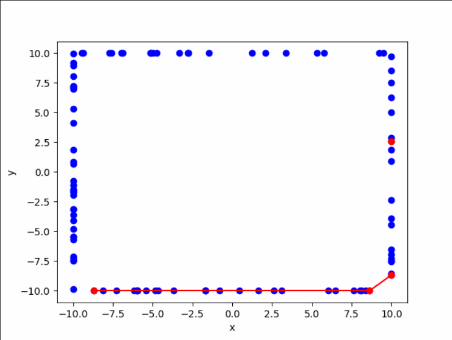
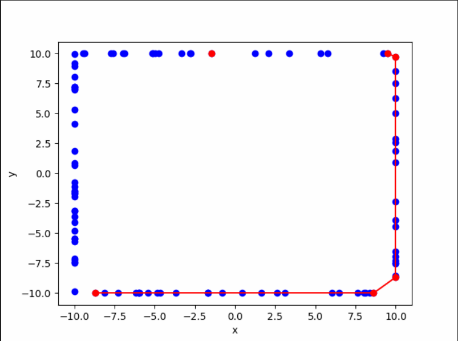
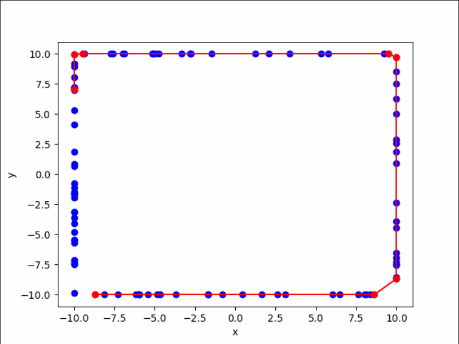
**Wykres 3.4 otoczka wypukła zbioru B**

Obraz zawierający zrzut ekranu, krąg, tekst, diagram

Opis wygenerowany automatycznie

* + 1. **Zbiór C**

**Wykres 4.1 Wykres 4.2 Wykres 4.3**

**  **

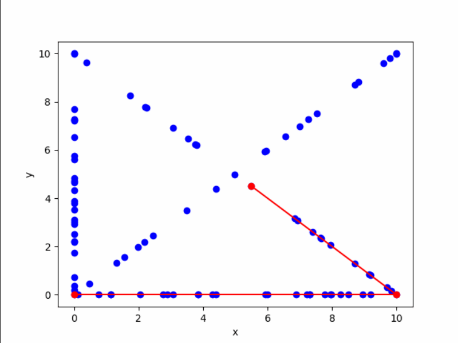
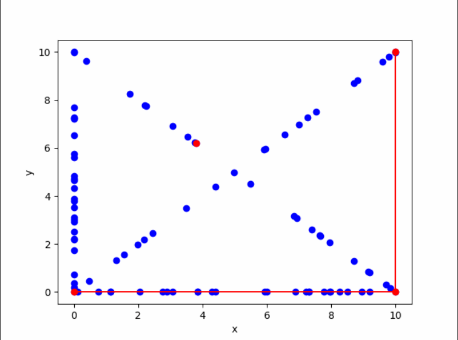
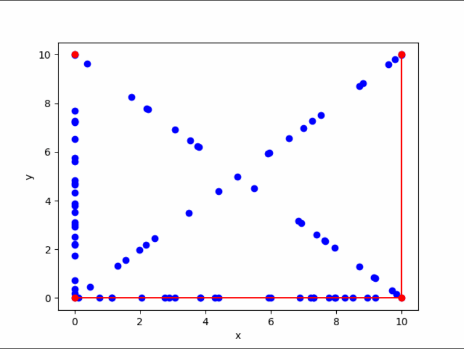
**Wykres 4.4 otoczka wypukła zbioru C**

**Obraz zawierający tekst, zrzut ekranu, linia, Wykres

Opis wygenerowany automatycznie**

* + 1. **Zbiór D**

**Wykres 5.1 Wykres 5.2 Wykres 5.3**

**  **

**Wykres 5.4 otoczka wypukła zbioru D**

**Obraz zawierający zrzut ekranu, linia, diagram, Wykres

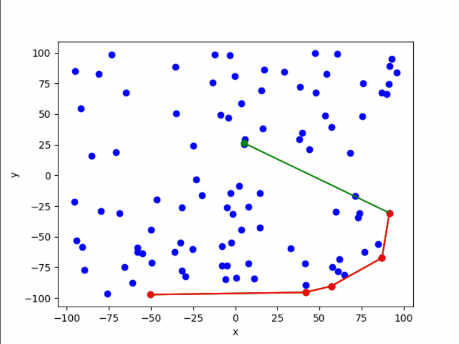
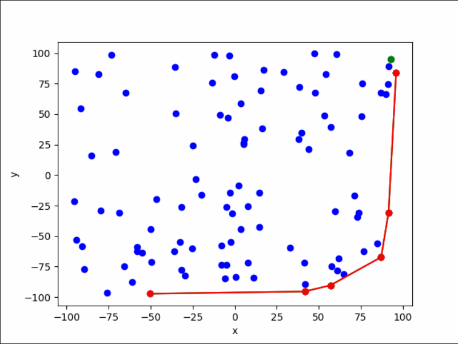
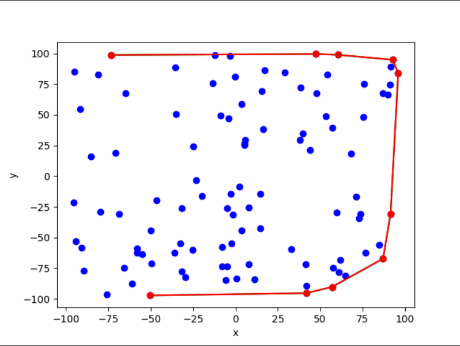
Opis wygenerowany automatycznie**

* 1. **Algorytm Jarvis’a**

**Dla algorytmu Jarvis wedle oczekiwań otoczki wypukłe były tożsame tym uzyskanym przy użyciu Graham’a (wykresy 2.4, 3.4, 4.4, 5.4) jednak oczywiście różniły się sposobem ich otrzymania. Na wykresach kolorem zielonym oznaczono punkty, które są w danym momencie sprawdzane jako potencjalne punkty otoczki.**

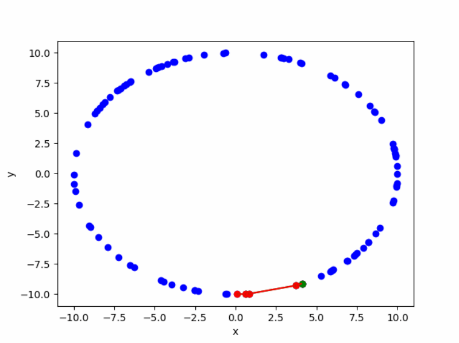
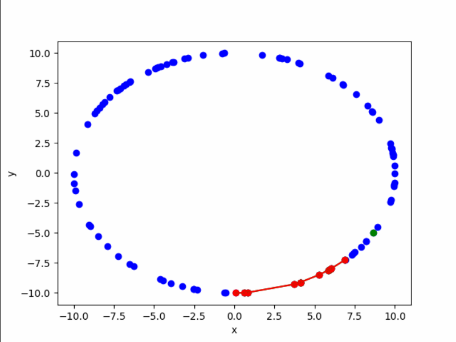
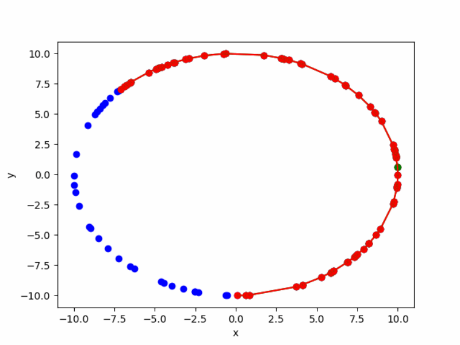
* + 1. **Zbiór A**

**Wykres 6.1 Wykres 6.2 Wykres 6.3**

**  **

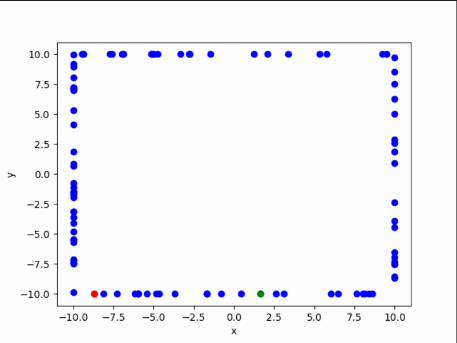
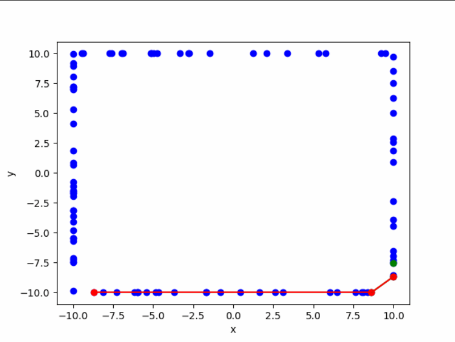
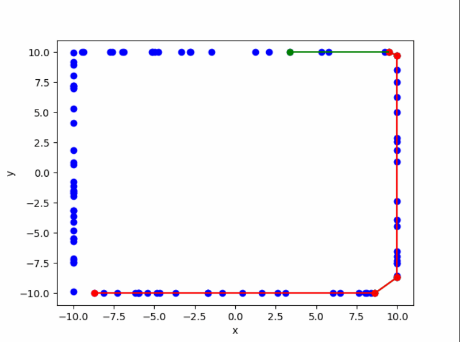
* + 1. **Zbiór B**

**Wykres 7.1 Wykres 7.2 Wykres 7.3**

**  **

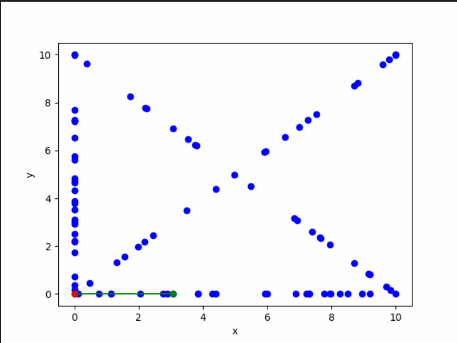
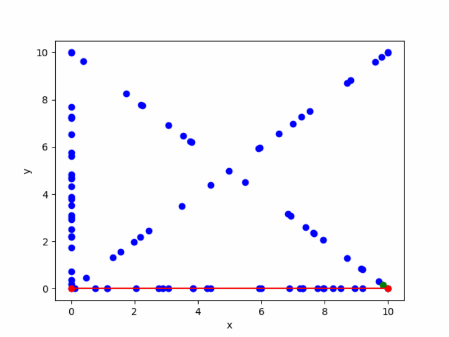
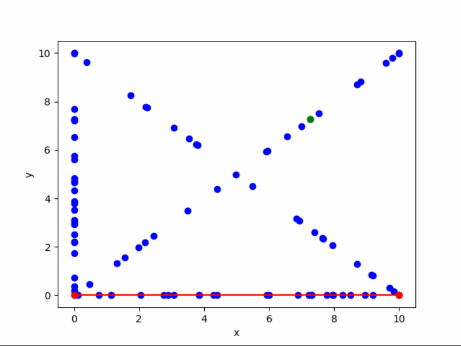
* + 1. **Zbiór C**

**Wykres 8.1 Wykres 8.2 Wykres 8.3**

**  **

* + 1. **Zbiór D**

**Wykres 9.1 Wykres 9.2 Wykres 9.3**

**  **

**Wszystkie powyższe wykresy są krokami algorytmów kolejno Graham’a i Jarvis’a pełna wizualizacja w postaci GIFa zawarta jest w Jupyter notebook, zrealizowana jest ona przy pomocy dostarczonego narzędzia Visualizer().**

1. **Porównanie czasów działania algorytmów**

Porównanie czasów działania algorytmów zostało zrealizowane na zbiorach nowo wygenerowanych informacja o owych zbiorach znajduje się w punkcie 2 (Generowanie punktów zbiorów).

Czas działania algorytmów zmierzony został przy pomocy biblioteki time a mianowicie funkcji time\_ns(), która to zawraca czas w nanosekundach [ns] w tych też jednostkach będzie on podawany od tego momentu. Wyniki dla każdego zbioru w tabelach.

* 1. **Wyniki dla zbioru A generowanego funkcją :**

**generate\_uniform\_points(left=-100, right=100, n)**

**Tabela 1 Zmierzone czasy dla zbioru A**

|  |  |  |  |
| --- | --- | --- | --- |
| **Liczba punktów** | **Graham [ns]** | **Jarvis [ns]** | **Różnica czasu [ns] (Graham-Jarvis)** |
| 25 | 253293 | 692411 | -439118 |
| 50 | 185063 | 865751 | -680688 |
| 75 | 323186 | 1200861 | -877675 |
| 100 | 383030 | 3937568 | -3554538 |
| 200 | 821938 | 5542191 | -4720253 |
| 300 | 1293908 | 9309187 | -8015279 |
| 400 | 3149587 | 12641733 | -9492146 |
| 500 | 2589860 | 12584304 | -9994444 |
| 1000 | 5825593 | 28240596 | -22415003 |

Wykres 10.1 Obrazuje czas działania obu algorytmów dla zbioru A w zależności od liczby punktów

**Obraz zawierający tekst, diagram, Wykres, linia

Opis wygenerowany automatycznie**

Łatwo zauważyć, iż czas działania algorytmu Grahama jest znacznie mniejszy niż algorytmu Jarvisa, krzywa ukazująca czasy działania algorytmu Jarvisa przypomina kawałek funkcji kwadratowej co może świadczyć, iż dla zbioru A czas działania owego algorytmu jest przybliżony do kwadratu liczby punktów zbioru.

* 1. **Wyniki dla zbioru B generowanego funkcją :**

**generate\_circle\_points(O, R, n) , środek okręgu oraz promień wybierany losowo**

**Tabela 2 Zmierzone czasy dla zbioru B**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Liczba punktów** | **Promień** | **Środek okręgu** | **Graham [ns]** | **Jarvis [ns]** | **Różnica czasu [ns] (Graham-Jarvis)** |
| 25 | 17 | (-55, -66) | 128064 | 1151767 | -1023703 |
| 50 | 7 | (23, 37) | 155015 | 4860160 | -4705145 |
| 75 | 8 | (90, 98) | 308739 | 9856197 | -9547458 |
| 100 | 4 | (26, -39) | 373493 | 15576080 | -15202587 |
| 200 | 11 | (29, -81) | 783324 | 61639555 | -60856231 |
| 300 | 1 | (-65, -13) | 1285842 | 112107355 | -110821513 |
| 400 | 20 | (54, -89) | 1361016 | 197154989 | -195793973 |
| 500 | 22 | (8, -63) | 1676398 | 295348911 | -293672513 |
| 1000 | 4 | (-54, 3) | 4563044 | 1959128676 | -1954565632 |

Wykres 10.2 Obrazuje czas działania obu algorytmów dla zbioru C w zależności od liczby punktów

Obraz zawierający tekst, zrzut ekranu, Wykres, diagram

Opis wygenerowany automatycznie

Podobnie jak w poprzednim przypadku (na wykresie 10.1) taki tutaj (wykres 10.2) widać, znaczną różnicę jeśli chodzi o czas działania tych algorytmów. Ponownie obserwujemy kształt przypominający część funkcji kwadratowej w przypadku wykresu dla algorytmu Jarvis’a. Różnice w wartościach są tak ogromne, że czas działania algorytmu Graham’a wydaje się niemal liniowy, lecz wiemy, że tak nie jest.

* 1. **Wyniki dla zbioru C generowanego funkcją :**

**generate\_rectangle\_points(a=(-10,-10), b=(10,-10), c=(10,10), d=(-10,10), n)**

**Tabela 3 Zmierzone czasy dla zbioru C**

|  |  |  |  |
| --- | --- | --- | --- |
| **Liczba punktów** | **Graham [ns]** | **Jarvis [ns]** | **Różnica czasu [ns] (Graham-Jarvis)** |
| 25 | 390004 | 435310 | -45306 |
| 50 | 639711 | 795597 | -155886 |
| 75 | 1190661 | 1383359 | -192698 |
| 100 | 1765718 | 1238091 | 527627 |
| 200 | 3338487 | 2593567 | 744920 |
| 300 | 4744790 | 4431462 | 313328 |
| 400 | 7629553 | 5934118 | 1695435 |
| 500 | 11620515 | 8934462 | 2686053 |
| 1000 | 33396142 | 19504447 | 13891695 |

Wykres 10.3 Obrazuje czas działania obu algorytmów dla zbioru C w zależności od liczby punktów

Obraz zawierający tekst, zrzut ekranu, Wykres, linia

Opis wygenerowany automatycznie

W tym przypadku widzimy, że wykres 10.3 różni się znacznie od 10.1 oraz 10.2, czasy działanie są zbliżone, a dla większych zbiorów danych nawet korzystniejsze dla algorytmu Jarvisa, wynika to zapewne z specyfiki rozmieszczenia punktów na prostokącie i w tym przypadku algorytm Jarvisa zyskuje przewagę w czasie działania nad algorytmem Grahama.

* 1. **Wyniki dla zbioru D generowanego funkcją :**

generate\_square\_points(a=(0,0), b=(10,0), c=(10,10), d=(0,10), axis\_n, diag\_n), axis\_n oraz diag\_n wyliczane są na podstawie kolejnych wartości 25, 50, 75, 100, 200, 300, 400, 500, 1000, w sposób opisany w punkcie 2

**Tabela 4 Zmierzone czasy dla zbioru D**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Liczba wszystkich punktów** | **Liczba punktów na boku** | **Liczba punktów na przekątnej** | **Graham [ns]** | **Jarvis [ns]** | **Różnica czasu [ns] (Graham-Jarvis)** |
| 24 | 3 | 9 | 343405 | 171096 | 172309 |
| 50 | 10 | 15 | 654659 | 373663 | 280996 |
| 74 | 14 | 23 | 1064440 | 455409 | 609031 |
| 100 | 24 | 26 | 1611343 | 1400366 | 210977 |
| 200 | 99 | 1 | 5392997 | 1422163 | 3970834 |
| 300 | 34 | 116 | 5586776 | 1541400 | 4045376 |
| 400 | 58 | 142 | 7905118 | 2059489 | 5845629 |
| 500 | 112 | 138 | 10672557 | 2769613 | 7902944 |
| 1000 | 135 | 365 | 22643433 | 6986349 | 15657084 |

Wykres 10.4 Obrazuje czas działania obu algorytmów dla zbioru D w zależności od liczby punktów

Obraz zawierający tekst, linia, Wykres, diagram

Opis wygenerowany automatycznie

W tym zaś przypadku widzimy znaczną przewagę algorytmu Jarvisa jeśli chodzi o szybkość działania zbiór D, który jest złożony z punktów leżących na dwóch przekątnych oraz na dwóch bokach kwadratu faworyzuje algorytm Jarvisa w swoją budową. Mniej iteracji potrzebne jest, aby odnaleźć otoczkę wypukłą właśnie tym sposobem w tym przypadku. Ciekawym jest też, fakt że oba wykresy przypominają funkcje liniowe.

1. **Wnioski**

**Jak widać na powyższych wykresach 10.1 do 10.4 dla zbioru A oraz B algorytm Graham’a działa szybciej niż algorytm Jarvis’a dzieje się tak zapewne przez specyfikę rozmieszczenia punktów w zbiorze, liczba punktów na otoczce jest całkiem duża zwłaszcza dla zbioru B. Ten fakt powoduje, że k zawarte w szacowaniu złożoności algorytmu Jarvis’a jest rzędu n więc złożoność tego algorytmu staje się zależnością kwadratową od liczby punktów w zbiorze. W dwóch ostatnich przypadkach (zbiór C oraz zbiór D) liczba punktów na otoczce jest niewielka co faworyzuje czasowo algorytm Jarvis’a, na wykresach 10.3 i 10.4, jak i w tabelach 3 i 4 widać wyraźną różnicę. W przypadku zbioru D otrzymane wykresy przypominają wręcz funkcje liniowe, powodem tego może być charakterystyczne rozmieszczenie punktów. Otrzymane wykresy (10.1-10.4) jak i tabele (1-4) potwierdzają założenia teoretyczne co do złożoności oraz czasów działania algorytmów Jarvis’a oraz Graham’a.**