Sprawozdanie z ćwiczenia drugiego

Algorytmy Geometryczne

Iwo Szczepaniak, Windows 11 – 4,2GHz

Jupyter Notebook, VS Code Python 3.9.12

1. **Wstęp**

Celem ćwiczenia jest wyznaczenie otoczki wypukłej(najmniejszego zbioru wypukłego zawierającego podzbiór wszystkich punktów na płaszczyźnie) dla konkretnego zbioru punktów. Do tego celu, pierwszym zadaniem jest wygenerowanie zadanych punktów i ich wizualizacja, a następnie umożliwienie zmiany ilości punktów oraz ich zakresu w zależności od życzenia użytkownika. Operując na tak wyznaczonych punktach, naszym zadaniem jest wyznaczyć otoczkę wypukłą. Do jej generowania służą dwa algorytmy – Grahama oraz Jarvisa.

Złożoność obliczeniowa :

n – szukanie minimum

nlogn – sortowanie

n – przejście po punktach sprawdzając ich położenia względem siebie

**Prognozowana złożoność: słabe nlogn**

1. **Opis algorytmów**

**- Algorytm Grahama:**

1. Wybierz *punkt startowy* o najmniejszej współrzędnej x; jeśli kilka punktów ma tę samą współrzędną y, wybierz spośród nich ten o największej współrzędnej y.
2. Posortuj punkty względem:
   * Orientacji względem *punktu startowego* -> funkcja orient(a,b,c) z poprzedniego laboratorium
   * Gdy „kąt” równy, względem odległości punktu {\displaystyle P\_{i}} od punktu startowego
3. Przeglądaj listę posortowanych punktów poczynając od *punktu startowego*:
   * Od bieżącej pozycji weź trzy kolejne punkty
   * Jeśli punkt trzeci punkt leży po prawej stronie od prostej na której leżą pierwsze dwa punkty, to może należeć do otoczki
   * Jeśli punkt B leży na lewo od prostej na której leżą pierwsze dwa punkty, to znaczy, że nie należy do otoczki. Wtedy usuń ten punkt ze stosu i cofnij się o jedną pozycję.
   * Powtarzaj aż dotrzesz z powrotem do punktu startowego

Złożoność obliczeniowa:

nh – n szukań dla h punktów otoczki(pesymistycznie n²)

**Prognozowana złożoność: wyniki pomiędzy n a n²**

**-Algorytm Jarvisa:**

1. Wybierz *punkt startowy* tak samo jak w algorytmie Grahama.
2. Przejrzyj wszystkie wierzchołki i dodaj na stos ten który jest najbardziej na lewo od ostatniego wierzchołka na stosie.
3. Powtarzaj aż dotrzesz z powrotem do punktu startowego.
4. **Generowanie i wizualizacja punktów**

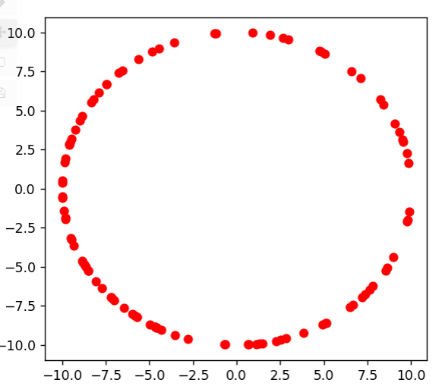
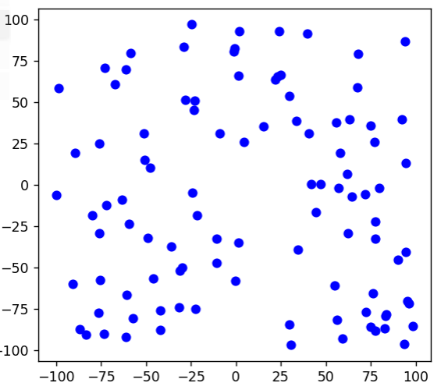
Zadaniem było przygotowanie:

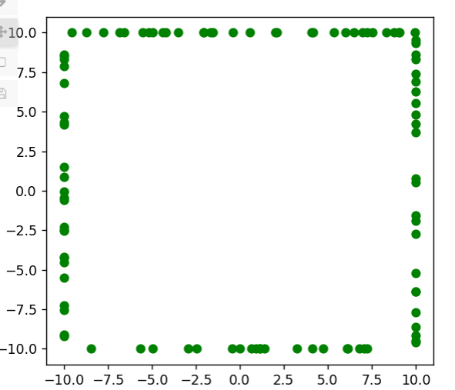
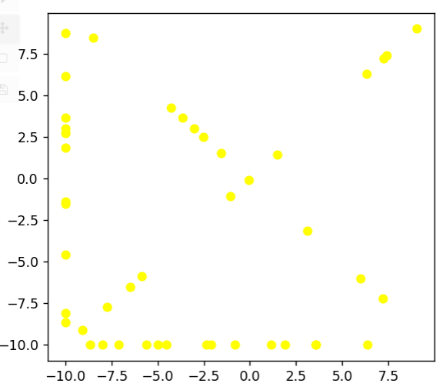
a) zawierający 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100],

b) zawierający 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10,

c) zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10),

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.

****

****

Wizualizacja zadania d)

*Rys. 4*

Wizualizacja zadania c)

*Rys. 3*

*Rys. 2*

*Rys. 1*

Implementacja pozwala też na zmianę ilości generowanych punktów oraz wielkości figur na których są generowane punkty – okręgu, kwadratu i wycinka kwadratu wraz z przekątną– lub zakresu generowanych punktów

Wizualizacja zadania b)

Wizualizacja zadania a)

1. **Wizualizacja działania algorytmów**

Pełna wizualizacja w Jupyter Notebook.

Do wizualizacji działania programu wykorzystano, zgodnie z poleceniem, trzy kolory. Czerwony odpowiada za punkty na otoczce, fioletowy za punkty aktualnie przetwarzane, a niebieski za pozostałe.

Aby zwizualizować konkretny podpunkt należy wprowadzić odpowiadającą mu literę.

Poniżej umieszczono jedynie niektóre interesujące wizualizacje w trakcie działania algorytmów.

**Słowniczek wizualizacji:**

n – liczba punktów

r – promień okręgu

S – środek okręgu

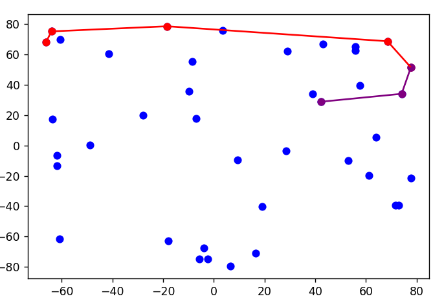
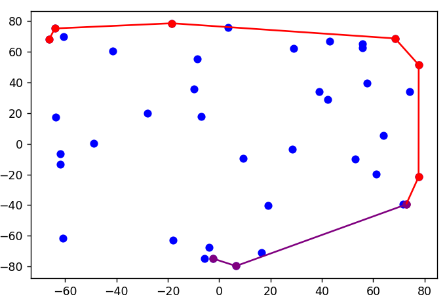
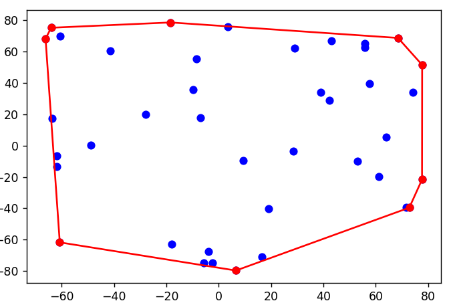
downLeft – lewy dolny róg prostokąta

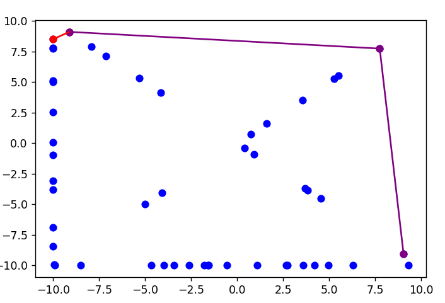
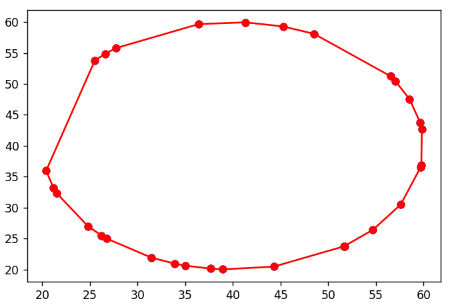
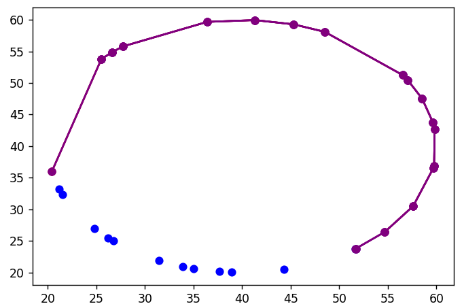
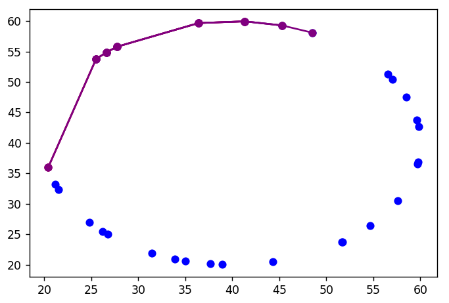
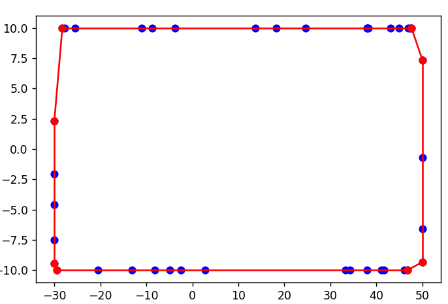
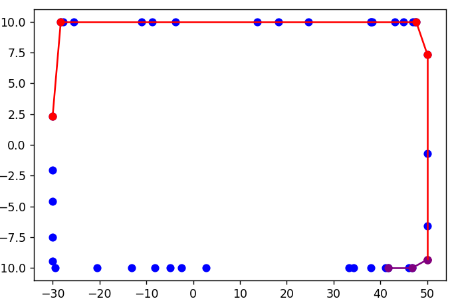
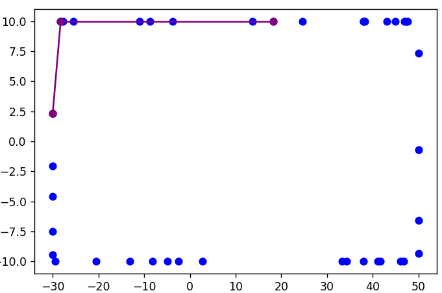
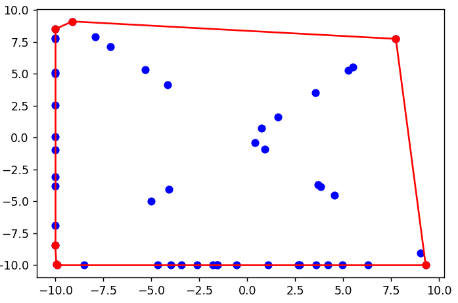
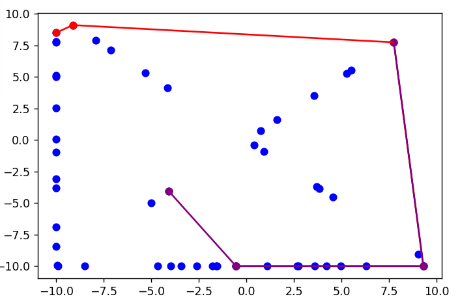
upRight – prawy górny róg prostokąta

a – długość boku w kwadracie w podpunkcie d

m – liczba punktów na przekątnej w podpunkcie d

**- Algorytm Grahama**

****

****

*Rys. 8c*

*Rys. 8b*

*Rys. 8a*

*Rys. 7c*

*Rys. 7a*

*Rys. 7b*

*Rys. 6c*

*Rys. 6b*

*Rys. 6a*

*Rys. 5c*

*Rys. 5b*

*Rys. 5a*

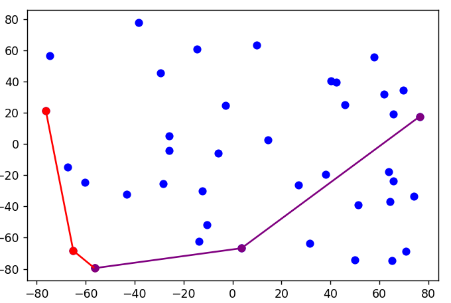
\*Podpunkt d) z parametrami: n = 30, m =20, downLeft = (-10,-10), a = 20

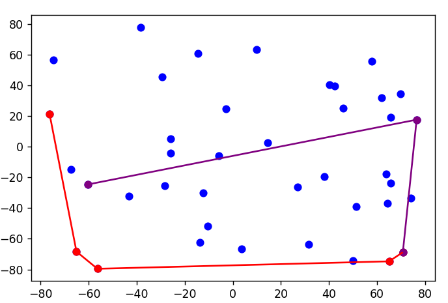
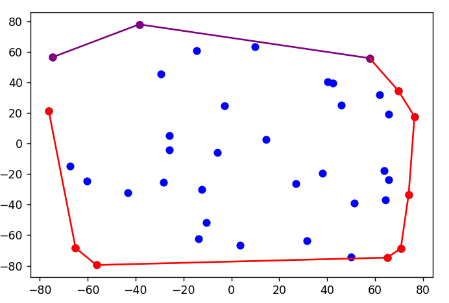
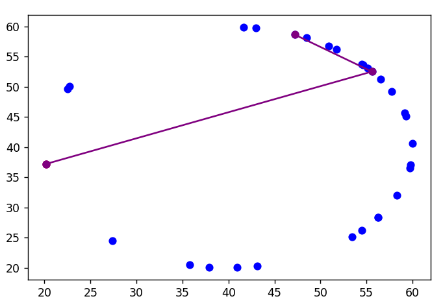
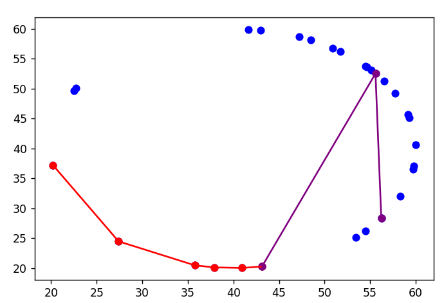
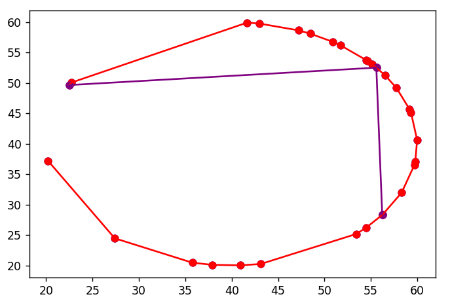
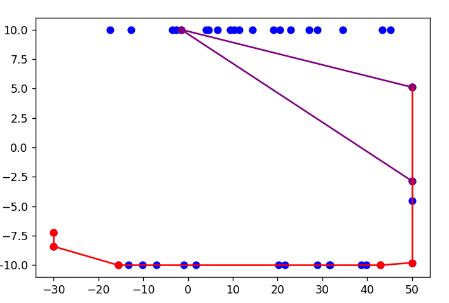
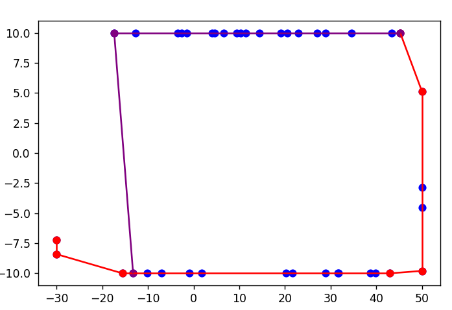
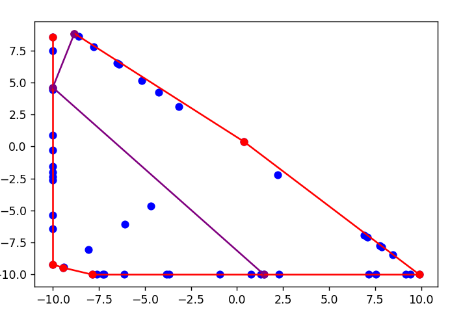
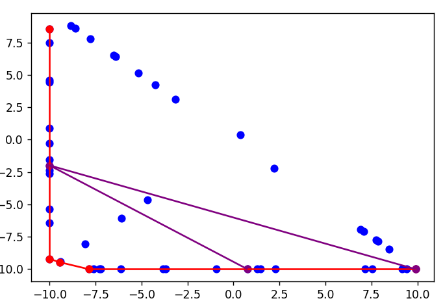
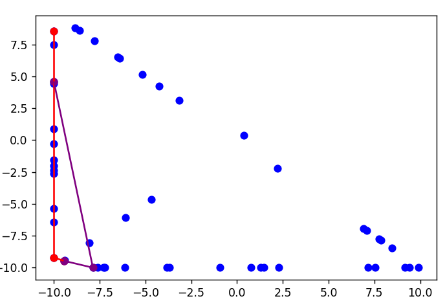
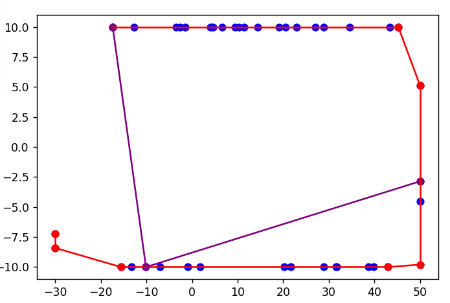
\*Podpunkt c) z parametrami: n = 40, downLeft = (-30,-10), upRight = (50,10)

\*Podpunkt a) z parametrami: n = 40, zakres [-80,80]

\*Podpunkt b) z parametrami: n = 30, r = 20, S = (40,40)

**- Algorytm Jarvisa**

****

****

*Rys. 12a*

*Rys. 12b*

*Rys. 12c*

*Rys. 11c*

*Rys. 11b*

*Rys. 11a*

*Rys. 10a*

*Rys. 10b*

*Rys. 10c*

*Rys. 9a*

*Rys. 9b*

*Rys. 9c*

\*Podpunkt d) z parametrami: n = 30, m =20, downLeft = (-10,-10), a = 20

\*Podpunkt c) z parametrami: n = 40, downLeft = (-30,-10), upRight = (50,10)

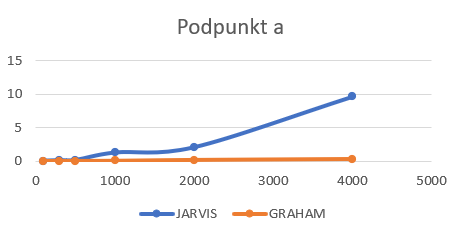
\*Podpunkt a) z parametrami: n = 40, zakres [-80,80]

\*Podpunkt b) z parametrami: n = 30, r = 20, S = (40,40)

1. **Czasochłonność ze względu na liczbę punktów algorytm wraz z wizualizacją:**

Podpunkt a)

|  |  |  |
| --- | --- | --- |
| n | JARVIS [s] | GRAHAM [s] |
| 100 | 0.02352047 | 0.005017996 |
| 300 | 0.098564863 | 0.008991718 |
| 500 | 0.128087521 | 0.018006086 |
| 1000 | 1.25450182 | 0.043254375 |
| 2000 | 2.045077801 | 0.177389622 |
| 4000 | 9.666536808 | 0.33026433 |

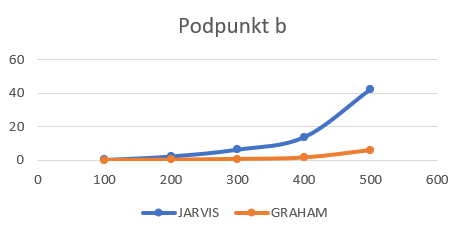
****

*Tab. 1*

*Rys. 13*

Podpunkt b)

|  |  |  |
| --- | --- | --- |
| n | JARVIS [s] | GRAHAM [s] |
| 100 | 0.192240477 | 0.0504632 |
| 200 | 2.231305361 | 0.24768424 |
| 300 | 6.34507966 | 0.719551086 |
| 400 | 13.7860055 | 1.574402332 |
| 500 | 42.40614486 | 6.080311537 |

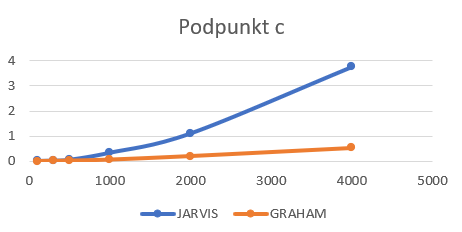
****

*Tab. 2*

*Rys. 14*

Podpunkt c)

|  |  |  |
| --- | --- | --- |
| n | JARVIS [s] | GRAHAM [s] |
| 100 | 0.015676737 | 0.005457401 |
| 300 | 0.031009436 | 0.012002468 |
| 500 | 0.064010859 | 0.02415061 |
| 1000 | 0.344103336 | 0.06397748 |
| 2000 | 1.113039494 | 0.205542088 |
| 4000 | 3.769374609 | 0.531229258 |

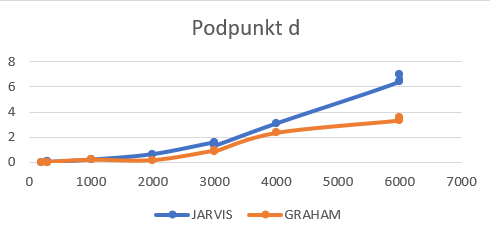
****

*Tab. 3*

*Rys. 15*

Podpunkt d)

|  |  |  |
| --- | --- | --- |
| n+m | JARVIS [s] | GRAHAM [s] |
| 100+100 | 0.013278008 | 0.009525299 |
| 100+200 | 0.025690079 | 0.010002136 |
| 200+100 | 0.024696589 | 0.014503241 |
| 500+500 | 0.203298807 | 0.190287113 |
| 1000+1000 | 0.639288664 | 0.156689405 |
| 1000+2000 | 1.57102704 | 0.922906399 |
| 2000+1000 | 1.308087111 | 0.846460342 |
| 2000+2000 | 3.076281786 | 2.347368002 |
| 4000+2000 | 6.400309324 | 3.344535828 |
| 2000+4000 | 7.627174616 | 3.548235416 |

****

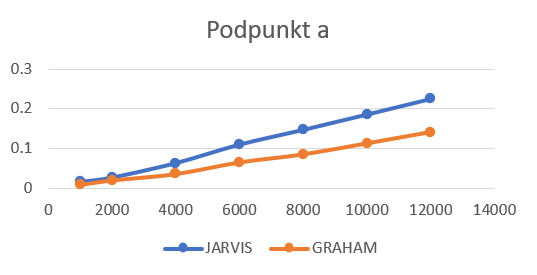
*Tab. 4*

*Rys. 16*

**CZASOCHŁONNOŚĆ – BEZ WIZUALIZACJI**

Podpunkt a)

|  |  |  |
| --- | --- | --- |
| n | JARVIS [s] | GRAHAM [s] |
| 1000 | 0.01548767 | 0.00914884 |
| 2000 | 0.02662659 | 0.02023649 |
| 4000 | 0.06312585 | 0.03620243 |
| 6000 | 0.10997272 | 0.06528592 |
| 8000 | 0.14719725 | 0.08533502 |
| 10000 | 0.18608737 | 0.11293578 |
| 12000 | 0.22539473 | 0.14110279 |

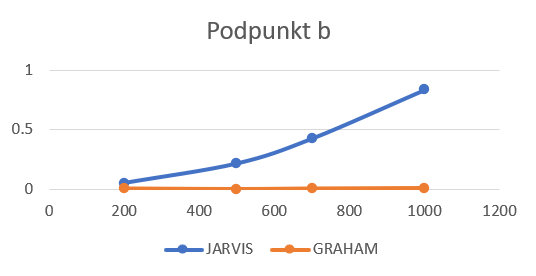
****

*Tab. 5*

*Rys. 17*

Podpunkt b)

|  |  |  |
| --- | --- | --- |
| n | JARVIS [s] | GRAHAM [s] |
| 200 | 0.04965997 | 0.00427485 |
| 500 | 0.21457386 | 0 |
| 700 | 0.42411566 | 0.00405645 |
| 1000 | 0.83552051 | 0.0080204 |

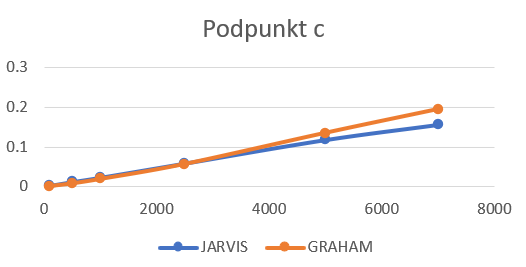
****

*Tab. 6*

*Rys. 18*

Podpunkt c)

|  |  |  |
| --- | --- | --- |
| n | JARVIS [s] | GRAHAM [s] |
| 100 | 0.00343275 | 0.0022285 |
| 500 | 0.01285267 | 0.00894427 |
| 1000 | 0.02377415 | 0.02002335 |
| 2500 | 0.05917048 | 0.05750895 |
| 5000 | 0.11804914 | 0.13523197 |
| 7000 | 0.15590477 | 0.19451022 |
| 10000 | 0.23317289 | 0.28978062 |

****

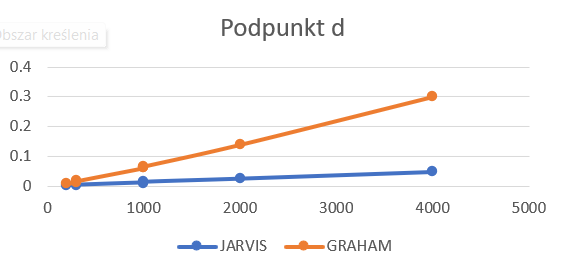
*Tab. 7*

*Rys. 19*

Podpunkt d)

|  |  |  |
| --- | --- | --- |
| d | JARVIS [s] | GRAHAM [s] |
| 100+100 | 0.00305986 | 0.00857973 |
| 100+200 | 0.00300097 | 0.01376009 |
| 200+100 | 0.00402522 | 0.01728296 |
| 500+500 | 0.01195645 | 0.06151152 |
| 300+700 | 0.00840282 | 0.06327271 |
| 700+300 | 0.01424789 | 0.06680799 |
| 1000+1000 | 0.0254457 | 0.13985467 |
| 2000+2000 | 0.04910636 | 0.30078149 |

*Tab. 8*

****

*Rys. 20*

**Wnioski**

Na podstawie otrzymanych wykresów można stwierdzić, że czas wykonywania był zgodny z przewidywaniami – Graham około nlogn, choć często wygląda na algorytm liniowy z dużym współczynnikiem a. Jarvis natomiast okazał się, jak podejrzewano, znacząco zależny od ilości punktów w otoczce – od złożoności n w podpunkcie d aż do n² w podpunkcie b.

**Wnioski dla testów z wizualizacją:**

W podpunktach a i c widać znaczącą różnicę między dwoma podejściami, ale w podpunkcie b różnica jest największa. Różnica w czasie wykonywania algorytmów wynosząca 35 s pojawia się już przy generowaniu 500 punktów. Jest to prawdopodobnie spowodowane tym, że w tym podpunkcie wszystkie punkty wygenerowane należą do otoczki(zbiorem punktów jest okrąg) więc współczynnik h jest równy n. W podpunkcie d różnica między algorytmami się znacznie zaciera i to z kolei jest to spowodowane stosunkowo małą ilością punktów wchodzących w skład otoczki na zadanej figurze, co zmniejsza współczynnik h. Mimo to algorytm Grahama nadal był zauważalnie wydajniejszy w tym podpunkcie.

Dla wszystkich zbiorów punktów algorytm Grahama okazał się szybszy od algorytmu Jarvisa.

**Wnioski dla testów bez wizualizacji:**

W podpunkcie a oba algorytmy wykazują niemal liniowy wzrost i różnica między nimi jest dość znikoma, chociaż rośnie stopniowo im więcej punktów - na korzyść algorytmu Grahama(z przewagi 0,006s dla 2000 punktów do 0,1s dla 12000 punktów).

W podpunkcie b przewaga algorytmu Grahama umacnia się znacznie szybciej – już przy 1000 punktów różnica to 1 sekunda. Jest to efektem pesymistycznej złożoności n² algorytmu Jarvisa - na okręgu wszystkie punkty należą do otoczki.

Podpunkt c to podpunkt w którym czasy ponownie wydają się rosnąć liniowo, jednak współczynnik prostej dla algorytmu Grahama jest mniej korzystny i dla około 2500 punktów zachodzi przełamanie, powyżej którego to algorytm Jarvisa staje się skuteczniejszy(to oznacza, że od tego punktu logn > k).

Podpunkt d to zupełne odwrócenie sytuacji z punktu b. Tym razem to algorym Jarvisa ma znaczną przewagę, ponieważ za każdym razem ilość punktów które tworzą obwódkę jest równa 4, więc algorytm rośnie liniowo i to z bardzo małym współczynnikiem. Dla 4000 punktów przewaga Jarvisa to około 0,25 sekundy.

**Podsumowanie:**

Uwzględnienie wizualizacji znacznie zakłamało wyniki na niekorzyść algorytmu Jarvisa – duża ilość dodawanych scen bardzo przeszkadzała w ukazaniu korzyści tego podejścia w podpunktach c i d. Dodatkowo pierwotnie w algorytmie generującym punkty w podpunkcie d brakowało wierzchołkowych punktów, co dodatkowo działało na niekorzyść tego algorytmu.

Analiza testów bez wizualizacji jasno pokazuje, że algorytmy te wykazują podobną złożoność dla większości zbiorów punktów, choć pewne skrajne przypadki takie jak generowanie punktów na okręgu czy gwarancja wystąpienia wierzchołków, znacznie przechyla korzyść z zastosowania na jeden z algorytmów - odpowiednio Grahama i Jarvisa.