**Algorytmy geometryczne, otoczka wypukła, ćwiczenie 1**

**Błażej Kapkowski, 12.11.2023r.**

1. **Dane techniczne:**

Język: Python

Translator: Visual Studio Code

Procesor: AMD Ryzen 7 5800H

System operacyjny: Windows 11

1. **Realizacja ćwiczenia:**

Otoczka wypukła to najmniejszy wielokąt wypukły, który obejmuje wszystkie punkty ze zbioru danych. Innymi słowy, jest to taki wielokąt wypukły, który zawiera wszystkie punkty wejściowe i nie ma w sobie żadnych wklęsłości. Otoczka wypukła jest istotna w geometrii obliczeniowej i znajduje zastosowanie w wielu dziedzinach, takich jak grafika komputerowa, przetwarzanie obrazów, czy algorytmy planowania trasy.

Istnieje kilka algorytmów służących do znajdowania otoczki wypukłej. Dwa z nich to algorytmy Grahama i Jarvisa. Oto krótkie wyjaśnienie działania każdego z nich:

**Algorytm Grahama:**

Sortowanie punktów:

Wybieramy punkt o najniższej wartości y (i minimalnej wartości x w przypadku punktów o tej samej wartości y) jako punkt początkowy (punkt bazowy).

Sortujemy pozostałe punkty według kąta, jaki tworzą z punktem bazowym.

Tworzenie otoczki:

Zaczynamy od punktu bazowego i bierzemy po kolei posortowane punkty.

Dla każdego punktu sprawdzamy, czy dodanie go do otoczki nie powoduje wklęsłości (tj. czy punkt ten jest po lewej stronie odcinka tworzonego przez poprzednie punkty otoczki).

Jeśli tak, to dodajemy punkt do otoczki.

Ostatecznie otrzymujemy otoczkę wypukłą zawierającą wszystkie punkty.

**Algorytm Jarvisa:**

Znalezienie punktu początkowego:

Znajdujemy punkt o najniższej wartości y (i minimalnej wartości x w przypadku punktów o tej samej wartości y) jako punkt początkowy (punkt bazowy).

Tworzenie otoczki:

Rozpoczynamy od punktu początkowego i znajdujemy kolejny punkt otoczki, który ma największy kąt pomiędzy odcinkiem łączącym bieżący punkt a wszystkimi innymi punktami.

Dodajemy ten punkt do otoczki.

Proces powtarzamy, aż wrócimy do punktu początkowego.

**Zbiory generowanych punktów:**

Zbiór a

Zbiór a jest to 100 losowych punktów, których współrzędna x ∈ [−100, 100] oraz y ∈ [−100, 100].

Zbiór b

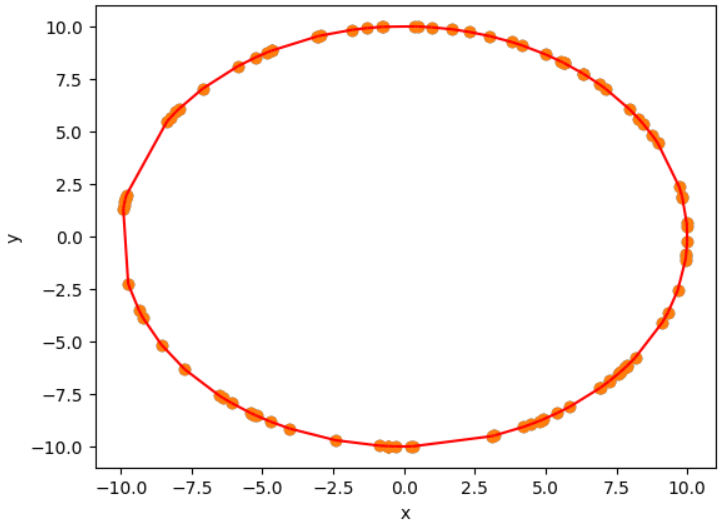
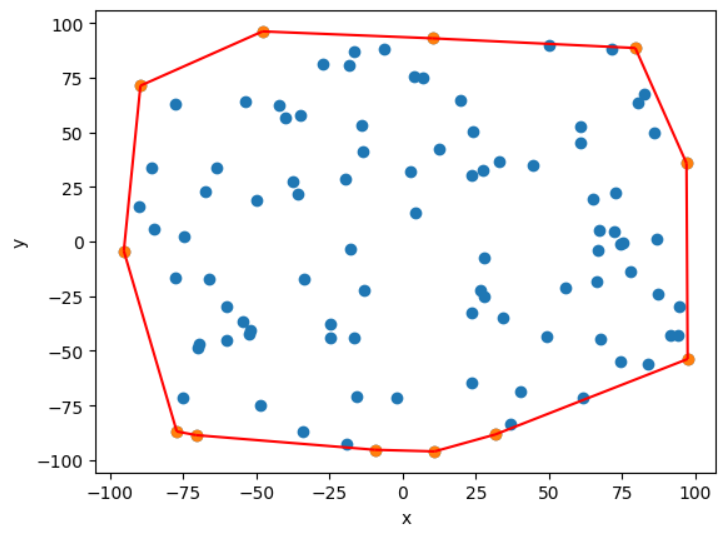
Zbiór b to 100 losowych punktów leżących na okręgu o środku O = (0, 0) oraz promieniu R = 10.

Zbiór c

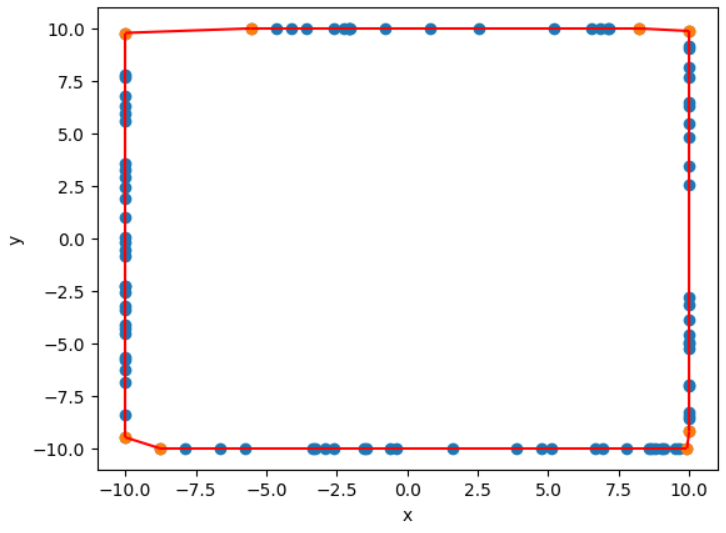
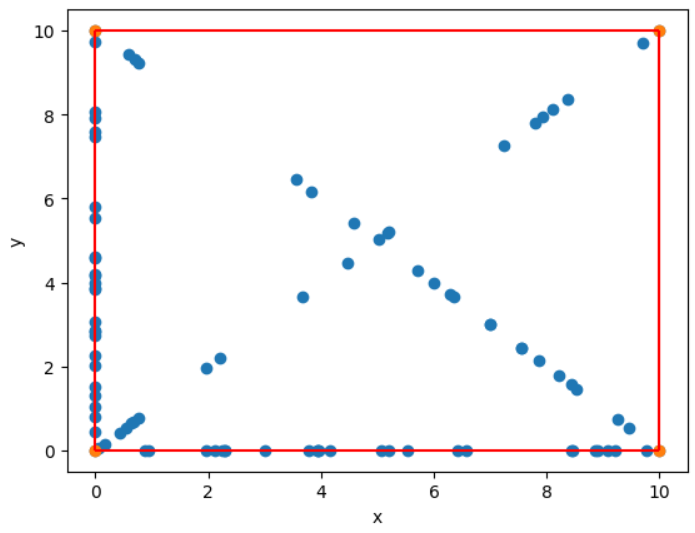
Zbiór punktów c to 100 losowych punktów leżących na obwodzie prostokąta A = (−10, 0), B = (10, 0), C = (10, 10), D = (−10 , 10).

Zbiór d

Zbiór punktów d to 90 losowych punktów leżących na kwadracie, z czego po 25 pkt leży na osiach X i Y oraz po 20 pkt leży na przekątnych kwadratu A = (0, 0), B = (10, 0), C = (10, 10), D = (0 , 10), dodatkowo 4 punkty A, B, C, D także należą do zbioru.



Rysunek 1: Otoczka zbioru punktów a Rysunek 2: Otoczka zbioru punktów b

Rysunek 3: Otoczka zbioru punktów c Rysunek 4: Otoczka zbioru punktów d

1. **Analiza wyników:**

Podczas testów analizujemy zbiory, które zostały wcześniej wspomniane, uwzględniając ich modyfikowaną liczbę elementów na , , .

**Zbiór a**

|  |  |  |
| --- | --- | --- |
| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa |
| 1000 | 0.0257 s | 0.0366 s |
| 10000 | 0.3909 s | 0.6959 s |
| 100000 | 5.1928 s | 7.9740 s |

Tabela 1: Porównanie czasu działania algorytmów dla różnej ilości punktów w zbiorze a

W tabeli 1 można zauważyć że dla ogólnego przypadku oba algorytmy działają w podobnym czasie jednak algorytm Grahama jest trochę szybszy.

**Zbiór b**

|  |  |  |
| --- | --- | --- |
| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa |
| 1000 | 0.0310 s | 2.7911 s |
| 10000 | 0.4874 s | 250.5212 s |

Tabela 2: Porównanie czasu działania algorytmów dla różnej ilości punktów w zbiorze b

W tym przypadku ograniczono się do 10000 punktów, ponieważ algorytm Jarvisa potrzebował zbyt dużo czasu dla większego rozmiaru danych. Wynika to z tego że złożoność algorytmu Jarvisa jest związana z ilością punktów na otoczce, a w przypadku okręgu wszystkie punkty należą do otoczki.

W takich przypadkach algorytm Grahama jest znacznie lepszy od algorytmu Jarvisa.

**Zbiór c**

|  |  |  |
| --- | --- | --- |
| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa |
| 1000 | 0.0446 s | 0.0267 s |
| 10000 | 0.6879 s | 0.3432 s |
| 100000 | 8.8856 s | 3.2027 s |

Tabela 3: Porównanie czasu działania algorytmów dla różnej ilości punktów w zbiorze c

W tym przypadku w tabeli 3 możemy zauważyć że algorytm Jarvisa jest szybszy niż algorytm Grahama, jest to związane z tym, że otoczka będzie posiadać mało punktów i punkty są ustawione w liniach prostych.

**Zbiór d**

|  |  |  |
| --- | --- | --- |
| Liczba punktów | Algorytm Grahama | Algorytm Jarvisa |
| 1000 | 0.0862 s | 0.0235 s |
| 10000 | 1.0758 s | 0.1671 s |
| 100000 | 13.8374 s | 1.5858 s |

Tabela 4: Porównanie czasu działania algorytmów dla różnej ilości punktów w zbiorze c

W tym przypadku w tabeli 3 możemy zauważyć że algorytm Jarvisa jest znacznie szybszy niż Grahama, ponieważ do otoczki zaliczają się tylko 4 wierzchołki kwadratu i prędkość algorytmu Jarvisa jest ograniczona tylko przez szybkość sortowania punktów.

W przypadku zbiorów b i c ważna jest także aproksymacja zera, ponieważ musimy odpowiednio wykrywać punkty leżące na jednej prostej.

|  |  |  |
| --- | --- | --- |
| Wartość epsilon | Liczba pkt otoczki zbioru b | Liczba pkt otoczki zbioru c |
|  | 83996 | 8 |
|  | 94405 | 8 |
|  | 98429 | 1330 |

Tabela 5: Porównanie ilości pkt otoczki dla różnych wartości epsilon dla 100000 pkt

W tabeli 5 możemy zauważyć, że aproksymując 0 nie jesteśmy wstanie wykryć wszystkich punktów otoczki dla zbioru b ponieważ niektóre punkty zaliczane są do tej samej prostej. Próbując zmniejszyć wartość epsilon napotykamy na problem w zbiorze c, gdzie przez małą wartość epsilon punkty które leżą na jednej prostej nie są do niej zaliczane. Rozwiązanie może być takie żeby wybierać różne wartości epsilon dla różnych zbiorów.

1. **Wnioski:**

Wyniki przeprowadzonego eksperymentu sugerują, że algorytm Grahama charakteryzuje się wyższą szybkością dla losowych zestawów punktów. Zauważamy, że algorytm Jarvisa osiąga najlepsze rezultaty w sytuacjach, gdzie duża liczba punktów nie należy do otoczki. W przypadku, gdy sytuacja jest odwrotna, algorytm Jarvisa osiąga niższą efektywność. Rozpatrując rezultaty algorytmu Jarvisa, można zauważyć, że dla zbiorów c i d jego złożoność jest proporcjonalna do prędkości sortowania punktów. W przypadku zbioru a tempo wzrostu jest nieznacznie większe, natomiast dla zbioru b algorytm wykazuje złożoność kwadratową. Ogólnie można powiedzieć, że lepiej stosować algorytm Grahama ponieważ w większości przypadków będzie sprawniejszy. Algorytm Jarvisa należy stosować tylko do specyficznych danych dla których wiemy, że poradzi sobie szybciej niż algorytm Grahama. Ważne jest też, żeby pamiętać o odpowiedniej wartości epsilon, która może wpłynąć na ilość punktów otoczki.