Akademia Górniczo-Hutnicza
Wydział Informatyki, Elektroniki i Telekomunikacji
Kierunek Informatyka

Algorytmy geometryczne

Laboratorium 2

Otoczka wypukła

Kyrylo Iakymenko

Kraków, 7 listopada 2023

1 Wprowadzenie

Otoczka wypukła stanowi fundamentalne pojęcie w dziedzinie algorytmów geometrycznych oraz obliczeń geometrycznych. Jest to zbiór punktów, które tworzą najmniejszy wielokąt wypukły, który obejmuje wszystkie punkty z danego zbioru. W praktyce znajduje ona zastosowanie w wielu dziedzinach, takich jak grafika komputerowa, przetwarzanie obrazów, planowanie trasy, a nawet w optymalizacji geometrii konstrukcji.

W niniejszym sprawozdaniu skupimy się na omówieniu dwóch popularnych algorytmów służących do wyznaczania otoczki wypukłej: algorytmu Grahama oraz algorytmu Jarvisa. Obie metody pozwalają na efektywne rozwiązanie tego problemu i zostały szeroko wykorzystane w praktyce.

2 Opis wykorzystanych algorytmów

2.1 Algorytm Grahama

Algorytm Grahama to inny popularny algorytm do wyznaczania otoczki wypukłej dla zbioru punktów w przestrzeni dwuwymiarowej. Ten algorytm jest bardziej efektywny niż algorytm Jarvisa i ma złożoność czasową wynoszącą O(n*log(n)), gdzie "n"to liczba punktów w zbiorze. Algorytm Grahama opiera się na wykorzystaniu sortowania punktów względem ich kąta względem punktu startowego.

Poniżej przedstawiam kroki algorytmu Grahama:

Wybierz punkt startowy: Wybieramy punkt startowy, który będzie częścią otoczki wypukłej. Podobnie jak w algorytmie Jarvisa, punkt ten może być dowolnym punktem z dostępnego zbioru punktów. W praktyce często wybiera się punkt o najniższej wartości współrzędnej y (i ewentualnie najniższej wartości x jako tie-breaker).

Sortowanie punktów: Sortujemy pozostałe punkty ze zbioru według kąta, jaki tworzą względem punktu startowego. Dzięki temu punkty są rozmieszczone w kolejności od najmniejszego kąta do największego kąta. W przypadku, gdy wiele punktów ma ten sam kąt, sortowane są względem odległości od punktu startowego (wzrostowo).

Główna pętla: Algorytm przetwarza punkty w kolejności rosnących kątów. Początkowo dodajemy punkt startowy do otoczki wypukłej.

Sprawdź kąt: Dla każdego punktu, który przetwarzamy, sprawdzamy, czy tworzy on odcinek wypukły w stosunku do ostatnich dwóch punktów na otoczce. Jeśli tak, dodajemy ten punkt do otoczki wypukłej.

Aktualizacja otoczki: Jeśli punkt nie tworzy odcinka wypukłego, to usuwamy ostatni punkt otoczki i sprawdzamy punkt obecny ponownie w kontekście poprzedniego punktu na otoczce.

Zakończenie: Algorytm kończy działanie, gdy wszystkie punkty zostały przetworzone, a otoczka wypukła została zakończona.

Algorytm Grahama jest bardziej efektywny niż algorytm Jarvisa, szczególnie dla większych zbiorów punktów, dzięki zastosowaniu sortowania. Jest szeroko stosowany w praktyce do wyznaczania otoczki wypukłej i znajduje zastosowanie w grafice komputerowej, przetwarzaniu obrazów, a także w problemach planowania tras i analizy danych przestrzennych.

2.2 Algorytm Jarvisa

Algorytm Jarvisa, znany również jako algorytm "Zawijania prezentu"(ang. "Gift Wrapping"), to jedna z popularnych metod wyznaczania otoczki wypukłej dla zbioru punktów w przestrzeni dwuwymiarowej. Ten algorytm jest stosunkowo prosty w implementacji, choć ma złożoność czasową wynoszącą O(nh), gdzie "n"to liczba punktów w zbiorze, a "h"to liczba punktów na otoczce wypukłej.

Poniżej przedstawiam kroki algorytmu Jarvisa:

Wybierz punkt startowy: Na początek należy wybrać punkt startowy, który będzie częścią otoczki wypukłej. To może być dowolny punkt z dostępnego zbioru punktów, ale

często wybiera się punkt o najniższej wartości współrzędnej y (jeśli istnieje więcej niż jeden taki punkt, wybiera się ten z najniższą wartością współrzędnej x).

Inicjalizacja otoczki wypukłej: Ustalamy punkt startowy jako pierwszy punkt otoczki wypukłej.

Znajdź kolejny punkt: W tej fazie algorytmu próbujemy znaleźć punkt z otoczki, który znajduje się na prawo od odcinka łączącego bieżący punkt otoczki z poprzednim punktem otoczki. To oznacza, że nowy punkt musi tworzyć odcinek wypukły w stosunku do bieżącej otoczki.

Główna pętla: Algorytm działa w pętli, w której znajduje się następny punkt otoczki, który spełnia kryterium odcinka wypukłego, aż wróci do punktu początkowego, co oznacza, że otoczka została zamknięta.

Aktualizacja otoczki: Po znalezieniu kolejnego punktu, dodajemy go do otoczki wypukłej.

Zakończenie: Algorytm kończy działanie, gdy wróci do punktu początkowego, tworząc zamkniętą otoczkę wypukłą.

Algorytm Jarvisa jest stosunkowo prosty, ale jego głównym ograniczeniem jest jego wysoka złożoność czasowa, co sprawia, że nie jest najbardziej wydajnym algorytmem w przypadku dużych zbiorów punktów. W praktyce bardziej efektywne są bardziej zaawansowane algorytmy, takie jak algorytm Grahama lub Quickhull. Jednak algorytm Jarvisa nadal znajduje zastosowanie w edukacji oraz w sytuacjach, gdy prostota implementacji jest ważniejsza niż wydajność.

3 Generowanie zbiorów punktów

3.1 Opis zbiorów testowych

Na potrzeby pomiarów wydajności obu algorytmów zmodyfikujemy nasze zbiory testowe opisane wyżej w następujący sposób.

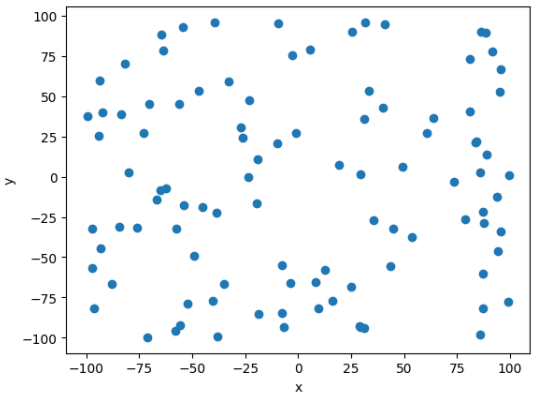
a) Losowe punkty (x, y) w przestrzeni $\mathbb{R}^2$.

b) Losowe punkty (x, y) w przestrzeni $\mathbb{R}^2$, położone na okręgu.

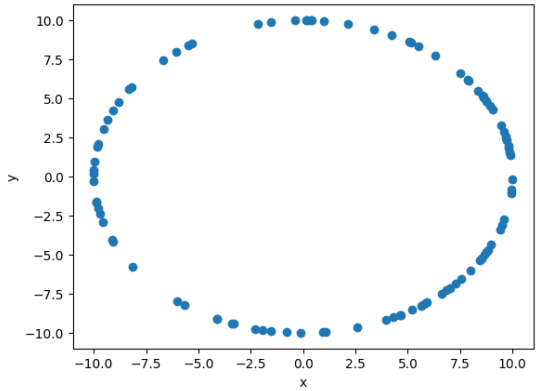
c) Losowe punkty w przestrzeni $\mathbb{R}^2$, położone na prostokącie.

d) Losowe punkty w przestrzeni $\mathbb{R}^2$, położone na dolnym i lewym bokack kwadratu i jego przekątnych.

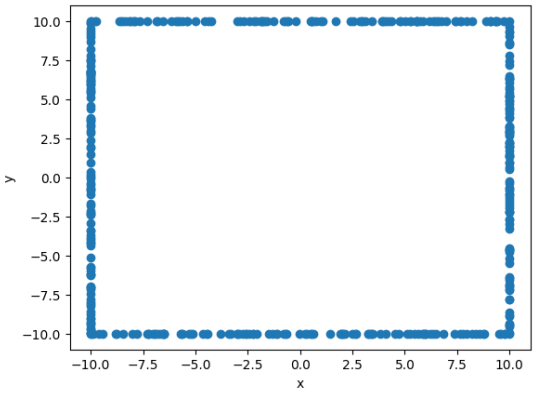
3.2 Wykresy zbiorów



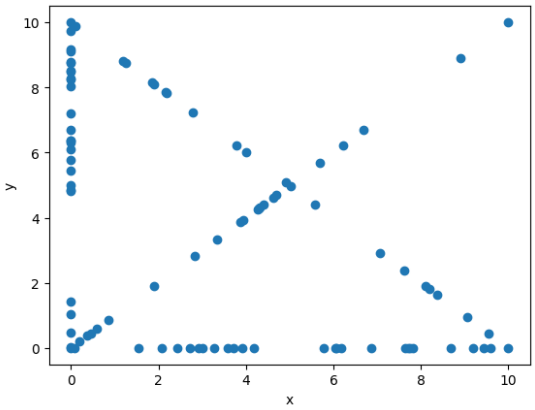
Rys. 1: Zbiór *a*.



Rys. 2: Zbiór *b*.



Rys. 3: Zbiór c.



Rys. 4: Zbiór d.

3.3 Algorytmy generacji zbiorów

1. Dla zbioru a. Osobna generacja każdego z lososwych punktów.
2. Dla zbioru b. Parametryzacja punktów na okręgu za pomocją funkcji trygonometrycznych sin i cos.
3. Dla zbioru c. Osobna generacja każdego z lososwych punktów leżących na bokach prostokąta.
4. Dla zbioru d. Generacja punktów leżących na bokach kwadratu w podobny sposób do zbioru c, dla przekątnych - generacja parametryzacja jednej ze wspólrzędnych za pomocą drugiej.

4 Wydajność algorytmów

4.1 Opis zbiorów testowych

Na potrzeby pomiarów wydajności obu algorytmów zmodyfikujemy nasze zbiory testowe opisane wyżej w następujący sposób.

a) Losowe punkty (x, y) w przestrzeni $\mathbb{R}^2$, gdzie $(x, y) \in \left[-10^4, 10^4\right]^2$.

b) Losowe punkty (x, y) w przestrzeni $\mathbb{R}^2$, położone na okręgu o promieniu $R = 10^4$.

c) Losowe punkty w przestrzeni $\mathbb{R}^2$, położone na prostokącie o wierzcholku w $(-10^4, -10^4)$ i $(10^4, 10^4)$.

d) Losowe punkty w przestrzeni $\mathbb{R}^2$, położone na dolnym i lewym bokack kwadratu o wierzcholku w $(-10^4, -10^4)$ i $(10^4, 10^4)$ i jego przekątnych.

4.2 Tabele

Czasy wykonania algorytmów na zbiorach testowych w zależności od ilości punktów podane w sekundach $[s]$.

	Liczba punktów			
Algorytm	10^3	10^4	10^5	10^6
Graham	0.007	0.088	1.222	15.013
Jarvis	0.010	0.107	1.771	25.266

Tabela 1: Czasy wykonania algorytmów na zbiorze testowym a.

	Liczba punktów		
Algorytm	100	10^3	10^4
Graham	0.001	0.011	0.130
Jarvis	0.008	0.678	70.709

Tabela 2: Czasy wykonania algorytmów na zbiorze testowym *b*.

	Liczba punktów				
Algorytm	100	10^3	10^4	10^5	10^6
Graham	0.005	0.064	0.739	9.177	113.494
Jarvis	0.001	0.009	0.115	1.242	13.464

Tabela 3: Czasy wykonania algorytmów na zbiorze testowym *c*.

	Liczba punktów				
Algorytm	100	10^3	10^4	10^5	10^6
Graham	0.002	0.027	0.212	2.816	37.919
Jarvis	0.001	0.011	0.077	0.921	9.709

Tabela 4: Czasy wykonania algorytmów na zbiorze testowym *d*.

5 Omówienie otrzymanych wyników

Krótko omówimy dlaczego zostały wybrane takie zbiory do testowania algorytmów oraz jakie problemy były tym spowodowane.

Zbiór *a* to po prostu zbiór losowych, więc w naszy przypadku jest przydatny, jako weryfikujący działanie algorytmów w przypadku najbardziej ogólnym. Zbiór *b* (okrąg) jest zbiorem, który stwarza najwięcej problemów dla algorytmu Jarvisa (dokładniej omówione poniżej) i potrzebny dla demonstracji różnicy pomiędzy dwoma algorytmami w przypadku skrajnym. Zbiory *c* i *d* charakteryzują się małą otoczką wypukłą oraz dużą ilością punktów współliniowych. To jest dobrą demostracją prtzypadku, dla którego algorytm Jarvisa jest bardziej wydajny, oraz testuje naszą możliwość wyznaczania punktów współliniowych w algorytmie Grahama.

Omówimy teraz otrzymane wyniki czasów wykonania obu algorytmów dla zbiorów testowych.

a) Dla zbiorów lososwych w których ilość punktów była mniejsza od 10^4 algorytm Grahama i Jarvisa mają prawie identyczne czasy wykonania. Dla zbiorów o większej mocy Graham mający mniejszą złożoność, jest zdecydowanie szybszy, co nie powinno nas dziwić.

b) Dla punktów na okręgu widzimy jeszcze wiekszą przewagę algorytmu Grahama niż z poprzednim zbiorem. Spowodowane to jest tym, że w okręgu każdy z punktów należy do otoczki wypukłej, a złożoność Jarvisa $O(nh)$, zależy od ilości punktów w otoczce h. Dlatego w pewnym sensie to jest najgorszy przypadek dla algorytmu Jarvisa i jego złożoność staje się $O(n^2)$ w porównaniu do złożoności Grahama $O(nlog(n))$.

c) Sytuację odwrotną widzimy dla prostokąta. Jego otoczka składa się tylko z 4 punktów (wierzchołków). i w tym przypadku Jarvis faktycznie ma złożoność liniową, a

Graham nadal pozostaje ze złożonością liniowo-logarytmiczną, co wyraźnie wydać po wynikach.

d) Dla kwadratu i jego przekątnych mamy podobną sytuację, jak i w przypadku z prostokątem. Otoczka wypukła tego zbioru punktów składa się z małej ilości punktów w porównaniu do mocy całego zbioru. To powoduje, że Jarvis jest znacznie bardziej wydajny i llepiej radzi sobie z problemem. Graham zaczynając od lewego dolnego wierzchołka p_0 sortuje wszystkie punkty względem kątów nachylenia prostych do nich prowadzących od p_0, to też może powodować niektóre problemy, gdzyż w tym przypadku, jak i w zbiorze c, mamy bardzo dużo punktów współliniowych i ich poprawne posortowanie dla zbiorów na dużych zakresach nie jest takie łatwe ze względu na możliwe błędy spowodowane precyzją obliczeń arytmetycznych.

6 Podsumowanie

W trakcie prowadzenia tego badania przeprowadzona została analiza i porównanie dwóch popularnych algorytmów służących do wyznaczania otoczki wypukłej, tj. algorytmów Grahama i Jarvisa. Celem było zrozumienie działania tych algorytmów oraz porównanie ich wydajności przy użyciu różnych zbiorów punktów. Poniżej przedstawione są główne wnioski z badań:

Algorytm Grahama:

Algorytm Grahama charakteryzuje się złożonością czasową wynoszącą $O(nlog(n))$, co sprawia, że jest bardziej wydajny niż algorytm Jarvisa dla większości zbiorów losowych, azckolwiek są wyjątki co widać po zbiorach c i d.

Algorytm Jarvisa:

Algorytm Jarvisa jest prosty w implementacji, ale jego złożoność czasowa wynosi $O(nh)$, gdzie n to liczba punktów, a h to liczba punktów na otoczce wypukłej. Jego wydajność maleje znacząco w przypadku zbiorów z otoczką wypukłą o dużym rozmiarze, co sprawia, że może być mniej praktyczny w niektórych sytuacjach. Porównanie:

W naszych badaniach zaobserwowaliśmy, że algorytm Grahama osiąga lepsze wyniki wydajnościowe niż algorytm Jarvisa, zwłaszcza w przypadku dużych zbiorów punktów. Algorytm Grahama jest bardziej efektywny dzięki zastosowaniu sortowania punktów względem kąta względem punktu startowego, co pozwala na znaczne przyspieszenie procesu wyznaczania otoczki wypukłej. Podsumowując, zarówno algorytm Grahama, jak i algorytm Jarvisa znajdują zastosowanie w wyznaczaniu otoczki wypukłej, ale wybór między nimi zależy od konkretnych potrzeb i rozmiaru zbioru punktów. Algorytm Grahama jest bardziej efektywny i zwykle bardziej polecany w praktyce, zwłaszcza w przypadku dużych zbiorów punktów, gdzie złożoność czasowa odgrywa kluczową rolę. Jednak algorytm Jarvisa pozostaje przydatnym narzędziem, zwłaszcza w celach edukacyjnych i w przypadku mniejszych zbiorów punktów, gdzie prostota implementacji może mieć znaczenie.