Szymon Nowak-Trzos 411246

Informatyka rok II, gr.2 środa nieparzysta 13:00-14:30

Algorytmy Geometryczne – Ćwiczenie 2 Sprawozdanie

1. Wstęp
   1. Cel Ćwiczenia

Celem ćwiczenia była implementacja dwóch algorytmów generujących otoczkę wypukłą oraz ich analiza na podstawie zbiorów danych podanych w treści zadania

* 1. Użyte Algorytmy

Do generacji otoczek wypukłych użyto dwóch algorytmów:

­- Algorytm Jarvisa

- Algorytm Grahama

* 1. Użyte Narzędzia

Ćwiczenie zostało wykonane dzięki zmodyfikowanemu narzędziu graficznemu załączonego na stronie UPEL. Do niezbędnych obliczeń został wykorzystany język programowania Python oraz biblioteka numpy. Ponadto wykresy zostały wygenerowano przy pomocy biblioteki matplotlib, a czas został wyznaczony poprzez bibliotekę time. Do wygenerowania gifów użyto bibiloteki Pillow. Obliczenia, których wyniki zostały uwzględnione w tabelach, oraz wszystkie wykresy zostały wygenerowane na procesorze Intel Core i5-1135G7 2.40GHz.

1. Szczegóły wykonywania ćwiczenia
   1. Zbiory Danych

Wygenerowane zostały cztery zbiory danych:

1. 100 losowych punktów o współrzędnych z przedziału [-100, 100]
2. 100 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
3. 100 losowych punktów leżących na bokach prostokąta o wierzchołkach

(-10, 10), (-10, -10), (10, -10), (10, 10),

1. Zbiór zawierający: 4 wierzchołki kwadratu o lewym dolnym rogu w (-10, 10) i o boku a=10,

Po 25 punktów na lewym i dolnym boku,

Po 20 punktów leżących na przekątnych kwadratu

Do wygenerowania zbiorów wykorzystana została funkcja random.uniform z biblioteki random języka Python. Zbiór B uzyskany został przez parametryzację okręgu o zadanym środku. W kodzie znajduje się funkcja ‘generate\_datasets’, dzięki której można wygenerować zbiory danych o innych parametrach. Wszystkie zbiory zostały zapisywane oraz odczytywane z plików .json.

|  |  |
| --- | --- |
| Rysunek . | Rysunek . |
| Rysunek . | Rysunek . |

|  |  |
| --- | --- |
| Rysunek 5. | Rysunek 6. |
| Rysunek 7. | Rysunek 8. |

Algorytm Grahama we wszystkich zbiorach dobrze wyznaczył otoczkę, zatem dalej zostały umieszczone wykresy wygenerowane za jego pomocą.

* 1. Algorytmy

W obu algorytmach zastosowano własną implementację wyznacznika 3x3 do określania położenia punktu względem prostej. Za tolerancje przyjęto **ε** = 10^(-12).

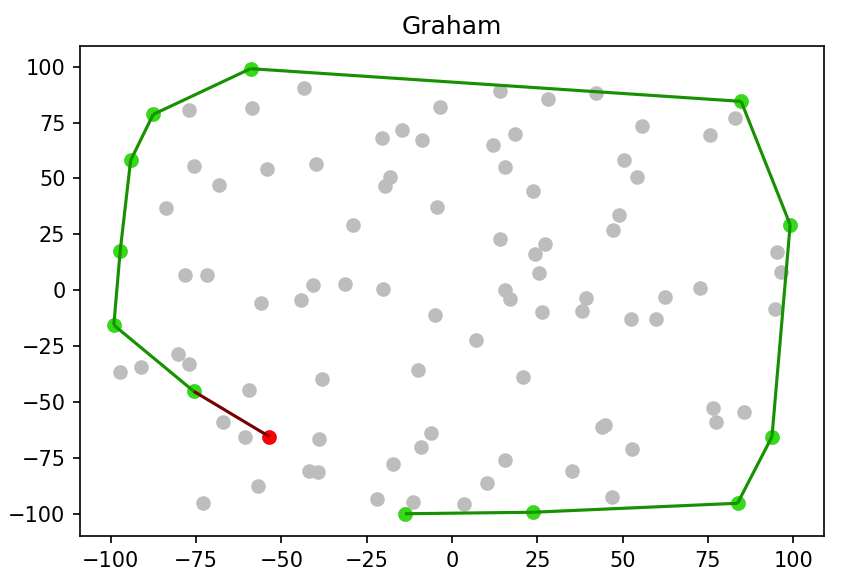
W implementacji algorytmu Grahama zamiast usuwania punktów współliniowych zaimplementowana została pętla, która pomija punkty współliniowe i dodaje do otoczki punkt o największej odległości od poprzedniego wierzchołka otoczki.

Zaś w algorytmie Jarvisa punkty współliniowe są usuwane z otoczki w trakcie jej generowania. Pętla usuwa ostatni punkt otoczki dopóki ostatnie trzy punkty nie są współliniowe.

* 1. Animacje

Animacje zostały załączone w postaci łączy do strony [giphy.com](https://giphy.com/).

Kolorem szarym zostały zaznaczone wszystkie punkty należące do danego zbioru punktów, na zielono punkty oraz krawędzie należące do otoczki. Czerwonym kolorem oznaczono rozpatrywaną krawędź i punkt.



Rysunek 9. Przykładowa klatka [animacji](https://giphy.com/gifs/AzbT7lB50PSQ9SiEnh) wykonani

algorytmu Grahama na zbiorze A

1. Analiza Danych
   1. Zbiory z treści zadania

Oba algorytmy działały poprawnie na zbiorach z treści zadania.

* + 1. Zbiór A

Zbiór ten jest złożony z punktów wygenerowanych losowo, więc jest bardzo małe prawdopodobieństwo wystąpienia wyjątków (np. trzech punktów współliniowych). Większość punktów występuje wewnątrz otoczki.

* + 1. Zbiór B

Otoczka tegoż zbioru zawiera wszystkie jego punkty, więc algorytm Grahama sprawdza się najlepiej. Złożoność algorytmu Jarvisa wynosi O(n^2).

* + 1. Zbiór C

Do otoczki tego zbioru należą po 2 punkty leżące na bokach prostokąta, reszta punktów jest współliniowa. Tym samym jest to dobry zbiór do sprawdzania zachowania algorytmów w przypadku współliniowości. Algorytm Jarvisa zachowuje się tu najlepiej, gdyż nie musi sprawdzać wszystkich punktów.

* + 1. Zbiór D

Otoczka składa się tylko z 4 wierzchołków. Znów algorytm Jarvisa okazuje się być tym sprawniejszym, bo tylko 4 razy musi znaleźć punkt o minimalnym kącie.

* 1. Zbiory większe

Wygenerowane zostały cztery zbiory danych:

1. 1000 losowych punktów o współrzędnych z przedziału [-100, 100]
2. 1000 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
3. 1000 losowych punktów leżących na bokach prostokąta o wierzchołkach

(-10, 10), (-10, -10), (10, -10), (10, 10),

1. Zbiór zawierający: 4 wierzchołki kwadratu o lewym dolnym rogu w (-10, 10) i o boku a=10,

Po 250 punktów na lewym i dolnym boku,

Po 200 punktów leżących na przekątnych kwadratu

|  |  |
| --- | --- |
| Rysunek 10. | Rysunek 11. |
| Rysunek 12. | Rysunek 13. |

Nie ma zauważalnych różnic w zachowaniu się algorytmów pomiędzy zbiorami większymi a standardowymi.

Zbiór A: [Algorytm Jarvisa](https://giphy.com/gifs/Utb1Xe0LBm933pEDJE); Zbiór D: [Algorytm Grahama](https://giphy.com/gifs/bc4It4YQ3yDlCC9zkL), [Algorytm Jarvisa](https://giphy.com/gifs/UrGhYjvYGbqVhbq63j)

* 1. Zbiory Mniejsze

Wygenerowane zostały cztery zbiory danych:

1. 100 losowych punktów o współrzędnych z przedziału [-10^(-4), 10^(-4)]
2. 50 losowych punktów leżących na okręgu o środku (1,0) i promieniu R=10^(-4),
3. 100 losowych punktów leżących na bokach prostokąta o wierzchołkach

(-10^(-4), 10^(-4)), (-10^(-4), -10^(-4)), (10^(-4), -10^(-4)), (10^(-4), 10^(-4)),

1. Zbiór zawierający: 4 wierzchołki kwadratu o lewym dolnym rogu w (1, 0), boku a=10^(-4),

Po 25 punktów na lewym i dolnym boku,

Po 20 punktów leżących na przekątnych kwadratu

|  |  |
| --- | --- |
| Rysunek 13. | Rysunek 13. |
| Rysunek 14. | Rysunek 15. |

Algorytm Jarvisa niedokładnie wyznacza minimalny element zbioru, przez co generuje błędy w otoczce. Na wygenerowanych danych w zbiorze C algorytm Jarvisa wchodził w nieskończoną pętlę, gdyż po wygenerowaniu już całkowitej otoczki nie powracał do początkowego punktu przez to, że nie wykrywał, że to on jest wierzchołkiem tworzącym następny najmniejszy kąt. Przy testowaniu innych zbiorów algorytm Grahama zawsze dobrze wskazywał otoczkę, a algorytm Jarvisa potrafił dawać błędny wynik w zbiorach B oraz C.

Zbiór A: [Algorytm Grahama](https://giphy.com/gifs/SYsYUuNZRm5PkrreOh), [Algorytm Jarvisa](https://giphy.com/gifs/nqYrW8MmALdiZ7l4kN)

Zbiór B: [Algorytm Grahama](https://giphy.com/gifs/nqjyoR7FHMZEr13wqX), [Algorytm Jarvisa](https://giphy.com/gifs/MUTKuOGUTWwdsvzaxX)

Zbiór C: [Algorytm Grahama](https://giphy.com/gifs/16RHEnlVJ05ZmsaRWR), [Algorytm Jarvisa](https://giphy.com/gifs/LFZmMSaInM4zST0bex)

Zbiór D: [Algorytm Grahama](https://giphy.com/gifs/sT9u0hnkhIhCZOt7Qg), [Algorytm Jarvisa](https://giphy.com/gifs/jEPkWwZihTZSzP9HO9)

1. Pomiary czasowe funkcji

Czas został zmierzony za pomocą funkcji time.time oraz jest on średnią ze 100 powtórzeń działania danego algorytmu na nieposortowanym zbiorze.

Można zauważyć, że algorytm Grahama jest zawsze szybszy w zbiorach A i B. W zbiorze większym D algorytm Jarvisa jest szybszy, gdyż ma do znalezienia tylko 4 punkty. Jarvis nie znalazł otoczki w zbiorze mniejszym C, gdyż wykonywał nieskończoną pętlę opisaną w punkcie wyżej. W zbiorze większym Jarvis jest szybszy w zbiorach C i D.

1. Wnioski

Zadanie to prezentuje różnice w sposobach rozwiązywania jednego problemu. Dwa różne podejścia mogą zwracać ten sam prawidłowy wynik, aczkolwiek zupełnie różnie się zachowywać. W treści zadania wybrano takie zbiory punktów by zaznaczyć wady i zalety danych algorytmów. Algorytm Grahama zawsze sortuje listę punktów, potem przechodzi po niej liniowo. Podejście to dobrze nadaje się do zbiorów danych, o których nic nie wiemy. Złożoność obliczeniowa to zawsze O(nlogn). Algorytm Jarvisa zaś ma złożoność O(n\*k), więc jeśli jesteśmy wcześniej przybliżyć ilość punktów oczekiwanej otoczki oraz będzie ona wystarczająco mała to powinniśmy skorzystać z tego algorytmu.