

Грищенко Юрій ІПС-32
Лабораторна робота №5

Постановка задачі: Знайти опуклу оболонку для множини точок X — мінімальну опуклу множину, що містить X .

Опис алгоритму:

“Швидкий” метод побудови опуклої оболонки - підхід, схожий на швидке сортування (quicksort), звідси назва ШвидкОбол (QuickHull).

Метод розбиває множину S із N точок на дві підмножини, кожна з яких міститиме одну із двох ламаних, з'єднання яких дає багатокутник опуклої оболонки.

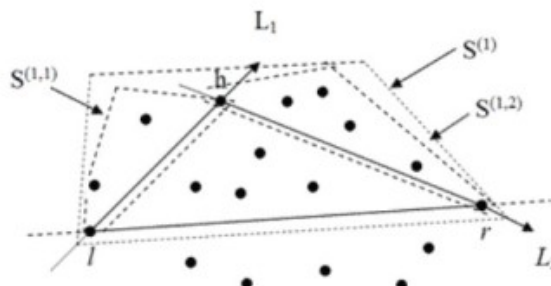
Складність алгоритму:

Розбиваємо множину точок S на підмножини в залежності від того, знаходяться вони зліва чи справа від лінії (ті, що справа, умовно відкидаються):

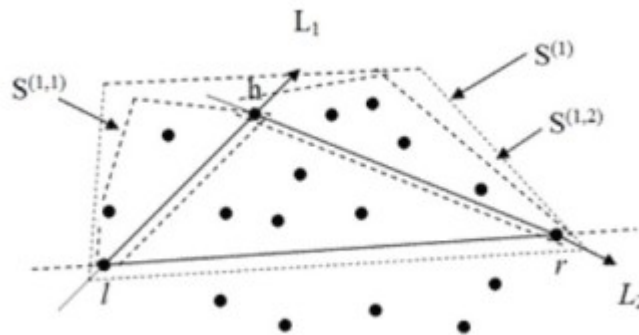
- Виберемо початкові значення $\{l_0, r_0\}$ точок l та r :
 - за l_0 – точку з найменшою абсцисою (x_0, y_0);
 - за r_0 – точку ($x_0, y_0 - \varepsilon$), де $\varepsilon > 0$ – мале число.
- Через це за початкову пряму, яка розбиває множину на частини, обирається вертикальна пряма, що проходить через точку l_0 .
- Після завершення алгоритму точка r_0 вилучається.

Далі рекурсивно оброблюємо підмножини точок зліва від лінії lr :

- Розглянемо підмножину S_1 .
- Визначимо точку h , для якої трикутник (hlr) має максимальну площу серед усіх трикутників $\{(plr): p \in S_1\}$.
- Якщо таких точок більше однієї, то вибираємо найлівішу.
- Точка h гарантовано належить опуклій оболонці.



- Побудуємо дві прямі: L_1 – спрямована із l в h ; L_2 – спрямована із h в r .
- Для кожної точки множини S_1 визначається її положення відносно цих прямих.
- Важливо помітити, що жодна з точок не знаходиться одночасно ліворуч від L_1 та від L_2 .
- Всі точки, розташовані праворуч від обох прямих, є внутрішніми точками трикутника $(l r h)$ і тому можуть бути вилучені із подальшої обробки.



- Точки, розташовані ліворуч від L_1 або на ній (і розташовані праворуч від L_2), утворюють підмножину $S_{1,1}$. Аналогічно утворюється підмножина $S_{1,2}$.
- Утворені підмножини передаються на наступний рівень рекурсивної обробки.
- Опукла оболонка для S_1 утворюється склейкою впорядкованих списків вершин опуклих оболонок для $S_{1,1}$ і $S_{1,2}$.

Складність алгоритму:

Складність схожа на QuickSort — в загальному випадку $O(n \log n)$, але в найгіршому випадку $O(n^2)$.

На кожному кроці робимо $O(n)$ операцій, і розбиваємо множину на дві підмножини, але не гарантовано, що підмножини рівні. (це гарантує алгоритм “Розділяй та володарюй” Шеймоса, він завжди виконується за час $O(\log n)$).

Також існують алгоритми:

Метод Грехема — $O(n \log n)$, якщо точки задані в полярних координатах і відсортовані, то $O(n)$

Метод Ендрю — аналогічно, з декартовими координатами

Метод Джарвіса — $O(hn)$ взагалі, в найгіршому випадку $O(n^2)$

Реалізовано на мові Python.

Інтерфейс користувача: набір точок задається файлом points.txt.
Програма у вікні показує всі задані точки та малює опуклу оболонку.