УДК 681.3

Терещенко В.М., д. ф.-м.н., професор

Один метод тріангуляції множини точок на площині

Київський національний університет імені Тараса Шевченка, 83000, м. Київ, пр-т. Глушкова 4д,
e-mail: v_ter@ukr.net

V.M. Tereshchenko, Dr. Sci.(Ph.-M.), Professor

One method for triangulation a set of points in the plane

Taras Shevchenko National University of Kyiv, 83000, Kyiv, Glushkovast., 4d,
e-mail: v_ter@ukr.net

У роботі пропонується метод тріангуляції множини точок на площині. В основі методу лежить ідея побудови опуклих шарів обходом Грехема, що дозволяє розробити алгоритм з оптимальною складністю θ(NlogN) та простою реалізацією. На множині S із N точок будуються опуклі оболонки, які утворюють k шарів, кожен з яких тріангулюється за один обхід сусідніх опуклих оболонок. Алгоритм легко розпаралелюється: кожен із шарів можна тріангулювати незалежно. Головна особливість запропонованого алгоритму у тому, що він має дуже просту реалізацію, а елементи (трикутники) одержаної тріангуляції представляються у вигляді простих і водночас швидких структур даних: зчеплена черга трикутників чи дерево трикутників. Це робить алгоритм зручним при розв'язанні широкого класу прикладних задач обчислювальної геометрії та комп'ютерної графіки, зокрема: перетин, рендеринг та морфінг.

Ключові слова: тріангулція, опукла оболонка, множина точок , опуклі шари, обхід Грехема.

In this paper, we propose a triangulation method for a set of points in the plane. The method is based on the idea of constructing convex layers by Graham's scan. It allows to develop an algorithm with the optimal complexity of θ(NlogN) and an easy implementation. First, convex hulls are constructed for the set S of N points, forming k layers. Then, each layer is triangulated in one scan of the adjacent convex hulls. Algorithm is easily parallelized: each layer can be triangulated independently. The main feature of the proposed algorithm is that it has a very simple implementation and the elements (triangles) of the resulting triangulation are presented in the form of simple and at the same time fast data structures: concatenable triangle queue or triangle tree. This makes the algorithm convenient for solving a wide range of applied problems of computational geometry and computer graphics, including intersection, rendering and morphing.

KeyWords: triangulation, convex hull, set of points, convex layers, Graham scan .

Статтю представив д.ф.-м.н., проф. Анісімов А.В.

1. **Вступ**

Постановка проблеми. В роботі пропонується оптимальний алгоритм тріангуляції множини точок на площині. Головною перевагою тріангуляції є те, що від об'єкта, який є потенційно дуже складний, ми можемо перейти до більш простих полігонів (трикутників), для їх подальшого дослідження.

Актуальність. На сьогоднішній день, для розв'язання задачі тріангуляції на множині точок існує ряд ефективних алгоритмів [1]. Розрізняють такі групи алгоритмів побудови тріангуляції: ітеративні алгоритми (найменш ефективні і досить складні в реалізації) [2], алгоритми на основі стратегії «розділяй та володарюй» (найшвидші і відносно прості в реалізації) [2-5], алгоритми прямої побудови (дають непогану (навіть лінійну) швидкість побудови в середньому, прості в реалізації) [6] та

двопрохідні алгоритми (найбільш складні в реалізації, не дуже ефективні) [7].

Найбільш ефективними, на мій погляд, є методи тріангуляції на основі стратегії «розділяй та пануй», які дають час виконання $O(N\log N)$ у гіршому і середньому випадках [2-5]. А з поміж них варто виділити алгоритми, які на кроці злиття використовують структури даних у вигляді зчеплених черг [4, 5]. Ці алгоритми дають оптимальний результат, що стосується оцінок складності $\theta(N\log N)$, проте, бажано б мати більш простішу реалізацію. Тому природно постає питання – а чи можна розробити алгоритм, який давав би високу ефективність і , водночас, був би простий в реалізації? Це, ще раз підкреслюю, важливо з точки зору практичного використання алгоритму. Таким прикладом може бути метод Грехема побудови опуклої оболонки, який дає оптимальну часову складність і в той же час дуже простий в реалізації [8]. І тому цей метод надихнув на ідею розробити алгоритм тріангуляції такий же ефективний і простий в реалізації, як і алгоритм побудови опуклої оболонки.

Мета. Розробити алгоритм тріангуляції на основі методу Грехема, який би давав оптимальну оцінку складності по часу ($\theta(N\log N)$) і водно час був простий в реалізації.

Новизна. В роботі запропоновано новий алгоритм тріангуляції множини точок на основі методу Грехема, який дає оптимальну оцінку складності $\theta(N\log N)$.

2. Постановка та метод розв'язання задачі

Постановка. На множині S із N точок на площині побудувати тріангуляцію, використовуючи обхід Грехема з часовою складністю $\theta(N\log N)$.

Метод розв'язання задачі

Нехай задана множина S із N точок на площині, рис. 1.

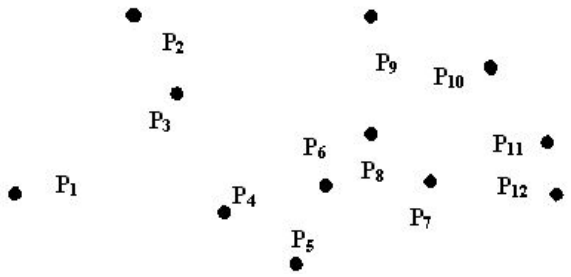

Рис.1. Приклад вхідної множини точок.

1. Згідно методу Грехема знаходимо центроїд q (x_q, y_q) перших трьох неколінеарних точок. Для прикладу на рис. 11 це буде центроїд точок P_1, P_2, P_3, з координатами x_q, y_q , відповідно, рис.2.:

$$x_q = \frac{x_1 + x_2 + x_3}{3}, \; y_q = \frac{y_1 + y_2 + y_3}{3}.$$

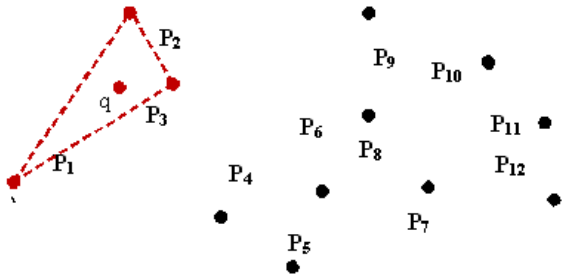

Рис.2. Визначення точки q.

2. Сортуємо відносно точки q задану множину S із N точок за полярним кутом (проти годинникової стрілки) одержавши упорядкований список U. Для прикладу з рис. 1 матимемо U= {P_4, P_5, P_6, P_7, P_{12}, P_8, P_{11}, P_3, P_{10}, P_9, P_2, P_1}.

3. До списку U застосовуємо обхід Грехема, в результаті чого одержимо опуклу оболонку для множини S з границею CH(S) = {P_5, P_{12}, P_{11}, P_{10}, P_9, P_2, P_1}, рис.3.

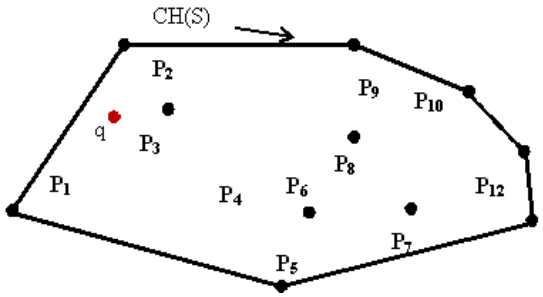

Рис. 3. Опукла оболонка для S.

4. До множини точок S_1, що залишилася в середині опуклої оболонки будуємо опуклу оболонку $CH(S_1)$ обходом Грехема. Аналогічно будуємо опуклі оболонки $CH(S_2)$, …, $CH(S_k)$ для наступних множин S_2, …, S_k , доки це можливо, рис.4.

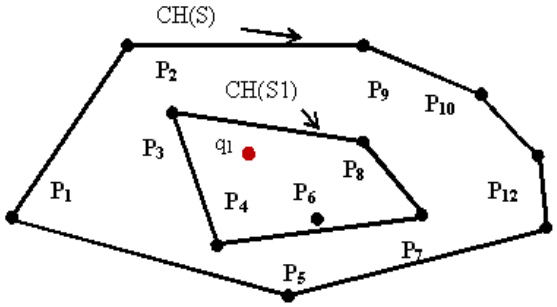

Рис. 4. Побудова опуклих оболонок $CH(S_2)$, …, $CH(S_k)$

5. Шари утворені сусідніми границями опуклих оболонок тріангулюємо відповідним чином, рис. 5. Це можна робити, навіть, підчас кожного наступного обходу, а при наявності можливості паралельної обробки, тріангулювати незалежно кожен шар.

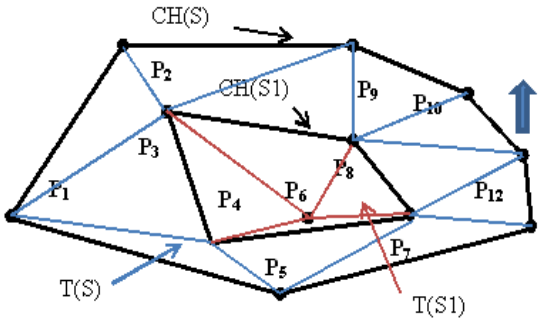

Рис.5 . Триангуляція опуклих шарів

3. **Побудова структури даних**.

Постає питання підтримки одержаної тріангуляції у вигляді певної структури даних, яку можна було б використовувати для обробки та розв'язання наступних задач (розфарбування, пошук перетинів, морфінг, рендеренінг, булеві операції і т. і.). І в цьому випадку процедура побудови тріангуляції надає нам і зручний її підтримки, а саме:

1) під час обходу побудови кожного нового трикутника ми присвоюємо йому ім'я і додаємо його до створеного циклічно упорядкованого списку таких трикутників. На рис. 6 показано відповідний приклад.

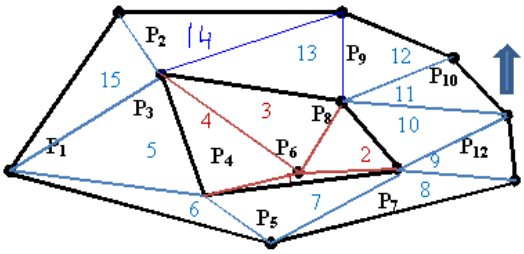

Рис.6. Побудова списку трикутників: T= { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,15}

2) утворений список ми можемо представити у вигляді зчепленої черги (рис.7), яка зберігає зв'язність в кожному із шарів (сині та коричневі лінії), а також містить вказівники на зв'язок між гранями сусідніх шарів (зелені лінії).

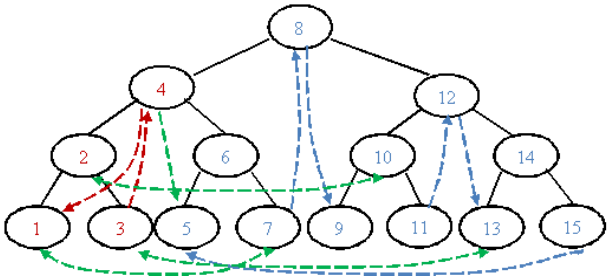

Рис. 7. Структура даних у вигляді зчепленої черги для рис. 6. Червоним помічені грані першого шару {1,2,3,4} синім-другого шару {5, 6, 7, 8, 9, 10, 11, 12, 13, 14,15}

Це дозволяє здійснювати логарифмічний пошук по тріангуляції у будь-якому шарі та напрямку.

3) саму пошарову тріангуляцію можна представити у вигляді бінарного дерева граней (рис.8), яке зберігає зв'язність .

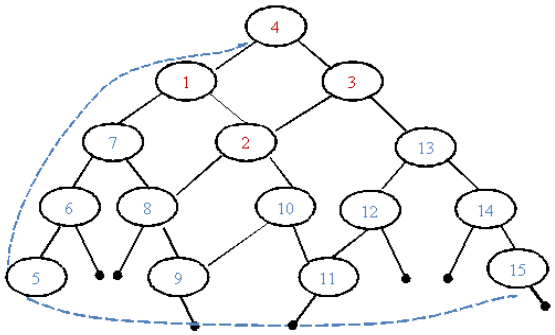

Рис. 8. Структура даних у вигляді бінарного дерева граней.

За допомогою цієї структури даних можна здійснювати логарифмічний пошук з будь-якого трикутника.

3. Обґрунтування складності

Твердження. Запропонований алгоритм має часову складність як і метод Грехема побудови опуклої оболонки – $O(N\log N)$ та використовує лінійну пам'ять.

Доведення. 1-й, 2-й та 3-й кроки це кроки алгоритму Грехема і вони виконуються за час $O(N)$, $O(N\log N)$, $O(N)$, відповідно. 4-й крок – це обхід Грехема і виконується він за час $O(N)$, але уже над множиною точок, що лишились після 3-го кроку. 5-й крок- тріангуляція шарів, яка здійснюється (з врахуванням упорядкованості точок на границі кожної опуклої оболонки за полярним кутом) за один прохід з часом $O(N)$.

4. Висновки

У роботі запропоновано оптимальний алгоритм тріангуляції на множині точок з оцінкою складності $O(N\log N)$. В основі цього алгоритму лежить алгоритм обходу Грехема побудови опуклої оболонки. На множині S із N точок будуються опуклі оболонки. Які утворюють k шарів, кожен з яких тріангулюється за один обхід сусідніх опуклих оболонок. Алгоритм легко розпаралелюється: кожен із шарів можна тріангулювати незалежно. Саме головна особливість запропонованого алгоритму у тому, що він має дуже просту реалізацію.

Список використаних джерел

1. *Скворцов А.В.*Триангуляция Делоне и её применение / А. В. Скворцов. – Томск:Том. Ун-т. – 2002. – 128 с.
2. *Скворцов А.В.* Эффективные алгоритмы построения триангуляции Делоне / А. В. Скворцов, Ю. Л. Костюк // Геоинформатика. Теория и практика. –Томск: Изд-во Том. ун-та. –1998. – Вып. 1.– С. 22–47.
3. *Guibas L.* Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams / L.Guibas, J. Stolfi // ACM Transactions on Graphics. –1985. –Vol. 4, N. 2. – P. 74–123.
4. Терещенко В.М. Узагальнений підхід розв'язання деяких задач обчислювальної геометрії на основі рекурсивно-паралельної технології / В.М. Терещенко // Наукові нотатки. –2008. – Випуск 22 (частина 2). – С. 344–349.
5. Терещенко В.Н. Рекурсия и параллельные алгоритмы в задачах геометрического моделирования / В.М. Терещенко, А.В. Анисимов // Кибернетика и системный анализ. – 2010. – №2.– С. 9 – 22.
6. Kirkpatrik D.G. Optimal search in planar subdivisions / D.G. Kirkpatrik // SIAM J. Comput. – 1983. – Vol. 12, N. 1. – P. 28–35.
7. Lewis B. Triangulation of planar regions with applications/ B. Lewis, J. Robinson // The Computer Journal. – 1978. – Vol. 21, N. 4. – P. 324–332.
8. Graham R. L. An efficient algorithm for determining the convex hull of a finite planar set / R. L. Graham // Info. Proc. Lett., 1. – 1972. – P. 132– 133.

References

1. SKVORCOV A.B.(2002) Trianguliacia Delone i eyio primenenie. Tomsk: Tom. Un-t. 128 p.
2. SKVORCOV A.B., KOSTIUK Y.L. (1998) Effectivnie algoritmi postroenia trianguliacii Delone // Geoinformatika. Teoria i praktika.- Tomsk: Un-t. - P. 22–47.
3. GUIBAS L., STOLFI J.(1985) Primitives for the manipulation of general subdivisions and the computation of Voronoi diagrams // *ACM Transactions on Graphics*.- 4, N 2. - P. 74–123.
4. TERESHCHENKO V. (2008) The generalized approach of the decision of some problems of computing geometry on the basis of recursively-parallel technology // Naukovi notatki, Luck-2008. - 22 , N 2.- P. 344-349.
5. TERESHCHENKO V.N., Anisimov A.V. (2010) Recursion and parallel algorithms in geometric modeling problems // *Journal: Cybernetics and Systems Analysis:*- 46, N 2**.**- P. 173 - 184.
6. KIRPATRIK D.G.(1983) Optimal search in planar subdivisions // *SIAM J. Comput.* – 12, N 1. – P. 28–35.
7. LEWIS B., ROBINSON J. (1978) Triangulation of planar regions with applications // *The Computer Journal.* – 21, N 4. - P. 324–332.
8. GRAHAM R. L.(1972) An efficient algorithm for determining the convex hull of a finite planar set // *Info. Proc. Lett*.- 1.- P. 132-133.

Надійшла до редколегії 03.12.14