Метод Форчуна

У багатьох випадках природа задачі, що розв’язується, така, що при тріангуляції необхідно ураховувати різного роду обмеження, тобто при постановці задачі на множину ребер тріангуляції накладаються певні обмеження. Типовий приклад такої ситуації дає задача тріангуляції всередині простого багатокутника.

Для розв’язування таких задач з обмеженнями можна було б використати "жадний" метод, але він має один недолік: ефективність отримання розв’язку далека від оптимальної. З іншої сторони, не видно способу застосувати для розв’язання таких задач метод тріангуляції Делоне [1]. Отже, необхідно пошукати новий метод.

Зазвичай на площині задано множину S із N точок. Крім того, на S задано множину E із M ребер, що не перетинаються (обмеження), які повинні ввійти в число ребер тріангуляції. Умовимось, що область, яка підлягає тріангуляції, є найменшим прямокутником, сторони якого паралельні координатним осям, і цей прямокутник містить всі точки множини S. Відмітимо, що у прямокутнику існують принаймні дві вершини, які мають найбільше значення y-координати, і принаймні дві вершини з найменшою y-координатою.

Мета полягає у тому, щоб ефективно виконати розбиття прямокутника на багатокутники меншого розміру(монотонні, див. означення 6.1[1, стор.290] та рис.1), тріангуляція яких не буде викликати ускладнень.

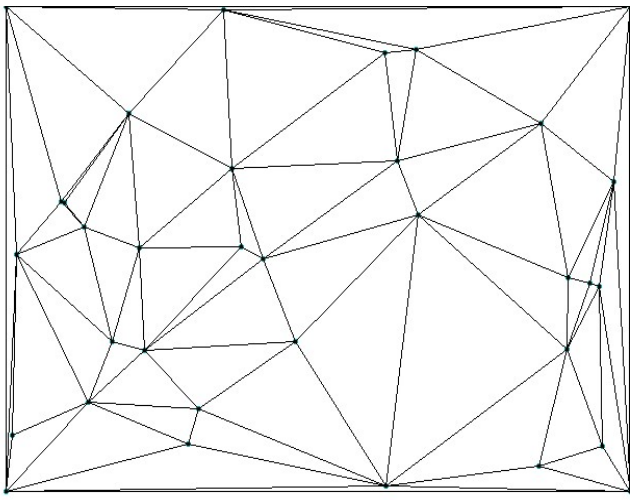


Рис. 1

Метод, який розглядається нижче, належить класу методів, які базуються на замітанні площини, і по суті співпадають із процедурою "регуляризації" планарного графа, описаної в [1]. Процедура регуляризації за час $O(N \log N)$ додає ребра так, що виконуються наступні умови:

(1) ніякі два ребра не перетинаються (за винятком вершин);

(2) кожна вершина (за винятком вершин з найбільшою *y*-координатою) безпосередньо з'єднана принаймні з однією вершиною, яка має більшу *y*-координату;

(3) кожна вершина (за винятком вершин з найменшою *y*-координатою) безпосередньо з'єднана принаймні з однією вершиною, яка має меншу *y*-координату.

В [1] доведено, що кожна область планарного графа, отримана у результаті процедури регуляризації, є монотонним багатокутником.

Постановка задачі

Написати програму, яка дозволяє розв'язати наступну задачу:

На заданій множині *S(N)* точок площини, обмеженої прямокутником, побудувати триангуляцію.

Програма має брати з входу координати точок і на виході видавати список ребер триангуляції. Для розв’язання слід обрати алгоритм з складністю $O(N\log N)$.

Обгрунтування оцінки складності

Загальна оцінка складності побудови триангуляції є результатом виконання таких етапів алгоритму:

1) Сортування початкової множини точок за допомогою методу швидкого сортування[2], складність якого O(N*log(N)).

2) Побудова діаграми Вороного методом Форчуна, що вимагає O(N*log(N)) елементарних дій. Це випливає з того, що сайти "берегової" лінії зберігаються в збалансованому бінарному дереві, в якому кожна операція пошуку, вставки чи видалення сайту виконується за час O(log(N)). Враховуючи загальну кількість точок N, отримуємо складність O(N*log(N)).

3) Побудова триангуляції на основі діаграми Вороного шляхом з'єднання відповідних вершин за лінійний час O(3N-6)=O(N), де 3N-6 - кількість ребер діаграми.

У результаті отримуємо загальну оцінку складності O(N*log(N)).

Характеризація вводу-виводу даних

Вхідні дані програма читає з файлу, ім’я якого задається з клавіатури, або автоматично генерує випадковий набір даних і зберігає згенерований набір в файл.

Побудований граф триангуляції програма

1) виводить на екран у вигляді малюнку в графічному режимі (при цьому малюнок масштабується таким чином, що заданий прямокутник, в межах якого будувалася трангуляція, повністю виводиться на екран разом з усіма елементами триангуляції);
2) записує в файл 'delone.txt' у вигляді таблиці трійок чисел, що є номерами сайтів з вхідного списку сайтів. При цьому трійки сайтів і є трикутниками побудованої триангуляції Делоне.

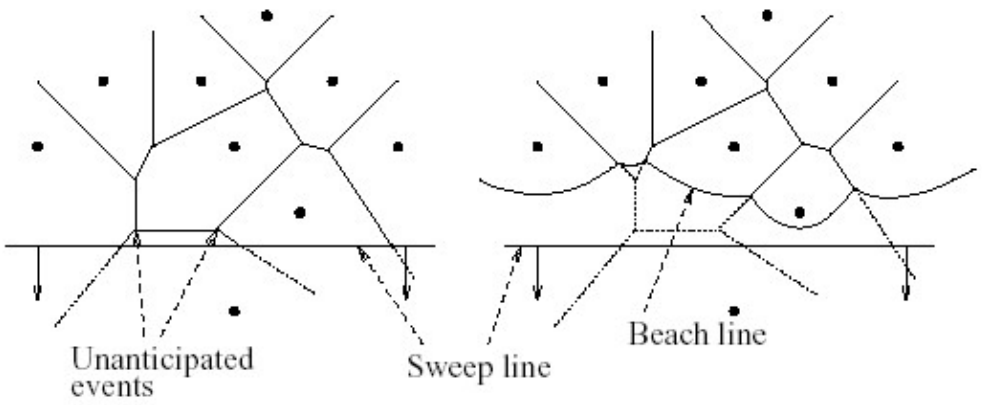
Науковий висновок

Існує багато різних підходів розв’язання задачі триангуляції. Зокрема, методи, які будують триангуляцію безпосередньо, та методи, що базуються на попередній побудові діаграми Вороного. Серед існуючих методів побудови діаграми Вороного ми обрали метод Форчуна з використанням таких структур даних як збалансоване бінарне дерево і двозв’язний список, що мають складність обходу O(N*log(N)) та O(N) відповідно. Є дві причини обрання нами саме цього метода: по-перше, його оптимальна оцінка складності O(N*log(N)), що виконується і в найгіршому випадку вхідних даних; по-друге, простота ідеї і можливість ефективного програмування на відміну від інших методів тієї ж складності, наприклад методу, що базується на принципі «Розділяй та володаруй»[1].

Серед перспектив подальшого розробки алгоритму можна вказати можливість узагальнення його на проблеми для тривимірного простору.

Алгоритм Форчуна побудови діаграми Вороного [3] – алгоритм планарного замітання, що дає можливість розв'язати задачу за час O(N*log(N)). Ключовим моментом алгоритму планарного замітання є можливість знаходження всіх подій, що надходять, ефективним способом. Наприклад, в алгоритмі перетину відрізків ми розглядаємо всі пари сусідніх відрізків і заносимо їх точки перетину в чергу подій, що надходять.

Замість того, щоб будувати діаграму Вороного за допомогою планарного замітання в своїй загальній формі, розраховують викривлену, але топологічно еквівалентну версію діаграми. Остаточна діаграма має таку ж структуру, як і діаграма Вороного, але з гранями, що є параболічними дугами. На основі такої діаграми легко згенерувати правильну діаграму Вороного з прямими відрізками.



У нашому випадку замість викривленої діаграми будемо реалізовувати алгоритм за допомогою викривленої замітаючої прямої. Фактично ми будемо розглядати два об'єкти, що контролюють процес замітання: горизонтальна замітаюча пряма, що рухається згори вниз, і "берегова лінія" (beach line) – монотонна по *x* крива, що складається з параболічних дуг. Як і замітаюча пряма, берегова лінія рухається донизу, але її форма буде залежати від розташування точок на площині, які ми будемо називати сайтами. Ми побачимо, що діаграма Вороного «виростає» з берегової лінії.

Відмітимо, що в задачі із звичайним планарним замітанням сайти, що лежать нижче замітаючої прямої, можуть впливати на отриману діаграму, що

знаходиться над нею. Щоб уникнути цього, використаємо тільки частину діаграми, на яку не впливають елементи, що знаходяться нижче замітаючої прямої. Для цього поділимо півплощину над прямою на дві області: ті точки, що ближчі до деякого сайта p з цієї півплощини ніж до замітаючої прямої, і точки, що є ближчими до замітаючої прямої ніж до сайта з півплощини.

Множина точок q, рівновіддалених від замітаючої прямої до свого найближчого сусіда і є "береговою лінією". Частина діаграми Вороного над "береговою лінією" *захищена* в тому розумінні, що ми маємо всю необхідну для її побудови інформацію, ігноруючи відомості про точки, що мають з'явитися за замітаючою прямою.

Відомо, що множина точок, рівновіддалених від сайта, що лежить над горизонтальною прямою, і від самої прямої формує параболу, відкриту згори. Легко показати, що парабола, наближаючись до прямої, стає «тоншою» і вироджується у вертикальний промінь з початком в даному сайті. Звідси, "берегова лінія" складається з нижніх частин цих парабол. Відмітимо, що вершиною є перетин двох кривих берегової лінії. Ця точка рівновіддалена від двох сайтів і замітаючої прямої і тому має лежати на ребрі діаграми Вороного. Зокрема, якщо криві "берегової лінії", що відповідають точкам p_i та p_j, мають спільну точку перетину , тоді ця точка лежить на ребрі Вороного між p_i та p_j.

З рухом замітаючої прямої параболи "берегової лінії" неперервно змінюють свою форму. Як і в інших алгоритмах планарного замітання, нас буде цікавити «значуща подія» точок, що змінюють топологічну структуру берегової лінії. Маємо два види таких подій:

Подія сайту: Коли замітаюча пряма зустрічає новий сайт, відбувається вставка нової дуги в берегову лінію.

Подія вершини: Коли довжина параболічної дуги скорочується до нуля, дуга зникає і в цій точці створюється нова вершина Вороного.

Алгоритм базується на дослідженні цих двох типів подій. На основі отримання вершин Вороного корегується двозв’язний список граней для діаграми і діаграма будується.

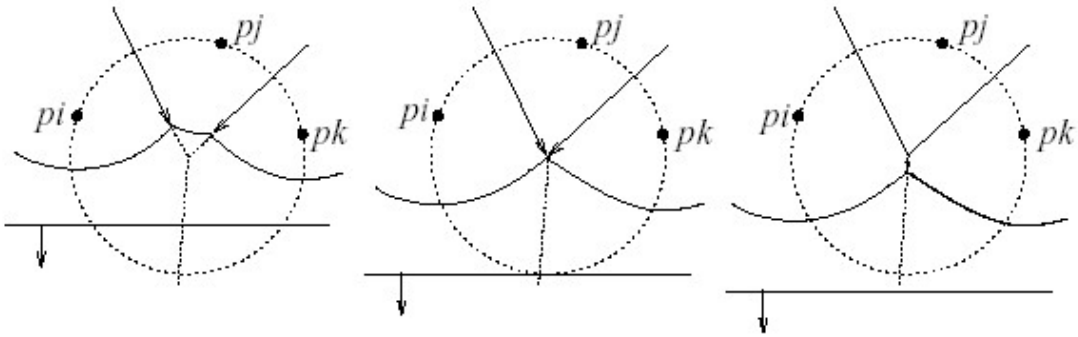
Далі, більш детально, розглянемо появу і обробку подій.

Подія сайту: Отже, подія сайту відбувається в момент проходження замітаючої прямої через сайт. З подальшим рухом прямої вироджена парабола (вертикальний промінь з початком в сайті) розширюється в дугу вздовж "берегової лінії". Нехай ця дуга лежить над новим сайтом. Тоді вона розщеплюється на дві дуги шляхом вставки нової невизначено малої параболи нового сайту. Паралельно замітаючому процесу ця парабола буде розширюватися і з часом долучиться до ребер діаграми.

Подія вершини: на відміну від знаходження сайтів, генерування вершин відбувається динамічно протягом реалізації алгоритму. Головною ідеєю є те, що кожна така подія генерується сусідніми за "береговою лінією" точками. Зокрема, розглядаються три сусідні точки p_i, p_j та p_k , дуги яких послідовно з’являються на береговій лінії зліва направо. Припустимо, що описане відносно цих трьох точок коло лежить принаймні частково нижче замітаючої прямої (тобто, вершину Вороного ще не побудовано), і що це описане коло не містить точок, нижчих за пряму (тобто жодна майбутня точка не заважатиме створенню вершини).

i

У момент дотику найнижчою точкою кола замітаючої прямої його центр є рівновіддаленим від усіх трьох сайтів і замітаючої прямої. Тобто, всі параболічні дуги проходять через його центральну точку (дуга точки p_j видаляється з "берегової лінії"). У термінах діаграми Вороного бісектриси (p_i, p_j) та (p_j, p_k) зустрінуться у вершині Вороного, причому залишиться лише бісектриса (p_j, p_k).



Представимо алгоритм більш детально.

Діаграма Вороного: Побудована діаграма Вороного буде зберігатися у вигляді двозв’язного списку граней. Є одна технічна складність, пов’язана з необмеженими гранями діаграми. Для її розв'язання ми припускаємо, що вся діаграма розглядається в обмеженому прямокутнику, достатньо великому для того, щоб всі вершини Вороного знаходилися всередині цього прямокутника.

"Берегова лінія": Сайти берегової лінії зберігаються в збалансованому бінарному дереві. Важливим є те, що ми не зберігаємо самі параболічні дуги. Замість цього для кожної дуги відповідної "берегової лінії" ми зберігаємо сайт, що дає початок даній параболі. Відмітимо, що сайт може виникати не один раз на "береговій лінії" (насправді лінійно n разів). Але загальна довжина "берегової лінії" не буде перевищувати $2n - 1$.

Між кожною послідовною парою сайтів p_i та p_j є точка зламу кривої, що рухається як функція замітаючої прямої, відмітимо можливість знаходження

«адреси» точки зламу кривої як функції p_i, p_j і поточної y-координати замітаючої прямої. Отже, як і у випадку "берегової лінії", ми неявно зберігаємо точки зламу. Скоріше ми розраховуємо їх тільки у випадку необхідності.

Проводимо такі важливі операції щодо "берегової лінії":

1) Зафіксувавши місцезнаходження замітаючої прямої; визначаємо дугу "берегової лінії", що перетинає отриману вертикальну лінію. Це може бути зроблено за допомогою бінарного пошуку точок зламу, що обчислюються під час реалізації алгоритму.
2) Розраховуємо попередні і наступні точки "берегової лінії".
3) Вставляємо нову параболічну дугу точки p_i всередину вже існуючої дуги p_j, що розбиває останню на дві. Таким чином створюються три параболи точок p_j, p_j та p_j.
4) Видаляємо дугу з "берегової лінії".

Не складно модифікувати збалансоване бінарне дерево для реалізації цих операцій за час O(log n).

Маємо пріоритетну чергу із здатністю вставляти і видаляти нові події, а також видаляти подію з найбільшою y-координатою. Для кожного сайту зберігаємо в черзі його y-координату.

Для кожної послідовної трійки p_i, p_j, p_k "берегової лінії" знаходимо описане коло. Якщо найнижча точка кола (його мінімальна y-координата) знаходиться нижче замітаючої прямої, створюємо вершину, y-координата якої є y-координатою найнижчої точки описаного кола. Зберігаємо цю подію в пріоритетній черзі. Кожна така подія в цій черзі має обернений перехресний зв'язок із трійками сайтів, що генерують ці події, і кожна послідовна трійка сайтів має перехресний зв'язок із згенерованою подією в пріоритетній черзі.

Цей алгоритм реалізується як будь-який інший алгоритм замітання. Ми виділяємо подію, обробляємо її і переходимо до наступної. Кожна подія може бути отримана в модифікованій діаграмі Вороного і "береговій лінії", а також може бути отримана шляхом створення або видалення існуючої події.

Кожна подія вимагає $O(1)$ часу плюс число доступів (константа) до різних структур даних. Кожний такий доступ займає $O(\log n)$ часу. Оскільки розмір структури даних $O(n)$, то загальний час обробки дорівнює $O(n \log n)$.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Ф.Препарата, М.Шеймос. Вычислительная геометрия: Введение. – М.Мир, 1989. -476 с.
2. Дональд Э.Кнут. Искусство программирования. Т.3. Сортировка и поиск. Издательский дом "Вильямс", 2000. – 822 с.
3. David M.Mount. Computer Graphics. Lecture notes. Dept. of Computer Science. University of Maryland, 1997.