Ускорение алгоритмов построения триангуляции Делоне/диаграммы Вороного

Амосов Федор

15 февраля 2014 г.

Итак, вспомним наш алгоритм построения графа Делоне D на наборе точек P через склеивание меньших подграфов.

1. Пусть S_i — случайный поднабор точек из P размера m;
2. Пусть P_i — разбиение точек P по клеткам Вороного C_i набора точек S_i (каждое P_i — это множество точек);
3. Построим граф Делоне D_i на каждом множестве P_i рекурсивным вызовом этого алгоритма. Пусть наш алгоритм возвращает помимо построенного графа еще и точки границы выпуклой оболочки. Тем самым, мы получим еще и H_i — точки границы выпуклой оболочки D_i (P_i);
4. Найдем все граничные треугольники. Мы их легко получим, зная все граничные точки H_i. Запустим dfs, описанный в предыдущем отчете, на графе соседних треугольников. С помощью него мы найдем все «плохие» треугольники. Выбросим все плохие треугольники из D_i;
5. Сконструируем множество точек V. Добавим в него все множества H_i. Так же добавим в него все точки найденных плохих треугольников;
6. Построим G — граф Делоне на V. Сделаем мы это каким–нибудь **другим** построителем графов Делоне;
7. Найдем в G те ребра, которые связывают вершины разных D_i;
8. Получим итоговый граф Делоне D вставкой этих ребер в объединение D_i.

На сей момент, корректность этого алгоритма была «проверена» только многочисленными экспериментами. Но сейчас нам будет интересен другой вопрос. Сколько этот алгоритм работает (в количестве операций)?

Итак, пусть $T(n)$ — время работы этого алгоритма на наборе из n точек. Пусть двумерных. Составим рекуррентное соотношение на $T(n)$. Предположения, в которых мы будем это делать,

- Все $ConvP_i$ имеют высокую выпуклость (не выстраиваются в линии)
- Все P_i имеют похожие размеры

Добиться этого можно взяв m (число множеств P_i) достаточно большим. Будем считать m константой.

Итак, из чего складывается $T(n)$,

1. Выбор m случайных точек — $O(m) = O(1)$
2. Разбиение всех точек по m клеткам. С учетом того, что m — константа, мы можем это сделать за $O(nf(m)) = O(n)$, где $f(m)$ — некоторая малая функция типа $\log m$ и т.п.
3. Построение всех D_i — $mT(\frac{n}{m})$

4. Запуск всех dfs — $O(mg(\frac{n}{m})\log n)$, где $g(n)$ — ориентировочное число точек границы выпуклой оболочки случайного множества из n точек. Для двумерного случая $g(n) = O(\sqrt{n})$. $\log n$ вылезает из–за поиска точки в круге при проверке треугольника на «хорошесть». Н.У.О. у нас уже построен поисковый индекс на P (или его частях). Тем самым, сложность получается такой, $O(n^{\frac{1}{2}}\log n)$.

5. Конструирование V — $O(mn^{\frac{1}{2}}) = O(n^{\frac{1}{2}})$, т.к. все плохие треугольники будут вдоль границ.

6. Пусть внешний алгоритм построения графа Делоне работает за $O(n^k)$. Тогда мы получим время работы на этом этапе $O((n^{\frac{1}{2}})^k) = O(n^{\frac{k}{2}})$.

7. Нахождение нужных ребер — $O(n^{\frac{1}{2}})$

8. Вставка ребер (удаление уже было произведено на этапе нахождения плохих треугольников) — $O(n^{\frac{1}{2}})$

Итого,

$$T(n) \le O(1) + O(n) + mT\left(\frac{n}{m}\right) + O(n^{\frac{1}{2}}\log n) + O(n^{\frac{1}{2}}) + O(n^{\frac{k}{2}}) + O(n^{\frac{1}{2}}) + O(n^{\frac{1}{2}})$$

$$T(n) \le mT\left(\frac{n}{m}\right) + O(n) + O(n^{\frac{k}{2}})$$

$$T(n) \le \begin{cases} mT\left(\frac{n}{m}\right) + O(n), & k \le 2 \\ mT\left(\frac{n}{m}\right) + O(n^{\frac{k}{2}}), & k > 2 \end{cases}$$

Вспомним основную теорему о рекуррентном соотношении.

Теорема
Если

$$T(n) \le aT\left(\frac{n}{b}\right) + O(n^d)$$

То

$$T(n) = \begin{cases} O(n^d), & d > \log_b a \\ O(n^d \log n), & d = \log_b a \\ O(n^{\log_b a}), & d < \log_b a \end{cases}$$

Воспользуемся ей для нашего случая. Итого получается,

$$T(n) \le \begin{cases} O(n\log n), & k \le 2 \\ O(n^{\frac{k}{2}}), & k > 2 \end{cases}$$

Тем самым, что мы сделали. Пусть у нас есть «тормознутый» алгоритм построения триангуляции Делоне. В нем много вычей, всяких непонятных штук и т.п. и он работает за $O(n^k)$. Мы научились «за дешево» его асимптотически ускорять по следующей схеме,

$$O(n^k) \to O(n\log n), \quad k \le 2$$

$$O(n^k) \to O(n^{\frac{k}{2}}), \quad k > 2$$

Более того, мы видим что этот переход сделан «с запасом», т.к. многие «хлипкие» $O(n^{\frac{1}{2}})$ были съедены ясной и понятной $O(n)$.

Тем самым, мы тупейший алгоритм $O(n^4)$ научились задаром превращать в $O(n^2)$, а какие–нибудь медленные, но надежные алгоритмы $O(n^2)$ сразу в $O(n\log n)$. Можем пойти еще дальше. Берем тупейший алгоритм $O(n^4)$, ускоряем его до $O(n^2)$, затем этот ускоренный алгоритм ускоряем до $O(n\log n)$. Как то странно это все, но весело =) Более того, это добро с виду хорошо параллелится. Еще можно подумать над тем, что строить граф Делоне G можно тоже без всяких внешних алгоритмов, а просто рекурсивным запуском нашего. Там тогда сразу получится сложность $O(n\log n)$, но это мутный момент, потому что становится непонятно, где мы вообще в таком случае что–то строим.

Осталось доказать два Утверждения для доказательства корректности алгоритма — и дело будет в шляпе =)