Алгоритм поиска ближайшей

пары точек методом

“разделяй и властвуй”

(Препарата, Шеймос)

Выполнил

Студент группы 11-002

Асадуллин Искандер Маратович

КФУ ИИТИС

Принцип устройства

Этот алгоритм был предложен Препаратой (Preparata) в 1975 г. Препарата и Шамос также показали, что в модели дерева решений этот алгоритм асимптотически оптимален.

Построим алгоритм по общей схеме алгоритмов **"разделяй-и-властвуй"**: алгоритм оформляем в виде рекурсивной функции, которой передаётся множество точек; эта рекурсивная функция разбивает это множество пополам, вызывает себя рекурсивно от каждой половины, а затем выполняет какие-то операции по объединению ответов. Операция объединения заключается в обнаружении случаев, когда одна точка оптимального решения попала в одну половину, а другая точка — в другую (в этом случае рекурсивные вызовы от каждой из половинок отдельно обнаружить эту пару, конечно, не смогут).

Итак, перейдём к построению алгоритма. Чтобы в будущем прийти к эффективной реализации стадии объединения, разбивать множество точек на два будем согласно их x-координатам: фактически мы проводим некоторую вертикальную прямую, разбивающую множество точек на два подмножества примерно одинаковых размеров. Такое разбиение удобно произвести следующим образом: отсортируем точки стандартно как пары чисел, т.е.:

 p_i < p_j \Longleftrightarrow (x_i < x_j) \lor \B[...]

Тогда возьмём среднюю после сортировки точку p_m(m = \lfloor n/2 \rfloor), и все точки до неё и саму p_mотнесём к первой половине, а все точки после неё — ко второй половине:

 A_1 = \{ p_i\ |\ i = 0 \ldots m \},   
 A_2 = \{ p_i\ |\ i = m+1 \ldots n-1 \}. 

Теперь, вызвавшись рекурсивно от каждого из множеств A_1и A_2, мы найдём ответы h_1и h_2для каждой из половинок. Возьмём лучший из них: h = \min (h_1, h_2).

Теперь нам надо произвести **стадию объединения**, т.е. попытаться обнаружить такие пары точек, расстояние между которыми меньше h, причём одна точка лежит в A_1, а другая — в A_2. Очевидно, что для этого достаточно рассматривать только те точки, которые отстоят от вертикальной прямой раздела на расстояние, меньшее h, т.е. множество Bрассматриваемых на этой стадии точек равно:

 B = \{ p_i\ |\ | x_i - x_m | < h \}. 

Для каждой точки из множества Bнадо попытаться найти точки, находящиеся к ней ближе, чем h. Например, достаточно рассматривать только те точки, координата yкоторых отличается не более чем на h. Более того, не имеет смысла рассматривать те точки, у которых y-координата больше y-координаты текущей точки. Таким образом, для каждой точки p_iопределим множество рассматриваемых точек C(p_i)следующим образом:

 C(p_i) = \{ p_j\ |\ p_j \in B,\ \ y_i - h < y_j \[...]

Если мы отсортируем точки множества Bпо y-координате, то находить C(p_i)будет очень легко: это несколько точек подряд до точки p_i.

Итак, в новых обозначениях **стадия объединения** выглядит следующим образом: построить множество B, отсортировать в нём точки по y-координате, затем для каждой точки p_i \in Bрассмотреть все точки p_j \in C(p_i), и каждой пары (p_i,p_j)посчитать расстояние и сравнить с текущим наилучшим расстоянием.

На первый взгляд, это по-прежнему неоптимальный алгоритм: кажется, что размеры множеств C(p_i)будут порядка n, и требуемая асимптотика никак не получится. Однако, как это ни удивительно, можно доказать, что размер каждого из множеств C(p_i)есть величина O(1), т.е. не превосходит некоторой малой константы вне зависимости от самих точек. Доказательство этого факта приведено в следующем разделе.

Наконец, обратим внимание на сортировки, которых вышеописанный алгоритм содержит сразу две: сначала сортировка по парам (x,y), а затем сортировка элементов множества Bпо y. На самом деле, от обеих этих сортировок внутри рекурсивной функции можно избавиться (иначе бы мы не достигли оценки O(n)для стадии объединения, и общая асимптотика алгоритма получилась бы O(n \log^2 n)). От первой сортировки избавиться легко — достаточно предварительно, до запуска рекурсии, выполнить эту сортировку: ведь внутри рекурсии сами элементы не меняются, поэтому нет никакой необходимости выполнять сортировку заново. Со второй сортировкой чуть сложнее, выполнить её предварительно не получится. Зато, вспомнив **сортировку слиянием** (merge sort), которая тоже работает по принципу разделяй-и-властвуй, можно просто встроить эту сортировку в нашу рекурсию. Пусть рекурсия, принимая какое-то множество точек (как мы помним, упорядоченное по парам (x,y)) возвращает это же множество, но отсортированное уже по координате y. Для этого достаточно просто выполнить слияние (за O(n)) двух результатов, возвращённых рекурсивными вызовами. Тем самым получится отсортированное по yмножество.

Оценка ассимптотики

Чтобы показать, что вышеописанный алгоритм действительно выполняется за O(n \log n), нам осталось доказать следующий факт: |C(p_i)| = O(1).

Итак, пусть мы рассматриваем какую-то точку p_i; напомним, что множество C(p_i)— это множество точек, y-координата которых лежит в отрезке [y_i-h; y_i], а, кроме того, по координате xи сама точка p_i, и все точки множества C(p_i)лежат в полосе шириной 2h. Иными словами, рассматриваемые нами точки p_iи C(p_i)лежат в прямоугольнике размера 2h \times h.

Наша задача — оценить максимальное количество точек, которое может лежать в этом прямоугольнике 2h \times h; тем самым мы оценим и максимальный размер множества C(p_i). При этом при оценке надо не забывать, что могут встречаться повторяющиеся точки.

Вспомним, что hполучалось как минимум из двух результатов рекурсивных вызовов — от множеств A_1и A_2, причём A_1содержит точки слева от линии раздела и частично на ней, A_2— оставшиеся точки линии раздела и точки справа от неё. Для любой пары точек из A_1, равно как и из A_2, расстояние не может оказаться меньше h— иначе бы это означало некорректность работы рекурсивной функции.

Для оценки максимального количества точек в прямоугольнике 2h \times hразобьём его на два квадрата h \times h, к первому квадрату отнесём все точки C(p_i) \cap A_1, а ко второму — все остальные, т.е. C(p_i) \cap A_2. Из приведённых выше соображений следует, что в каждом из этих квадратов расстояние между любыми двумя точками не менее h.

Покажем, что в каждом квадрате **не более четырёх** точек. Например, это можно сделать следующим образом: разобьём квадрат на 4 подквадрата со сторонами h/2. Тогда в каждом из этих подквадратов не может быть больше одной точки (т.к. даже диагональ равна h / \sqrt{2}, что меньше h). Следовательно, во всём квадрате не может быть более 4 точек.

Итак, мы доказали, что в прямоугольнике 2h \times hне может быть больше 4 \cdot 2 = 8точек, а, следовательно, размер множества C(p_i)не может превосходить 7, что и требовалось доказать.



Код программы и тесты находятся в репозитоиии

Источники

http://www.e-maxx-ru.1gb.ru/algo/nearest\_points