Математические заметки

Том 116 выпуск 4 октябрь 2024

УДК 519.163

Приближенный поиск k-го порядкового расстояния в системе точек единичного квадрата

К. В. Каймаков, Д. С. Малышев

Для заданных $P = (p_1, \dots, p_n)$ – набора точек единичного квадрата и числа $1 \leqslant k \leqslant \binom{n}{2}$ в данной работе рассматривается задача поиска k-го порядкового расстояния между элементами P в ℓ_s-норме, где $s \in \{1, \infty\}$. Иными словами, рассматривается задача поиска такого минимального d_k, что выполнено $\sum_{i<j} \mathbb{I}(\|p_i, p_j\|_s \leqslant d_k) \geqslant k$, где $\mathbb{I}$ – индикаторная функция и $s \in \{1, \infty\}$. В настоящей работе для любого $\epsilon > 0$ предлагается ϵ-приближенный алгоритм со сложностью $O(n \log n \log(1/\epsilon))$ для вычисления d_k.

Библиография: 12 названий.

Ключевые слова: вычислительная геометрия, поиск порядковых расстояний, эффективный алгоритм.

DOI: https://doi.org/10.4213/mzm14205

1. Введение

Классическая в вычислительной геометрии *задача о паре ближайших точек* для заданного множества n точек в d-мерном пространстве состоит в том, чтобы найти пару ближайших точек по какой-нибудь норме. Для офлайновой или онлайновой (когда точки заранее неизвестны и поступают одна за другой) постановок этой задачи тривиальный алгоритм перебора всех пар точек имеет сложность $O(n^2)$ при фиксированном d. Оказывается, что для любых фиксированных d и $s \geqslant 1$ по норме ℓ_s офлайновая задача может быть решена за время $O(n \log n)$ [1]. В модели вычислений, когда атомарной операцией считается вычисление целой части числа, она может быть решена за время $O(n \log \log n)$ [2]. В этой же модели вычислений офлайновая задача о паре ближайших точек решается рандомизированным алгоритмом (см, например, [3]) сложности $O(n)$. Для ее онлайновой постановки имеется алгоритм сложности $O(\log n)$ на обработку каждой точки [4].

Естественным образом определяемая *задача о k парах ближайших точек* является обобщением задачи о паре ближайших точек. Ее офлайновая версия может

Исследование выполнено при поддержке Министерства науки и высшего образования Российской Федерации (госзадание № 075-03-2024-107), номер проекта FSMG-2024-0025.

быть решена, см. работы [5], [6], за время $O(n \log n + k)$ при любом фиксированном d. Некоторые приложения задачи о k парах ближайших точек (такие как проектирование СБИС или геоинформационные системы) и алгоритмы для ее решения представлены в [7]–[9].

В данной работе мы рассматриваем специальную подзадачу офлайновой задачи о k парах ближайших точек, а именно, задачу вычисления k-го порядкового расстояния по норме ℓ_s, $s \in \{1, \infty\}$. Более формально, для заданных $P = (p_1, \ldots, p_n)$ – набора точек единичного квадрата и числа $1 \leqslant k \leqslant \binom{n}{2}$ мы рассматриваем и решаем следующую задачу:

$$\min_{d_k} \sum_{i<j} \mathbb{I}(\|p_i, p_j\|_s \leqslant d_k) \geqslant k,$$

где $\mathbb{I}$ – индикаторная функция и $s \in \{1, \infty\}$. Напомним, что ℓ_1-*норма Минковского* между точками $p, p' \in \mathbb{R}^2$ определяется как

$$\|p, p'\|_1 = |p_x - p'_x| + |p_y - p'_y|,$$

а ℓ_∞-*норма Чебышева* между ними определяется как

$$\|p, p'\|_\infty = \max(|p_x - p'_x|, |p_y - p'_y|).$$

Наивный алгоритм (вычисления всевозможных попарных расстояний и поиска k-го по порядку значения среди них) имеет сложность $O(n^2)$. В настоящей работе для любого $\epsilon > 0$ предлагается ϵ-приближенный алгоритм со сложностью $O(n \log n \log(1/\epsilon))$ для вычисления значения d_k. Иными словами, он возвращает такие конкретные значения $d^\ell_{k,\epsilon}$ и $d^r_{k,\epsilon}$, что выполнено

$$d_k - \epsilon \leqslant d^\ell_{k,\epsilon} \leqslant d_k \leqslant d^r_{k,\epsilon} \leqslant d_k + \epsilon,$$

и работает за время $O(n \log n \log(1/\epsilon))$. При фиксированном ϵ и $n \log n = o(k)$ наш алгоритм работает быстрее, чем известные алгоритмы для офлайновой задачи о k парах ближайших точек.

2. Предварительное алгоритмическое обеспечение

В этом разделе работы описываются сведéние линейного времени задачи для ℓ_1-нормы к той же задаче для ℓ_∞-нормы, а также некоторые алгоритмические техники, которые впоследствии будут использоваться при построении эффективного алгоритма.

2.1. Линейное сведéние задачи для ℓ_1-нормы к задаче для ℓ_∞-нормы. Напомним, что ℓ_1-*расстояние* между точками a и b на плоскости определяется как

$$\|a, b\|_1 = |a_x - b_x| + |a_y - b_y|,$$

что можно переписать следующим образом:

$$\|a, b\|_1 = \begin{cases} a_x + a_y - b_x - b_y, & a_x > b_x \quad \text{и} \quad a_y > b_y, \\ -a_x + a_y + b_x - b_y, & a_x < b_x \quad \text{и} \quad a_y > b_y, \\ a_x - a_y - b_x + b_y, & a_x > b_x \quad \text{и} \quad a_y < b_y, \\ -a_x - a_y + b_x + b_y, & a_x < b_x \quad \text{и} \quad a_y < b_y, \end{cases}$$

или

$$\|a,b\|_1 = \max\bigl(|(a_x - b_x) + (a_y - b_y)|, |(b_x - a_x) + (a_y - b_y)|\bigr).$$

Рассмотрим преобразование поворотной гомотетии – поворот на $\pi/4$ с последующим растяжением на $\sqrt{2}$:

$$p' = Ap, \qquad A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} p_x + p_y \\ -p_x + p_y \end{pmatrix},$$
$$p'_x = p_x + p_y, \qquad p'_y = -p_x + p_y.$$

После преобразования точка a перейдет в точку a', а точка b перейдет в точку b'. Нетрудно видеть, что выполнены равенства

$$\|a,b\|_1 = \max(|a'_x - b'_x|, |a'_y - b'_y|) = \|a',b'\|_\infty.$$

Отсюда следует, что задача для ℓ_1-нормы сводится за линейное время к той же задаче для ℓ_∞-нормы. Поэтому всюду далее мы будем рассматривать задачу только для ℓ_∞-нормы. Линиями уровня этой нормы являются квадраты.

2.2. Бинарный поиск по мощности покрытия квадратами. Для набора точек P определим на $[0,1]$ функцию

$$f_P(t) = \sum_{p_i,p_j \in P,\, i \neq j} \mathbb{I}(\|p_i,p_j\|_\infty \leqslant t).$$

Иными словами, $f_P(t)$ – это удвоенное количество точек из P, покрываемых квадратами размера $2t$ с центрами в элементах P, не считая центров самих квадратов. Нас интересует поиск такого минимального d_k, что выполнено неравенство $f_P(d_k) \geqslant 2k$.

Если имеется алгоритм, который вычисляет значение $f_P(\,\cdot\,)$ в любой точке отрезка $[0,1]$ за время $O(T)$, то ϵ-приближенный поиск точки d_k может быть организован за время $O(T\log(1/\epsilon))$ бинарным поиском. На каждой его итерации выполняется дихотомическое деление текущего отрезка, содержащего d_k, сравнение значения $f_P(\,\cdot\,)$ в его середине m с $2k$, выбор первой половины отрезка, если $f_P(m) > 2k$, или второй половины, если $f_P(m) \leqslant 2k$. Процесс повторяется до тех пор, пока текущий отрезок не станет достаточно маленьким.

Далее будет показано, что значение функции $f_P(\,\cdot\,)$ в каждой точке отрезка $[0,1]$ вычисляется за время $O(n\log n)$, т.е. что $T = O(n\log n)$. Поэтому ϵ-приближенный алгоритм вычисления d_k имеет сложность $O(n\log n\log(1/\epsilon))$.

2.3. Метод сканирующей прямой. Метод сканирующей прямой – это способ обработки точек на плоскости, состоящий в их упорядочивании по координатной прямой (или каких-нибудь событий, ассоциированных с точками) и последующем проходе по ним. Иными словами, рассматриваются положения некоторой прямой, параллельной одной из осей координат, а также множество событий, порождаемых каждым таким положением. Например, в известном алгоритме Бентли–Оттмана [10] для поиска всех пар пересекающихся отрезков среди заданных отрезков на плоскости такими событиями являются “начало отрезка”, “конец отрезка” и “пересечение отрезков”. Сложность алгоритма Бентли–Оттмана для n отрезков есть $O((n+k)\log n)$, где k – количество пересечений, что при $k = o(n^2/\log n)$ лучше квадратичного алгоритма перебора всех пар отрезков.

Для нашей задачи точки из $P^* = P_{+t} \cup P \cup P_{-t}$, где $t \in [0,1]$ и

$$P_{\pm t} = (p_1 \pm t, p_2 \pm t, \ldots, p_n \pm t),$$

упорядочиваются по неубыванию значения их абсцисс, причем для одинакового значения абсциссы сначала идут точки из P_{+t}, затем точки из P, а затем точки из P_{-t}. Тем самым, каждое из трех типов событий "открытие квадрата", "центр квадрата" и "закрытие квадрата" имеет свой уникальный номер от 1 до $3n$ и, наоборот, каждому такому номеру соответствует свое событие относительно некоторого квадрата с центром в точке из P.

2.4. Эффективный запрос суммы на отрезке изменяющегося массива. Предположим, что задан некоторый числовой массив $A[1,\ldots,n]$. Требуется так предобработать этот массив, чтобы по результату предобработки T за время $O(\log n)$:

1) для каждых заранее неизвестных $1 \leqslant i \leqslant n$ и $\mathrm{val} \in \mathbb{R}$ перестраивать T после присваивания $A[i] := \mathrm{val}$;
2) для каждого заранее неизвестного запроса $1 \leqslant i \leqslant j \leqslant n$ вычислять значение суммы
$$A[i] + A[i+1] + \cdots + A[j].$$

Для решения поставленной задачи можно использовать дерево отрезков [11] с дополнительной памятью/временем предобработки $O(n)$ или дерево Фенвика [12] с дополнительной памятью $O(n)$ и временем предобработки $O(n \log n)$.

3. Алгоритм

Основная идея нашего алгоритма похожа на основную идею алгоритма Бентли–Оттмана. Мы используем метод сканирующей прямой по оси абсцисс и поддерживаем события "открытие квадрата"/"закрытие квадрата" с помощью массива, индексы которого соответствуют точкам из P. Мощность покрытия квадратами вычисляется при наступлении событий "центр квадрата" с помощью бинарного поиска соответствующего интервала в индексах массива и отклика дерева отрезков/Фенвика запросом суммы элементов массива с индексами на этом интервале.

В целом наш алгоритм сложности $O(n \log n \log(1/\epsilon))$ выглядит следующим образом.

Шаг 1. Если рассматривается ℓ_1-норма, то перейти к ℓ_∞-норме путем поворота направо на $\pi/4$ с последующим растяжением на $\sqrt{2}$:

$$\forall\, p_i \in P \qquad \text{вычислить} \quad p_i := \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix} p_i.$$

Шаг 2. Положить $d^\ell_{k,\epsilon} := 0, d^r_{k,\epsilon} := 1$. Пока выполнено $d^r_{k,\epsilon} - d^\ell_{k,\epsilon} > 2\epsilon$ применять следующие действия:

Шаг 2.1. Для текущей оценки $[d^\ell_{k,\epsilon}, d^r_{k,\epsilon}]$ величины d_k найти множество

$$P^* := P_{+t} \cup P \cup P_{-t},$$

где $t := (d^r_{k,\epsilon} + d^\ell_{k,\epsilon})/2$. Создать массив A_P из нулей, индексы которого соответствуют значениям ординат точек из P, упорядоченных по неубыванию. По массиву A_P

построить соответствующее дерево отрезков/дерево Фенвика T_P. Положить счетчик k' мощности покрытия квадратами равным нулю, т.е. $k' := 0$.

Шаг 2.2. Упорядочить P^* по неубыванию значений абсцисс и сформировать массив из $3n$ соответствующих событий. Каждое из них хранит соответствующую точку из P и информацию об одном из трех типов событий ("открытие квадрата" или "центр квадрата" или "закрытие квадрата").

Шаг 2.3. Итерация по P^*:

Шаг 2.3.1. Для каждого события "открытие квадрата" с центром в точке p выполнить присваивание $A_P[p.y] := A_P[p.y] + 1$ и перестроить дерево T_P.

Шаг 2.3.2. Для каждого события "закрытие квадрата" с центром в точке p выполнить присваивание $A_P[p.y] := A_P[p.y] - 1$ и перестроить дерево T_P.

Шаг 2.3.3. Для каждого события "центр квадрата" найти бинарным поиском по индексам массива A_P такой минимальный индекс ℓ_p, что $p_y - t \leqslant \ell_p$, и такой максимальный индекс r_p, что $r_p \leqslant p_y + t$. Затем сделать запрос по дереву T_P: вернуть S_p – сумму элементов A_P с ℓ_p-го по r_p-й и выполнить $k' := k' + S_p - 1$.

Шаг 2.4. Если $k' > 2k$, то положить $d^\ell_{k,\epsilon} := d^\ell_{k,\epsilon}$ и $d^r_{k,\epsilon} := (d^\ell_{k,\epsilon} + d^r_{k,\epsilon})/2$. В противном случае положить $d^\ell_{k,\epsilon} := (d^\ell_{k,\epsilon} + d^r_{k,\epsilon})/2$ и $d^r_{k,\epsilon} := d^r_{k,\epsilon}$.

Шаг 3. Вернуть $d^\ell_{k,\epsilon}$ и $d^r_{k,\epsilon}$.

Оценим вычислительную сложность каждого из шагов. Ясно, что шаг 1 выполняется за время $O(n)$, а шаг 3 выполняется за время $O(1)$. Количество итераций бинарного поиска шага 2 можно оценить сверху как $\lceil \log_2(1/(2\epsilon)) \rceil$. Шаг 2.1 имеет сложность $O(n)$, а шаг 2.2 имеет сложность $O(n \log n)$. Каждый из шагов 2.3.1–2.3.3 имеет сложность $O(\log n)$, следовательно, шаг 2.3 имеет сложность $O(n \log n)$. Шаг 2.4 имеет сложность $O(1)$. Таким образом, итоговая сложность нашего алгоритма есть $O(n \log n \log(1/\epsilon))$.

4. Заключение

В данной работе рассматривается и решается задача поиска k-го порядкового расстояния между элементами множества P из n точек единичного квадрата в ℓ_s-норме, где $s \in \{1, \infty\}$. Для любого $\epsilon > 0$ для решения этой задачи в настоящей работе предлагается ϵ-приближенный алгоритм сложности $O(n \log n \log(1/\epsilon))$.

Для случаев $P \subset \mathbb{R}^2$ и $P \subset \mathbb{Z}^2$ наш алгоритм дает, соответственно, ϵ-приближенное и точное решения со сложностями $O(n \log n \log(D_P/\epsilon))$ и $O(n \log n \log D_P)$. Через D_P обозначен диаметр (=наибольшее из попарных расстояний) множества P в норме ℓ_∞, причем D_P вычисляется за $O(n)$ время поиском минимального и максимального значений по каждой из координат. Улучшение предложенного нами решения является интересной исследовательской задачей для возможной будущей работы.

СПИСОК ЦИТИРОВАННОЙ ЛИТЕРАТУРЫ

[1] K. L. Clarkson, "Fast algorithms for the all nearest neighbors problem", *Proceedings of 24th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Washington, 1983, 226–232.

[2] S. Fortune, J. E. Hopcroft, "A note on Rabin's nearest-neighbor algorithm", *Information Processing Letters*, **8**:1 (1979), 20–23.

[3] S. Khuller, Y. Matias, "A simple randomized sieve algorithm for the closest-pair problem", *Information and Computation*, **118**:1 (1995), 34–37.

[4] S. N. Bespamyatnikh, "An optimal algorithm for closest-pair maintenance", *Discrete and Computational Geometry*, **19** (1998), 175–195.

[5] H.-P. Lenhof, M. Smid, "Enumerating the k closest pairs optimally", *Proceedings of* 33*rd Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, Washington, 1992, 380–386.

[6] H.-P. Lenhof, M. Smid, "Sequential and parallel algorithms for the k closest pairs problem", *International Journal of Computational Geometry and Applications*, **5**:3 (1995), 273–285.

[7] H. Kurasawa, A. Takasu, J. Adachi, "Finding the k-closest pairs in metric spaces", *Proceedings of the* 1*st Workshop on New Trends in Similarity Search*, Association for Computing Machinery, New York, 2011, 8–13.

[8] A. Corall, A. Manolopoulos, Y. Theodoridis, M. Vassilakopoulos, "Closest pair queries in spatial databases", *ACM SIGMOD Record*, **29**:2 (2000), 189–200.

[9] M. Aumüller, M. Ceccarello, "Solving k-closest pairs in high-dimensional data", *Similarity Search and Applications*, Springer, Berlin, 2023, 200–214.

[10] J. L. Bentley, T. A. Ottman, "Algorithms for reporting and counting geometric intersections", *IEEE Transactions on Computers*, **C-28**:9 (1979), 643–647.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, The MIT Press, Cambridge, Massachusetts London, England, 2022.

[12] P. M. Fenwick, "A new data structure for cumulative frequency tables", *Software*: *Practice and Experience*, **24**:3 (1994), 327–336.

К. В. Каймаков
Национальный исследовательский университет
"Высшая школа экономики", г. Москва;
ООО "Коулмэн Тех", г. Москва
E-mail: kaymakov.kirill@huawei-partners.com

Д. С. Малышев
Национальный исследовательский университет
"Высшая школа экономики"
(Нижегородский филиал);
Московский физико-технический институт
(национальный исследовательский университет),
Московская область, г. Долгопрудный
E-mail: dsmalyshev@rambler.ru

Поступило
27.11.2023
Принято к публикации
17.04.2024