

Железная няня

Будем делать корневую по запросам. Выберем константу K и будем отвечать сразу на блок запросов размера K .

Ближайшая точка к текущей будет или в наборе, или среди добавленных в блоке запросов. Второе множество точек маленькое, размера $O(k)$, там точки можно просто перебрать. С изначальным набором сложнее. К нему также будут поступать запросы удаления, но не будет запросов добавления.

Поделим плоскости на квадраты со стороной B . Точка с координатами (x, y) будет принадлежать квадрату $(\lfloor \frac{x}{B} \rfloor, \lfloor \frac{y}{B} \rfloor)$. Определим квадрат, в котором находится текущая точка, и будем расширять «радиус» квадратов, в которых будет ближайшая точка.

Утверждение: если между квадратами Манхэттенское расстояние r , то расстояние между точками в них лежит в диапазоне $[\max(0, (r - 2)B + 2), (r + 2)B - 2]$. Это очевидно, достаточно лишь посмотреть на расстояние между точками в ближайших и в самых далёких углах.

Таким образом, если мы найдём минимальный радиус r_{min} , на котором есть хотя бы один непустой квадрат, нам нужно будет рассмотреть только квадраты с радиусами от r_{min} до $r_{min} + 3$. Получается, нам нужно рассмотреть не более чем $4 \cdot 4 \cdot B = 16B$ квадратов.

Сначала разберёмся с тем, как искать r_{min} , потом с тем, как в квадрате искать ближайшую точку.

На эту задачу можно посмотреть, как на поиск ближайшей точки, если координаты не больше чем $\lceil \frac{A}{B} \rceil$. Обозначим эту границу как S . Пусть точек не больше S , тогда переберём все точки. Пусть точек больше S . Посчитаем префиксные суммы по всем диагоналям. Затем перебираем радиус и проверяем, есть ли точка. Если нужно удалить точку, пересчитаем префсуммы в двух диагоналях. Таким образом умеем $O(S)$ делать запрос и удаление. В случае с поиском r_{min} получаем $O(\lceil \frac{A}{B} \rceil)$.

Теперь мы нашли r_{min} . Переберём все квадраты на расстоянии от r_{min} до $r_{min} + 3$. Есть три случая с квадратами:

1. Координаты квадратов не совпадают ни по x , ни по y . Тогда оба модуля в формуле расстояния раскрываются одинаково для всех точек, и нам нужно минимизировать одно из 4 выражений: $x + y, x - y, -x + y, -x - y$. Их можно явно хранить в векторе. Таким образом за $O(1)$ мы отвечаем на запрос минимума. А квадратов будет $O(B)$.

2. У квадрата совпадает только одна из координат. Таких квадратов будет 16. Однако одна из координат в формуле расстояния имеет фиксированный знак. Перебираем вторую и выбираем $\min / \max$ по второй. $O(B)$.

3. У квадрата совпали обе координаты (иными словами, это квадрат, в котором точка находится). Эта задача аналогична поиску r_{min} , но координаты до B . Таким образом, эта часть решения работает за $O(B)$.

Однако нам ещё поступают запросы удаления точек. Мы можем просто пометить, что точка удалена, тогда при запросе минимума по некоторому выражению мы можем просто удалять верхний элемент из вектора, пока он помечен как удалённый. Это просто отложенные операции, так что время работы можно оценивать так, как будто бы мы удаляли элемент из вектора размера $O(K)$. То есть удаление точки мы делаем за $O(K)$.

Однако векторы, в которых мы делаем бинпоиски (задача про поиск ближайшей точки в небольшом квадрате), надо поддерживать в явном виде, их размеры $O(B)$, для них удаление работает за $O(B)$.

Таким образом получаем время работы $O(q(\lceil \frac{A}{B} \rceil + B + K))$ на все запросы.

Реконструкция можно делать за линейное время от количества точек, несмотря на то что для квадрата нужно много отсортированных векторов. Их все можно поддерживать, сортируя блок новых точек и сливая его с текущим порядком.

Время на реконструкцию $O(q)$, хоть и очень долгий. Всего реконструкций будет $O(\frac{q}{K})$. Важно, что префсуммы мы не перестраиваем, а добавляем в квадрат точки по одной.

$K = \sqrt{q}, B = \sqrt{A}$. Итоговое время работы составит $O(q(\sqrt{A} + \sqrt{K}))$.