Спрятавшееся множество

Шатунов Леонид, Никитин Артем

14 апреля 2025 г.

Задача. Назовем множество точек $\mathcal{B} := \{B_1, \ldots, B_n\}$ спрятавшимся относительно точки A, если все $n+1$ точки находятся в общем положении и можно соединить точки множества $\mathcal{B}$ некоторой ломаной так, чтобы каждый отрезок вида AB_i пересекал ломаную по некоторому звену. На вход подаются координаты точки A и точек множества $\mathcal{B}$. Проверьте, является ли множество $\mathcal{B}$ спрятавшимся относительно A.

Определение. Направленным углом между двумя лучами OX и OY будем называть нименьший угол, на который нужно повернуть луч OX против часовой стрелки вокруг O, чтобы получить луч OY. Будем обозначать его $\angle_r(OX, OY)$.

Решение. Поместим точку A в начало координат.

Шаг 1: отсортируем точки множества $\mathcal{B}$ по возрастанию $\angle_r(AX, AB)$, где $X = (1, 0)$ и $B \in \mathcal{B}$. Не умаляя общности, получим список $b\_sort := (B_1, \ldots, B_n)$.

Шаг 2: найдем $\triangle B_i B_j B_k$ такой, что внутри него лежит точка A и нет точек из $\mathcal{B}$.

Алгоритм: Зациклим список b_sort. Заведем список номеров вершин $hide\_array := (1, 2)$. Пусть на каком-то шаге алгоритма $hide\_array = (i_1, i_2, \ldots, i_k)$. Положим $i_{k+1} := i_k + 1$. Смотрим на ребро i_k, i_{k+1}. Если оказалось, что четырехугольник $AB_{i_{k-1}}B_{i_k}B_{i_{k+1}}$ — невыпуклый и несамопересекающийся, то переходим к следующему шагу (т.е. рассматриваем $i_{k+2} := i_{k+1} + 1$). Если оказалось, что четырехугольник $AB_{i_{k-1}}B_{i_k}B_{i_{k+1}}$ — выпуклый несамопересекающийся, то выкидываем из нашего списка индекс i_k. Проверяем на выпуклость следующий четырехугольник $AB_{i_{k-2}}B_{i_{k-1}}B_{i_{k+1}}$ и так далее, пока можем выкидывать индексы. Как только более ничего выкинуть не можем, переходим к следующей точке. Продолжаем так, пока не дойдем снова до точки с индексом 1 и не проведем для нее процедуру выкидывания индексов. Если получили в какой-то момент самопересекающийся четырехугольник при попытке добавить новую точку, то множество $\mathcal{B}$ не является спрятавшимся, завершаем алгоритм. Таким образом, каждая точка у нас войдет в список ровно один раз за исключением точки 1 и выйдет не более одного раза, тем самым мы сделаем порядка n операций. Заметим, что точка 1 у нас будет начинать и замыкать наш список.

Предположим, что точка A лежит внутри выпауклой оболочки множества $\mathcal{B}$. Тогда, так как точка A всегда находилась по одну сторону от прямых $i_{k-1}i_k$ (ребер ломаной, построенной на вершинах списка, на каждом шаге в силу прохода по отсортированному списку направленных углов из точки A и принадлежания выпуклой оболочке), то в конце A так же будет лежать внутри ломаной. Предположим, что в списке оказалось более 4 точек. Тогда найдется выпуклый многоугольник, что будет противоречить выполнению шагаи алгоритма (нетрудно увидеть, нарисовава картинку и посчитав сумму углов многоугольника). Следовательно, точек будет не более 4, при этом первая повторится дважды. Если точек меньше 4, то мы либо не выполним условие принаджлежание A внутренности ломаной, либо нарушим шаг построения, положив точку A на какой-то отрезок, соединяющий точки из списка. Получаем треугольник $B_{i_1}B_{i_2}B_{i_3}$ такой, что внутри него нет точек множества $\mathcal{B}$, но точка A находится внутри него.

Если же точка A не принадлежит выпуклой оболочке, то на каком-то шаге точка A окажется по другую сторону от прямой $i_{k-1}i_k$ и мы получим самопересекающийся четырехугольник. И так же из предположения следует, что точку A можно отделить от $\mathcal{B}$, что гарантирует отсутствие свойства спрятанности.

Шаг 3: После нахождения треугольника из Шага 2 необходимо и достаточно проверить, что есть три пары чисел $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, что отрезки $B_{x_1}B_{y_1}, B_{x_2}B_{y_2}, B_{x_3}B_{y_3}$ пересекают отрезки с выколотыми концами $(AB_i), (AB_j), (AB_k)$ соответственно.

Алгоритм: для каждой точки $B_x \neq B_i$ найдем бинарным поиском в списке c границы подмассива из точек $B_y \neq B_i$ таких, что луч AB_i пересекает отрезок B_xB_y и $\angle_r(AB_x, AB_y) < 180°$. Сохраним эти индексы границ в массив $borders$.

Корректность: нужно найти $y, z \neq i$ такие, что $\angle_r(B_xAB_y) < \angle_r(B_xAB_i) < \angle_r(B_xAB_z) < 180°$. При этом $\angle_r(B_xAB_z)$ должен быть максимальным из таких углов, а $\angle_r(B_xAB_y)$ минимальным.

Теперь отсортируем точки множества $\mathcal{B}_i := \mathcal{B} \cup A \setminus B_i$ по возрастанию $\angle_r(B_iX, B_iB)$, где $X = B_i + (0, 1)$ и $B \in \mathcal{B}_i$. Получим список $b\_sort\_i = (B_{1,i_1}, \ldots, B_{n,i_n})$, где первый индекс совпадает с порядком сортировки, а второй индекс означает $B_x = B_{y,x}$. Аналогично алгоритму выше получим массив границ $borders\_i(z)$, где для каждой точки $B_{z,x} \neq A$ из b_sort_i сохраним границы (по y) подмассива из точек $B_{y,t} \neq A$ таких, что луч B_iA пересекает отрезок B_xB_t и $\angle_r(B_iB_x, B_iB_t) < 180°$.

Сделаем пересечение элементов полученных списков попарно для точек из $\mathcal{B}$ — получим границы подмассивов для которых пересекаются именно отрезки. Если есть хотя бы один непустой, то переходим к следующей точке B_j и B_k. Иначе $\mathcal{B}$ спрятать невозможно.

Сложность. Дополнительная память $O(n)$. Подсчет времени:

Шаг 1: $O(n \log n)$ на сортировку и подсчет углов.

Шаг 2: $O(n)$ по доказанному выше.

Шаг 3: $O(n \log n)$ на бинарный поиск и сортировку точек.

Итог: $O(n \log n)$ времени и $O(n)$ памяти.