«Титаник»

Автор задачи и разработчик: Александр Заварин

Для начала поймем, что, так как крюк может выстреливать только перпендикулярно движению лодки, то существует всего 1 точка на прямой, проведенной через точки A и B. Это значит, что мы можем вычислить отрезок на прямой, на котором крюк будет заблокирован, если мы зацепим шлюпку в начальной точке. Для того чтобы вычислить точку, в которой нужно зацепить шлюпку, нужно провести перпендикуляр к прямой AB, основание перпендикуляра и будет нужной точкой. Чтобы вычислить точку, где крюк освободится, необходимо из начальной точки провести вектор в направлении движения спасательной лодки с длиной, равной длине перпендикуляра. Будем называть для шлюпки начальную точку этого отрезка — *точкой захвата*, а конечную точку — *точкой спасения*.

Рассмотрим 1 подгруппу, в ней есть ограничение на координату y у точек A и B, она у них равна 0. Б. о. о. пусть спасательная лодка плывет слева направо (иначе можно просто отзеркалить координаты относительно прямой $x = 0$). Тогда, если i-я шлюпка находится в точке (x_i, y_i), то точка захвата у нее имеет координаты $(x_i, 0)$, а точка спасения — $(x_i + y_i, 0)$, т.к. расстояние между точкой захвата и шлюпкой было ровно y_i. Мы можем отбросить все шлюпки, у которых x-координата точки захвата меньше a_x или больше b_x, потому что одни мы не сможем захватить крюком, а другие не успеем притянуть к себе — игра закончится. Теперь отсортируем все шлюпки по x-координате точки захвата, чтобы воспользоваться динамическим программированием. Посчитаем для каждой шлюпки i величину dp_i — максимальное число шлюпок среди тех, у которых номер в отсортированном порядке меньше i, и при этом спасти i-ю шлюпку. Теперь просто одним циклом будем перебирать номер лодки в отсортированном порядке, пусть сейчас это i, а теперь переберем все $j < i$, для j-й лодки проверим, что ее x-координата точки спасения не больше x-координаты точки захвата нашей i-й лодки, иначе мы не сможем спасти ее. Вычислим $dp_i = \max_{j<i}(dp_j) + 1$ среди подходящих j. Это работает, потому что для dp_i среди спасенных лодок с номерами $j < i$ есть лодка с максимальным номером j_{max}, и для нее, если убрать спасенную i-ю лодку, $dp_{j_{max}} = dp_i - 1$ по определению dp_i. Тогда, так как ответ тоже содержит в себе некоторую лодку i_{max} с наибольшим номером, то ответ — это $dp_{i_{max}}$, т.е. его можно вычислить как $\max_i(dp_i)$.

Перейдем ко 2 подгруппе, теперь отсортировать лодки так удобно не выйдет, однако это можно сделать, используя теперь в качестве параметра сортировки расстояние между точкой A и точкой захвата у шлюпки. Это можно вычислить, используя формулы построения перпендикуляра к прямой и вычисления расстояний между точками, однако авторское решение использует векторные и скалярные произведения для подсчета этого расстояния. Пусть шлюпка находится в точке P, тогда расстояние от A до точки захвата можно вычислить как $|AP| \cdot \cos\alpha$, где α — это угол между AP и AB. Это можно выразить через скалярное произведение векторов как $|AP| \cdot \cos\alpha = \frac{AB \cdot AP}{|AB|}$. Длина перпендикуляра рассчитывается подобным образом, но через векторное произведение: $|AP| \cdot \sin\alpha = \frac{|AB \times AP|}{|AB|}$, важно не забыть взять модуль от векторного произведения: в первом случае модуль не использовался, чтобы шлюпки вне доступности спасательной лодки имели отрицательное расстояние от точки A для удобства. Теперь, используя эти две величины, мы можем посчитать расстояние от точки A до точки спасения шлюпки как их сумму. Заметим, что у расстояний общий знаменатель — длина отрезка AB. Давайте тогда просто домножим все величины на это число, от этого порядок лодок и их сортировка не изменятся, но зато мы сможем перейти к вычислениям в целых числах. Теперь, имея две величины для каждой шлюпки — расстояние от точки A до точки захвата и точки спасения, а также используя идею с динамическим программированием из 1 подгруппы, можем сдать эту.

Перейдем к 3 подгруппе, и теперь подумаем, как улучшить подсчет величины dp_i. Для этого воспользуемся *scanline*-идеей, у нас будет 2 типа событий: "мы прибыли в точку захвата некоторой шлюпки" и "мы прибыли в точку спасения некоторой шлюпки". Мы отсортируем все события по расстоянию от точки A до них, теперь будем идти по ним и хранить ответ — максимальное число лодок, которое мы можем спасти, если мы прошли первые i событий, обозначим его $score$. Если мы встречаем событие 1-го типа для j-й шлюпки, то мы вычисляем $dp_j = score + 1$ по определению.

Если мы встречаем событие 2-го типа для j-й шлюпки, это значит, что, если бы мы эту шлюпку спасали, то крюк бы освободился, а значит, мы можем теперь использовать dp_j для обновления $score$: $score = \max(score, dp_j)$. Важно, что, если несколько событий находятся в одной точке, то сначала нужно обработать события 2-го типа, а потом уже первого типа. После обработки всех событий $score$ и будет нашим ответом.

Подгруппы 4 и 5 используют объединенные идеи подгрупп 2 и 3.