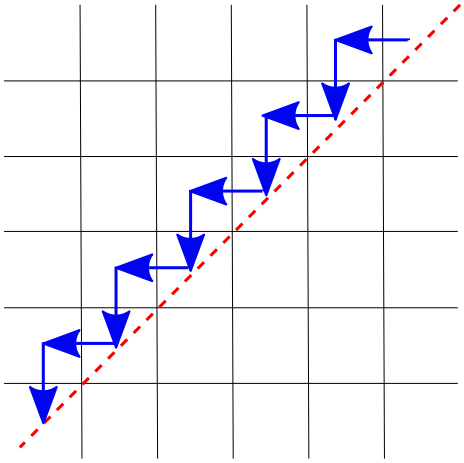
Разбор задачи «Поиски Трезубца»

Заметим, что в задаче требовалось обойти отмеченные клетки в порядке следования побочных диагоналей матрицы.

Для того, чтобы получить **57 баллов** по этой задаче, можно было отсортировать все комнаты с «X» по возрастанию суммы $i + j$, где i – номер ряда, а j — номер столбца данной комнаты. Далее с помощью «L» и «R» перемещаемся по первому ряду, чтобы выбрать нужный столбец, после чего приходим в нужную комнату, а затем возвращаемся на первый ряд. Это можно было сделать, например, n раз добавив в ответ «D», так как поле циклично.

Для **полного** решения задачи нужно понять, что на самом деле не важно, где подсказки, достаточно просто обойти все клетки в порядке следования диагоналей. Это можно сделать конструктивно, возвращаясь на предыдущую диагональ.

На рисунке пример последовательности ходов, обходящих диагональ, отмеченную красным пунктиром.



Разбор задачи «Атакующие пары»

Для решения первой подзадачи достаточно просто перебрать все пары чисел (a_i, a_j) и проверить, что $1 \leqslant |i - j| \leqslant k$, а также что $l \leqslant |a_i - a_j| \leqslant r$. Такое решение имеет асимптотику $O(n^2)$.

Для решения второй подзадачи достаточно перебирать только пары индексов (i, j), удовлетворяющие условию $1 \leqslant |i-j| \leqslant k$. Для этого достаточно перебрать $1 \leqslant i \leqslant n$ и $\max(0, i-k) \leqslant j \leqslant i-1$. Такое решение имеет асимптотику $O(n \cdot k)$.

Третья подзадача может быть решена отдельно от предыдущих двух: для ее решения нам нужно найти хотя бы одну пару чисел $a[i] = a[j]$, таких, что $1 \leqslant |i-j| \leqslant k$. Для этого будем идти по массиву указателем $1 \leqslant i \leqslant n$, а также поддерживать множество последних k чисел $a[i-k..i-1]$ (например, используя структуру `std::set` в C++ или `Set` в Java). Для каждого числа a_i будем искать число a_i в текущем множестве к ответу — таким образом, пройдя по всем a_i мы найдем ответ на задачу.

Для решения четвертой подзадачи для каждого a_i в множестве последних k чисел $a[i-k..i-1]$ нужно проверить наличие a_j такого, что $|a_i - a_j| \leqslant r$, то есть что $a_i - r \leqslant a_j \leqslant a_i + r$. Это можно сделать например с помощью метода `std::set::lower_bound`.

Для решения последней подзадачи достаточно добавить еще одну проверку с помощью `std::set::lower_bound`, чтобы проверить, что $l \leqslant |a_i - a_j|$.

Разбор задачи «Минное поле»

Будем поддерживать для каждой клетки ближайшую клетку, в которой еще не выкопана бомба, в каждом из четырех направлений. Когда поступает запрос на нахождение клетки с бомбой, пройдемся по этим переходам в нужном направлении, пока не дойдем до клетки с бомбой или до границы поля. После этого, во всех клетках, по которым мы прошли, обновим ссылку, теперь они будут указывать на клетку, которая сейчас является ответом. Таким образом, получился аналог системы непересекающихся множеств с эвристикой переподвешиваний. Значит, амортизированное время работы — $O(\log n)$ на запрос.

В первой подгруппе можно было поддерживать клетки с бомбами, и при каждом запросе перебирать клетки в нужном направлении, пока не встретится клетка с бомбой. Асимптотика решения — $O(q \cdot (n + m))$.

Во второй подгруппе можно было обработать все запросы на выкапывание бомб, а потом с помощью метода динамического программирования найти ближайшую клетку с бомбой для каждой клетки в каждом направлении. Рассмотрим, как найти ближайшую клетку с бомбой для каждой клетки в направлении вверх. Будем перебирать клетки по строкам от верхней до нижней. Для очередной клетки, если соседняя сверху клетка содержит бомбу, она является ближайшей, содержащей бомбу. Иначе, ответ для текущей клетки совпадает с ответом для соседней сверху клетки. Направления вниз, влево и вправо решаются аналогично. Асимптотика решения — $O(n \cdot m + q)$.

Разбор задачи «Оптимальное перестроение»

Решение 1. Наивное. Чтобы решить задачу при маленьких n, достаточно было перебрать значение x, после этого построить новый ряд из рыб по правилам, указанным в условии, после чего для каждого из получившихся рядов перебрать все пары рыб в нем и проверить, не образуют ли они пару, где более сильная рыба стоит раньше менее сильной.

Асимптотика времени работы данного решения — $\mathcal{O}(n^3)$, этого достаточно для получения 40 баллов.

Ключевое замечание. Пусть в исходной перестановке была пара позиций i, j, для которой выполнялось $i < j$ и $a_i > a_j$. Пусть также Аквамен отдал команду перестроения и назвал число x. Заметим, что эта пара рыб будет стоять относительно друг друга **не** в таком же порядке после перестроения тогда и только тогда, когда выполняется неравенство $a_i \geqslant x \geqslant a_j$ (так как в этом случае первая рыба с номером j встанет до рыбы x, а рыба с номером i — после).

Таким образом, чтобы посчитать ответ для каждого x, надо посчитать количество пар номеров i, j, для которых верно, что $i < j$ и либо $x > a_i > a_j$, либо $a_i > a_j > x$. Пусть a^{-1} — обратная перестановка к перестановке a, то есть такая последовательность из n различных чисел от 1 до n, для которой верно $a^{-1}_{a_i} = i$. Теперь введем новые обозначения: $i' = a_i$, $j' = a_j$. Тогда $i = a^{-1}_{i'}$, $j = a^{-1}_{j'}$. Таким образом, нам надо посчитать количество пар i', j', где $a^{-1}_{i'} < a^{-1}_{j'}$, а также либо $i' > j' > x$, либо $x > i' > j'$. Заметим, что это количество инверсий на префиксе до числа x и на суффиксе после числа x в перестановке a^{-1}. Инверсией мы называем пару индексов x, y, где $x < y$ и $a^{-1}_x > a^{-1}_y$.

Теперь в перестановке a^{-1} посчитаем для каждого элемента количество инверсий, которые он образует с элементами после него и до него. Таким образом, для каждого x ответ — это сумма этих чисел на префиксе до числа x и после числа x.

Существует множество способов считать для каждого элемента количество инверсий, которые он образует. Если делать это наивно, с помощью цикла, получается решение с асимптотикой времени работы $\mathcal{O}(n^2)$, которое получает 60 баллов. Если применить одну из структур данных с запросами на отрезке, таких как корневая декомпозиция, дерево отрезков или дерево Фенвика, можно было получить решение с асимптотикой времени работы до $\mathcal{O}(n \log n)$, и получить от 80 до 100 баллов.