Нечестная игра

Автор задачи и разработчик: Даниил Орешников

Заметим несколько важных критериев:

1. чтобы фишка могла находиться на удаляемой клетке $(i_{\text{cur}}, j_{\text{cur}})$, должна быть возможность за $\sum\limits_{t \leqslant \text{cur}} k_t$ шагов добраться из (r, c) до $(i_{\text{cur}}, j_{\text{cur}})$;

2. если по пути мы не оказываемся в клетке без неудаленных соседей, то каждый шаг меняет четность суммы номера строки и столбца, поэтому четность клетки $(i_{\text{cur}}, j_{\text{cur}})$ должна быть соответствующей;

3. если $\sum\limits_{t \leqslant \text{cur}} k_t$ строго больше длины какого-то пути из (r, c) в $(i_{\text{cur}}, j_{\text{cur}})$, и у конечной клетки есть хотя бы один сосед, то можно дойти до этой клетки по найденному пути, а после делать парные ходы в соседнюю клетку и обратно;

4. альтернативно — если в клетке $(i_{\text{cur}}, j_{\text{cur}})$ можно было оказаться ровно в тот момент, когда был удален последний ее сосед, то можно в ней находиться и в любой момент после, так как больше из нее перемещений не будет.

Таким образом, для одного из частичных решений **третьей подзадачи**, в которой компьютер выполняет случайные ходы, можно было находить расстояние между клетками как $|r-i_{\text{cur}}|+|c-j_{\text{cur}}|$ за исключением случаев, когда клетки имеют общую координату. Если ходы случайны, то с большой вероятностью в первые несколько ходов найдется клетка нужной четности, достижимая за пройденное расстояние, и при этом все оценки расстояний будут верны, так как количество удаленных клеток не позволяет «перекрыть» все кратчайшие маршруты между стартовой клеткой и конечной.

Первая подзадача позволяла не беспокоиться об отслеживании пройденного расстояния: все клетки становятся достижимыми в первый же ход. Если четность удаляемой клетки совпадает с четностью пройденного числа шагов (с учетом четности стартовой клетки), то можно сразу вывести, что фишка находилась на удаляемой клетке. Такое решение набирало неполный балл, потому что не учитывало клетки, в которых фишка могла оказаться «запертой». Больше баллов можно было получить, проверяя также при удалении клетки, что число ее неудаленных соседей равно нулю — в таком случае мы можем выиграть вне завивимости от ее четности.

Во **второй подзадаче** перемещения всегда происходят на один шаг. Из-за этого, во-первых, четность клетки меняется с каждым ходом, а во-вторых, легко пересчитывать, в каких клетках мы бы могли сейчас находиться. Достаточно поддерживать известные расстояния до каждой клетки и массив клеток, которые стали достижимы на последнем ходу. Тогда для нового хода достаточно перебрать их соседей и обновить тех из них, до которых расстоянее ранее не было известно. Оставшиеся детали решения совпадают с описанным выше. Аналогичные рассуждения также работали и в **пятой подзадаче**.

Четвертая подзадача выделялась тем, что любые две удаляемые подряд клетки были соседними по стороне. Поскольку во вводе перечислены все клетки, они образуют «змейку», покрывающую все поле. А в таком случае можно показать, что фишка не может оказаться «запертой», и с каждым ходом будет менять четность клетки, на которой она находится, в соответствии с четностью совершенных перемещений. Таким образом, эту подзадачу проходила версия полного решения, не учитывающая возможность фишки оказаться заблокированной в клетке без соседей.

В **шестой подзадаче** поле представляет собой полосу, поэтому достаточно просто отслеживать, на какие отрезки ее разбивают удаленные клетки, а также в каких отрезках и на клетках какой четности может находиться фишка. Если не придумать полное решение, то можно было набрать баллы за эту подзадачу, поддерживая эти отрезки с помощью декартова дерева, либо с помощью системы непересекающихся множеств, обрабатывая запросы с конца.

Полное решение сочетает в себе все описанные выше общие идеи. Только для поиска расстояний между клетками требуется делать поиск в ширину из стартовой клетки. Поскольку в момент,

когда мы доходим до клеток с расстоянием $> \sum_{t \leqslant \text{cur}} k_t$, мы еще не знаем, можно ли будет после следующего хода компьютера через них проходить, будем

- поддерживать глобальную очередь `bfs` с еще не обработанными клетками;
- на каждом ходу продолжать `bfs` с того состояния, на котором остановились в прошлом, при чем будем пропускать уже удаленные клетки;
- останавливать `bfs` при достижении им клеток на расстоянии больше пройденного;
- для всех клеток, соседних с удаляемой, проверять, сколько из их соседей еще не удалены — если все соседи удалены, и фишка может находиться в данной клетке в этот момент (расстояния хватает и совпадает четность), запомним эту клетку в отдельное множество.

Тогда ответом будет первая удаляемая клетка, для которой выполнятся описанные в начале этого разбора критерии. **Седьмая подзадача** позволяла не заметить, что на все решение можно сделать только один «общий» `bfs`, и запускать на каждый запрос новый `bfs` заново. **Восьмая подзадача** ограничивала стартовую клетку, что могло помочь пройти тесты решениям, некорректно обрабатывающим какие-то из описанных случаев или критериев.

Общее время работы полного решения — $\mathcal{O}(nm)$.