

Человек-паук Нуар и кубик Рубика

Авторы задачи: Даниил Орешников и Егор Юлин, разработчик: Даниил Орешников

Задачи о поиске пути в матрице из левого-верхнего в правый нижний угол часто оказываются либо на динамическое программирование, либо на потоки. В случае, когда еще и ходить разрешается только вправо или вниз, это с очень большой вероятностью динамика.

Перед тем, как описать, какие состояния в данной динамике понадобятся, отметим несколько ключевых идей.

1. Если есть возможность попасть в клетку (i, j) , и мы знаем, был ли инвертирован столбец j , то мы однозначно восстанавливаем, была ли инвертирована строка j . Действительно, только эти две инверсии могли менять значение клетки (i, j) , а в конце ее значение должно было стать равно 1, что задает черный цвет клетки.
2. Соответственно, обратное тоже верно — по факту, инвертировался ли столбец, мы однозначно восстанавливаем, была ли инвертирована строка, если клетка лежит на пути.
3. В клетку (i, j) можно попасть только из $(i - 1, j)$ или $(i, j - 1)$.
4. Если в клетку (i, j) мы пришли из $(i - 1, j)$, то, зная, была ли инвертирована строка i , мы можем восстановить, была ли инвертирована строка $i - 1$ по цепочке $\text{inv_row}[i] \rightarrow \text{inv_column}[j] \rightarrow \text{inv_row}[i - 1]$.
5. Аналогично для столбцов при переходе из $(i, j - 1)$ в (i, j) .

Теперь, когда мы понимаем, что достаточно знать только факт инвертирования столбца или строки какой-то клетки, чтобы восстановить такую же информацию при переходах, можно посчитать динамику: $\text{dp}[i][j][c]$ — это минимальное количество инверсий, которое надо сделать, чтобы существовал путь из левого верхнего угла таблицы в клетку (i, j) , **при условии** c ($c = 0$ означает, что столбец j не был инвертирован, $c = 1$ — был).

Посмотрим, какие переходы возможны и через какие состояния можно обновить $\text{dp}[i][j][c]$.

1. Мы могли прийти в (i, j) из $(i - 1, j)$. Поскольку они находятся в одном и том же столбце, то флаг его инвертирования не мог поменяться и равен c . Тогда флаг инвертирования строки i равен $r = \text{a}[i][j] \oplus c \oplus 1$, где \oplus — операция логического «исключающего ИЛИ» (**xor**). Поскольку это новая строка, то надо учесть ее инвертирование:

$$\text{dp}[i][j][c] = \min(\text{dp}[i][j][c], \text{dp}[i - 1][j][c] + r).$$

2. Мы могли прийти в (i, j) из $(i, j - 1)$. Тогда флаг инвертирования строки i равен $r = \text{a}[i][j] \oplus c \oplus 1$, а флаг инвертирования столбца $j - 1$ можно посчитать как $c_{j-1} = \text{a}[i][j - 1] \oplus r \oplus 1$. Обновляем аналогично:

$$\text{dp}[i][j][c] = \min(\text{dp}[i][j][c], \text{dp}[i][j - 1][c_{j-1}] + c).$$

Теперь, когда мы посчитали минимальное количество инверсий $\min(\text{dp}[n][m][0], \text{dp}[n][m][1])$, достаточно с помощью стандартных техник восстановления ответа «вернуться» по оптимальному пути в клетку $(1, 1)$ и выписать по пути все инвертированные строки и столбцы, после чего вывести. Проще всего по ходу подсчета динамики также запоминать, из какого состояния было самое оптимальное обновление, и для восстановления ответа возвращаться по этим состояниям назад. Общее время работы алгоритма — $\mathcal{O}(nm)$.