

Борской Мой

Автор задачи и разработчик: Владимир Рябчун

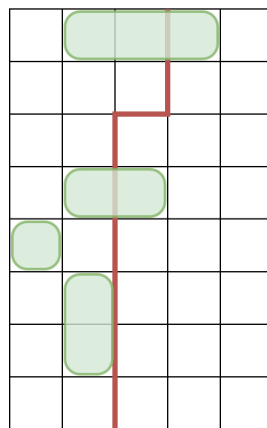
Для решения данной задачи применим динамическое программирование по изломанному профилю.

Будем постепенно заполнять поле кораблями, двигаясь по столбцам слева направо. Чтобы поставить какой-то корабль, необходимо знать, сколько кораблей данного вида осталось. Кроме этого необходимо хранить информацию о предыдущих размещенных кораблях — они могут пересекаться или касаться того, который мы хотим поставить.

Будем идти по полю слева направо и сверху вниз. У нас в каждый момент будет одна клетка, про которую мы определяем, что с ней произойдет. Есть четыре варианта:

1. Оставить клетку пустой. Для этого в ней не должно стоять символа 'х' и не должно быть уже поставленных кораблей, которые покрывают эту клетку.
2. Продолжить вправо корабль, размещенный до этого. Для этого нужно как-то помнить, а не «торчит» ли вперед какой-то старый корабль. О том, как это делать будет сказано далее.
3. Разместить в этой клетке корабль вертикально. Тогда он покроет эту клетку и несколько клеток ниже. Чтобы это было возможным, нужно проверить, не пересекаемся ли мы с уже размещенными кораблями и не пытаемся ли мы покрыть клетки, в которых записано 'о'.
4. Разместить в этой клетке корабль горизонтально. Нужно опять же проверить, не мешает ли нам какой-то прошлый корабль и не нарушаем ли мы информацию про поле.

В итоге нужно помнить профиль кораблей — насколько они «торчат» за уже рассмотренные клетки. Предлагается для этого хранить битовую маску, однако каждой клетке будет отводиться два бита, кодирующие значения от 0 до 3. Значение 0 означает, что на границе нет клеток корабля, значение 1 означает, что на границе есть клетка, 2 — что корабль выпирает на одну клетку за границу, а 3 — что корабль выпирает на две клетки за границу.



В данном примере красной линией обозначен излом профиля, а зелеными фигурами — корабли. Состояние поля будет соответствовать паре чисел $(2, 0110202_4)$, где первое значение пары обозначает позицию излома профиля, а второе — описанную выше маску.

Итоговая динамика имеет вид $dp[c][ship_1][ship_2][ship_3][r][mask]$, где c — столбец, а r — излом.

Данная задача требовала довольно аккуратной реализации. Общее число масок для каждого профиля было не больше 2400, хотя числовые значения маски могли быть очень большими. Чтобы уменьшить потребление памяти, необходимо было сжать значения масок перед вычислением динамики. Кроме того, имело смысл делать динамику вперед и избавиться от первого измерения, храня только два последних столбца.

Для дальнейшего ускорения можно было для каждой клетки и для каждого из пяти возможных расположений кораблей на ней посчитать, а можно ли там разместить этот корабль. Это значительно ускоряет решение. Также можно для каждой маски и излома заранее посчитать, какой будет

маска при каждом переходе. Последняя оптимизация, примененная автором задачи, заключалась в смене индексов динамики и порядке вычислений: возможность поставить корабль какого-то типа однозначно определяется уже при фиксированных (c, r, mask) , поэтому стоило перебирать сначала их, а потом безусловно обновлять значения динамики в циклах по $\text{ship}_1, \text{ship}_2, \text{ship}_3$.

Итоговая асимптотика решения — $\mathcal{O}(wh \cdot s_1 s_2 s_3 \cdot \text{States})$, где $\text{States} \approx 2400$.