Задача A. Черно-белая таблица

Можно получить решение, встроив информацию о 15×15 сетке в исходный код, но это занимает много времени. Поскольку сетка симметрична относительно центрального квадрата (8-я строка и 8-й столбец), мы можем решить задачу более простым способом.

В данной сетке квадрат окрашен в черный цвет, если "расстояние" от центрального квадрата до него нечетное, и в белый, если четное. В частности, квадрат на r-й строке и c-м столбце черный тогда и только тогда, когда расстояние Чебышёва (шахматное расстояние) от центрального квадрата, $\max(|r-8|, |c-8|)$, нечетное. (Здесь $|x|$ обозначает абсолютное значение x.)

Задача B. Високосный год

Давайте используем условный оператор для реализации логики, описанной в условии задачи. Чтобы определить, делится ли x на y, можно проверить, выполняется ли условие $x\%y == 0$. Здесь $x\%y$ обозначает остаток от деления x на y, где % — оператор, который находит остаток.

Если x делится на y, то остаток от деления x на y равен 0, поэтому приведенный выше код определяет делимость.

В нашем случае может быть несколько условных операторов, что может привести к глубокой вложенности. В таких ситуациях можно сразу вывести результат и завершить выполнение функции, как только условие выполнится, чтобы избежать глубокой вложенности.

Задача C. Робот-манипулятор

Начнем с теста 1.

Переходим к тесту a_1, затем к тесту a_{a_1}, и так далее, пока не достигнем теста n.

Каждый переход соответствует одному запросу подсказки у Даши.

Используем цикл для моделирования переходов.

Если мы попадаем в цикл (например, $a_i = i$), то Борис никогда не получит «Принято», и количество подсказок будет бесконечным.

Если мы достигаем теста n, то количество подсказок равно количеству переходов.

Задача D. Сборка робота

Задача заключается в подсчете количества троек (i, j, k), таких что $A_i < B_j < C_k$. Если зафиксировать j, то искомое количество будет равно произведению количества i, для которых $A_i < B_j$, и количества k, для которых $B_j < C_k$.

Таким образом, для каждого j можно подсчитать количество подходящих i и k, перемножить их и сложить результаты.

Как же подсчитать количество i и k? Если просто перебирать все элементы массива, то сложность будет $O(N^2)$, что неприемлемо.

Предварительно отсортируем массивы A и C. Тогда количество i и k можно найти с помощью бинарного поиска. В C++ для этого удобно использовать стандартные функции `lower_bound` и `upper_bound`.

Итоговая временная сложность составит $O(N \log N)$, что является приемлемым.

Задача E. Две матрицы

Вместо того чтобы удалять строки и столбцы по одному, как в исходной постановке задачи, мы можем рассмотреть следующую операцию, выполняемую ровно один раз: "выбрать некоторые (возможно, ни одной) строки и столбцы исходной матрицы A и удалить выбранные строки и столбцы одновременно". Например, для следующей матрицы A, приведённой в Sample Input 1:

```
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
```

мы можем выбрать 1-ю строку, 3-ю строку, 2-й столбец и 5-й столбец и удалить их одновременно, чтобы получить следующую матрицу B, приведённую в Sample Input 1:

```
6 8 9
16 18 19
```

Чтобы определить, можно ли сделать матрицу A равной матрице B с помощью таких операций, достаточно перебрать все 2^{H_1} способов выбора "удалять ли каждую из H_1 строк" и все 2^{W_1} способов выбора "удалять ли каждый из W_1 столбцов" (всего $2^{(H_1+W_1)}$ способов) и проверить, делает ли какой-либо из них матрицу A равной матрице B.

Перебор всех способов

Для перебора всех $2^{(H_1+W_1)}$ способов выбора строк и столбцов можно использовать метод битового перебора (bit brute-forcing), который мы опишем ниже:

- Каждый способ выбора "удалять ли каждую из H_1 строк" представляется H_1-битовым двоичным неотрицательным числом X, где:
  - Для $i = 0, 1, 2, \ldots, H_1 - 1$:
    - Если i-я строка удаляется, то i-й бит (считая с младшего разряда) числа X равен 1.
    - Если i-я строка не удаляется, то i-й бит равен 0.
- Например, выбор удаления 1-й и 3-й строк из 5 строк может быть представлен числом 00101 в двоичной системе. Таким образом, все 2^{H_1} способов выбора строк могут быть представлены числами $000\ldots00$, $000\ldots01$, $\ldots$, $111\ldots11$ в двоичной системе или числами $0, 1, \ldots, 2^{H_1} - 1$ в десятичной системе.
- Аналогично, способы выбора "удалять ли каждый из W_1 столбцов" могут быть представлены числами $0, 1, \ldots, 2^{W_1} - 1$.

Реализация перебора

Таким образом, мы можем перебрать все комбинации способов выбора строк и столбцов с помощью двойного вложенного цикла, где:

- Первый цикл перебирает способы выбора строк: $0, 1, \ldots, 2^{H_1} - 1$.
- Второй цикл перебирает способы выбора столбцов: $0, 1, \ldots, 2^{W_1} - 1$.

Задача F. Хитрый пароль

Дан прямоугольник размером $n \times m$, где каждая ячейка раскрашена в один из цветов. Требуется подсчитать количество прямоугольников, раскрашенных ровно в один цвет.

Для каждой строки d и каждого столбца l перебираем все возможные прямоугольники, начиная с (d, l). Для каждого прямоугольника находим минимальную высоту столбцов в диапазоне $[l, r]$ и добавляем её к ответу.

- n — количество строк.
- m — количество столбцов.
- a — матрица размера $n \times m$, содержащая цвета ячеек.
- h — массив высот столбцов, где $h[i]$ — количество подряд идущих ячеек одного цвета сверху в столбце i.
- ans — итоговый ответ (количество одноцветных прямоугольников).

1. Инициализируем массив h единицами.
2. Для каждой строки d от 0 до $n - 1$:

(a) Для каждого столбца l от 0 до $m-1$:

i. Инициализируем $\min\_h = h[l]$ и $r = l$.

ii. Пока $r < m$ и $a[d][r] = a[d][l]$:

A. Обновляем $\min\_h = \min(\min\_h, h[r])$.

B. Добавляем $\min\_h$ к ans.

C. Увеличиваем r на 1.

(b) Если $d \neq n-1$, обновляем массив h:

i. Для каждого столбца i от 0 до $m-1$:

A. Если $a[d][i] \neq a[d+1][i]$, сбрасываем $h[i] = 1$.

B. Иначе увеличиваем $h[i]$ на 1.

3. Выводим ans.

Сложность

- Временная сложность: $O(n \cdot m^2)$.
- Пространственная сложность: $O(m)$.