Задача A. Треугольники

Первое, что необходимо научиться делать в данной задаче - определять «остроту» угла треугольника. Напомним, что острым называется угол строго меньше 90 градусов.

Углы треугольников в ответе являются вписанными, а значит величина таких углов равна половине дуги, на которую они опираются. Отсюда следует, что любой угол, опирающийся на дугу, строго меньшую половины окружности (180 градусов) будет острым.

Наша окружность разделена на 12 одинаковых сегментов, откуда размер одного сегмента равен $\frac{360}{12} = 30$ градусов. Следовательно, любой угол, опирающийся на 5 сегментов или менее, будет острым.

Расположение точек является симметричным относительно поворота на 90 градусов, поэтому общее количество треугольников равно количеству треугольников с фиксированной вершиной 1 (аналогично можно зафиксировать вершину 2 или вершину 3), умноженному на количество точек 1 на окружности.

Для фиксированной вершины 1 существует $4 \times 4 = 16$ возможных комбинаций вершин 2 и 3. На этом этапе можно остановить размышления и просто проверить каждый треугольник.

Если занумеровать точки на окружности от 1 до 12 так, что зафиксированная вершина 1 треугольника будет иметь номер 1, то список возможных треугольников имеет вид:

1. 1 2 3
2. 1 2 6
3. 1 2 9
4. 1 2 12
5. 1 5 3
6. 1 5 6
7. 1 5 9
8. 1 5 12
9. 1 8 3
10. 1 8 6
11. 1 8 9
12. 1 8 12
13. 1 11 3
14. 1 11 6
15. 1 11 9
16. 1 11 12

Существует способ сократить количество проверок. Обратим внимание, что если две из трёх вершин треугольника являются соседними на окружности (противолежащий угол опирается ровно на один сегмент), то два других угла в треугольнике будут опираться суммарно на 11 сегментов. Так как угол не может опираться на нецелое число сегментов, то хотя бы один из углов будет опираться на 6 сегментов или более, то есть не будет острым.

Остаются следующие треугольники:

1. 1 5 3 - угол $1 - 3 - 5$ опирается на 8 сегментов - не является острым.

2. 1 5 9
3. 1 8 3
4. 1 8 6
5. 1 11 3 - угол $11 - 1 - 3$ опирается на 8 сегментов - не является острым.
6. 1 11 6
7. 1 11 9 - угол $9 - 11 - 1$ опирается на 8 сегментов - не является острым.

Видно, что подходящими являются 4 треугольника для фиксированной вершины 1. Суммарно для всех четырех точек 1 получается $4 \times 4 = 16$ треугольников, что и является ответом на задачу.

Задача B. Шахматная доска

Итоговую сумму можно разделить на две независимые суммы — в белых и в черных клетках.

Клетки фиксированного цвета в рамках строки заполняются подряд слева направо. Пусть значение в самой левой клетке цвета равно S, а количество клеток такого цвета в строке равно K. В таком случае в данном ряду будут значения S, $S + 1$, $S + 2$, ..., $S + K - 1$.

Данные значения можно представить как арифметическую прогрессию с шагом 1, поэтому их сумма Sum будет равна $\frac{(S + (S + K - 1)) \cdot K}{2}$. Соответственно, для вычисления ответа нам необходимо найти величины S и K для белых и черных клеток, соответственно.

В каждой строке будет $L = \left\lfloor \frac{N}{2} \right\rfloor$ клеток одного цвета и $G = N - L$ клеток другого цвета.

- в нечетных строках $(1, 3, \dots)$ L черных и G белых;
- в четных строках $(2, 4, \dots)$ наоборот — L белых и G черных.

Величина S для заданного цвета в фиксированной строке i равна $B + P(i) + 1$, где:

1. B — начальный сдвиг значений в клетках такого цвета;
2. $P(i)$ — суммарное количество клеток данного цвета в строках выше i.

Величина B для белого цвета всегда фиксирована и равна 0. Для черного цвета значение B равно суммарному количеству белых клеток на доске:

- для четных N количество черных и белых клеток совпадает и равно $\frac{N^2}{2}$;
- для нечетных N количество белых и черных клеток отличается на 1, откуда несложно получить формулу: $\frac{N^2 + 1}{2}$ белых клеток и $\frac{N^2 - 1}{2}$ черных клеток на доске.

Исходя из факта, что в двух соседних строках доски размера $N \times N$ всегда расположено ровно N белых и N черных клеток, несложно найти формулу для функции $P(i)$:

- для нечетных строк $(i = 1, 3, \dots)$ $P(i) = N \cdot \left\lfloor \frac{i}{2} \right\rfloor$;
- для четных строк $(i = 2, 4, \dots)$ $P(i) = P(i - 1) + K_{i-1}$, где K_{i-1} — количество клеток данного цвета в строке $i - 1$.

В рамках задачи нам необходимо вычислить сумму в строке 7 таблицы размером 15×15.

Подставляя данные значения в полученные выше формулы, получаем следующие значения:

$L = \left\lfloor \frac{15}{2} \right\rfloor = 7$, $G = N - L = 15 - 7 = 8$.

Для белого цвета:

- $K = G = 8$,
- $S = 1 + 0 + 15 \cdot \left\lfloor \frac{7}{2} \right\rfloor = 1 + 15 \cdot 3 = 46$,
- $Sum = \frac{(46 + (46 + 8 - 1)) \cdot 8}{2} = 396$;

Для черного цвета:

- $K = L = 7$,
- $B = \frac{15^2 + 1}{2} = 113$,
- $S = 1 + 113 + 15 \cdot \left\lfloor \frac{7}{2} \right\rfloor = 114 + 15 \cdot 3 = 159$,
- $Sum = \frac{(159 + (159 + 7 - 1)) \cdot 7}{2} = 1134$;

Итоговый ответ равен $396 + 1134 = 1530$.

Задача C. Тролль Сева

Первым делом обратим внимание, что по сути операция переворота стула увеличивает итоговое количество стульев на $4 - 1 = 3$.

Соответственно, вам требуется проверить, существует ли среди элементов арифметической прогрессии $a_i = (2, 5, 8, \dots)$ число, кратное N.

Отсюда сразу видно решение — перебрать в цикле все элементы прогрессии до достижения определенного числа итераций. Если не нашли кратное — рассадить троллей нельзя.

Если обозначить количество итераций за K, то сложность алгоритма будет $O(K)$. Но как связаны K и N?

Так как речь идёт о делимости, то правильнее всего перейти к арифметике остатков: число X кратно числу N, если остаток от деления X на N равен 0. Очевидно, что числа X и $X + 3 \cdot N$ имеют одинаковый остаток от деления на N, поэтому нет смысла совершать более N итераций — если не нашли остаток 0 среди первых N, то дальше его точно не будет в силу периодичности.

Отсюда следует, что $K = N$, то есть итоговая асимптотика предложенного решения получается $O(T \cdot N)$. В худшем случае $T \cdot N = 10^{13}$ — слишком неэффективно по времени.

Для получения эффективного решения запишем формулу нашей прогрессии $a_i = 3 \cdot i + 2$ ($i = 0, 1, \dots$). Заметим, что остаток от деления на 3 равен 2 для всех членов прогрессии.

Рассмотрим различные случаи относительно остатка от деления N на 3:

- Все числа вида $N = 3 \cdot k$ по определению делятся на 3, поэтому не могут быть делителями чисел, дающих в остатке 2 при делении на 3;
- Если число N имеет вид $3 \cdot k + 1$, то существует число $2 \cdot N = 3 \cdot (2 \cdot k) + 2$, которое подходит под формулу прогрессии;
- Если число N имеет вид $3 \cdot k + 2$, то оно встречается в прогрессии само по себе.

Видно, что рассадить N троллей можно при условии, что N не делится на 3. Итоговое решение имеет сложность $O(1)$ на один набор данных, что даёт суммарную сложность $O(T)$.

Задача D. (Не)достижимый идеал

Первое решение, которое приходит в голову при прочтении задачи — в цикле перебрать все числа X от L до R и для каждого проверить выполнение условия $\gcd(N, X) = \min(N, X)$.

Сложность вычисления $\gcd(A, B)$ составляет $O(\log(\min(A, B)))$ в худшем случае (два соседних числа Фибоначчи) и $O(1)$ в лучшем (один параметр кратен другому).

Даже в лучшем случае оценка сложности будет $O(R-L)$ операций на один запрос, что в худшем случае ($L = 1$, $R = 10^{18}$) составит порядка 10^{18} итераций, что является слишком неэффективным (а это еще и оценка gcd взята по лучшему случаю).

Для начала заметим, что $N \leqslant L < R$, откуда следует, что $\min(N, X) = N$ для X от L до R. Если $\gcd(N, X) = N$, то по определению наибольшего общего делителя X делится на N без остатка.

Если остановиться на этом шаге и реализовать проверку кратности каждого X из промежутка $[L; R)$, то получится та же сложность $O(R - L)$.

Для полного решения надо сделать следующее наблюдение: если обозначить через $f(N, L, R)$ количество чисел в полуинтервале $[L; R)$, кратных N, то выполняется следующее равенство:

$f(N, L, R) = f(N, 1, R) - f(N, 1, L) = p(N, R) - p(N, L)$.

Здесь применяется обозначение $f(N, 1, S) = p(N, S)$ — количество положительных целых чисел строго меньших S, кратных числу N.

Несложно понять, что $p(N, S) = \left\lfloor \dfrac{S-1}{N} \right\rfloor$, так как кратным является каждое N-е число (N, $2 \cdot N$, $3 \cdot N$, ...).

Итоговый ответ равен $p(N, R) - p(N, L) = \left\lfloor \dfrac{R-1}{N} \right\rfloor - \left\lfloor \dfrac{L-1}{N} \right\rfloor$.

Задача E. Хакерская Атака

В условии описан итеративный процесс размножения вирусов: каждый вирус каждую секунду создаёт $(q - 1)$ своих копий. Если обозначить начальное количество вирусов за b, то итоговое количество вирусов через t секунд будет равно $v(t) = b \cdot q^t$.

На первый взгляд может показаться, что в задаче не хватает входных данных — не дано ни начального количества вирусов b, ни множителя вирусов в секунду q.

Всё, что дано — два момента времени ($t = 3$ и $t = 6$) и два количества вирусов ($v_3 = v(3) = 1029$ и $v_6 = v(6) = 352947$). Можно ли вычислить b и q с помощью этих значений?

Запишем систему уравнений:

- $b \cdot q^3 = v_3$ (1)
- $b \cdot q^6 = v_6$ (2)

Разделим уравнение (2) на (1), получим $\dfrac{v_6}{v_3} = \dfrac{b \cdot q^6}{b \cdot q^3} = q^3$.

Зная q^3, можно вычислить $q = \sqrt[3]{\dfrac{v_6}{v_3}}$.

b вычисляется из (1): $b = \dfrac{v_3}{q^3}$.

Подставляя числа из условия, получаем:

- $q^3 = \dfrac{352947}{1029} = 343$;
- $q = \sqrt[3]{343} = 7$;
- $b = \dfrac{1029}{343} = 3$.

Итоговая формула равна $3 \cdot 7^t$.

Значения b и q можно было вычислить самостоятельно на бумаге / на калькуляторе или воспользоваться встроенными математическими функциями языков программирования.

P.S. Самой частой ошибкой среди участников на python было использование формулы $21 \cdot 7^{t-1}$ вместо оригинальной. Может показаться, что формулы дают одинаковые результаты, но есть одно большое но:

При $t = 0$ ответ будет равен $21 \cdot 7^{-1} = 21 \cdot \dfrac{1}{7}$. Промежуточный результат $\dfrac{1}{7}$ является по определению нецелым, поэтому весь результат принимает вещественный тип. По итогу неверные решения выводили «3.0» заместо «3», из-за чего приходил вердикт «Неправильный формат вывода».

Задача F. Метро

Для начала определимся с точной величиной суммарного количества пассажиров за смену F. Смена задаётся тремя величинами:

- s — момент времени, когда поезд отправился в первый маршрут за день;
- k — количество маршрутов за смену;
- d — расстояние по времени между соседними маршрутами в смене.

Так как k и d фиксированы для заданного набора входных данных, то по сути мы решаем задачу выбора s для максимизации величины $F(s)$:

$F(s) = a_s + a_{s+d} + \cdots + a_{s+(k-1)\cdot d}$

Так как за смену необходимо проехать все k маршрутов, то возникает дополнительное условие: $s + (k-1)\cdot d \leqslant N$, что равносильно $s \leqslant N - (k-1)\cdot d$.

Соответственно, самое простое решение — перебрать в цикле все s от 1 до $N - (k-1)\cdot d$, для каждого с шагом d вычислить описанную сумму и найти итоговый максимум.

Сложность такого решения будет $O((N-(k-1)\cdot d)\cdot k) = O(N\cdot k - k^2\cdot d + k\cdot d)$. Учитывая, что в худшем случае $d = 1$, а слагаемое $k^2\cdot d$ «доминирует» над $k\cdot d$, получаем сложность $O(N\cdot k - k^2)$.

Очевидно, что максимум функции $N\cdot k - k^2$ достигается при $k = \dfrac{N}{2}$, откуда итоговая сложность для худшего случая получается $O(N\cdot\dfrac{N}{2} - (\dfrac{N}{2})^2) = O(N^2)$.

Для данных ограничений $N \leqslant 2\cdot 10^5$, поэтому N^2 будет порядка 10^{10} — недостаточно эффективно в общем случае.

Рассмотрим упрощенную версию задачи — $d = 1$: в таком случае $F(s) = a_s + a_{s+1} + \cdots + a_{s+k-1}$ - сумма подотрезка массива фиксированного размера k.

Данная задача решается за $O(N)$ двумя способами:

- Префикс-суммами: $F(s) = P_{s+k} - P_s$, где $P_i = a_{i-1} + a_{i-2} + \cdots + a_1$.

  Данный массив вычисляется за $O(N)$ с помощью формулы $P_i = P_{i-1} + a_{i-1}$.
- Суммой в окне: если известна величина $F(s)$, то величина $F(s+1) = F(s) - a_s + a_{s+k}$.

Хорошо, для случая $d = 1$ решение за $O(N)$ существует. Но что делать для случаев $d > 1$?

На самом деле легко понять, что общий подход не меняется совершенно - только теперь вместо шага 1 у нас получается шаг d, порождающий d независимых между собой подзадач. В самом деле, элемент a_i будет участвовать только в тех суммах $F(s)$, у которых $s\%d = i\%d$, где $\%$ - остаток от деления.

Поэтому существует три итоговых способа решения:

- Построить d независимых массивов B таких, что B_j $(0 \leqslant j < d)$ содержит только элементы массива a_i при $i\%d = j$. Для каждого массива B_j можно решить описанную выше задачу для $d = 1$, после чего найти максимум среди ответов.

  Суммарный размер $|B_0| + |B_1| + \cdots + |B_{d-1}| = N$, поэтому сложность будет $O(N)$.
- Префикс-суммами с шагом d: $F(s) = P_{s+k\cdot d} - P_s$, где $P_i = a_{i-d} + a_{i-2\cdot d} + \cdots + a_{i\%d}$.

  Данный массив вычисляется за $O(N)$ с помощью формулы $P_i = P_{i-d} + a_{i-d}$.
- Суммой в окнах с шагом d: если известна величина $F(s)$, то величина $F(s+d) = F(s) - a_s + a_{s+k\cdot d}$.

Итоговая сложность решения $O(N)$ независимо от выбранного способа.

Задача G. Неожиданный кроссовер

Данная задача представляет собой классический пример игры на ориентированном ациклическом графе:

- вершина графа совпадает с состоянием игры и задаётся тремя параметрами (x, y, $step$) — координаты клетки с фишкой и длина шага из текущей клетки (2, 3 или 5).

  Для упрощения записи перейдем от длины шага $step$ к позиции k ($0 \leqslant k < 3$) в массиве длин $steps$.

- ребро между вершинами (xf, yf, kf) и (xt, yt, kt) задаёт корректный ход по правилам игры и существует при условии $kt = (kf + 1)\%3$ ($\%$ — остаток от деления) и выполнении одного из следующих условий:

  1. $xt = xf + steps_{kf}$, $yt = yf$;
  2. $xt = xf$, $yt = yf + steps_{kf}$.

- вершина (x, y, k), у которой $N < x + steps_k$ и $M < y + steps_k$, является проигрышной по определению — назовем такую вершину финальной.

- игрой является путь от вершины (1, 1, 0) до любой финальной вершины.

Классическое решение таких задач сводится к одному из трёх способов:

1. ретроспективный анализ — состояние считается выигрышным, если существует переход в проигрышное состояние.

2. теорема Шпрага-Гранди (каждому состоянию сопоставляется значение специальной функции, связанное с игрой Ним);

3. формула или правило, выведенные из анализа процесса игры (часто основывается на анализе результатов из пунктов 1 и/или 2);

Ретроспективный анализ является самым простым и базовым подходом, поэтому начнём с него. Простейшим решением является простейшая рекурсивная функция, перебирающая все возможные варианты игры без какого-либо запоминания промежуточных результатов:

$win(x, y, k, N, M) = \overline{win}(x + steps_k, y, (k + 1)\%3, N, M)$ *or* $\overline{win}(x, y + steps_k, (k + 1)\%3, N, M)$, где $\overline{win}$ — логическое отрицание, а *or* — логическое «или».

N и M передавать необходимо, так как они нужны для определения финальности вершины.

Сложность такой функции будет прямо пропорциональна, как минимум, количеству различных путей в описанном графе.

Для целей данного разбора достаточно только нижней оценки количества путей: предположим, что фишка каждый раз передвигается ровно на 5. В таком случае любой путь будет иметь длину $(D = \frac{N}{5}) + (R = \frac{M}{5})$ и может быть задан последовательностью из D переходов вниз и R переходов вправо.

Очевидно, что количество таких путей будет равно $C(D + R, D)$. В худшем случае $D = R = \sqrt{\frac{3 \cdot 10^4}{5 \cdot 5}} \sim 34$, а $C(68, 34) \sim 28 \cdot 10^{18}$. Причем это оценка снизу с увеличенным размером шагов (очевидно, что для шагов 2, 3, 5 длина и вариантов пути будет больше), а также будет еще и дополнительный множитель, связанный с перемещением по путям туда и обратно.

Общий вывод — рекурсивный перебор без дополнительных оптимизаций не обладает достаточной эффективностью.

Обратим внимание, что при заданном размере доски тип вершины (проигрышная / выигрышная) зависит только от (x, y, k), но не от предыдущих вершин в пути. Соответственно, можно применить принцип динамического программирования и запоминать типы состояний после первого вычисления.

Такое запоминание обычно делается одним из двух способов:

1. Создать внешнее хранилище типов для вершин и проверять наличие типа для заданной вершины в хранилище; рекурсивное вычисление будет производится только, если для данной вершины тип ранее не вычислялся;

2. Так как игра происходит на ориентированном ациклическом графе, то можно вычислить все возможные состояния в порядке топологической сортировки без рекурсии.

   Топологическая сортировка в данном случае достаточно очевидна: вершина P вычисляется ранее вершины Q, если $x_p \leqslant x_q$ и $y_p \leqslant y_q$. Вершины, не связанные подобным соотношением, могут вычисляться в любом порядке.

На самом деле способ (1) так же производит топологическую сортировку, но в неявном виде.
Необходимо четко понимать, что сложность такого подхода равна $O(V \cdot E)$, где

- V — количество возможных достижимых состояний (вершин);
- E — количество возможных переходов из состояния (исходящих ребер из вершины).

E в данной задаче не превосходит $2 = O(1)$.

А какая же оценка на V? Воспользуемся приёмом, аналогичным описанному выше, и заменим все длины шагов на 2. В таком случае каждая достижимая вершина задаётся двумя уже «привычными» параметрами:

- количество шагов вниз D;
- количество шагов вправо R;

D может принимать значения от 0 до $\frac{N}{2}$, R — аналогично от 0 до $\frac{M}{2}$. Итоговая **верхняя оценка** на количество достижимых состояний V равна $O(D \cdot R) = O(N \cdot M)$.

Легко понять, что нижняя оценка так же равна $O(N \cdot M)$ (заменяем все длины шагов на 5).

Соответственно, по памяти получается $O(N \cdot M)$, но остаётся проблема с эффективностью по времени, которая равна $O(T \cdot S)$ в худшем случае (S — максимальное возможное произведение $N \cdot M$, равное $3 \cdot 10^4$).

Обратим внимание, что ходы в игре абсолютно симметричны относительно направления перемещения фишки — перемещения вниз и вправо с началом в вершине (1, 1, 0) можно заменить на перемещения влево и вверх с началом в вершине (N, M, 0) и ничего не изменится.

У данной формулировки есть одно важное преимущество: вершина (x, y, k) теперь является финальной в случае, если $steps_k \geqslant x$ и $steps_k \geqslant y$ — условие НЕ ЗАВИСИТ от размера доски!

А это означает, что сокращается количество параметров у функции вычисления выигрышности состояния:

$win(x, y, k) = \overline{win}(x - steps_k, y, (k+1)\%3)$ *or* $\overline{win}(x, y + steps_k, (k+1)\%3)$.

Данная оптимизация является очень важной, так как теперь значения функции win являются одинаковыми для игр на всех возможных размерах доски.

Оценим общее количество достижимых клеток (x, y) по всем возможным играм в запросе. Так как любая игра однозначно задаётся своей стартовой клеткой (N, M), то нас интересует общее количество пар целых положительных чисел (N, M) таких, что:

$$1 \leqslant N \leqslant S$$

$$1 \leqslant M \leqslant S$$

$$1 \leqslant N \cdot M \leqslant S.$$

Одной из верхних оценок является $O(S\sqrt{S})$: пусть $N \leqslant M$, тогда $N^2 \leqslant N \cdot M \leqslant S$, а значит $N \leqslant \sqrt{S}$ и $\sqrt{S} \leqslant M \leqslant S$.

На самом деле такого способа оценки уже будет достаточно по памяти — трехмерный массив размера $3 \times \sqrt{S} \times S$ заходит без особых проблем.

Существует ли асимптотически ниже верхняя оценка? Да, существует. Так как $N \cdot M \leqslant S$, то $M \leqslant \frac{S}{N}$, а итоговое количество пар будет равно $\frac{S}{1} + \frac{S}{2} + \frac{S}{3} + \cdots + \frac{S}{S}$.

Сумма $\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{S}$ является частичной суммой бесконечного гармонического ряда. Можно доказать, что она равна $O(\log S)$. Соответственно, оценка для количества пар равна $O(S \cdot \log S)$.

Итоговая сложность решения равна $O(S \cdot \log S \cdot K + T)$ ($K = 3$ — количество различных длин шагов), так как вычисление типов состояний происходит один раз по всем запросам.

Чтобы достичь такой оценки по памяти, хранить состояния необходимо или в векторе векторов (i-й вектор имеет длину $\frac{S}{i}$), или в словаре любого типа.

P.S. Одним из способов доказательства оценки суммы гармонического ряда является ограничение суммы сверху и снизу специальными суммами:

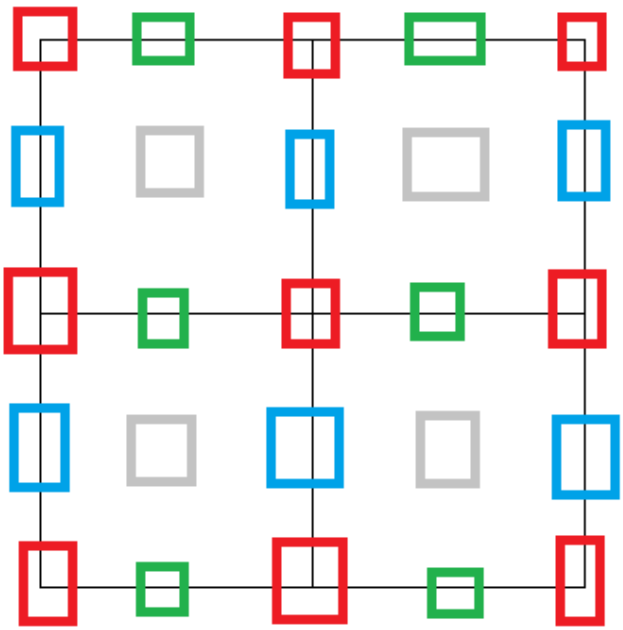
- Верхняя сумма $\frac{1}{1}+\frac{1}{2}+\frac{1}{2}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{8}+\frac{1}{8}+\ldots$.
- Нижняя сумма $\frac{1}{1}+\frac{1}{2}+\frac{1}{4}+\frac{1}{4}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{8}+\frac{1}{16}+\ldots$.

Легко понять, что обе суммы имеют оценку $O(\log S)$, поэтому и исходная сумма имеет такую же оценку.

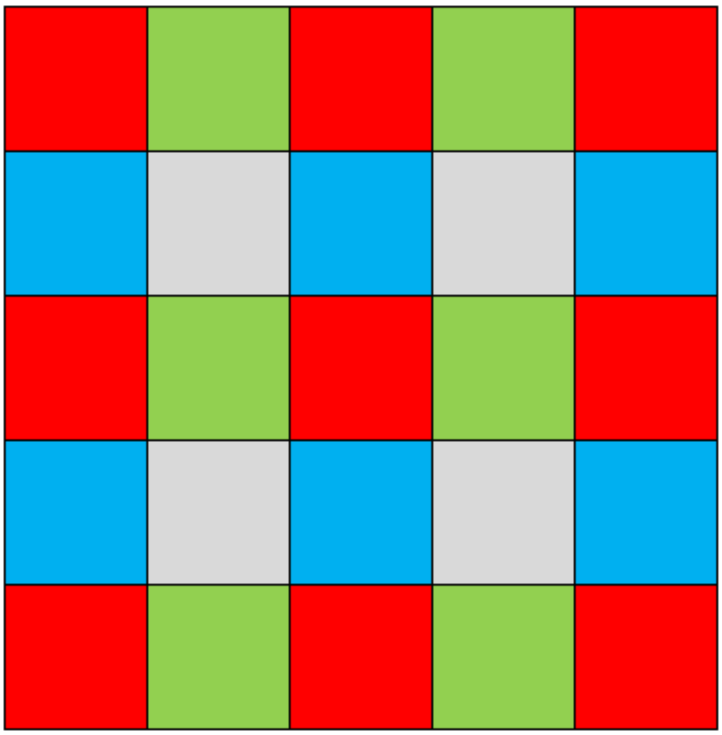
P.P.S. Автор задачи утверждает, что при достаточно полном анализе таблицы ответов / функций Шпрага-Гранди можно увидеть закономерность и решить задачу для $N \cdot M \leqslant 10^{18}$. Вам не требовалось совершать такой подвиг на соревновании, но можете попробовать проделать это в свободное время.

Задача H. Осознание десятого уровня

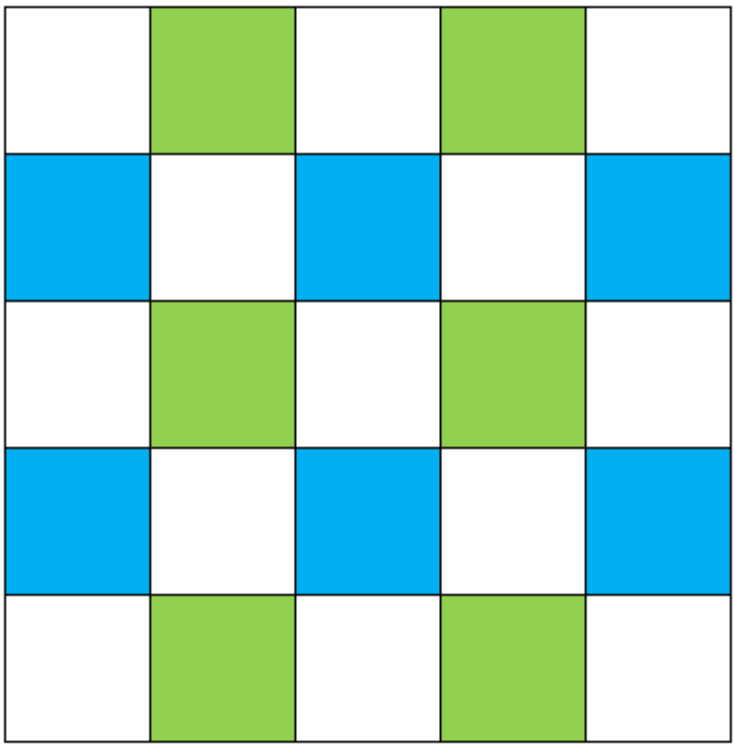
Обычное клетчатое поле разбивается на несколько элементов:



Серым цветом обозначены внутренности клеток, красным — углы, зеленым — горизонтальные ребра, голубым — вертикальные. Элементы клетчатого поля снова образуют клетчатое поле:



В задаче нас интересуют только ребра (зеленые и голубые клетки). Они образуют шахматную раскраску:



Если клетчатое поле состоит из $H \times V$ клеток, то у него $(2 \cdot H + 1) \times (2 \cdot V + 1)$ элементов (углов, ребер и внутренностей клеток). В данной задаче число строк $H = 2^n$, а число столбцов $V = 2^m$. Обратим внимание, что количество строк и столбцов в новой раскраске всегда нечетное.

Предлагается преобразовать координаты ребер к координатам клеток в приведенной шахматной раскраске следующим образом:

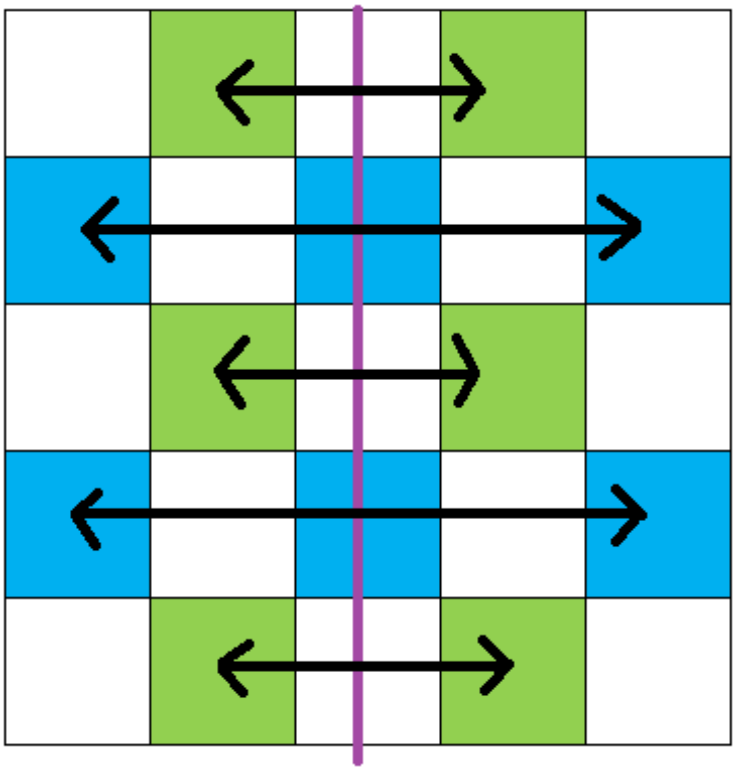
- вертикальный сгиб: $x = 2 \cdot r - 1$, $y = 2 \cdot i$;
- горизонтальный сгиб: $x = 2 \cdot j$, $y = 2 \cdot c - 1$.

Сам ответ на запрос предлагается искать, применяя поочередно все сгибы с первого до последнего, оставляя в рассмотрении каждый раз только половину с искомой клеткой.

Введем для каждого направления (v и h) три величины $left$, $right$ и inv — нижняя и верхняя граница рассматриваемого интервала для соответствующего направления, а так же показатель инверсии сгиба.

Что такое инверсия сгиба? Каждый сгиб бумаги изменяет направление будущих сгибов для одной из двух половин.

На примере показан вертикальный сдвиг. У клеток на сгибе (фиолетовая линия) направление сгиба уже определено и равно D по умолчанию. Остальные же клетки разбиваются на пары переходящих друг в друга под действием центральной симметрии клеток (черные стрелочки):



Направления сгибов внутри пары переходящих друг в друга клеток противоположны. После сгиба одна из половин окажется перевернутой, а другая — сохранит своё направление. К примеру, если будет производится сгиб RL, то сохранится направление для левой половины, а при LR — для правой.

Изначально для каждого направления $left = 0$, $right = 2^{b+1}$, $inv = false$, где b — количество сгибов соответствующего направления.

На каждом шаге вначале вычисляется координата текущего сгиба $middle = \dfrac{left + right}{2}$.

Если клетка располагается на текущем сгибе (соответствующая координата равна $middle$), то спуск прекращается — ответом является D, если $inv_v = inv_h$, иначе — U.

В ином случае происходит переход в половину с искомой клеткой, а так же обновляется величина inv для направления текущего сгиба.

Назовём интервал ($left$, $middle$) **младшим**, а ($middle$, $right$) соответственно старшим.

Обозначим через lc факт нахождения искомой клетки в младшем интервале.

Переход в половину происходит максимально просто: если $lc = true$, то происходит переход в младший интервал, иначе — в старший.

Обозначим через lb факт инверсии текущим сгибом направления в младшем интервале. Такую инверсию выполняют сгибы LR и UD.

Величина inv по итогу равна факту равенства величин lc и lb.

Итоговая сложность решения — $O(q \cdot (n + m))$.

P.S. Может показаться неочевидным, что величина inv никак не зависит от предыдущих сгибов в том же направлении. Для строгого доказательства достаточно рассмотреть направление на 4 полосах бумаги при всех возможных комбинациях двух сгибов RL и/или LR (аналогично можно проверять для UD и DU).

Это доказательство является по сути проверкой всех случаев, поэтому не приводится здесь развернуто. Можете самостоятельно проверить этот факт, разделив лист бумаги на 4 или 8 полос, после чего пронаблюдать за «судьбой» направления той или иной полосы.

P.P.S. Ограничение 29 вместо 30 выбрано по следующей причине: при $n = 30$ или $m = 30$ правая граница интервала будет равна 2^{31}, что ровно на 1 переполняет 32-битный целочисленный знаковый тип данных.