Задача A. Отгадай-ка

Поскольку $A = X + Y$ и $B = X - Y$, то $X = \frac{A+B}{2}$ и $Y = \frac{A-B}{2}$. При этом известно, что X и Y целые. Однако это возможно только если $A + B$ и $A - B$ чётные. Это, в свою очередь, возможно только тогда, когда у A и B одинаковая чётность.

Обратите внимание, что числа могут быть очень большой длины, поэтому надо вводить эти числа в виде строк, а затем проверять чётность только у последней цифры каждого числа.

Задача B. Сделай 99

У этой задачи есть два решения:

$$9 + 8 + 7 + 6 + 5 + 43 + 21 = 99$$

$$9 + 8 + 7 + 65 + 4 + 3 + 2 + 1 = 99$$

Задача C. Детский сад

Пусть x_i — количество конфет, съеденное i-м ребёнком. Заметим, что в сумме $a_1 + a_2 + \ldots + a_n$ каждое из x_i встречается ровно $N - 1$ раз, потому что входит во все a_j кроме $j = i$. Отсюда можно понять, что суммарное число съеденных конфет $s = \frac{a_1+a_2+\ldots+a_n}{n-1}$, а $x_i = s - a_i$. Так можно вычислить все значения x_i, найти среди них максимальное, если оно единственное, то вывести его индекс, а иначе -1.

Альтернативный подход состоит в том, что нам на самом деле не важно конкретное значение s, потому что оно одинаковое для всех i, и величина x_i зависит исключительно от a_i, а именно: чем меньше было a_i, тем больше x_i, так как a_i входит в него с отрицательным знаком. То есть надо было найти минимальное a_i и проверить, что оно единственное.

Задача D. Саморезы в доме

Заметим, что использовать шуруповёрт всегда выгоднее, чем отвёртку, так как $l_i \geqslant 1$. Если $K \geqslant N$, то полностью закрутить все саморезы можно с помощью шуруповёрта, и ответ равен N. Если же $K < N$, то рассмотрим один из саморезов. Сокращение времени в закручивании, если мы используем шуруповёрт, а не отвёртку, равно $l_i - 1$. Так как мы хотим сократить время как можно сильнее, нам нужно максимизировать все суммарные сокращения времени для K саморезов. Для этого мы можем отсортировать все саморезы по возрастанию их высоты недокрученности, закрутить последние K с помощью шуруповёрта, а оставшиеся $N - K$ — с помощью отвёртки. Ответом будет K плюс сумма $N - K$ наименьших l_i. Сложность решения этой задачи равна $O(N \log N)$.

Задача E. Подстольный теннис

Основная суть задачи состоит в том, чтобы реализовать ввод и вывод в задаче. Считаем ввод нашей ракетки в массив, при этом верхушку и рукоятку можно не сохранять. Будем идти с конца этого массива:

- Если секция ракетки была с неровностью «`(..)`», то в ответе должна быть секция «`)..(`».
- Если секция ракетки была с неровностью «`)..(`», то в ответе должна быть секция «`(..)`».
- Если секция ракетки была с неровностью «`/..\`», то в ответе должна быть секция «`/..\`» (так как ракетка перевёрнута).
- Если секция ракетки была с неровностью «`\../`», то в ответе должна быть секция «`\../`» (так как ракетка перевёрнута).

Перед выводом первой секции надо вывести верхушку:

- Если первая секция ракетки для вывода с неровностью «`(..)`» или «`/..\`», то верхушка ответа должна быть «`.__.`».
- Если первая секция ракетки для вывода с неровностью «`)..(`» или «`\../`», то верхушка ответа должна быть «`____`».

После вывода последней секции надо вывести рукоятку:

- Если последняя секция ракетки для вывода с неровностью «`(..)`» или «`\../`», то рукоятка ответа должна быть «`.II.`».
- Если последняя секция ракетки для вывода с неровностью «`)..(`» или «`/..\`», то рукоятка ответа должна быть «`IIII`».

Задача F. Торжественные массивы

Выберем первые $N-1$ элементов массива $a_1, a_2, \ldots, a_{N-1}$, пусть их сумма $a_1+a_2+\ldots+a_{N-1} = X$, тогда из того, что сумма $(a_1 + a_2 + \ldots + a_N)$ равна нулю по модулю K, однозначно определяется $a_N \equiv -X \pmod K$, то есть для каждого массива длины $N-1$ однозначно определяется массив длины N с суммой, кратной K. Получается, количество различных торжественных массивов равно количеству массивов длины $N-1$, то есть K^{N-1}.

Задача G. Шахматный турнир

Давайте мысленно проведём ребро от победителя к проигравшему, тогда для $a_i = 0$ они будут выглядеть так: $\to a_i \leftarrow$, для $a_i = 2$ так: $\leftarrow a_i \to$, а для $a_i = 1$ возможны два варианта: либо $\leftarrow a_i \leftarrow$, либо $\to a_i \to$. Ответ «`NO`» будет только в том случае, если какое-то ребро должно быть направлено в две стороны.

Заметим, что две двойки не могут стоять рядом, потому что тогда ребро между ними должно быть направлено вправо, потому что слева стоит 2, но одновременно и влево, потому что справа тоже стоит 2. Аналогично два нуля не могут стоять рядом.

А для единиц будем делать следующее: удалим единицу вместе с её левым ребром: $a_{i-1} \leftarrow a_i \leftarrow a_{i+1}$ превратится в $a_{i-1} \leftarrow a_{i+1}$, либо $a_{i-1} \to a_i \to a_{i+1}$ превратится в $a_{i-1} \to a_{i+1}$. В обоих случаях направление ребер ни для правого, ни для левого соседа не изменится, значит, ответ будет такой же, как в исходном массиве. Удалим так последовательно все единицы и посмотрим, что никакие две двойки или два нуля не стоят рядом, не забыв, что первый и последний элементы тоже являются соседними.

Задача H. Чудесная находка

Заметим, что в задаче нам дается корневое дерево с корнем в вершине 1. Алгоритм передачи шара, описанный в задаче, называется поиском в глубину или же *Depth-first search (DFS)*, а последовательность, записанная на шаре в конце — это порядок, в котором шар переходил между людьми. Иными словами, ответ — это количество различных обходов DFS на данном дереве.

Пусть $deg(v)$ — количество детей вершины v. $n! = \prod_{i=1}^{n} i$ — факториал n.

Докажем, что ответ для дерева на N вершинах равен $\prod_{i=1}^{N}(deg(i)!)$ по индукции.

База:

- Для дерева из 1 вершины ответ равен $1 = deg(1)! = 0!$.

Шаг:

- Пусть корень дерева имеет номер 1. Зафиксируем какой-нибудь порядок детей корня. Заметим, что, пойдя из корня в его ребенка, мы обойдем все вершины в поддереве этого ребенка и только потом вернемся в корень. Следовательно, обходы различных детей не пересекаются и ответ — это произведение ответов для детей корня, что по индукции равно $\prod_{i=2}^{N}$. Всего различных перестановок детей корня $deg(1)!$, следовательно, ответ равен $\prod_{i=1}^{N}$, что и требовалось доказать.

Предподсчитать факториалы всех целых чисел от 0 до N можно за линию, следовательно асимптотика решения $O(N)$.

Задача I. Большой лифт

Сначала поймём, что лифты, у которых $l_i = r_i$, не влияют на ответ, потому что в них нет смысла заходить, а значит, нет смысла и ставить камеры. Такие лифты при вводе будем игнорировать, теперь у всех лифтов $l_i < r_i$.

При $N = 1$ ответ -1, потому что человек, зашедший в здание, сразу попадает на этаж N, не используя лифты, значит, заметить его невозможно.

Теперь решение при $N \geqslant 2$. Если мы поставим камеры во все лифты, которые могут приехать на некоторый этаж x, то любой человек будет замечен, поскольку он когда-то будет проезжать этот этаж, и тогда он будет использовать один из лифтов с камерой. Но это не оптимальное решение: заметим, что для всех этажей, кроме первого и последнего, мы можем убрать камеры из тех лифтов, у которых самый верхний доступный этаж это x, и условие задачи всё ещё будет выполнено, потому что мы не сможем приехать на этаж x незамеченными, аналогично мы можем убрать все камеры из лифтов, у которых самый нижний доступный этаж это x. Важно, что мы убираем камеры только у лифтов, у которых самый нижний этаж это x или только у тех, у которых x это самый верхний этаж, но не одновременно.

Получаем следующее решение: для каждого этажа посчитаем три числа cnt_x — количество лифтов, которые могут приехать на этаж x, d_x — количество лифтов, у которых x это самый нижний этаж, u_x — количество лифтов, у которых x это самый верхний этаж. Тогда мы можем выполнить условие задачи, используя $cnt_x - max(d_x, u_x)$ камер, где $2 \leqslant x \leqslant N - 1$, переберём x и найдём минимум m, также мы можем выполнить условие задачи, используя cnt_1 или cnt_N камер, поставив камеры во все лифты на первом этаже или на последнем соответственно. Получаем, что ответ это минимум из cnt_1, cnt_N и m.

Теперь докажем, что решение, использующее меньше камер, невозможно. От обратного, пускай мы использовали меньше камер, тогда для любого этажа x существует или лифт без камеры, который ходит с этажа ниже x на этаж выше x, или два лифта без камер, один из которых приезжает на этаж x, а второй уезжает с этажа x. Значит, на каком бы этаже мы не были, мы всегда можем поехать на один этаж выше, используя лифт без камеры.

Задача J. Двоичная сортировка

Заметим, что в задаче операцию сортировки подотрезка можно представить в виде нескольких операций сортировки подотрезка длины два (например, сортировкой пузырьком). Перейдём к задаче, где мы в строке S заменяем только вхождения 10 на 01. Получается, мы можем двигать единицу вправо до тех пор, пока она не столкнётся с другой единицей.

Пусть в строке k единиц. Разберём случай, где k нечётное. Если N чётное, то ответ сразу нет, так как для одной из единиц не будет пары в палиндроме. Если N нечётное, то в центре строки должна стоять единица. В палиндроме справа и слева от центра должно быть поровну единиц, поэтому мы должны проверить, что $\frac{k+1}{2}$-я по счёту единица находится в центре строки или левее центра.

Мы можем разбить единицы на пары (если k нечётное, то $\frac{k+1}{2}$-я по счёту единица будет без пары, но мы уже выбрали ей позицию), так что i-я по порядку единица в паре с $k-i+1$-й по порядку. Если мы преобразуем строку в палиндром, то для каждой единицы её пара должна лежать напротив в этом палиндроме, то есть, если позиция единицы x, то позиция её пары в палиндроме должна быть $N-x+1$. Мы можем двигать единицы только вправо, поэтому если в изначальной строке у единицы позиция x и у её пары позиция y больше, чем $N - x + 1$, то ответ нет, ведь какие бы действия мы не совершали, мы только увеличим y или x, не изменив неравенство $y > N - x + 1$.

Теперь мы можем превратить изначальную строку в палиндром, для этого переместим последнюю по порядку единицу так, чтобы она была напротив первой, предпоследняя должна быть напротив второй, ..., i-я напротив $k - i + 1$-й, ..., $\lceil \frac{k}{2} \rceil + 1$-ю напротив $\lfloor \frac{k}{2} \rfloor$-й, и если k нечётное, то $\frac{k+1}{2}$-ю по порядку поместим в центр. В итоге мы превратили изначальную строку в палиндром, и ответ да.

Задача K. Пробег

Описанное в условии расстояние называется манхэттенским. То есть в задаче требуется для каждого человека найти количество компьютеров на заданном манхэттенском расстоянии.

Пусть мы хотим найти все компьютеры на расстоянии d от точки (x_0, y_0). Если $d = 0$, то нужно просто проверить, есть в этой точке компьютер или нет. Если $d > 0$, то все подходящие компьютеры находятся на периметре квадрата, повёрнутого на 45° и имеющего координаты вершин $[(x_0 + d, y_0); (x_0, y_0 + d); (x_0 - d, y_0); (x_0, y_0 - d)]$.

Пусть i-й компьютер стоит в точке (x_i, y_i). Заметим, что эта точка находится на прямой с уравнением $x+y = x_i+y_i$, а также на прямой с уравнением $x-y = x_i-y_i$. Для каждой прямой $x+y = C$ и для каждой прямой вида $x-y = C$ запишем в массив, какие точки на них находятся. Таким образом, мы запишем i-ю точку в общий массив для двух прямых. После записи каждого компьютера отсортируем все массивы по возрастанию x-ов.

Теперь довольно понятно, как найти количество компьютеров для человека в точке (x_0, y_0) и с расстоянием d — необходимо посмотреть на компьютеры на прямых:

- $x + y = x_0 + y_0 + d$
- $x + y = x_0 + y_0 - d$
- $x - y = x_0 - y_0 + d$
- $x - y = x_0 - y_0 - d$

А чтобы определить количество подходящих компьютеров, необходимо запустить по два бинарных поиска с соответствующими границами в массиве каждой из этих прямых. Также нужно не забыть про пересечения этих прямых: если там стоит какой-то компьютер, то мы его посчитали дважды, и его надо вычесть.

Задача L. Философский вопрос

Ключевой факт: если до какой-то вершины смогли добраться за x шагов, то сможем добраться и за $x+2$, дойдя за x шагов и пройдя вперёд и назад по любому из рёбер этой вершины. Поэтому на самом деле для каждой вершины нам важно, за сколько минимум шагов можно добраться, совершив четное или нечетное число шагов. Для этого посчитаем стандартную динамику: $dp_{0,v}$ — минимальная чётная длина пути до вершины v, $dp_{1,v}$ — минимальная нечётная длина пути до вершины v. Переходы в такой динамике следующие: из вершины v и чётности p можно перейти в вершину u и чётность $p \oplus 1$ (противоположная чётность), если между вершинами u и v есть ребро.

Давайте для массива $a_1, a_2, \ldots, a_n$ посчитаем $maxDist_{p,i}$ — какое максимальное расстояние чётности p мы можем пройти, выпив **не больше**, чем i стаканчиков. Заметим, что для фиксированной чётности с увеличением i значение $maxDist_{p,i}$ монотонно неубывает. В дальнейшем это позволит делать бинарный поиск по ответу. Как посчитать эти значения? Давайте посчитаем вспомогательный массив $maxExactDist_{p,i}$ — какое максимальное расстояние чётности p мы можем пройти, выпив **ровно** i стаканчиков. Тогда понятно, что $maxDist_{p,i} = \max_{0 \leqslant j \leqslant i} maxExactDist_{p,j}$.

Как считать значения $maxExactDist_{p,i}$? Без ограничения по чётности нам надо было бы взять i самых больших элементов. Пусть $prefSum_i$ — сумма i самых больших элементов. Тогда $maxExactDist_{prefSum_i \bmod 2,i} = prefSum_i$. Как посчитать ответ для i элементов, но противоположной чётности? Надо заменить чётность у чётного числа элементов. Так как мы взяли i самых больших элементов, то убирание любого элемента из взятых и добавление любого из невзятых может лишь уменьшить сумму, поэтому для максимизации суммы надо сделать как можно меньше таких операций, то есть одну. То есть надо либо добавить один чётный и убрать один нечётный, либо добавить один нечётный и убрать один чётный. Понятно, что всегда выгодно убирать самый маленький из возможных, а добавлять самый большой. То есть в отсортированном по невозрастанию массиве надо посчитать $next_{p,i}$ и $prev_{p,i}$ — ближайший слева и справа соответственно к i элемент чётности p, тогда $maxExactDist_{(prefSum_i \bmod 2) \oplus 1,i} = max(prefSum_i - prev_{0,i} + next_{1,i+1}, prefSum_i - prev_{1,i} + next_{0,i+1})$, если нужные предыдущий и следующий элементы существуют.

Как, имея описанные выше значения, считать ответ для какой-то вершины v? Понятно, что в ответе для этой вершины мы сделали либо чётное, либо нечётное число шагов. Посчитаем отдельно эти два случая и выберем лучший. Пусть сейчас хотим посчитать ответ для чётности p. Мы знаем, за сколько минимум шагов можем дойти до этой вершины с этой чётностью — это значение $d = dp_{p,v}$. Нас интересует минимальное такое i, что $maxDist_{p,i} \geqslant d$. В силу монотонности такое i можно искать бинарным поиском.

Задача M. Обратный к НОДу

От нас требуется найти среднее арифметическое значение указанных дробей. Для этого нужно знать их сумму и количество. Количество находится легко по формуле $\frac{N\cdot(N+1)}{2}$. Будем считать сумму дробей. Запишем требуемую сумму в более формальном виде:

$$\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{1}{(i,j)}$$

Преобразуем внутреннюю сумму. Нетрудно заметить, что если вынести $\frac{1}{i}$ за скобки, то каждая дробь преобразуется в $\frac{i}{(i,j)}$, что является делителем i. Рассмотрим один из делителей i (назовём его d). Заметим, что этот делитель встречается в сумме ровно $\varphi(d)$ раз. Таким образом можно преобразовать исходное выражение в:

$$\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{1}{(i,j)}=\sum_{i=1}^{N}\frac{1}{i}\cdot\sum_{d|i}d\cdot\varphi(d)$$

Рассмотрим произвольное число k ($1\leqslant k\leqslant N$). Посчитаем, сколько раз в общую сумму войдёт значение $k\cdot\varphi(k)$ и с какими множителями $\frac{1}{i}$. Оно войдёт в сумму ровно столько раз, сколько существует чисел i таких, что $k|i$. Известный факт — таких чисел ровно $\lfloor\frac{N}{k}\rfloor$, при этом каждое из них имеет форму $k\cdot x$, где $1\leqslant x\leqslant\lfloor\frac{N}{k}\rfloor$. С помощью этого преобразуем выражение ещё раз:

$$\sum_{i=1}^{N}\frac{1}{i}\cdot\sum_{d|i}d\cdot\varphi(d)=\sum_{k=1}^{N}k\cdot\varphi(k)\cdot\sum_{x=1}^{\frac{N}{k}}\frac{1}{k\cdot x}$$

Вынесем из внутренней суммы $\frac{1}{k}$ за скобки, тем самым получится:

$$\sum_{k=1}^{N}k\cdot\varphi(k)\cdot\sum_{x=1}^{\frac{N}{k}}\frac{1}{k\cdot x}=\sum_{k=1}^{N}k\cdot\varphi(k)\cdot\frac{1}{k}\cdot\sum_{x=1}^{\frac{N}{k}}\frac{1}{x}=\sum_{k=1}^{N}\varphi(k)\cdot\sum_{x=1}^{\frac{N}{k}}\frac{1}{x}$$

Будем вычислять последнее выражение сразу же по модулю $P=10^9+7$.

Если мы знаем все значения $\varphi(k)$ и все значения $\frac{1}{x}$, то найти ответ можно будет предельно легко. Пройдёмся решетом Эратосфена по всем числам от 2 до N и запомним для каждого наименьший простой делитель lp_i. Здесь сильнее всего подойдёт алгоритм решета Эратосфена с линейным временем работы, так как значения lp_i уже используются в этом алгоритме.

После этого посчитаем все $\varphi(i)$. Это можно сделать по следующей формуле за линейное время:

$$\varphi(i)=\begin{cases}1, & \text{если } i=1\\ \varphi(\frac{i}{lp_i})\cdot lp_i, & \text{если } lp_i^2|i\\ \varphi(\frac{i}{lp_i})\cdot(lp_i-1), & \text{если } lp_i^2\nmid i\end{cases}\tag{1}$$

Осталось научиться считать все $\frac{1}{i}$ по модулю P. Зная их все, любое $\sum_{i=1}^{\frac{N}{k}}\frac{1}{i}$ по модулю P можно найти с помощью префиксных сумм $pr_x=\sum_{i=1}^{x}\frac{1}{i}$. Это можно сделать по следующей формуле за линейное время:

$$\frac{1}{i}\equiv_P\begin{cases}1, & \text{если } i=1\\ \frac{1}{P\bmod i}\cdot(P-\lfloor\frac{P}{i}\rfloor), & \text{в противном случае}\end{cases}\tag{2}$$

Таким образом, требуемая сумма дробей равна:

$$\sum_{i=1}^{N}\sum_{j=1}^{i}\frac{1}{(i,j)}=\sum_{i=1}^{N}\varphi(i)\cdot pr_{\lfloor\frac{N}{i}\rfloor}$$

Осталось поделить посчитанную сумму на $\frac{N\cdot(N+1)}{2}$ по модулю P. Это делается с помощью малой теоремы Ферма домножением суммы на $(\frac{N\cdot(N+1)}{2})^{P-2}$. Чтобы быстро это посчитать, необходимо воспользоваться алгоритмом быстрого возведения в степень за $O(\log P)$.

Суммарная сложность данного решения равна $O(N + \log P) = O(N)$.

Задача N. Convex Hull Treap

Вначале отсортируем все точки по возрастанию x. Будем решать задачу рекурсивно — вначале найдём выпуклую оболочку множества слева, затем справа и объединим их.

Заметим, что в выпуклую оболочку точно входят самая левая точка множества (L), самая правая точка множества (R) и самая верхняя точка множества (U). Будем хранить выпуклую оболочку в трёх структурах данных: все точки выпуклой оболочки, находящиеся от L до U, в одном стеке (они будут в порядке по часовой стрелке), назовём этот стек *левой огибающей*; все точки выпуклой оболочки, находящиеся от R до U, в другом стеке (они будут в порядке против часовой стрелки), назовём этот стек *правой огибающей*; и все точки выпуклой оболочки, находящиеся от L до R, в деке (они будут в порядке против часовой стрелки), назовём этот дек *нижней огибающей*. Заметьте, что если в множестве одна точка, то во всех этих структурах лежит ровно одна эта точка, размер выпуклой оболочки равен 1, а если несколько, то L, R и U лежат в двух структурах, поэтому ответом будет сумма размеров этих структур минус 3.

При объединении есть четыре случая:

1. Если слева и справа от делегата пустые множества, то создаём два стека и дек с одной точкой, запоминаем, что ответ 1.

2. Если слева от делегата пустое множество, то создаём левую огибающую с одной точкой. После этого пытаемся добавить эту точку к правой огибающей правого множества, удаляя вершины из верха стека до тех пор, пока порядок не будет против часовой стрелки. Затем пытаемся добавить эту точку к нижней огибающей правого множества, удаляя вершины из начала дека до тех пор, пока порядок не будет против часовой стрелки. В конце пересчитываем ответ.

3. Если справа от делегата пустое множество — симметричный прошлому случай: нужно создать правую огибающую, обновить левую огибающую левого множества, обновить нижнюю огибающую левого множества, пересчитать ответ.

4. И слева, и справа от делегата непустые множества. В этом случае надо обновить левую огибающую левого множества, обновить правую огибающую правого множества. Затем надо каким-то образом объединить нижние огибающие двух множеств. Очевидно, что никакие из точек двух множеств, которые не входили в нижнюю огибающую своего множества, не могут оказаться в новом деке. Также очевидно, что делегат также не может быть частью нижней огибающей. Значит, необходимо удалить какое-то (возможно, нулевое) количество точек из конца нижней огибающей левого множества и из начала нижней огибающей правого множества.

   Будем по очереди пытаться добавить первую точку правой нижней огибающей в левую (но не добавлять на самом деле), удаляя вершины из конца левого дека до тех пор, пока порядок не будет против часовой стрелки, а затем пытаться добавить последнюю точку левой нижней огибающей в правую (но не добавлять на самом деле), удаляя вершины из начала правого дека до тех пор, пока порядок не будет против часовой стрелки. Будем делать эти удаления до тех пор, пока в каждом деке не станет по одной вершине, либо до момента, когда все точки двух деков станут в порядке против часовой стрелки. После этого надо сложить вместе эти два дека и пересчитать ответ.

С помощью подобного алгоритма можно найти ответ для каждой точки. Однако есть несколько замечаний:

- Если запоминать все стеки и деки для каждой точки, то в худшем случае на это уйдёт $O(N^2)$ памяти. Поэтому необходимо переиспользовать уже посчитанные ранее структуры. Заметьте, что если мы посчитали ответ для левого и правого множества, то конкретные структуры для

их делегатов поддерживать не нужно. Просто возьмём левую огибающую левого множества и правую огибающую правого, а в качестве нижней огибающей — одну из двух. Оставшиеся структуры мы просто не будем больше использовать. Тогда нетрудно заметить, что каждая точка побывала ровно один раз в левой огибающей, побывала ровно один раз в правой огибающей и побывала ровно один раз в нижней огибающей. При этом, если точку удаляли из одной из этих структур, то она больше не появлялась в других такого же типа. Значит, весь алгоритм с переиспользованием структур работает за $O(N)$ времени и $O(N)$ памяти.

- Самый последний шаг четвёртого случая подразумевает сложение двух деков. Однако подобное сложение, если даже оно и реализовано в языке программирования, просто добавит все элементы второго дека к первому, что противоречит нашему расчёту, что каждая точка побывала ровно в одной нижней огибающей. Вместо такого сложения необходимо как-то «подвязывать» деки друг к другу. Поэтому необходимо реализовать собственный дек на основе двусвязного списка с сложением деков за $O(1)$.

- Для удобства реализации на всех точках можно построить декартово дерево. Поскольку изначально мы отсортировали точки по возрастанию x, это можно сделать за $O(N)$. Это позволит с удобством передавать необходимые структуры вверх по дереву.

Таким образом, весь алгоритм работает за $O(N \log N)$ на изначальную сортировку точек и за $O(N)$ на нахождение ответа.

Задача O. Cavalry Battle Advanced

Решим эту задачу с помощью метода «Оценка и пример».

Если $N \leqslant 4$, можно запустить полный перебор расстановок всех коней и убедиться, что вариантов с бóльшим количеством коней нет. Ниже приведены примеры:

0
0
1

1	0
0	0
0	1

1	1	1
1	0	1
1	1	0

1	0	1	1
1	0	1	1
1	1	1	1

Для остальных случаев вначале приведём оценку. Заполним всю доску конями и будем удалять из графа вершины со всеми исходящими рёбрами. Посчитаем, сколько рёбер было изначально. Заметим, что клетки по углам поля имеют только двух соседей; клетки, которые находятся в средней строке и находятся рядом с краем поля либо через клетку от края, имеют только двух соседей; клетки, которые находятся в верхней или нижней строке и находятся через клетку от левого или правого, имеют трёх соседей; все остальные клетки имеют четырёх соседей. Покажем на примере поля 3×7:

2	3	4	4	4	3	2
2	2	4	4	4	2	2
2	3	4	4	4	3	2

Отсюда получается, что сумма степеней всех вершин в изначальном графе равна $2 \cdot 8 + 3 \cdot 4 + 4 \cdot 3 \cdot (N - 4) = 12 \cdot N - 20$, а количество рёбер тогда будет равно $6 \cdot N - 10$.

Пусть мы суммарно удалили a вершин с 4 соседями (на момент удаления), b вершин с 3 соседями, c вершин с 2 соседями, d вершин с одним соседом и e вершин без соседей, причём получилось дерево. Поскольку в дереве вершин ровно на одну больше, чем рёбер, то верно равенство:

$$(6 \cdot N - 10 - 4 \cdot a - 3 \cdot b - 2 \cdot c - 1 \cdot d - 0 \cdot e) + 1 = 3 \cdot N - a - b - c - d - e$$

$$3 \cdot N - 9 = 3 \cdot a + 2 \cdot b + c - e$$

При этом мы стремимся минимизировать количество удалённых вершин, то есть минимизировать величину $a + b + c + d + e$, но при этом так, чтобы второе равенство было верным. В самом лучшем случае, величины b, c, d и e равны 0. Тогда $a = N - 3$. Значит, нам нужно удалить хотя

бы $N - 3$ вершины (причём не из первого, второго, предпоследнего и последнего столбцов). Таким образом, ответ на задачу не больше $2 \cdot N + 3$.

Теперь приведём примеры на N от 5 до 7:

1	1	0	1	1
1	1	0	1	1
1	1	1	1	1

1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	1	1	1

1	1	1	0	1	1	1
1	1	0	0	0	1	1
1	1	1	1	1	1	1

Для $N = 8$, к сожалению, расстановки 19 коней не существует, этот случай исключение. Проверить это возможно полным перебором удаления 5 коней из 12 вершин с 4 соседями в изначальном графе (таких вариантов всего 792), ни один из них не подходит. Таким образом, для $N = 8$ ответ 18. Пример:

1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1
1	1	0	0	1	0	0	0

Для $N > 8$ ответ всегда $2 \cdot N + 3$. Есть много различных примеров, которые создаются повторением подобной расстановки одной части доски несколько раз подряд. Ниже приведён наш:

- Если $N \underset{4}{\equiv} 2$, то ответ строится подобным образом:

1	1	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1
1	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1
1	1	0	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1

- Если $N \underset{4}{\equiv} 3$, то ответ строится подобным образом:

1	1	1	1	1	1	1	0	1	1	1	1	1	1	0	1	1	1
1	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1
1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1

- Если $N \underset{4}{\equiv} 0$, то ответ строится подобным образом:

1	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
1	1	1	0	1	0	1	0	0	0	1	0	0	0	1	0	0	0	1	1
1	1	0	1	0	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1

- Если $N \underset{4}{\equiv} 1$, то ответ строится подобным образом:

1	1	0	1	0	1	0	1	1	1	1	1	0	0	0	1	1	1	1	1	1	1
1	1	1	0	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1	1	1
1	1	1	1	0	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	1	1

Здесь в каждом типе примера самая левая часть (отделена жирной линией) всегда такая для своего типа, затем идут чередующиеся блоки таблиц 3×4, причём закончить можно любым из них, а в конце каждого примера самая правая часть (отделена жирной линией) — блок таблицы 3×2, состоящий из единиц — в конце каждой таблицы.

P.S. Доказательство того, что приведённые примеры являются деревьями, оставляем читателям в качестве упражнения.

P.P.S. Придумать свой собственный пример можно было, запустив динамику по профилю с восстановлением ответа. Это решение не найдёт ответ для больших N, но с его помощью можно найти закономерность для ответов большего размера.