Разбор задачи «Лара Крофт и клад»

Самая простая задача тура. В ней надо просто вывести 0.

Начертив траекторию или сложив соответствующие векторы, несложно убедиться, что Лара Крофт пришла в ту же точку. Пример к задаче мог служить подсказкой.

Разбор задачи «История ICPC»

Так как в предыдущем сезоне в команде участвовало 4 участника, то всего на данный момент тренируются $4K - L + M$ участников. Количество участников, остающихся без команды — это остаток от деления $4K - L + M$ на 3. Если остаток равен 0, никого дополнительно привлекать не надо. Если остаток равен 1, надо привлечь двух участников, если остаток равен 2 — одного участника.

Также ответ можно записать формулой через операцию взятия остатка, но в этом случае надо учитывать особенности реализации этой операции в используемом языке программирования.

Разбор задачи «Петя, Маша и верёвочки»

В случае, если длина наибольшей найденной на столе верёвочки строго больше, чем сумма остальных, Маша могла взять одну из коротких верёвочек, и ответом будет разность между длиной максимальной верёвочки и суммой остальных. Иначе Маша могла взять только оставшуюся длинную верёвочку, и ответом будет сумма длин всех найденных на столе верёвочек.

Разбор задачи «Многозначность»

Наименьшее K-значное число в B-ичной системе счисления имеет вид «единица и $K-1$ нулей»; соответственно, равно B^{K-1}. Аналогично, наименьшее $K+1$-значное равно B^K, следовательно, предшествующее ему число $B^K - 1$ является наибольшим K-значным.

Тем самым задача сводится к следующей: даны два отрезка с концами $B_1^{D_1-1}$ и $B_1^{D_1} - 1$ и $B_2^{D_2-1}$ и $B_2^{D_2} - 1$, найти количество целых точек в их пересечении. Для этого найдём самый правый из левых концов отрезка L_r и самый левый из правых R_l. Если первый левее правее второго, ответ 0, иначе ответ $R_l - L_r + 1$.

Разбор задачи «Неактуальный баннер»

Для каждой буквы C количество способов прочтения слова ACM, в которую она входит, равно количеству соседних с буквой C букв A, умноженному на количество соседних с ней букв M (любые два способа, в которых различается первая буква, различны, равно как и любые два способа, в которых различается последняя буква; так как выбор из возможных соседних A и M независим, то для получения общего количества прочтений для заданной буквы C результаты подсчёта надо перемножить). Если просуммировать получившиеся результаты для всех букв C, то мы получим требуемый ответ.

Разбор задачи «Гиперсферы»

Обозначим расстояние между центрами за D.

Возможны следующие случаи:

- $D = 0$. Тогда если радиусы равны, гиперсферы совпадают и ответ равен -1, если радиусы различны, гиперсферы не пересекаются и ответ равен 0.
- Расстояние между центрами меньше одного из радиусов или равно ему. Тогда центр одной из гиперсфер находится внутри или на границе другой. В этом случае если разность большего и меньшего радиусов равна D, гиперсферы касаются изнутри, и ответ равен 1, если разность больше D, гиперсферы не имеют общих точек, и ответ равен 0, а если разность меньше D, ответ равен 2 в случае двумерного пространства (так как окружности пересекутся в двух точках) и -1 в случае большей размерности (так как пересечение будет по гиперсфере размерности $N-1$, содержащей бесконечное количество точек).

- Расстояние между центрами больше любого из радиусов. Тогда центр каждой гиперсферы находится вне другой. Если сумма радиусов равна D, имеем внешнее касание и ответ 1, если сумма радиусов меньше D, гиперсферы не пересекаются, если сумма радиусов больше D, гиперсферы пересекаются в двух точках в случае двумерного пространства и в бесконечном количестве случае в случае $N > 2$.

Отметим, что так как в сравнениях всегда в одной из частей находится D, а в другой — сумма или разность целых чисел и обе части сравнения неотрицательны, можно возвести обе стороны в квадрат и работать в 64-битных целых (так как максимальное значение суммы разностей координат равно $100 \cdot (2 \cdot 10^5)^2) = 4 \cdot 10^{12}$, что с гарантией помещается в 64-битный целочисленный тип.

Разбор задачи «Баскетбольная статистика»

Для решения задачи было достаточно прочитать протокол, преобразовать данные в целочисленный тип (то есть заменить имена игроков их номерами), для всех видов бросков подсчитать их результат (0, если не попал, или набранное количество очков, если попал), после чего пройти по полученному массиву, выполняя следующие действия:

- если событие — бросок с ненулевым результатом, то сделавшему бросок игроку увеличивается сумма очков, и в случае, если два предыдущих события — владение с одним и тем же номером команды, игроку из самого раннего из этих событий засчитываем передачу;
- если событие — владение, то если предыдущее событие — бросок с нулевым результатом, то засчитываем подбор, а если владение с другим номером команды — перехват.

Разбор задачи «Игра с матрицей»

Назовем дополнительным минором $M_{i,j}$ матрицы A определитель матрицы, полученной вычеркиванием из матрицы A i-й строки и j-го столбца. Обозначим определитель матрицы A как $\det(A)$. Можно доказать, что $\det(A) = \sum_{i=1}^{n} (-1)^{1+i} A_{1,i} \cdot M_{1,i}$.

Давайте прибавим единицу к элементу $A_{1,j}$ и посмотрим, как изменится определитель. Из приведённой выше формулы очевидно, что он изменится на $D = (-1)^{1+j} \cdot M_{1,j}$. Мы можем легко найти D как разность нового и старого определителя матрицы.

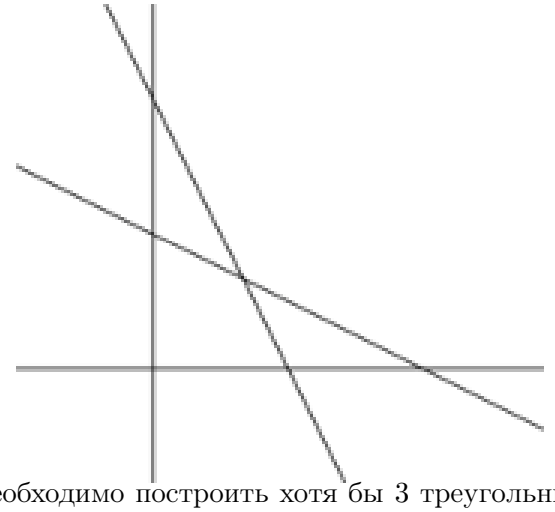
Пусть $D \neq 0$. Обозначим за B матрицу, полученную после нашей операции. Нам известно, что $\det(A) + D = \det(B)$. Давайте теперь к элементу $B_{1,j}$ прибавим число $-\frac{\det(B)}{D}$. Так как мы меняли только элементы первой строчки, то все используемые нами дополнительные миноры не изменились, а значит определитель полученной матрицы будет равен $\det(B) - \frac{\det(B)}{D} \cdot (-1)^{1+j} \cdot M_{1,j} = \det(B) - \frac{\det(B)}{D} \cdot D = \det(B) - \det(B) = 0$, что и требовалось.

Так как изначально определитель был не равен 0, то существует такое j, что $M_{1,j} \neq 0$, нужно лишь его найти. Для этого достаточно попытаться прибавить единицу к каждому элементу первой строки. Итоговое количество действий не превышает 21, что укладывается в ограничения задачи.

Разбор задачи «Прямые и треугольники»

Если $n = 3$, то единственный случай, когда ответ существует, это $k = 1$. Доказательство этого факта очевидно, поэтому перейдем к следующему случаю.

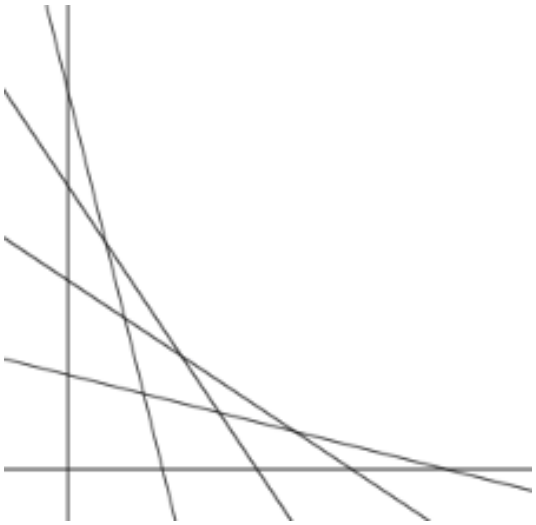
Если $n = 4$, то единственный случай, когда ответ существует, это $k = 2$. Пример:



Если $k > 2$, то нам необходимо построить хотя бы 3 треугольника, которые суммарно имеют 3 ребра. Значит, всего 9 ребер и, по принципу Дирихле, будет существовать прямая, содержащая в себе 3 ребра. Однако это невозможно, поскольку прямую три другие могут разбить не более чем на два отрезка конечной длины.

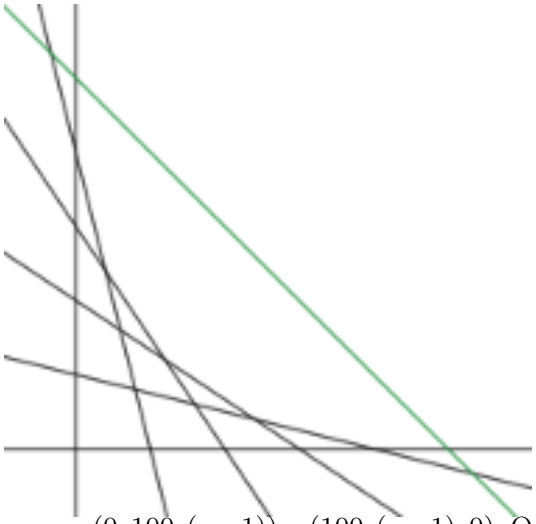
Во всех остальных случаях ответ существует.

Пусть $k = n - 2$. Тогда пример строится следующим образом:



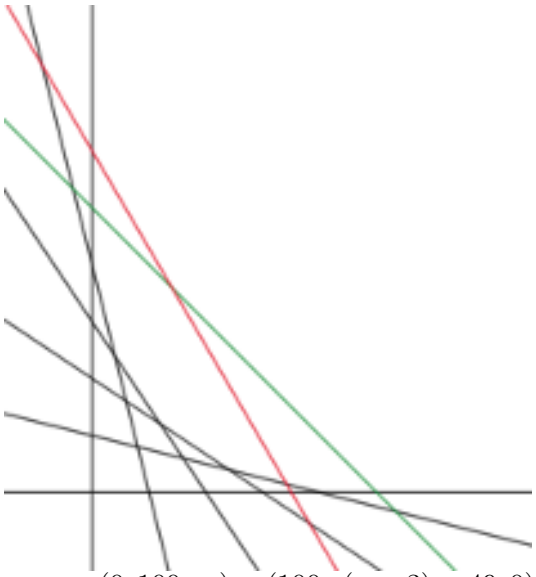
Проводятся две прямые, совпадающие с осями координат. Затем проводится $n - 2$ прямые, i-я из которых проходит через точки с координатами $(0, 100 \cdot i)$ и $(100 \cdot i, 0)$. Нетрудно убедиться, глядя на картинку, что всего будет ровно $n - 2$ треугольника.

Пусть $k = n - 1$. Тогда пример получается из предыдущего добавлением одной прямой:



Прямая проходит через точки $(0, 100 \cdot (n-1))$ и $(100 \cdot (n-1), 0)$. Однако следует быть осторожным, поскольку новая прямая может быть параллельна одной из ранее проведённых. Поэтому имеет смысл передвинуть одну из ее точек, например первую: $(0, 100 \cdot (n-1) - 1)$.

Пусть $k = n$. Тогда пример получается из предыдущего добавлением одной прямой:



Прямая проходит через точки $(0, 100 \cdot n)$ и $(100 \cdot (n-3) - 49, 0)$.

Разбор задачи «Камни, компьютер и вечность»

Для начала, научимся определять, является ли позиция выигрышной для игрока.

Смоделируем первый ход компьютера. Пусть количество оставшихся камней в кучках равно `a b c`, причем, без ограничения общности, $a \leqslant b \leqslant c$. Есть три случая:

1. a. В таком случае компьютер сразу выиграл.
2. $2^t \cdot a = b = c$, для некоторого $t \geqslant 0$. Тогда, если игрок своим ходом выберет кучку b или c, то последовательность ходов компьютера будет следующей:

$$(a; 2 \cdot b; b) \to (a; b - a; b) \to (a; b - a; 0)$$

   То есть, компьютер, в таком случае, выиграет.

   Если же игрок своим ходом будет все время выбирать первую кучку, то через t шагов он попадет в ситуацию, где количество камней во всех кучках одинаково, и проиграет, так как после любого его хода компьютер ту кучку, которую удвоил игрок.

3. В остальных случаях игрок может играть так, чтобы компьютер не сделал ни одного хода. Заметим, что если компьютер больше ходить не будет, то достичь одинакового количества камней во всех кучках не получится, так как для этого их количество должно различаться в $k = 2^t$ раз. Случай, когда $b = c$ рассмотрен во втором пункте, поэтому $b < c$. Но тогда $a, b \leqslant \frac{c}{2}$, то есть компьютер мог сделать еще один ход, противоречие.

   Заметим, что компьютер не может ходить тогда и только тогда, когда количество камней в кучках удовлетворяет неравенству треугольника. На каждом ходу игроку следует удваивать кучку, количество камней в которой минимально. Тогда, если $2 \cdot a \leqslant c$, то $2 \cdot a + b > c$ и неравенство треугольника выполнено. Иначе $2 \cdot a < b + c$, так как $a \leqslant b$ и $a \leqslant c$, и хотя бы одно из неравенств строгое.

   Получается, что в этом случае игрок побеждает.

Давайте теперь посчитаем количество позиций, в которых побеждает компьютер. Нам известно, что после первого хода все такие позиции переходят в позицию одного из двух типов:

1. $(a; a \cdot 2^t; a \cdot 2^t)$
2. $(0; a; a)$

Для каждой позиции можно посчитать, из какого количества позиций она достижима. Для этой цели можно завести функцию f, которая по трём числам посчитает количество искомых позиций простым перебором.

Мы знаем общее количество позиций, а также количество позиций, в которых побеждает компьютер. Несложно посчитать количество позиций, в которых побеждает игрок.

Разбор задачи «Игра с массивом»

Воспользуемся классической для данного типа задач идеей восстановления k-го комбинаторного объекта.

Для каждого суффикса и для каждого элемента посчитаем, сколько существует подпоследовательностей на текущем суффиксе и которые начинаются на выбранный элемент. Это можно делать при помощи динамического программирования на суффиксе.

Очевидно, что когда мы к суффиксу добавляем слева элемент a, меняется только количество подпоследовательностей, начинающихся на a. Любую подпоследовательность, начинающуюся с числа a можно перенумеровать так, чтобы она начиналась именно с добавленного элемента. Поэтому новое количество подпоследовательностей, начинающихся на a есть общее количество подпоследовательностей предыдущего суффикса + 1 — пустая подпоследовательность. Количество подпоследовательностей предыдущего суффикса можно посчитать как сумму элементов нашей динамики.

Зная эту динамику для каждого суффикса можно легко восстановить искомую подпоследовательность. На каждом шагу можно бинарным поиском на префиксных суммах динамики искать очередной элемент подпоследовательности и переходить к нему.

Однако, такое решение работает за $O(n^2)$.

Можно заметить, что на очередном шагу подсчёта динамики меняется только одно число, поэтому достаточно хранить один массив для динамики, и на каждом шагу помнить, какое число мы изменили.

Чтобы ускорить бинарный поиск на префиксных суммах нашего массива можно использовать структуру данных «дерево отрезков», которая позволяет делать эти операции за $O(\log(n))$.

Итоговая асимптотика решения получается $O(n \cdot \log(n))$

Разбор задачи «Граф. Просто граф...»

Пусть $G = (V, E)$ - это исходный граф. Запустим в нем поиск в глубину, и пусть $T = (V, E')$ - корневое дерево, которое обошел поиск. Тогда по свойствам поиска, каждое из ребер E E' соединяет какую-то из вершин с ее предком в T. Назовем такие ребра *дополнительными*. Также назовем вершину *помеченной*, если она инцидентна одному из дополнительных ребер и/или является наименьшим общим предком двух инцидентных дополнительным ребрам вершин. Так как $|E'| = n - 1$,

а $|E| \leqslant n + 4$, то дополнительных ребер не более пяти, а помеченных вершин - не более 14, или $3(m - n + 1) - 1$; назвем это число k.

Помеченные вершины устроены таким образом, что любой путь между двумя помеченными вершинами, не содержащий промежуточных вершин, есть либо дополнительное ребро, либо путь по T, причем такой путь единственнен. Найдем все такие пути и построим граф $H = (V_H, E_H)$, вершинами которого будут помеченные вершины, а ребра будут соответствовать путям без промежуточных вершин. Отметим, что две вершины G' могут быть соединены не более чем двумя кратными ребрами. Таким образом, в G' будет k вершин и не более чем $k - 1 + (m - n + 1) = O(k)$ ребер.

Заметим, что с помощью стандартной динамики по профилю мы за $O(2^k * k)$ сможем найти количества простых путей в G' от одной вершины до всех остальных. Насчитаем следующие массивы:

- $paths[v_1][v_2]$, $v_1, v_2 \in V_H$ - количества простых путей в G' от v_1 до v_2;
- $paths2[e][v]$, $e \in E_H$, $v \in V_H$ - сумма количеств простых путей от v_1^e до v и до v_2^e до v, не проходящих через ребро e, соединяющее вершины v_1^e и v_2^e;
- $paths3[e_1][e_2]$, $e_1, e_2 \in E_H$ - количества простых путей, соединяющих один из концов ребра e_1 с одним из концов ребра e_2 и не проходящих через ребра e_1 и e_2.

Теперь временно удалим помеченные вершины из графа G; тогда граф распадется на компоненты связности, каждая из которых является поддеревом T. Найдем для каждой компоненты $comp$ список $lst = lst(comp)$ помеченных вершин, соединенных ребром с вершинами $comp$ в графе G. Из устройства помеченных вершин очевидно, что $lst(comp)$ содержит не более двух вершин. Также определим $comp$ для помеченной вершины v как $\{v\}$. Пусть также $v(comp)$ есть вершина списка $lst(comp)$, если она единственна; если же $|lst(comp)| = 2$, то пусть $e(comp)$ есть ребро в графе G', соединяющее вершины $lst(comp)$.

Поймем теперь, как ответить на запрос для вершин v_1 и v_2. Пусть $comp_1$ и $comp_2$ есть компоненты, в которых лежат v_1 и v_2 соответственно. Тогда ответ может быть получен следующим образом:

- Если $|lst(comp_1)| = |lst(comp_2)| = 1$, то ответ есть $paths[v(comp_1)][v(comp_2)]$;
- Если $|lst(comp_1)| = 1$, а $|lst(comp_2)| = 2$, то ответ есть $paths2[e(comp_2)][v(comp_1)]$;
- Если $|lst(comp_1)| = |lst(comp_2)| = 2$, то:
  - если $comp_1 \neq comp_2$, то ответ - это $paths3[e(comp_1)][e(comp_2)]$;
  - если $comp_1 = comp_2$, то пусть $w_1 \in V$ есть минимальный предок v_1, лежащий на пути в T, соединяющем вершины $lst(comp_1)$. Аналогично определим w_2 для v_2. Тогда если $w_1 = w_2$, то тогда любой путь между v_1 и v_2 будет проходить через наименьшего общего предка v_1 и v_2, являющегося потомком w_1 или самой v_1; в силу этого, ответ 1. Если же $w_1 \neq w_2$, то можно единственным способом провести два непересекающиеся вершинно пути из v_1 и v_2 в вершины $lst(comp_1)$; если это u_1 и u_2, то ответ есть $1 + (paths2[e(comp_1)][u_2] - 1) = paths2[e(comp_1)][u_2]$ (мы учли путь «напрямую», а также вычли путь, соединяющий u_2 с собой же).

Итоговая асимптотика решения $O(n\ log\ n + 2^k * k^3)$ (массивы $paths$, $paths2$, $paths3$ ищем за $O(2^k * k^3)$).