Задача A. Катя и сломанная клавиатура

Автор задачи: Екатерина Ведерникова, разработчик: Николай Ведерников

Проходя по строке будем поддерживать $cur[c]$ — сколько раз встретилась буква c. Всего возможно три случая:

1. буква c не сломанная, тогда к ответу добавляем 1,
2. буква c сломана и количество $cur[c]$ делится нацело на $(x[c]-1)$, тогда следующее нажатие не сработает, значит к ответу добавляем 2,
3. буква c сломана и количество $cur[c]$ не делится нацело на $x[c]-1$, тогда следующее нажатие сработает,значит к ответу добавляем 1.

Задача B. Кооперативная игра на дереве

Автор задачи: Екатерина Шиляева, разработчик: Иван Волков

Для решения данной задачи можно воспользоваться методом динамического программирования (более конкретно — методом динамического программирования по поддеревьям). Введем dp_v следующим образом:

- $dp_v = 0$, если v — лист
- иначе, dp_v равняется ответу на задачу, если игра начинается в вершине v.

Обозначим теперь за d_a глубину вершины a, а за $S_{v,d}$ — множество вершин, лежащих в поддереве вершины v и находящихся на глубине d (все глубины будем считать относительно всего дерева). Несложно убедиться, что если начинать игру в вершине v, и первым листом, в который придет красная фишка, окажется вершина r, то максимальное количество фишек, которое можно получить за игру, есть $1 + \max\limits_{b \in S_{v,d_r}} dp_b$ (в данной формуле b отвечает за ту вершину, в которой будет синяя фишка, в момент когда красная придет в r). Значит, dp_v будет равен максимальному значению этого выражения по всем листьям r из поддерева v.

Заметим теперь, что если b_1 — предок (не обязательно непосредственный) b_2, то выполнено $dp_{b_1} \geqslant dp_{b_2}$. Действительно, если существует стратегия, при которой начиная из вершины b_2 можно добиться в конце k фишек на дереве, то для вершины b_1 можно добиться такого же результата, переведя обе фишки в b_2 и воспользовавшись далее этой стратегией. Из этого следует, что для ответа достаточно рассмотреть $1 + \max\limits_{b \in S_{v,dr_v}} dp_b$, где dr_v — наименьшая глубина из всех листьев поддерева v (т.к. все вершины, в которых могла переродиться красная фишка, либо лежат в S_{v,dr_v}, либо являются потомками какой-то вершины из этого множества, и тогда значение динамики для них не оптимально).

Итого, чтобы посчитать dp_v, достаточно найти глубину dr_v самого неглубокого листа в поддереве v, а дальше взять максимальное значение динамики по всем вершинам поддерева v, лежащим на глубине dr_v и прибавить к нему 1. Значения dr_v легко насчитать для всех вершин дерева одним обходом в глубину. Далее, можно по ходу пересчета динамики для каждой глубины исходного дерева хранить массив, в который выписывать значения динамики по всем вершинам на этой глубине в порядке обхода. Тогда для всякой вершины v, S_{v,dr_v} будет соответствовать некоторому подотрезку такого массива для глубины dr_v. Точные границы этого подотрезка можно найти бинарным поиском, для это также понадобится для каждой вершины насчитать ее времена входа и выхода tin_v и $tout_v$. Для быстрого поиска минимума на отрезке необходимо воспользоваться структурой данных, например, деревом отрезков. В таком случае итоговое время для подсчета всех значений динамики составит $\mathcal{O}(n \log n)$, ответом на задачу будет являться dp_1.

Задача C. Экзамен в ГИББД

Автор и разработчик задачи: Никита Голиков

Для начала, давайте научимся проверять, что текущего количества льда достаточно для покрытия всех дорог. Построим двудольный граф: в левой доле будут все вершины из отрезка, в правой доле будут все ребра из отрезка. Для каждой вершины и ребра создадим количество копий, равное значению.

Добавим ребра между копиями вершины и копиями инцидентных ей ребер. Тогда покрытие дорог льдом эквивалентно нахождению паросочетания в этом графе. Тогда получается, что возможно покрыть дороги, если существует паросочетание в этом графе, покрывающее всю правую долю.

Для проверки этого утверждения можно воспользоваться леммой (или теоремой) Холла. Зафиксируем подмножество вершин правой доли (иными словами, ребер), тогда количество вершин в левой доле, из которых есть ребро в выбранное подмножество, должно быть не меньше его размера. Заметим, что если мы взяли какую-то копию вершины в правую долю, то можно взять все, так как это только усилит утверждение. Так же заметим, что выбранные нами вершины правой доли разбиваются на циклические подотрезки, и для них множества соседей не пересекаются. Тогда достаточно проверить утверждение для всех циклических подотрезков. Получается следующее утверждение: можно покрыть все дороги, если для любого циклического подотрезка ребер сумма их значений d_i не больше, чем сумма значений w_i соответствующих им вершин.

Для решения исходной задачи, обобщим утверждение леммы Холла. Найдем множество вершин в правой доле, для которого разность его размера и количества соседей максимальна, обозначим эту разность за x. Если $x \leqslant 0$, то мы можем покрыть все дороги. Иначе, нам нужно добавить суммарно хотя бы x единиц льда. Утверждается, что этого количества достаточно. Это утверждение уже менее известно, иногда его называют обобщенной леммой Холла. Например, это можно доказать, используя теорему Форда-Фалкерсона о равенстве максимального потока и минимального разреза.

Используя ранние наблюдения, получается, что нам нужно максимизировать по всем способам выбрать непересекающиеся циклические подотрезки ребер суммы $d_i - w_j$ по ребрам и вершинам выбранных отрезков. Для решения данной задачи можно использовать дерево отрезков. Заметим, что если мы будем сканировать ребра отрезка слева направо, то каждый раз у нас либо открыт текущий отрезок, либо закрыт, и в зависимости от этого мы пересчитываем текущую сумму, и либо закрываем, либо открываем новый отрезок. Так же, нужно не забыть учесть ребро, замыкающее цикл. Тогда в вершине дерева отрезков достаточно хранить матрицу 2×2, хранящую ответ, при условии, что до отрезка ДО у нас либо открыт, либо закрыт отрезок, и после обработки он либо открыт, либо закрыт. Для объединения двух вершин нужно перебрать тип начала левой вершины, конца правой, и конца левой. Получается решение за $O(q \log n)$.

Задача D. Посчитай Ёлки

Автор и разработчик задачи: Егор Юлин

Решим более простую задачу: посчитаем количество способов добавить n-й слой, если у нас уже есть $n-1$.

Будем добавлять рёбра слева-направо. Тогда пусть уже проведено i ребер из j вершин $n-1$ слоя, если j не последняя вершина $n-1$ слоя, то из $j+1$-й вершины мы можем провести от 0 до 2-х рёбер, то есть сделать i, $i+1$ или $i+2$ рёрер и $j+1$ вершин.

В таком случае будем считать следующее двухмерное дп: $\texttt{dp}[i][j]$, где i – количество проведённых рёбер, а j – количество вершин, из которых были проведены рёбра. Пересчёт динамики будет следующим

$$\texttt{dp}[i][j] = \texttt{dp}[i][j-1] + \texttt{dp}[i-1][j-1] + \texttt{dp}[i-2][j-1]$$

Ответ для такой задачи будет находится в $\texttt{dp}[n][n-1]$.

Решим исходную версию задачи: нужно посчитать количество ёлок высоты n. Мы умеем считать количество способов добавить новый слой в ёлку. Заметим, что слои независимы, тогда ответом

будет произведение количество способов добавить $2, 3, \ldots, n$ слои. Через дп это значение равно

$$\texttt{dp}[2][1] \cdot \texttt{dp}[3][2] \cdot \ldots \cdot \texttt{dp}[n][n-1]$$

Таким образом можно предпосчитать значения дп, и вывести нужное произведение.

Задача E. Не всё так многозначно

Автор и разработчик задачи: Олег Христенко

Пусть x — загаданное число, b — основание системы счисления, n — количество цифр в b-ичной записи числа x.

Тогда найдём двоичным поиском такое минимальное d_0, что $x + d$ имеет больше знаков, чем x. Очевидно, что это будет число вида b^n (то есть первое $n+1$-значное число в b-ичной системе счисления).

Докажем, что в качестве верхнего предела достаточно взять 10^{12}: если $b = 1000$, а $x = 10^9$, то $n = 4$ и ближайшее пятизначное число как раз 10^{12}; если $b < 1000$, то, так как число $x \cdot b$ будет иметь на единицу больше знаков, чем x, то $x \cdot b \leqslant 10^9 \cdot b < 10^9 \cdot 1000 = 10^{12}$. Если $b > 1000$, то число не может иметь более трёх разрядов, то есть $x + d$ будет не более $b^3 \leqslant 2023^3 < 10^{12}$.

После этого ищем двоичным поиском минимальное D такое, что $x + d_0 + D$ имеет больше знаков, чем $x + d_0$. Это будет b^{n+1} (то есть первое $n+2$-значное число в b-ичной системе счисления). Тогда $b^{n+1} - b^n = D$, зная n и D, находим b, а затем находим x. При втором двоичном поиске в качестве максимума достаточно взять 10^{15} (доказательство полностью аналогично доказательству для предыдущей верхней границы).

Тем самым при первом двоичном поиске получается не более $log_2(10^{12}) < log_2(2^{40}) = 40$ запросов, при втором — не более $log_2(10^{15}) < log_2(2^{50}) = 50$, в сумме не более 90 запросов.

Задача F. Магический квадрат

Автор и разработчик задачи: Николай Ведерников

Заметим, что сумма в каждой строке и в каждом столбце равна $\frac{n \cdot (n^2+1)}{2}$.

Будем называть строку или столбец *плохим*, если сумма не будет равна $\frac{n \cdot (n^2+1)}{2}$.

После того, как поменяли два числа местами, *плохими* могли стать максимум два столбца или две строки, потому что остальные не изменялись.

Тогда возможны три случая:

1. Два *плохих* столбца и ноль *плохих* строк, тогда числа которые поменял Гаргамель находятся в одной строке. Переберём строку и проверим, что числа которые стоят на пересечении строки и двух *плохих* столбцов, при обмене сделают квадрат вновь магическим.

2. Ноль *плохих* столбца и две *плохие* строки, аналогично предыдущему пункту.

3. Два *плохих* столбца и две *плохие* строк. Тогда рассмотрим четыре числа, которые стоят на пересечении *плохих* строк и *плохих* столбцов. Среди них поменяли числа расположенные в противоположных углах получившегося прямоугольника. Выберем те два числа, которые при обмене сделают квадрат вновь магическим.

Задача G. Случай в казино

Автор и разработчик задачи: Александр Понкратов

Разобьем все числа на классы по остатку от деления на 3. Обозначим за cnt_i количество чисел с остатком i $(0 \leqslant i < 3)$. Будем считать, что чисел с остатком равным 0 пока нет.

Если заменить все числа на их остаток от деления на 3, то первому элементу последовательности однозначно задается и вся последовательность. Возможных вариантов всего два: $1, 1, 2, 1, 2, 1, 2, \ldots$

и $2, 2, 1, 2, 1, 2, 1, \ldots$. Последовательность начинается с одного из чисел такого класса, что их количество больше. Способов расставить $cnt_1! \cdot cnt_2!$. Также отметим, если $cnt_1 = cnt_2$ или разница между cnt_1 и cnt_2 больше 2, то ни один способ не подойдет.

Вернем в массив числа с остатком, равным 0, такие числа не изменяют сумму на префиксе по модулю 3, поэтому их можно расставить на любые позиции, кроме первой. Таких способов $cnt_0! \cdot C_{n-1}^{cnt_0}$.

Итоговый ответ: $cnt_1! \cdot cnt_2! \cdot cnt_0! \cdot C_{n-1}^{cnt_0}$.

Задача H. Самокатные номера

Автор задачи: Константин Кохась, разработчик: Маргарита Саблина

Пусть d_{nk} — число разбиений числа n на слагаемые, для которых $mex \geqslant k$. В таких разбиениях обязательно присутствуют числа от 1 до $k-1$.

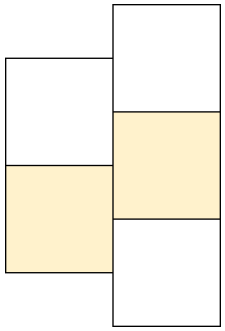
При $k = 1$ в d_{n1} каждое разбиение с $mex = 1$ встречается однократно. При $k = 2$ в d_{n1} каждое разбиение с $mex = 2$ встречается однократно, в d_{n2} каждое разбиение с $mex = 2$ встречается однократно. При $k = p$ каждое разбиение с $mex = p$ встречается однократно с d_{n1} по d_{np}, поэтому, если суммировать $\sum_{i=1}^{p} dp_{ni}$, то каждое значение i будет повторяться в сумме столько столько раз, сколько будет разбиений с $mex = i$.

Подсчет можно осуществить методом динамического программирования. $d[i][k]$ — число разбиений числа i на слагаемые, максимальное число в разбиении $\leqslant k$. Формула пересчёта $d[i][k] = d[i-k][k] + d[i][k-1]$. Искомая в задаче сумма для всех $k \geqslant 1$, $n - \frac{k\cdot(k-1)}{2} \geqslant 0$ — $\sum_k dp[n - \frac{k\cdot(k-1)}{2}][n]$ — разбиение с $mex = k$ содержит все числа от 1 до $k-1$, поэтому для каждого k нужно суммировать разбиения числа $n - \frac{k\cdot(k-1)}{2}$, максимальное число в разбиении $\leqslant n$.

Задача I. Квадратики

Автор и разработчик задачи: Александр Бабин

Для начала нам нужно понять, как выглядит *покрытие*. Четыре простых варианта получаются, если все квадратики располагаются ровной сеткой. Иначе найдутся два квадратика, которые соприкасаются неровно, например, как на картинке:



Если есть пара квадратов, как на картинке, то в двух столбцах, где находятся эти квадраты, все остальные квадраты должны идти последовательно, примыкая друг к другу, с шагом в 2 единицы. Таким образом нашелся столбец, который заполнен столбцами. Рассуждая в таком ключе и дальше можно заметить, что все квадраты из *покрытия* разбиваются на столбцы. Нетрудно понять, что и в других случаях *покрытие* будет разбиваться на строки или столбцы, покрытые квадратами.

Итак, вся наша бесконечная таблица разбивается на строки или столбцы 4-мя способами (выбираем, на что разбивается таблица: строки или столбцы, и в каждом из двух выбранных вариантов также надо выбрать какие столбцы/строки встретятся в покрытии: четные или нечетные). Мы будем искать наибольшее *хорошее* подмножество, которое дополняется до каждого из четырех вариантов *покрытия* независимо.

Если таблица разбивается на столбцы, то для каждого из столбцов мы можем поддерживать количество квадратов, которые встречаются в нем на четной и нечетной высоте. Тогда понятно, что

в наибольшем *хорошем* подмножестве в этом случае и в этом столбце встретится наибольшее количество из двух посчитанных. Далее задача добивается простой техникой и решается за $O(n \log n)$, если используется `std::map` и решается за $O(n)$, если используется хеш-таблица.

Задача J. Улицы Флатландии

Автор задачи и разработчик: Даниил Орешников

Первый факт, который следует заметить — количество магистралей в точности равно количеству дорог $n - 1$ минус количество пар совместимых дорог, которые мы помещаем в одну магистраль. Действительно, каждое действие вида «соединить магистраль, заканчивающуюся в дороге (u, v) и магистраль, начинающуюся в дороге (v, w), в одну» уменьшает количество магистралей на 1. Будем последовательно выбирать пары совместимых дорог и объединять их магистрали — тогда количество магистралей будет равно $(n-1) -$ `<кол-во действий>`, а каждое действие соответствует паре совмещаемых дорог.

Таким образом, мы свели задачу к поиску максимального количества пар дорог, которые можно совместить по описанным в условии правилам. Более того, эту задачу можно решать независимо в окрестности каждой вершины v: если дороги (a, b), (b, c) и (c, d) образуют одну магистраль, то при фиксированной (b, c) выбор a не зависит от выбора d. То есть, проще говоря, достаточно для каждой вершины v найти максимальное паросочетание на дорогах, исходящих из нее.

Чтобы найти максимальное паросочетание на дорогах, исходящих из вершины v, стоит сразу отметить, что у каждой дороги есть не более двух совместимых с ней. Действительно, определение совместимости простыми словами формулируется как «вектор $\overrightarrow{vw}$ должен быть ближайшим к $\overrightarrow{uv}$ при повороте вокруг v». Ближайший может быть только один при повороте по часовой стрелке и один — против.

Теперь просто отдельно рассмотрим каждую вершину v и дороги вокруг нее. Построим на этих дорогах граф, где две дороги будут связаны ребром, если они совместимы. Для этого достаточно отсортировать вектора дорог по их углу, после чего либо воспользоваться методом двух указателей, либо искать для каждого вектора $\overrightarrow{uv}$ соседние по углу дороги с помощью двоичного поиска. После остается только найти максимальное паросочетание на полученном графе. Но в нем степень каждой вершины не превосходит 2, а значит граф разбивается на пути и циклы. В таком графе размер максимального паросочетания в каждой компоненте связности размера x равен $\left\lfloor \frac{x}{2} \right\rfloor$.

Общее время работы такого решения — $\mathcal{O}(n \log n)$.

Задача K. Строчечка

Автор и разработчик задачи: Александр Бабин

Введем две стратегии для угадывания строки: F (если про строку ничего не известно) и D (если известно, что первый символ строки равен `b` или `c`). Максимальное количество запросов для строки длины n, если пользуемся соответствующей стратегией обозначим как F_n и D_n.

- $n = 0$. Фиктивные состояния F и D требуют по 0 действий;
- $n = 1$. Мы понимаем, что слева от последнего символа есть еще один, который мы уже знаем. Соответственно в стратегии F надо сделать 2 запроса, прикладывая к суффиксу строки `ab` и `ac`, а в стратегии D достаточно одного запроса `ab`. То есть $F_1 = 2$ и $D_1 = 1$.
- $n \geqslant 2$, стратегия F. Сделаем пару запросов `aa` и `ba`, приложив эти строки к началу строки. Исходя из ответов на эти запросы мы однозначно восстанавливаем первый символ строки понимаем, равен ли второй символ `a`, таким образом $F_n = 2 + \max\{F_{n-2}, D_{n-1}\}$.
- $n \geqslant 2$, стратегия D. Сделаем запрос `ba`.
  - Получили 0, тогда мы восстановили первый символ и можем перейти к стратегии D_{n-1};
  - Получили 1, тогда возможны следующие варианты для двух первых символов строки: ca, bc, bb, сделаем запрос `bb` и восстановим первые два символа строки, перейдя к стратегии F_{n-2}.

– Получили 2, тогда мы можем перейти к стратегии F_{n-2}.

Таким образом $F_n = 2 + \max\{F_{n-2}, D_{n-1}\}$ и $D_n = \max\{2 + F_{n-2}, 1 + D_{n-1}\}$. Но учитывая, что $F_1 \leqslant F_2 \leqslant F_3 \leqslant \ldots$ можем заключить, что:

$$F_n = 2 + max\{F_{n-2}, D_{n-1}\} = \max\{2 + F_{n-2}, 4 + F_{n-3}, 3 + D_{n-2}\} = \\ = \max\{2 + F_{n-2}, 4 + F_{n-3}\}, n \geqslant 3$$

Получаем $(F_0, F_1, F_2) = (0, 2, 3)$ и $F_n = 4 + F_{n-3}$, откуда легко понять, что $F_n < 4/3n + 1$, что означает $F_n = \lceil \frac{4}{3}n \rceil$.

Задача L. a, ab, ba строки

Автор задачи: Александр Чистяков, разработчик: Александр Понкратов

Разобьем строку на подстроки, которые начинаются с «b», заканчиваются на «b» и символы перед и после тоже равны «b». Среди всех подстрок можно выбирать подстроки минимальной длины, дающие в объединении исходную строку. Каждую такую подстроку можно корректно разбить тогда и только тогда, когда количество символов «b» не превосходит количество символов «a». Тогда задача сводится к поддержке таких подстрок и проверке корректности разбиения. Это можно сделать, поддерживая все подстроки в `std::set` и сохраняя неразбиваемые в отдельном множестве; при ответе на запрос проверять, что отрезок запроса не содержит неразбиваемых подстрок; при изменении символа проверять, что какие-то подстроки могли объединиться в одну или распасться на несколько. Считать количество символов на отрезке можно с помощью дерева Фенвика.

Также существуют альтернативные решения, использующие дерево отрезков.

Задача M. Три чемодана

Автор и разработчик задачи: Екатерина Ведерникова

Для решения задачи рассмотрим 5 случаев отпрваки чемоданов:

1. Каждый чемодан пробить своим чеком;
2. 1й и 2й чемоданы пробить одним чеком, а 3й отдельно;
3. 1й и 3й чемоданы пробить одним чеком, а 2й отдельно;
4. 2й и 3й чемоданы пробить одним чеком, а 1й отдельно;
5. Пробить все три чемодана одним чеком.

В каждом случае посчитаем сколько нужно заплатить за чеки. И выберем тот случай, в котором мы заплатим меньше денег.