Разбор задачи «Арифметическая магия»

Обозначим числа за X и Y. Тогда значение равно $(X+1)(Y+1) - XY - X - Y = 1$. 1 в любой целой неотрицательной степени равно 1. Так что для решения этой задачи надо просто вывести 1.

Разбор задачи «Большие перемены»

Максимальное значение доступности равно $N-1$ и достигается, если соединить какой-то город (назовём его столицей) со всеми остальными. Действительно, все требования выполняются (перелететь между любыми двумя городами можно с пересадкой в столице). Доступность не может превышать $N-1$, так как для каждого города существует $N-1$ город, не совпадающий с ним. Поэтому количество способов при $N=2$ равно количеству вариантов выбрать столицу, то есть N. При $N=2$ существует единственный способ соединения городов, который и будет максимальным, так что при $N=2$ ответ равен 1.

Разбор задачи «Совпадения»

Очевидно, что если у участника номер паспорта больше N, то совпадение невозможно. Если несколько участников имеют номер, не превосходящий N, то в соответствующую комнату можно заселить только одного участника, для остальных совпадение уже невозможно (комната занята). Таким образом, ответ — количество различных чисел среди a_i, не превосходящих N. Подсчитать их можно, например, заведя булевский вектор из N элементов, помечая там те числа, которые уже были, и в случае изменения значения элемента увеличивая счётчик возможных совпадений на единицу.

Разбор задачи «Драфт НБА»

В этой задаче надо просто сделать то, что описано в условии: посчитать для каждого параметра, в какую категорию он попадает, и затем в нужном порядке проверить все варианты.

Разбор задачи «Думский регламент»

Обратим внимание на следующие факты:

1. Пустая запись является корректной.
2. Если две записи являются корректными, то их слияние тоже является корректным. Действительно, получится заседание, в котором к моменту окончания первой записи быди приняты все внесённые законопроекты, а затем заседание продолжилось внесением очередного законопроекта и также корректно завершилось.
3. Если запись является корректной, её можно “вставить” внутрь обсуждения одного законопроекта и получить корректную запись. Это прямо следует из третьего пункта регламента.

Несложно показать, что любую запись можно получить из пустой с помощью действий 2 или 3. Действительно, пусть у нас есть некая корректная запись. Рассмотрим первый законопроект. Он или был поставлен на голосование последним (тогда был применено третье действие — вся оставшаяся запись была вставлена в обсуждение этого законопроекта), или был поставлен на голосование в какой-то момент раньше. Тогда к моменту после голосования по первому законопроекту все внесённые законопроекты уже “закрыты”, то есть часть записи от внесения законопроекта на обсуждение до голосования представляет собой корректную запись. Аналогично оставшаяся часть представляет собой корректную запись (так как к моменту её начала никаких законопроектов на повестке дня не стоит, то стартовые условия выполняются, остальные же требования следуют из корректности первоначальной записи), и в этом случае было применено второе действие. В каждом случае получаем переход к записи меньшей длины; будем продолжать действовать таким образом, пока не дойдём до пустой записи.

Тем самым мы показали, что условие задачи эквивалентно правильной скобочной последовательности с 26 типами скобок. Для проверки того, что скобочная последовательность является правильной, лучше всего использовать стек: в начале обсуждения законопроекта мы кладём на стек

букву, соответствующую внёсшей его партии, по окончании — сравниваем вершину стека с буквой, соответствующей партии, законопроект которой поставлен на голосование (если не равны — запись некорректна) и снимаем букву со стека. Если в какой-то момент времени была попытка взять вершину пустого стека — запись некорректна. Если по завершении всей записи стек не пуст — какие-то законопроекты не поставлены на голосование и запись некорректна. Во всех остальных случаях запись корректна.

Разбор задачи «Правильный подмногоугольник»

Если N делится на $k > 2$, то можно выбрать k вершин так, чтобы они образовывали правильный многоугольник. Таким образом, нужно найти наименьшее $k > 2$, на которое делится N.

Если N делится на 3, ответ равен 3.

Иначе, если N нечётно, перебором до $\sqrt{N}$ ищем наименьший простой делитель, он и будет ответом. Если не нашли, ответ равен N.

Иначе, если N делится на 2, но не делится на 4, перебором до $\sqrt{N}$ ищем наименьший нечётный простой делитель. Если не нашли, ответ равен $N/2$.

Иначе N делится на 4 и ответ равен 4.

Разбор задачи «Есть ли делитель?»

Если строка состоит из одного символа, то ответ -1 в случае, если число не является составным (1 2 3 5 7), и 10 (к примеру) в случае, если является (то есть 4 6 8 9).

Если строка состоит из более чем одного символа, то работает ответ ($B = S + 1$, $X = S$), где S — сумма цифр (так как $X^k - 1 = (X - 1) \cdot (X^{k-1} + \ldots + X + 1)$, то остаток числа в $S + 1$-ичной системе от деления на S равен остатку от деления на S суммы его цифр, то есть 0 по построению).

Разбор задачи «ICPC»

Рассмотрим все K-значные слова. «`icpc`» может находиться на одной из $K-3$ позиций, при этом все остальные буквы могут быть произвольными (при таком подходе мы считаем вхождения на конкретной позиции, что автоматически решает вопрос учёта слов со многими «`icpc`»). Таким образом, слов с «`icpc`» будет $26^{K-4} \cdot (K-3)$. Кроме того, возможны три стыка: $cXicp|cXicq$, $pcXic|pcXid$, $cpcXi|cpcXj$, где X — произвольный набор из $K-4$ букв, то есть ещё добавляется $26^{K-4} \cdot 3$ слова. Итого получается $26^{K-4} \cdot K$ вхождений на блок K-значных чисел.

Просуммировав все такие значения для K от 4 до N по заданному модулю, получим ответ к задаче. Покажем, что сумма сворачивается в формулу. Обозначим за $F(N)$ сумму $1 + 26 + \ldots + 26^N$; $F(N) = (26^{N+1} - 1)/(26 - 1)$. Тогда наша сумма равна $3 \cdot F(N-4) + \sum\limits_{i=0}^{N-4} 26^i \cdot (i+1) = 3 \cdot F(N-4) + F(N-4) + 26 \cdot F(N-5) + 26^2 \cdot F(N-6) \ldots$.
Но $26^k \cdot F(N-4-k) = (26^{N-4-k+1} \cdot 26^k - 26^k)/(26-1) = (26^{N-3} - 26^k)/(26-1)$. Таким образом, после суммирования получим в числителе $(N-3) \cdot 26^{N-3} - 1 - 26 - 26^2 \ldots = (N-3) \cdot 26^{N-3} - F(N-4)$.

Разбор задачи «Составление задач»

Очевидно, что нас интересуют только задачи, которые знает как можно меньше людей. Две задачи из этого набора могут войти в контест только если их знает одинаковый набор человек. Поэтому мы для каждого набора человек считаем, какие задачи известны именно этому набору и берем максимальное по размеру получившееся множество. Асимптотика времени работы оценивается как $O(m \cdot log(n))$

Разбор задачи «Порталы»

Разобьём таблицу на компоненты связности так, будто у нас нет портальной пушки. Между компонентами связности мы можем перемещаться только с ее помощью. Будем каждую компоненту связности считать отдельной вершиной. Добавим все необходимые рёбра между компонентами. Заметим, что каждая клетка даёт нам не более 4 ребёр — вдоль каждого из возможных направлений. Поэтому ребер в данном графе будет не более $4 \cdot n \cdot m$. В полученном графе длина пути до финальной вершины есть количество использований портальной пушки. Это позволяет нам запустить на таком

графе обход в ширину для поиска кратчайшего расстояния. Затем остаётся только восстановить ответ.

Отдельным случаем стоит выделить ситуацию, когда комната, в которой игрок начинает целиком окружена стеклянными стенами. Тогда игрок никак не может ее покинуть, потому что ему некуда поставить первый портал, и если конечная клетка находится в другой компоненте, то попасть в нее никак не получится.

Разбор задачи ««Контакт» для двоих»

Построим бор на словах из словаря; пусть для каждой вершины бора v $sz[v]$ есть количество слов, заканчивающихся в поддереве вершины v, включая саму вершину v. Все значения $sz(v)$ можно насчитать за $O(SUMLEN)$, где $SUMLEN$ - суммарная длина всех строк.

Пусть задан запрос (s, k), где s - строка словаря длины n, а ее путь в боре состоит из вершин $v_0, v_1, \ldots, v_{|s|}$. Заметим, что ответ для нее не превосходит $k*|s|$. Предположим, что мы можем назвать $c \leqslant k * |s|$ слов в процессе игры. Тогда очевидно, что $sz(v_1) \geqslant c$, $sz(v_2) \geqslant c-k$, $sz(v_3) \geqslant c-2*k$, $\ldots$, $sz(v_{|s|}) \geqslant c-(|s|-1)*k$ (ибо мы обязаны назвать как минимум $c-(i-1)*k$ в поддереве вершины v_i, $i = 1, 2, \ldots, |s|$).

С другой стороны, докажем, что эти условия не только необходимы, но и достаточны. Заметим, что оптимальной с точки зрения максимизации слов стратегией для второго игрока является называть каждый раз слово с как можно меньшим общим префиксом с угадываемым словом, который только возможен с учетом уже названного лосем Валерой слова. Предположим, что второй игрок, используя эту стратегию, назвал некоторое количество слов $c+1 \leqslant k*|s|$, остановившись на l-ом раунде, $1 \leqslant k \leqslant |s|$. Пусть $0 \leqslant l' < l$ - номер последнего раунда, в котором назывались только слова, "ответвляющиеся" от s непосредственно после l'-го раунда, или 0, если таких раундов не было. Тогда в поддереве вершины $v_{l'+1}$ были названы все слова, причем исключительно в раундах, начиная с $l'+1$, и их оказалось ровно $c+1-l'*k$; следовательно, $sz(v_{l'+1}) = \ +1-l'*k > c-l'*k$. Следовательно, если c слов назвать невозможно, то и вышеописанные условия не выполняются. Таким образом, эквивалентность доказана.

Вышедоказанное утверждение дает возможность отвечать на запрос за $O(|s|)$, находя минимум из чисел $sz(v_i)-k*(i-1)$ на пути строки s, что, конечно, недостаточно эффективно. Однако заметим, что $sz(v_i) \geqslant sz(v_{i+1})$ для любого i, причем строгое неравенство выполняется, только если в вершине v_i заканчивается некоторое слово и/или происходит ответвление. Таких событий на пути не более $2 * \sqrt{(SUMLEN)}$ (в противном случае, суммарная длина слов в словаре окажется больше, чем $SUMLEN$, что противоречит условию); следовательно, если для каждой вершины предподсчитать заранее следующую вершину с ответвлением (отметим, что она не зависит от самого слова s), то время ответа на запрос сократится до $O(\sqrt{SUMLEN})$. Таким образом, итоговая сложность решения $O(SUMLEN + Q * \sqrt{SUMLEN})$.

Разбор задачи «Ближайшие точки»

Как известно из школьной геометрии, геометрическое множество точек, более близких к точек p, нежели к точек q, есть открытая полуплоскость, граница которой является серединным перпендикуляром отрезка pq. За $O(n\ log\ n)$ можно найти пересечение этих полуплоскостей и прямоугольник, после чего с помощью сканирующей прямой легко вычислить для каждого целого x, $0 \leqslant x \leqslant X$, диапазон y-ов, т.ч. точка (x, y) лежит строго внутри; такое решение работает за $O(n\ log\ n + X)$. Однако этот алгоритм является не самым приятным и простым в реализации.

На наш взгляд, более приятным является использование дерева Ли-Чао. С его помощью легко для каждого x от 0 до X найти самый высокий y, т.ч. (x, y) лежит на границе полуплоскости, "смотрящей вверх", а также самый низкий y на границе полуплоскости, "смотрящей вниз"; это можно сделать за $O(n\ log\ X)$. После этого, ответ можно найти тривиальным образом за $O(X)$, а общая асимптотика такого решения - $O(n\ log\ X + X)$.

Отметим, что решение с использованием типов данных с плавающей точкой, наподобие double или даже long double в C++ могли приводить к потере точности в ответе. В силу этого, рекомендуется выполнять все вычисления в рациональной арифметике. Ограничения в задаче подобраны таким образом, что при аккуратной реализации, достаточно было хранить числитель и знаменатель в виде 64-битных целых чисел.