Технокубок 2019 - Финал(Разбор)

A. Технокубок Огня

1 секунда, 256 мегабайт

Во-первых, когда мы переводим человека в другую школу, количество школ, содержащих избранных, не может увеличиться больше, чем на 1. Во-вторых, если некоторая школа содержит $c>0$ избранных учеников, но никто из них не является сильнейшим в этой школе, то мы должны произвести операцию перевода человека в другую школу хотя бы c раз, чтобы все эти избранные ученики были выбраны. Комбинируя эти два утверждения, можно заметить, что ответом на задачу является количество избранных учеников, которые изначально не выбраны.

B. Системное тестирование

1 секунда, 256 мегабайт

Сначала определим для каждого решения, когда оно начнёт тестироваться. Это можно сделать, например, следующим алгоритмом: для каждого процесса будем хранить момент времени, когда он освободится (изначально все эти k чисел равны нулю), затем пройдёмся по всем решениям в очереди и для каждого из них выберем процесс с минимальным временем, когда он освободится; скажем, что это решение начнёт тестироваться в этот момент времени, и обновим время освобождения для этого процесса.

После того, как мы это определим, мы легко сможем для каждого момента времени сказать, сколько решений были полностью протестированы до него, а затем проверить необходимое условие интересности на каждом тесте каждого решения.

C. Диана и Лиана

2 секунды, 256 мегабайт

Для начала, научимся определять, может ли данный отрезок $[l,r]$ цветов превратиться в идеальный венок (то есть, можно ли удалить какие-нибудь цветы, чтобы то, что осталось от подотрезка $[l,r]$, было идеальным венком, и при этом чтобы он был вырезан). Во-первых, количество остальных венков должно равняться $n-1$, поэтому неравенство $\left\lfloor\frac{l-1}{k}\right\rfloor \cdot k + \left\lfloor\frac{m-r}{k}\right\rfloor \cdot k \geq n - 1$ должно быть выполнено. Во-вторых, отрезок $[l,r]$ должен содержать цветы всех необходимых типов. Наконец, $r-l+1$ должно быть не меньше k. Легко видеть, что этих условий достаточно для того, чтобы $[l,r]$ мог стать идеальным венком.

Научимся теперь для каждого l находить такое минимальное r, что для отрезка $[l,r]$ выполнено второе условие. Это можно сделать с помощью двух указателей: при увеличении l соответствующее r не может уменьшиться, а поддерживать количества

цветов разных типов на отрезке и количество типов, которых недостаёт, можно за $O(1)$ при увеличении как левой границы, так и правой.

Значит, осталось лишь проверить, найдётся ли такое l, что отрезок $[l,\max(r,l+k-1)]$ удовлетворяет первому условию, и если найдётся, то вывести какие-нибудь номера цветов перед l, которые мы удаляем (при этом количество оставшихся цветков перед l должно делиться на k и при этом не превосходить $(n-1)k$ и какие-нибудь цветы перед $[l,r]$, которые мы удаляем (не нарушая условий). Цветы после r удалять не обязательно.

D. Сжать строку

3 секунды, 256 мегабайт

Обозначим через $dp[p]$ наименьшую стоимость, необходимую для кодирования префикса строки s длины p, тогда ответ равен $dp[n]$. Если мы хотим закодировать префикс длины p, то последний символ в закодированной строке будет или равняться sp, или представлять некоторую подстроку $s[l,p]$, которая встречается в префиксе размера $l-1$. Таким образом,

$$dp[0]=0,$$
$$dp[p]=\min(a+dp[p-1],\min(b+dp[l-1] \mid s[l,p] \text{ является подстрокой } s[1,l-1])).$$

Один из способов посчитать dp заключается в том, чтобы обновлять состояния вперёд и использовать хеши, но в этом случае могут понадобиться дополнительные усилия, чтобы избежать коллизий и уложиться в ограничение по времени. Другой способ — для каждого p найти все подходящие l, посчитав z-функцию на развёрнутой строке $s[1,p]$. Итоговая сложность в этом случае — $O(n^2)$.

E. Случай в казино

1 секунда, 256 мегабайт

Поскольку каждая операция сохраняет знакочередующуюся сумму цифр, если она различна для a и b, то ответ '-1'. Докажем, что в противном случае необходимая последовательность действий существует.

Представим, что цифры могут быть отрицательными или превосходить 9 (то есть, например, число 19 может стать числом с цифрами 2 и 10). Обозначим через a_i i-ю цифру числа a (и аналогично для b). Теперь порядок операций, которые мы хотим совершить, не имеет значения, поэтому мы можем совершать их слева направо. После того, как мы сделаем все операции с первой цифрой (их как минимум $|a_1-b_1|$), a_2 будет равняться $a_2+b_1-a_1$. После того, как мы затем сделаем все операции с a_2 и a_3 (их как минимум $|a_2+b_1-a_1-b_2|$), a_3 станет равным $a_3+b_2-a_2-b_1+a_1$, и так далее. Таким образом мы можем найти наименьшее необходимое число шагов. Докажем теперь, что их можно совершить в каком-либо порядке, не нарушая условий на промежуточные цифры.

В самом деле, попробуем совершать эти операции слева направо. Предположим, что мы не можем совершить очередную операцию. Предположим, что мы не можем уменьшить пару цифр: одна из a_k и a_{k+1} сейчас равна 0. Легко видеть, что $a_k>0$, потому что в противном случае, если мы совершим операции, игнорируя правила с цифрами, то в конце a_k превратится в b_k, что неотрицательно. Значит, $a_{k+1}=0$. В таком случае мы обязаны увеличить a_{k+1} и a_{k+2} хотя бы раз (опять же, потому что мы можем совершить все оставшиеся операции, не заботясь о цифрах, и получить b_{k+1} в конце). Если мы можем это сделать, то сделаем это и затем уменьшим a_k и a_{k+1}, как и собирались, после чего продолжим выполнять операции. В противном случае $a_{k+2}=9$. Рассуждая аналогично, либо мы можем уменьшить a_{k+2} и a_{k+3} прямо сейчас, либо $a_{k+3}=0$, и так далее. Поскольку это не может продолжаться бесконечно, мы можем сделать какие-то необходимые ходы и затем уменьшить a_k и a_{k+1}. Таким образом, мы можем получить нужное число за наименьшее число операций.

F. Дерево власти

2 секунды, 256 мегабайт

Легко видеть, что задача не меняется, если мы будем пытаться уметь получать любые числа в листьях из одних нулей.

Решение 1. Пусть $v_1, v_2, ..., v_l$ — индексы всех листьев в порядке Эйлерова обхода. Обозначим через a_i число, записанное v_i. Положим $v_0=v_{l+1}=0$.
Обозначим $a_{i+1}-a_i$ через d_i. Для решения задачи необходимо и достаточно уметь получать все возможные комбинации $d_0, ..., d_l$ с нулевой суммой.

Предположим, мы купили вершину u, чьё поддерево содержит листья с v_i по v_j. Применяя операцию с числом x к этой вершине, мы увеличиваем разность d_{i-1} на x, уменьшаем d_j на x и не меняем остальные разности.

Построим новый граф с вершинами, пронумерованными от 0 до l. Каждой вершине u исходного графа соответствует ребро, соединяющее вершины $(i-1)$ и j и имеющее вес c_u. Чтобы уметь получать любую комбинацию с нулевой суммой, необходимо и достаточно, чтобы граф на рёбрах, соответствующих купленным вершинам, был связным. Для этого надо найти минимальный вес остовного дерева и все рёбра, которые могут в нём присутствовать. И то, и другое, может быть найдено с помощью алгоритма Краскала.

Решение 2. Скажем, что вершина u *покрывает* лист v, если этот лист лежит в поддереве вершины u.

Легко видеть, чтобы мы можем купить некоторое множество вершин тогда и только тогда, когда в поддереве любой вершины u есть не больше одного листа, который не покрыт ни одной купленной вершиной в этом поддереве. В самом деле, если таких листьев хотя бы два, то разность значений в них никогда не меняется. С другой стороны, если это условие выполняется, то мы можем выполнять операции с купленными вершинами в порядке удаления от корня, делая необходимую операцию всякий раз, когда в поддереве данной вершины есть лист, не покрытый ни одной из купленных вершин поддерева, кроме рассматриваемой.

Таким образом, можно считать значения $ans[v][k]$, при $0 \le k \le 1$, равные минимальной стоимости купленных вершин в поддереве вершины v таким образом, чтобы в нём было не больше k непокрытых листьев. Также можно показать, что этих значений достаточно, чтобы найти все вершины, которые могут лежать в ответе.

G. Тот самый Мюнхгаузен

1 секунда, 256 мегабайт

Назовём *балансом* числа x значение выражения $S(nx) \cdot n - S(x)$. Балансом строки, состоящей из цифр, будем называть баланс соответствующего ей числа.

Решение 1. План решения следующий:

- Узнать, существует ли решение. Если нет, то вывести -1 и завершить работу.
- Найти произвольную строку с отрицательным балансом.
- Найти произвольную строку с положительным балансом.
- Взять их линейную комбинацию с нулевым балансом (возможно, предварительно дописав лидирующие нули к числам). Линейной комбинацией нескольких строк мы называем их конкатенацию, где, возможно, каждая из них встречается несколько раз.

Понятно, как осуществить последний шаг. Для выяснения наличия решения удобно ввести следующие леммы:

Лемма 1. $S(a+b) \le S(a)+S(b)$.

Доказательство. Это довольно очевидно, поскольку при сложении столбиком сумма цифр результата будет равна $S(a)+S(b)$ минус число переходов через разряд, умноженное на 9.

Лемма 2. $S(ab) \le aS(b)$.

Доказательство. $S(ab)=S(b+b+\ldots+b) \le S(b)+S(b)+\ldots S(b)=aS(b)$. Неравенство выполнено по лемме 1.

Лемма 3. $S(ab) \le S(a)S(b)$.

Доказательство. Пусть $a=\overline{a_{n-1}a_{n-2}\ldots a_1a_0}$. Тогда $S(ab)=S(a_{n-1}b \cdot 10^{n-1}+a_{n-2}b \cdot 10^{n-2}+\ldots+a_0b) \le S(a_{n-1}b \cdot 10^{n-1})+S(a_{n-2}b \cdot 10^{n-2})+\ldots+S(a_0b)= S(a_{n-1}b)+S(a_{n-2}b)+\ldots+S(a_0b) \le (a_{n-1}+a_{n-2}+\ldots+a_0)S(b)=S(a)S(b)$. Рассмотрим теперь два случая.

- $a=2^{d1} \cdot 5^{d2}$.

Положим $b=10^{\max(d1,d2)}$. Легко видеть, что $a \cdot S(an)=a \cdot S(\frac{bn}{b/a})=\frac{a}{S(b/a)} \cdot S(b/a) \cdot S(\frac{bn}{b/a}) \ge \frac{a}{S(b/a)} \cdot S(bn)=\frac{a}{S(b/a)} \cdot S(n)$. Таким образом, если $a > S(b/a)$, то ответа не существует, иначе число b/a имеет неположительный баланс. Можно заметить, что число 1 всегда имеет неотрицательный баланс, поэтому в этом случае задача решена.

- a имеет простой делитель, отличный от 2 и 5.

Оказывается, в этом случае ответ всегда существует. В самом деле, десятичная дробь $1/a$ бесконечна, из чего следует, что $S(\frac{10^k}{a})$ неубывает и может быть сколь угодно велико; как и значения выражения $S(\frac{10^k}{a}+1)$, поскольку число 9-к в конце $\frac{10^k}{a}$ ограничено. С другой стороны $S(a \cdot (\frac{10^k}{a}+1))$ ограничено сверху, например, числом, $1+9 \cdot len(a)$, поэтому мы всегда можем найти строку с отрицательным балансом, и, как было замечено ранее, число 1 всегда имеет неотрицательный баланс.

Нам (авторам) известны минимум два способа реализовать второй шаг.

- Разделим **1** на a и найдём период. Обозначим его через строку (возможно, с лидирующими нулями) s, а предпериод — через t (возможно, пустой). Положим s' имеющей такую же длину, что и s, и равной $(s+1)$, если при сложении считать s числом. Мы ищем строчку вида $tsss\ldots ss'$ с отрицательным балансом. Можно посчитать вклад строк t, s и s' в баланс и тем самым найти минимальное необходимое число строк s.
- Построим взвешенный ориентированный граф. Его вершинами будут числа от **0** до $a-1$, и для любой пары чисел $(c,x)\neq(0,0)$, где $0\le c<a$ **and** $0\le x\le 9$, есть ребро из вершины c в $\left\lfloor\frac{ax+c}{10}\right\rfloor$ с весом $(ax+c)\pmod{10}$ и меткой x. Неформально, если пройти по произвольному пути из вершины **0**, и метки пути, выписанные справа налево, образуют число n, то сумма весов на пути есть не что иное, как текущий баланс числа an (слово «текущий» означает, что при подсчёте баланса мы рассматриваем только последние $len(n)$ цифр), а последняя вершина пути равняется текущему значению переноса. В полученном графе можно найти отрицательный цикл алгоритмом Форда-Беллмана и построить путь от **0** до этого цикла, пройти по этому циклу нужное число раз (чтобы сумма весов на итоговом пути была отрицательной), и вновь вернуться в нулевую вершину.

У этого решения есть недостаток: длина полученного по этому алгоритму ответа может достигать $S(n)\cdot n^2$. Чтобы этого избежать, можно, например, генерировать **300** различных строк с отрицательным балансом (если брать больше периодов/отрицательных циклов), а затем найти строку с положительным балансом следующим образом: попробовать все числа от **1** до **10000** и построить их линейную комбинацию с небольшим балансом, делящимся на какой-нибудь из отрицательных балансов. Это можно сделать с помощью рюкзака на полученных положительных балансах. Идея в том, чтобы сделать НОД найденных строк как можно больше, чтобы строку с отрицательным балансом не пришлось повторять много раз (поскольку именно она может иметь суперлинейную длину).

Решение 2. Представим, что в нашем распоряжении бесконечные память и время. Тогда мы можем завести состояния вида $(carry,balance)$ аналогично вершинам графа из решения 1, где для каждого состояния $(carry,balance)$ и каждой цифры x (кроме состояния (0,0) с цифрой 0) есть переход в $((carry+a\cdot x)\pmod{10},balance+a\left\lfloor\frac{carry+a\cdot x}{10}\right\rfloor-x)$. Наша цель — достигнуть (0,0) снова. Можно обойти этот (бесконечный) граф состояний в ширину, создавая новые состояния, когда нужно, однако это потребляет слишком много памяти.

Оказывается, что если не рассматривать состояния с $|balance|>a$, то решение всегда находится, и это решение относительно легко придумать (и точно легче сдать, чем предыдущее решение).

H. Секретные письма

2 секунды, 512 мегабайт

Рассмотрим какое-либо оптимальное решение. Рассмотрим все письма, которые посылал Пух между двумя походами Пятачка к Кролику. Легко видеть, что

- возможно, первое из этих писем было послано через Кролика,
- последние несколько (может, ни одного) писем также были посланы через Кролика,
- все остальные письма посланы через Сову.

Смысл посылать первое письмо через Кролика в том, чтобы забрать письма, которые на тот момент у него хранятся. Если первое письмо послано Совой, а какое-нибудь из остальных передано Кролику, то можно поменять эти два письма местами и тем самым заплатить не больше. С другой стороны, если нам нечего забирать у Кролика, то имеет смысл посылать письмо через Кролика если и только если $ct \leq d$, где t есть время между отправкой и тем, как Пятачок придёт к Кролику. То же самое верно, если поменять Пуха и Пятачка местами.

Обозначим через $ans[i]$ наименьшую возможную стоимость отправки первых $(i-1)$ писем, если i-е из них отправлено через Кролика.

Также обозначим через $ans\_delay[i]$ наименьшую возможную стоимость отправки первых $(i-1)$ писем, если i-е из них отправлено через Кролика, **и если Кролик на тот момент хранил ровно одно письмо**. Чтобы посчитать эту динамику, можно каждый раз перебирать суффикс писем, хранящихся у Кролика. Это занимает $O(n^2)$ времени.

С другой стороны, если зафиксировать левую границу писем, отправленных через Кролика, то стоимость их отправки зависит линейно от времени t_i, для которого мы это вычисляем, и, поскольку каждый раз нам нужно находить минимум линейных функций, мы можем использовать convex-hull trick, чтобы получить асимптотику $O(n)$ или $O(n \log n)$ complexity.