Задача A. Идеальный бутерброд

Заметим, что наибольшее количество идеальных бутербродов, которое может приготовить Вадим, не может быть больше, чем A, B div 10, C div 2 и D div 3. Это значит, что наибольшее количество равно $\min(A, B \text{ div } 10, C \text{ div } 2, D \text{ div } 3)$. Причём Вадим не сможет накормить больше N гостей, а значит ответ равен $\min(N, A, B \text{ div } 10, C \text{ div } 2, D \text{ div } 3)$.

Задача B. Поиск в Яндексе

В задаче требуется реализовать предложенный алгоритм. Для этого можно смотреть на каждое слово по очереди и обрабатывать его в порядке ввода. Посмотрим на последний символ, если это строчная латинская буква, то нужно вывести это слово, если это вопросительный знак, то ничего не вывести, если это восклицательный знак, то посчитать их количество в слове (пусть k), записать все остальные символы в новое слово и вывести его $k + 1$ раз.

Задача C. Деление с остатком

Заметим, что подходящих чисел бесконечно много, если A и B совпадают. Далее, пусть $A > B$, если это не так, можно их поменять местами. Заметим, что остатки от деления A и B на некоторое M совпадают тогда и только тогда, когда $A - B$ делится на M. $A - B > 0$, так как они различны, поэтому наибольшим делителем $A - B$ является само $A - B$, это значение и является ответом.

Задача D. Подарки для призёров

Будем выбирать подарки, начиная со стоимости 1 и увеличивая его стоимость на 1. В тот момент, когда суммарного бюджета перестанет хватать (пусть, на m-й стоимости), добавим остаток к последнему подарку. Очевидно, что все стоимости будут различные. Покажем, что больше $m - 1$ выбрать подарков невозможно. Пусть можно выбрать хотя бы m различных подарков, тогда затраченный бюджет на подарки не меньше $1 + 2 + 3 + \ldots + (m - 1) + m$, что невозможно, так как в построенном выше алгоритме было показано, что эта сумма больше N.

Задача E. Интересные инвестиции

Пусть стоимость одного компьютера в NAUMENcoin'ах будет T рублей. Тогда можем составить уравнение: $N \cdot T = A \cdot C + B$, где B — искомое возможное число монет, а $A \cdot C + B$ — общая стоимость покупки в этих монетах. Т.к. все числа целые, то ясно что $(A \cdot C + B)$ делится на N, т.е. $(A \cdot C + B) \mod N = 0$. А значит, $B \mod N = -A \cdot C \mod N$. Откуда следует, что $B = (-A \cdot C) \mod N + k \cdot N$. Т.к. в получившимся уравнении неизвестно только k, то минимальное значение выражение примет при $k = 0$, а максимальное значение — при максимальном k таком, что $B < C$. Значит $(-A \cdot C) \mod N + k \cdot N < C$, откуда $k = \left\lfloor \frac{C-((-A \cdot C) \mod N)-1}{N} \right\rfloor$. Подставив k в исходное выражение, получим максимально возможное значение B. Если же максимальное значение оказалось меньше минимального или максимальное значение оказалось больше C, то ответ -1.

Задача F. Wordle

Задача подразумевает переборное решение. Так как латинский алфавит имеет 26 букв, то всего есть $26^5 = 11\,881\,376$ вариантов слов. Нужно перебрать все слова, подходящие на первый взгляд, и каждое проверить явно. В зависимости от того, на каком языке вы пишите, и как быстро вы проверяете, вам могут понадобиться отсечения.

Воспользуемся тем, что во входных данных точно есть жёлтая и зелёная буква. Первое отсечение: не перебирать позицию, в которой есть зелёная буква. Это оставляет $26^4 = 456976$ вариантов для проверки. Можно также не перебирать буквы на позициях, где они были покрашены жёлтым или зелёным.

Второе отсечение: воспользуемся жёлтой буквой. Переберём только такие комбинации, где она есть на другой позиции. Это даёт нам $26^3 \cdot 4 = 52728$ комбинаций, кроме одного случая: когда жёлтая буква и есть та самая зелёная. Этот случай сложно обработать, но если вам удастся, и в слове нет других жёлтых букв, то ответ можно посчитать комбинаторно, перемножить количество оставшихся букв в каждой позиции.

Имеет смысл рассказать также, как лучше всего проверять подходящие слова. Можно генерировать ответ по правилам, описанным в условии, и проверять равенство покрасок. Но есть более простой метод.

Давайте у каждой попытки для каждой буквы посчитаем, сколько раз эта буква была покрашена в какой цвет. Если буква в пределах одной попытки была покрашена в зелёный g раз, в жёлтый y раз, в чёрный b раз, то в слове она встречается не менее $g + y$ раз. При этом если $b > 0$, то в слове она встречается не более $g + y$ раз.

Агрегировав эти данные для всех букв по всем попыткам, получим для каждой буквы минимальное и максимальное количество раз, которое оно встречается. Чтобы проверить слово, нужно посчитать для каждой буквы, что её количество лежит в этом диапазоне. Останется проверить, что все буквы обязательно стоят там, где они были зелёными, и не стоят там, где они были жёлтыми или чёрными — это можно учесть даже до перебора, поддерживая для каждой позиции множество всех букв, которые там могут находиться.

Задача G. Две дороги

Вначале посчитаем количество способов выбрать на карте прямоугольник размера $A \times B$, который будет содержать в себе две дороги. Таких ровно $(N - A + 1) \cdot (M - B + 1)$. После этого в таком прямоугольнике можно выбрать первую дорогу $B+1$ способами, а вторую — $A+1$ способами, причём они всегда будут пересекаться.

Покажем, что этот алгоритм покрывает все возможные варианты, а также, что никакие два различных способа не соответствуют одинаковому варианту. Легко заметить, что по любому варианту можно построить прямоугольник размера $A \times B$, где его границы будут содержать концы дорог, а значит в алгоритме этот вариант был учтён. Второе утверждение докажем от противного. Пусть два способа соответствуют одному варианту, тогда заметим, что прямоугольник для обоих способов совпадает, иначе дороги заканчиваются в различных местах. А из посчитанных способов в одном прямоугольнике ровно один соответствует этому варианту. Значит эти способы совпадают, противоречие.

В итоге, количество вариантов равно $(N - A + 1) \cdot (M - B + 1) \cdot (A + 1) \cdot (B + 1)$.

Задача H. Sherk 2: The Game

Заметим факт: пусть в оптимальном ответе мы перешли и ушли с частей Котока в разные моменты (пусть момент a и b), тогда рассмотрим максимальное значение на этом отрезке (пусть в позиции m), если мы перейдём в части Котока не с a-й части Шерка, а с m-й части, а также уйдём с частей котока не после b-й части Котока, а после m-й части, то ответ не ухудшится: $\max_{i\in[1,a]}(s_i) + \max_{i\in[a,b]}(k_i) + \max_{i\in[b,N]}(v_i) \leqslant \max_{i\in[1,m]}(s_i) + k_m + \max_{i\in[m,N]}(v_i)$

Значит, достаточно перебрать номер части Котока, на и которую и с которой мы будем переходить. Здесь нужно посчитать ответ. Это можно сделать быстро, предварительно посчитав максимум на префиксе у частей Шерка и максимум на суффиксе у частей Весла. Тогда ответ будет восстанавливаться за $O(1)$. Осталось найти максимум по ответам для каждой перебранной части Котока и вывести после него два одинаковых индекса, в котором максимум был достигнут. Суммарно это решение работает за $O(N)$ и занимает $O(N)$ памяти.

Задача I. Денежные переводы

Заметим ключевое утверждение: остаток при делении на числа на 9 такой же, как и у суммы его цифр при делении на 9. Поэтому у чисел $1, 11, 111, \ldots, 111\,111\,111$ остаток соответственно $1, 2, \ldots, 9$, а числа больше $111\,111\,111$ можно вообще не использовать.

Задачу можно решить динамическом программированием, но мы не можем использовать количество денег как измерение, потому что денег на счету может быть очень много. Вместо этого количество денег должно быть *значением* `dp`.

Пусть `dp[i][j]` — наименьшее количество денег, которое можно отослать за i переводов, и которое по модулю 9 равно j. Заполним массив очень большими значениями, кроме `dp[0][0] = 0`. Переходы считаются следующим образом:

```
dp[i + 1][j + 1] = min(dp[i + 1][j + 1], dp[i][j] + 1)
```

```
dp[i + 1][j + 2] = min(dp[i + 1][j + 1], dp[i][j] + 11)
dp[i + 1][j + 3] = min(dp[i + 1][j + 1], dp[i][j] + 111)
dp[i + 1][j + 4] = min(dp[i + 1][j + 1], dp[i][j] + 1111)
...
```

И так далее. Ответом будет наименьшее число i такое, что `dp[i][N mod 9]` $\leqslant N$.

Альтернативное решение:

Обозначим числа, состоящие из единиц, как (a_i), $i \leqslant 9$. Тогда $a_1 = 1, a_2 = 11, \ldots, a_9 = 111\,111\,111$, и $a_i \mod 9 = i$. Оказывается, имеет смысл переводить только суммы a_i и a_{i+1} для некоторого i. Если бы мы перевели a_i и a_j, где $j \geqslant i + 2$, то можно заменить их на a_{i+1} и a_{j-1} с явно меньшей суммой.

Это нам уже даёт переборное решение: можно рассмотреть все i и перебрать, сколько раз мы вычитаем a_i и a_{i+1}. А можно перебрать, сколько переводов мы совершим. Если мы совершим K переводов, то $i = \lfloor (N \mod 9)/K \rfloor$. Затем можно явно вычислить, сколько раз вы возьмём a_i и a_{i+1}, чтобы их сумма цифр равнялась $N \mod 9$.