Разбор дистанционного тура 1

Yandex Algo, C, 2023

17 ноября 2023 г.

A Кратность

Чтобы найти количество чисел, делящихся на Z, в отрезке $[X, Y]$, давайте воспользуемся стандартной идеей: найдём количество чисел, делящихся на Z, на префиксе $[1, Y]$ и вычтем из него количество чисел, делящихся на Z, на префиксе $[1, X-1]$. Итоговая формула: $\left[\frac{Y}{Z}\right] - \left[\frac{X-1}{Z}\right]$. Заметим, что при $X = 1$ решение также работает корректно.
В качестве частичного решения, проходящего тесты с небольшими X и Y, можно было проверить делимость всех чисел на Z линейным проходом по отрезку $[X, Y]$.

B Игра в напёрстки

Для начала отметим, что после N сдвигов шарик вернётся на прежнее место, поэтому достаточно всегда делать не X сдвигов, а лишь $X \mod N$: конечная позиция шарика от этого не поменяется. Обозначим $X \mod N$ за T. Также предлагается перейти в 0-нумерацию напёрстков для удобства вычислений.
Для решения на 60 баллов достаточно было наивно моделировать процесс за $O(N^2)$ или $O(N^3)$: создать вектор или дек из напёрстков, запомнить начальную позицию шарика и на очередном ходе по одному перемещать T напёрстков, удаляя первый элемент очереди/вектора и записывая его в конец.
Для полного решения заметим, что на каждом ходе мы можем вычислить новую позицию шарика за $O(1)$. Если текущая позиция шарика равна Y, и $Y - T \geq 0$, то новая позиция шарика равна $Y - T$. Иначе она равна $N - (T - Y)$. Нетрудно заметить, что можно объединить это одной формулой $(Y - T + N) \mod N$.

C Лиза и Потеря Времени

Важной частью задачи было заметить пункт «задержки при отправке и получении времени с сервера одинаковые». Это значит, что, чтобы вычислить длительность одной задержки, достаточно посчитать, сколько времени прошло между T_{send} и $T_{receive}$ и поделить пополам. Единственная дробная часть, которая при этом могла получиться — это ровно $\frac{1}{2}$, так что достаточно всегда округлять значение задержки вверх. Для того, чтобы получить текущее время, достаточно прибавить величину задержки к T_{real}.

Теперь разберёмся, как посчитать $T_{receive} - T_{send}$. Обозначим количество часов, минут и секунд в $T_{receive}$ за $H_{receive}, M_{receive}, S_{receive}$, аналогично для T_{send}. Давайте переведём оба времени в количество секунд, прошедшее от начала дня: $K_{receive} = H_{receive} \cdot 60^2 + M_{receive} \cdot 60 + S_{receive}$, аналогично для K_{send}. Если $K_{receive} - K_{send}$ получилось неотрицательным (то есть запрос происходил в одних сутках), то величина задержки равна $\lceil (K_{receive} - K_{send})/2 \rceil$. Иначе запрос происходил в разных сутках, тогда величина задержки равна $\lceil (24 \cdot 60^2 - K_{send} + K_{receive})/2 \rceil$. Нетрудно заметить, что эти случаи можно объединить формулой $\lceil (24 \cdot 60^2 + K_{send} - K_{receive}) \mod (24 \cdot 60^2)/2 \rceil$. Аналогично можно перевести в секунды время T_{real} и прибавить к нему задержку, получив текущее количество секунд от начала суток. Чтобы из него получить количество часов, нужно поделить на 60^2 и взять по модулю 24 (поскольку это время могло быть уже в новых сутках); количество минут — поделить на 60 и затем взять по модулю 60; секунд — взять по модулю 60.
Чтобы удобно считывать время в таком формате в C++, можно воспользоваться следующей конструкцией:

```
scanf("%d:%d:%d", &h, &m, &s);
```

где h, m, s — интовые переменные (для long long испльзуйте %lld вместо %d), куда вы хотите записать часы, минуты, секунды соответственно.

D Saratov City

Посчитаем суммарное количество людей, обозначим его за S. Ясно, что нам потребуется сделать не менее $\lceil \frac{S}{K} \rceil$ циклов подъёма и спуска, чтобы увезти всех людей. При этом мы хотим каждый раз подниматься на как можно более низкий этаж, чтобы затрачивать как можно меньше времени. Поэтому здесь применима жадная стратегия, попробуем сначала увезти всех людей с N-го этажа, затем с $(N-1)$-го, ..., в последнюю очередь с первого этажа. Если в какой-то момент лифт заполняется полностью, то добавляем к величине ответа $2\cdot$ номер максимального посещённого этажа на данном шаге и обнуляем количество людей в лифте. Количество пройденных тестов зависело от реализации данного решения.
Для прохождения тестов на 40 баллов достаточно было добавлять в лифт людей по одному либо пока он не заполнится, либо пока не закончатся люди на этаже.
Для решения на полный балл необходимо заметить, что могла возникнуть ситуация, когда грузопдъёмность лифта очень маленькая, а людей на этаже много. Тогда мы хотели бы быстро понимать, сколько раз придётся подняться на этот этаж и забирать людей только с него. Для этого, когда мы приезжаем на этаж i и не можем сразу увезти всех людей с него, необходимо добавить к ответу $2 \cdot \left[\frac{a_i - x}{K} \right] \cdot i$, где x — количество людей, которых мы увезём с i-го этажа, когда в первый раз приедем на него (у нас могло остаться ещё какое-то количество места в лифте с более верхних этажей).

E Чёрная пятница

Более формализованная версия условия: необходимо найти следующее по возрастанию после данного число, состоящее из K цифр такое, что сумма цифр в первой

половине равна сумме цифр во второй (или вывести число из K нулей, если такого нет).

Для прохождения тестов на 20 баллов достаточно было наивно пербирать числа в порядке возрастания. На 40 балов подобное решение тоже могло пройти, если перебирать не все числа, а только начиная с $N + 1$.

Для прохождения тестов на полный балл попробуем понять, как быстро конструировать число из фиксированного количества цифр $K/2$ с заданной суммой. Увеличим правую половину на 1 и попытаемся сначала изменить только правую половину, сделав сумму равной левой. Если не получится, увеличим левую половину на 1 и найдём минимальную подходящую правую половину (она точно найдётся, поскольку, можно, например, скопировать левую половину, и сумма будет такая же). Чтобы увеличить число, которое не помещается в стандартные типы данных, на 1, достаточно действовать, как в сложении столбиком: прибавить единицу к последнему разряду и осуществить нужное количество переносов при необходимости.

Итак, перейдём к нахождению правой части. Обозначим суммы в левой и правой части за S_l и S_r соответственно. Если $S_l \geq S_r$, то будем пытаться увеличить все разряды, начиная с младшего: увеличиваем каждое s_i до 9, а если разница d в суммах уже меньше, чем $9 - s_i$, то до $s_i + d$. Если же $S_l < S_r$, то для каждого $s_i < 9$, начиная с младшего, в правой половине, попробуем увеличить его на 1, а все последующие разряды обнулить. Если при этом у нас получилась сумма $\leq S_l$, то начнём увеличивать разряды с младшего, как в предыдущем случае. Если же мы перебрали все разряды до старшего, и всё ещё $S_l < S_r$, то необходимо увеличить левую половину на 1.