Задача A. Сумма простая

Задача была разобрана на лекции. Код для решения.

```
vector<int> pref_sum(n + 1);
for (int i = 0; i < n; i++) {
    pref_sum[i + 1] = pref_sum[i] + a[i];
}

// sum(l, r - 1): pref_sum[r] - pref_sum[l]
```

Задача B. Максимальная сумма

Построим префиксные суммы на данном массиве. Теперь мы знаем, что сумма на отрезке $[l, r)$ может быть представлена как $pref_r - pref_l$.

Переберем всевозможные правые границы отрезка r. Теперь для фиксированной правой границы мы хотим найти такую левую границу l, чтобы значение выражения $pref_r - pref_l$ было как можно больше. При этом в качестве левой границы l нам подходит любой индекс, меньший чем r. Поэтому для поиска оптимальной левой границы l достаточно поддерживать минимум среди всех рассмотренных ранее значений $pref_i$.

Время работы: $\mathcal{O}(n)$.

Задача C. Прибавляем, суммируем

Заведём массив d. Прибавление константы x на отрезке $[l, r)$ выглядит следующим образом: $d[l] += x$, $d[r] -= x$. Чтобы получить значение массива после всех операций, надо посчитать префиксные суммы на массиве d.

Задача D. Возрастающий массив

Заметим, что каждое значение массива после применения операции может быть равно a_i или $-a_i$. Также заметим, что чем меньше текущий элемент, тем лучше, потому что в качестве следующего элемента подходит больше различных значений. Отсюда возникает следующий жадный алгоритм.

В качестве первого элемента массива выберем число $b_1 = -|a_1|$. Далее предположим, что мы уже построили первые k элементов массива, и хотим построить $(k+1)$-й элемент. Если $-|a_{k+1}| \geqslant b_k$, то скажем, что $b_{k+1} = -|a_{k+1}|$. В противном случае, если $|a_{k+1}| \geqslant b_k$, то скажем, что $b_{k+1} = |a_{k+1}|$. Наконец, если оба условия не верны, построить отсортированный массив невозможно.

Время работы: $\mathcal{O}(n)$.

Задача E. Оптимальная пара

Воспользуемся той же идеей, которой мы пользовались при поиске отрезка с максимальной суммой. Переберем индекс j. Теперь мы хотим выбрать оптимальный индекс i, такой что $i < j$ и $a_j - a_i$ минимально. Для этого необходимо найти максимальный элемент среди всех элементов до j-го и выбрать его. Вместо того, чтобы каждый раз искать максимум, будем поддерживать его значение и позицию в процессе перебора j.

Время работы: $\mathcal{O}(n)$.

Задача F. Лето в Сочи

Воспользуемся фактом, что среднее арифметическое набора чисел $a_1, a_2, \ldots, a_k$ не меньше, чем минимальное из этих чисел. Более того, среднее арифметическое набора чисел $a_1, a_2, \ldots, a_k$ равно минимальному из этих чисел тогда и только тогда, когда $a_1 = a_2 = \ldots = a_k$.

Таким образом, сначала определим минимально возможное значение средней температуры: оно будет равно минимальному значению температуры в какой-то из дней, то есть $T = \min(t_1, t_2, \ldots, t_n)$.

Теперь необходимо найти подотрезок в массиве, состоящий из одинаковых элементов, равных T, максимальной длины. Для этого выделим все максимальные по включению подотрезки элементов, равных T.

Найдем первый элемент в массиве t_i, такой что $t_i = T$. После этого будем идти далее по массиву до тех пор, пока не встретим первый элемент t_j, такой что $t_j \neq T$. В этот момент нужно остановиться,

обновить ответ с значением $j-i$ и продолжить искать первый элемент в массиве, равный T, начиная с позиции j.

Так как данный цикл переберет каждый элемент массива по одному разу, время работы составит $\mathcal{O}(n)$.

Задача G. M — многомерность

Пересечение двух параллелепипедов — это либо тоже параллелепипед, либо пустое множество. Чтобы его найти, нужно независимо пересечь отрезки по каждой из M координат. Таким образом, мы можем пересечь два параллелепипеда за $O(M)$. Чтобы пересечь K параллелепипедов, нам потребуется $O(KM)$ времени.

Чтобы вычислить количество целочисленных точек внутри параллелепипеда, нужно посчитать количество таких точек на отрезке, соответствующем каждой из координат, и перемножить. Это можно сделать за $O(M)$.

Зафиксируем один параллелепипед A, которые мы не хотим включать в пересечение. Остальные параллелепипеды пересечём, назовём результат I. Как теперь посчитать количество точек в I, которые при этом не лежат в A? Очень просто: это количество точек в I минус количество точек в $I \cap A$.

Если мы будем перебирать A, то общее время работы составит $O(N^2M)$, что плохо. Но мы можем воспользоваться тем, что мы постоянно пересекаем одни и те же параллелепипеды. Посчитаем пересечения для всех префиксов и всех суффиксов параллелепипедов за $O(NM)$. Тогда найти пересечение всех, кроме одного, мы сможем за $O(M)$: достаточно пересечь префикс до исключённого, и суффикс до него же.

Общее время работы составит $O(NM)$.

Задача H. Антипа и бухгалтерия

В данной задаче необходимо восстановить массив префиксных сумм таким образом, чтобы элементы исходного массива были положительными.

Аналогично предыдущей задаче выделим все отрезки массива префиксных сумм, состоящие из значений -1. Рассмотрим один из таких отрезков $[l, r)$. Мы знаем, чему была равна сумма чисел исходного массива на этом отрезке:

- Если $r \leqslant n$, то сумма равна $a_r - a_{l-1}$;
- Иначе сумма может быть равна любому числу, что хорошо.

Теперь, если оказалось, что сумма элементов меньше, чем длина отрезка $r - l + 1$, то искомого массива не существует. В противном случае скажем, что в первые $r - l$ дней элементы исходного массива были равны 1, а весь остаток сложим в последний элемент отрезка. После этого перейдем к следующему отрезку, состоящему из -1, и обработаем его аналогично.

Время работы: $\mathcal{O}(n)$.

Задача I. Межпланетный лифт

Заметим, что нам не важны позиции лифтов, важна только разность их позиций. Далее будем считать, что первый лифт находится ниже второго (если это не так, то просто поменяем их программы и позиции). Тогда возможны 4 ситуации:

- Если первый лифт едет вниз, а второй вверх, то разность увеличивается на 2.
- Если первый лифт едет вверх, а второй вниз, то разность уменьшается на 2.
- Иначе она не изменяется.

Также понятно, то лифты ломаются, если разность их позиций становится отрицательной или нулевой. Также можно заметить, что если $K > M$, то программы всегда можно исправить. Действительно, если на текущей итерации лифты поехали друг навстречу другу, то просто поменяем

один символ в одной из программ, после чего разность их позиций не изменится. Также понятно, что нет смысла в том, чтобы менять один символ несколько раз.

Для решения первой подгруппы переберем, какие символы надо изменить в первой программе и проверим, что в таком случае лифты не столкнутся. Если они не сломаются, то проверим, что не превысили ограничение по изменениям. Сложность этого решения $\mathcal{O}(2^M \cdot M)$.

Для второй группы необходимо просто уметь проверять, что лифты не столкнутся — просто симулировать процесс передвижения лифтов и проверять, что разность позиций не становится неположительной. Сложность этого решения $\mathcal{O}(M)$.

В третьей подгруппе можно было полностью изменить строку. Алгоритм для этой подгруппы описан в начале разбора. Сложность этого решения $\mathcal{O}(M)$.

Для полного решения задачи заметим, что если в какой-то момент лифты столкнулись, то можно просто в этот момент изменить один символ так, чтобы они не столкнулись (тогда разность позиций не изменится). Просто будем делать это каждый раз, когда лифты сталкиваются. Почему это оптимально? Потому что если сейчас есть свободная операция, то ее нужно поставить не позже первого момента столкновения. Но тогда можно использовать ее в момент столкновения (предотвратив его) и тогда мы сделаем не хуже, чем если бы использовали ее раньше (потому что просто на суффиксе к разности позиций прибавляется число). Сложность этого решения $\mathcal{O}(M)$.

Задача J. Очередная задача про игру с камнями

Автор задачи: Никита Голиков, разработчики: Михаил Кондрашин и Михаил Первеев

Подзадача 1

В первой подзадаче достаточно реализовать перебор. Для каждой кучки переберем всевозможные способы выбрать некоторое количество камней из нее. Для каждого такого варианта вычислим суммарное количество выбранных камней. Для всех способов, когда суммарное количество взятых камней равно s, для каждой кучки запомним, сколько камней было выбрано из нее, чтобы вычислить ответ.

Асимптотика: $\mathcal{O}\left(n \cdot \prod_{i=1}^{n}(r_i - l_i + 1)\right)$.

Подзадача 2

Зафиксируем кучку i, для которой мы хотим вычислить ответ. После этого переберем количество камней $x \in [l_i, r_i]$, которое мы хотим взять из этой кучки, и выясним, можно ли сделать это, получив суммарное количество камней s. Для этого необходимо выяснить, можно ли, используя остальные кучки, набрать суммарное количество камней $s - x$.

Заметим, что если у нас есть две кучки, такие что из первой кучки можно взять любое количество камней из диапазона $[l_1, r_1]$, а из второй кучки — любое количество камней в диапазоне $[l_2, r_2]$, то, используя две кучки, можно получить любое количество камней в диапазоне $[l_1 + l_2, r_1 + r_2]$.

Используя это замечание, можно легко вычислить диапазон количеств камней, которые можно получить, используя все кучки, кроме i-й, после чего проверить, лежит ли число $s - x$ в данном диапазоне.

Асимптотика: $\mathcal{O}(n^2 \cdot m)$.

Подзадача 3

Для решения данной подзадачи можно оптимизировать решение второй подзадачи. Вместо того, чтобы перебирать число x, научимся за $\mathcal{O}(1)$ выяснять, сколько подходящих чисел x существует в диапазоне $[l_i, r_i]$. Мы знаем, что $x \in [l_i, r_i]$. Также мы знаем, что при помощи остальных кучек мы можем набрать произвольное количество камней $y \in [L, R]$ (числа L и R мы научились вычислять во второй подзадаче). Мы хотим вычислить количество чисел x, таких что существует число y, такое что $x + y = s$. Иными словами, $x = s - y$. Для того, чтобы сделать это, необходимо пересечь

диапазоны $[l_i, r_i]$ и $[s - R, s - L]$. Количество целых точек в пересечении этих диапазонов равно количеству вариантов выбрать число x.

Асимптотика: $\mathcal{O}(n^2)$.

Подзадача 4

Для решения данной подзадачи можно также оптимизировать решение второй подзадачи, но сделать это иным способом. Для начала один раз найдем диапазон количеств камней, которое можно набрать, используя все кучки. Данный диапазон равен $[L, R]$, где $L = \sum_{i=1}^{n} l_i$, а $R = \sum_{i=1}^{n} r_i$. Теперь заметим, что если мы хотим использовать все кучки, кроме i-й, то диапазон количеств будет равен $[L - l_i, R - r_i]$. Далее переберем число x из диапазона $[l_i, r_i]$ и проверим, лежит ли число $s - x$ в диапазоне $[L - l_i, R - r_i]$.

Асимптотика: $\mathcal{O}(n \cdot m)$.

Подзадача 5

Наконец, для того, чтобы получить полное решение, достаточно применить обе рассмотренные оптимизации.

Асимптотика: $\mathcal{O}(n)$.