Задача A. Соки

Апельсиновый сок расходуется последним, поэтому его останется $\min(x, c)$ (не больше, чем его было изначально, и чем осталось). Тогда яблочного и вишнёвого сока суммарно останется $y = x - \min(x, c)$. Аналогично вишнёвого сока останется $\min(y, b)$. Тогда яблочного сока осталось $x - \min(x, c) - \min(y, b)$.

Задача B. Хоровод

Для начала разберём случай $x = y$. Понятно, что максимальное количество пар соседей разного пола достигается при чередовании мальчиков и девочек: `BGBGBG...BG`. В данном случае количество пар равно $x + y$. Минимальное количество пар достигается, если все мальчики идут подряд, а потом все девочки идут подряд: `BB...BBGG...GG`, тогда количество пар равно 2.

Заметим, что количество пар соседей разного пола всегда чётно. Чтобы доказать это, можно рассмотреть блок подряд идущих мальчиков, с ним образуется ровно 2 пары с девочками (слева и справа). Если просуммировать это по всем блокам мальчиков, то мы получим ровно количество пар соседей разного пола, и оно будет чётным.

Теперь, чтобы получить меньше, чем $x + y$ пар соседей разного пола, можно начать объединять детей одного пола в блоки. Например:

- `GBGBGBGB` — 8 пар;
- `GGBBGBGB` — 6 пар;
- `GGGBBBGB` — 4 пары;
- `GGGGBBBB` — 2 пары.

То есть, мы каждый раз берём следующего мальчика и девочку и добавляем в блоки подряд идущих мальчиков и девочек, уменьшая количество пар на 2. Так можно получить любое чётное количество пар от 2 до $x + y$.

Теперь перейдём к случаю $x \neq y$. Понятно, что у нас не могут просто чередоваться мальчики и девочки. Однако максимальное количество пар достигается, когда они почти чередуются: `BGBGBG...BGBGGGGG` (в случае, если девочек больше). Количество пар соседей разного пола в этом случае равно $2 \cdot \min(x, y)$. Чтобы получить какое-то промежуточное количество пар (между 2 и $2 \cdot \min(x, y)$), можно так же, как описано выше, объединять мальчиков и девочек в блоки.

Задача C. Лавка с числами

Сначала разберём отдельный случай: с помощью суммы чисел на отрезке можно представить любое число, если $l = 1$. В противном случае в виде суммы нельзя представить число 1. То есть ответ — -1 тогда и только тогда, когда $l = 1$.

Далее, число k представимо в виде суммы x чисел из отрезка, если выполняется $l \cdot x \leqslant k \leqslant r \cdot x$. А значит число не представимо, если для какого-то x имеем $r \cdot x < k < l \cdot (x+1)$, то есть x слагаемых — мало, а $x + 1$ — много. Значит, нам нужно найти максимальный «пропуск» такого вида.

Имеем неравенство, где x — количество слагаемых: $r \cdot x + 1 < l \cdot (x + 1)$ (так как между ними должно быть хотя бы одно число). Решая его, получаем $x < \frac{a-1}{b-a}$. А так как нам нужен максимальный целый x, то он будет равен $\lfloor \frac{a-2}{b-a} \rfloor$. Мы нашли максимальный «пропуск», а максимальное число из него равняется $l \cdot (\lfloor \frac{a-2}{b-a} \rfloor + 1) - 1$.

Задача D. Крайний Банк

Пусть мы знаем все пары покупка-продажа, как трейдер дошел до состояния, описанного во входных данных. Заметим, что если есть пара вида (i, j) и в момент j у нас была какая-то незакрытая сделка, в которой стоимость покупки больше чем a_i, но при этом меньше a_j, то мы можем поменять моменты покупки в этих парах и всё останется корректным.

Из этого замечания сразу же следует жадный алгоритм для решения задачи. Давайте идти по моментам времени и если мы находимся в моменте где трейдер продавал, то будем ставить ему в

пару максимальный незанятый момент покупки, стоимость товара в котором не превышает текущей стоимости продажи. Таким образом мы либо получим корректное разбиение на пары, либо в какой-то момент столкнемся с противоречием (не будет подходящего момента покупки), что значит, что трейдер никак не мог исполнить условие, для того, чтобы быть потенциальным мошенником.

Для эффективной реализации данного решения предлагается для каждого значения a_i массив с ещё не занятыми моментами покупки, имеющими такую стоимость. Если нам встретится момент продажи, то нам нужно будет найти максимальный, по стоимости, не занятый момент покупки, не превышающий данного a_i. Это легко можно сделать используя массивы, который мы поддерживаем (достаточно пройтись от $a_i - 1$ до 1 и найти первый непустой массив). После этого мы создаем пару с любым элементом из этого массива и данным индексом. Удаляем данный индекс из этого массива и продолжаем процесс. Заметим, что для эффективной реализации следует всегда брать в пару последний элемент из массива, так как его удаление будет работать за $O(1)$. Таким образом мы получаем решение за $O(n)$.

Задача E. Максимальная попарная разница

Если раскрыть все модули в $|a_1 - b_{p_1}| + |a_2 - b_{p_2}| + \ldots + |a_n - b_{p_n}|$, то n чисел из $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ войдут в сумму со знаком плюс, а оставшиеся n чисел со знаком минус.

Поэтому, максимальный счёт точно не может быть больше чем «сумма n максимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ минус сумма n минимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$». И оказывается, что такого счёта всегда можно достичь.

Давайте отсортируем все числа $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ и посмотрим: сколько чисел из массива a окажется в первой половине отсортированного массива, пусть x. Тогда чисел из массива b в первой половине всего $n - x$ штук. А во второй половине: $n - x$ чисел из массива a, и x чисел из массива b. Поэтому мы можем образовать n пар: «число из массива a, число из массива b» так, что в каждой паре по одному числу из каждой половины отсортированного массива. И счёт в таком случае, будет в точности равен «сумма n максимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ минус сумма n минимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$».

А число способов это сделать будет $x! \cdot (n - x)!$, где $k! = 1 \cdot 2 \cdot \ldots \cdot k$. Так как нужно независимо сопоставить x наименьшим числам из a — x наибольших чисел из b, и независимо $n - x$ наибольшим числам из a — $n - x$ наименьших чисел из b.

При этом при любом другом способе, хотя бы одна из пар (a_i, b_{p_i}) будет содержать пару чисел из одной и той же половины отсортированного массива, и, так как все числа различны, в таком случае счёт будет не максимален.

Асимптотика решения: $O(n \log n)$, из-за сортировки массива.

Задача A. Хоровод

Для начала разберём случай $x = y$. Понятно, что максимальное количество пар соседей разного пола достигается при чередовании мальчиков и девочек: `BGBGBG...BG`. В данном случае количество пар равно $x + y$. Минимальное количество пар достигается, если все мальчики идут подряд, а потом все девочки идут подряд: `BB...BBGG...GG`, тогда количество пар равно 2.

Заметим, что количество пар соседей разного пола всегда чётно. Чтобы доказать это, можно рассмотреть блок подряд идущих мальчиков, с ним образуется ровно 2 пары с девочками (слева и справа). Если просуммировать это по всем блокам мальчиков, то мы получим ровно количество пар соседей разного пола, и оно будет чётным.

Теперь, чтобы получить меньше, чем $x + y$ пар соседей разного пола, можно начать объединять детей одного пола в блоки. Например:

- `GBGBGBGB` — 8 пар;
- `GGBBGBGB` — 6 пар;
- `GGGBBBGB` — 4 пары;
- `GGGGBBBB` — 2 пары.

То есть, мы каждый раз берём следующего мальчика и девочку и добавляем в блоки подряд идущих мальчиков и девочек, уменьшая количество пар на 2. Так можно получить любое чётное количество пар от 2 до $x + y$.

Теперь перейдём к случаю $x \neq y$. Понятно, что у нас не могут просто чередоваться мальчики и девочки. Однако максимальное количество пар достигается, когда они почти чередуются: `BGBGBG...BGBGGGGG` (в случае, если девочек больше). Количество пар соседей разного пола в этом случае равно $2 \cdot \min(x, y)$. Чтобы получить какое-то промежуточное количество пар (между 2 и $2 \cdot \min(x, y)$), можно так же, как описано выше, объединять мальчиков и девочек в блоки.

Задача B. Суммы из отрезка

Сначала разберём отдельный случай: с помощью суммы чисел на отрезке можно представить любое число, если $l = 1$. В противном случае в виде суммы нельзя представить число 1. То есть ответ — -1 тогда и только тогда, когда $l = 1$.

Далее, число k представимо в виде суммы x чисел из отрезка, если выполняется $l \cdot x \leqslant k \leqslant r \cdot x$. А значит число не представимо, если для какого-то x имеем $r \cdot x < k < l \cdot (x+1)$, то есть x слагаемых — мало, а $x + 1$ — много. Значит, нам нужно найти максимальный «пропуск» такого вида.

Имеем неравенство, где x — количество слагаемых: $r \cdot x + 1 < l \cdot (x + 1)$ (так как между ними должно быть хотя бы одно число). Решая его, получаем $x < \frac{a-1}{b-a}$. А так как нам нужен максимальный целый x, то он будет равен $\lfloor \frac{a-2}{b-a} \rfloor$. Мы нашли максимальный «пропуск», а максимальное число из него равняется $l \cdot (\lfloor \frac{a-2}{b-a} \rfloor + 1) - 1$.

Задача C. Крайний Банк

Пусть мы знаем все пары покупка-продажа, как трейдер дошел до состояния, описанного во входных данных. Заметим, что если есть пара вида (i, j) и в момент j у нас была какая-то незакрытая сделка, в которой стоимость покупки больше, чем a_i, но при этом меньше a_j, то мы можем поменять моменты покупки в этих парах и все останется корректным.

Из этого замечания сразу же следует жадный алгоритм для решения задачи. Давайте идти по моментам времени и если мы находимся в моменте, где трейдер продавал, то будем ставить ему в пару максимальные незанятые моменты покупки, стоимость товара в которых не превышает текущей стоимости продажи. Таким образом мы либо получим корректное разбиение на пары, либо в какой-то момент столкнемся с противоречием (не будет подходящих моментов покупки), что значит, что трейдер никак не мог исполнить условие, для того чтобы быть потенциальным мошенником.

Для эффективной реализации данного решения предлагается хранить multiset с парами вида (a_i, c_i), означающих, что у нас есть «незакрытая» покупка по цене a_i, в которую мы купили c_i

бипок. При продаже нужно делать lower bound и брать предыдущий индекс, после чего корректно пересчитать multiset, и так пока мы не закроем всю продажу. Такое решение работает за $O(n \log n)$.

Задача D. Максимальная попарная разница

Если раскрыть все модули в $|a_1 - b_{p_1}| + |a_2 - b_{p_2}| + \ldots + |a_n - b_{p_n}|$, то n чисел из $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ войдут в сумму со знаком плюс, а оставшиеся n чисел со знаком минус.

Поэтому, максимальный счёт точно не может быть больше чем «сумма n максимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ минус сумма n минимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$». И оказывается, что такого счёта всегда можно достичь.

Давайте отсортируем все числа $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ и посмотрим: сколько чисел из массива a окажется в первой половине отсортированного массива, пусть x. Тогда чисел из массива b в первой половине всего $n - x$ штук. А во второй половине: $n - x$ чисел из массива a, и x чисел из массива b. Поэтому мы можем образовать n пар: «число из массива a, число из массива b» так, что в каждой паре по одному числу из каждой половины отсортированного массива. И счёт в таком случае, будет в точности равен «сумма n максимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ минус сумма n минимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$».

А число способов это сделать будет $x! \cdot (n - x)!$, где $k! = 1 \cdot 2 \cdot \ldots \cdot k$. Так как нужно независимо сопоставить x наименьшим числам из a — x наибольших чисел из b, и независимо $n - x$ наибольшим числам из a — $n - x$ наименьших чисел из b.

При этом при любом другом способе, хотя бы одна из пар (a_i, b_{p_i}) будет содержать пару чисел из одной и той же половины отсортированного массива, и, так как все числа различны, в таком случае счёт будет не максимален.

Асимптотика решения: $O(n \log n)$, из-за сортировки массива.

Задача E. Рекордное удаление

1 группа

Для каждого запроса промоделируем процесс удаления. На каждом этапе будем искать последовательность рекордов и запоминать элементы, которые мы уже удалили. Получаем асимптотику $O(nk)$ на один запрос и $O(qnk)$ итоговую.

2 группа

В данной группе перестановка отсортирована по убыванию. Рассмотрим произвольный запрос. Заметим, что на каждом этапе удаления длина последовательности рекордов не превосходит 1. Если мы совершаем k этапов удаления, то ответ это минимум из k и количества чисел на суффиксе.

3 группа

Сохраним для каждой позиции перестановки все запросы, суффиксы которых начинаются в этой позиции. Будем перебирать индекс перестановки от конца к началу и отвечать сразу на все запросы для одной позиции. Для решения этой группы заведем стек рекордов. При переходе от позиции i к позиции $i - 1$ удалим с верхушки стека все элементы, меньшие a_{i-1} и положим a_{i-1} в стек (т.к. a_{i-1} точно окажется в последовательности рекордов на первом этапе удаления). Чтобы узнать длину последовательности рекордов, посмотрим на размер стека. Каждый элемент попадает в стек ровно один раз и будет удален из стека не более одного раза, поэтому итоговая асимптотика $O(n + q)$.

4 группа

Улучшим решение для предыдущей группы. Вместо одного стека будем поддерживать два — в первом будет храниться последовательность рекордов для текущего суффикса, а во втором последовательность рекордов, если совершить 1 этап удаления. Посмотрим на те элементы, которые были удалены из первого стека при переходе от i к $i - 1$. Заметим, что эти элементы неизбежно окажутся в последовательности рекордов после первого этапа удаления. Мы добавим их во второй стек, но перед этим нужно понять, какие элементы могут перестать входить в последовательность рекордов после первого этапа удаления. На самом деле, это все те элементы, что не превосходят максимум среди добавляемых элементов. Удалим все эти значения из второго стека и положим новые. Это легко сделать, так как значения в стеках идут в возрастающем порядке. Можно заметить, что после такой процедуры значения в стеках по-прежнему будут идти в возрастающем порядке.

5 группа

Воспользуемся тем, что k в любом запросе не превосходит 20. Обобщим идеи для прошлых групп и заведем сразу 20 стеков. В i-м стеке будет лежать последовательность рекордов после $i-1$ этапа удалений. При добавлении нового элемента обновим первый стек, а дальше будем идти стекам, начиная со второго, и обновлять их через те элементы, которые были выброшены из предыдущего стека. Если на каком-то шаге мы не выбросили ни одного элемента, прекратим этот процесс. Для ответа на запрос посчитаем сумму размеров первых k стеков. Каждый элемент добавится в каждый стек и удалится из каждого стека не более одного раза, поэтому итоговая асимптотика $O(nk+qk)$.

Задача F. Дорожный патруль

Заметим, что оптимальными значениями для ограничения являются числа a_i, поэтому таких значений не более n. Давайте отсортируем машины по убыванию скорости и будем при добавлении очередной машины заново считать ответ.

При $t \geqslant 3000$ заметим, что мы остановим не более $O(\frac{n}{t})$ машин, поэтому мы можем сложить позиции всех машин со скоростью больше данной в set, а следующую, которую нужно остановить, искать с помощью lower bound. Такое решение работает за $O(n\frac{n}{t}\log n)$.

В общем случае, давайте разделим весь массив на блоки по $\sqrt{n}$, для каждого элемента будем пересчитывать первую машину **вне его блока**, которую мы остановим после него. Аналогично будем пересчитывать сумму a_i всех машин между данной и следующей из другого блока, а также число таких машин. Заметим, что при добавлении новой машины мы можем пересчитать весь блок за его размер (т.е. за $O(\sqrt{n})$) проходом справа налево.

Осталось уметь пересчитывать ответ, используя такую структуру. Достаточно поддерживать первую машину, которую мы остановим, после чего, начиная с неё, переходить к машинам в следующих блоках, с помощью того, что мы насчитали. При таком процессе нужно поддерживать число остановленных машин и сумму их скоростей, после чего для данного ограничения скорости можно восстановить ответ. Такое решение работает за $O(n\sqrt{n})$.

Задача A. Суммы из отрезка

Сначала разберём отдельный случай: с помощью суммы чисел на отрезке можно представить любое число, если $l = 1$. В противном случае в виде суммы нельзя представить число 1. То есть ответ — -1 тогда и только тогда, когда $l = 1$.

Далее, число k представимо в виде суммы x чисел из отрезка, если выполняется $l \cdot x \leqslant k \leqslant r \cdot x$. А значит число не представимо, если для какого-то x имеем $r \cdot x < k < l \cdot (x+1)$, то есть x слагаемых — мало, а $x + 1$ — много. Значит, нам нужно найти максимальный «пропуск» такого вида.

Имеем неравенство, где x — количество слагаемых: $r \cdot x + 1 < l \cdot (x + 1)$ (так как между ними должно быть хотя бы одно число). Решая его, получаем $x < \frac{a-1}{b-a}$. А так как нам нужен максимальный целый x, то он будет равен $\lfloor \frac{a-2}{b-a} \rfloor$. Мы нашли максимальный «пропуск», а максимальное число из него равняется $l \cdot (\lfloor \frac{a-2}{b-a} \rfloor + 1) - 1$.

Задача B. Крайний Банк

Пусть мы знаем все пары покупка-продажа, как трейдер дошел до состояния, описанного во входных данных. Заметим, что если есть пара вида (i, j) и в момент j у нас была какая-то незакрытая сделка, в которой стоимость покупки больше, чем a_i, но при этом меньше a_j, то мы можем поменять моменты покупки в этих парах и все останется корректным.

Из этого замечания сразу же следует жадный алгоритм для решения задачи. Давайте идти по моментам времени и если мы находимся в моменте, где трейдер продавал, то будем ставить ему в пару максимальные незанятые моменты покупки, стоимость товара в которых не превышает текущей стоимости продажи. Таким образом мы либо получим корректное разбиение на пары, либо в какой-то момент столкнемся с противоречием (не будет подходящих моментов покупки), что значит, что трейдер никак не мог исполнить условие, для того чтобы быть потенциальным мошенником.

Для эффективной реализации данного решения предлагается хранить multiset с парами вида (a_i, c_i), означающих, что у нас есть «незакрытая» покупка по цене a_i, в которую мы купили c_i бипок. При продаже нужно делать lower bound и брать предыдущий индекс, после чего корректно пересчитать multiset, и так пока мы не закроем всю продажу. Такое решение работает за $O(n \log n)$.

Задача C. Максимальная попарная разница

Если раскрыть все модули в $|a_1 - b_{p_1}| + |a_2 - b_{p_2}| + \ldots + |a_n - b_{p_n}|$, то n чисел из $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ войдут в сумму со знаком плюс, а оставшиеся n чисел со знаком минус.

Поэтому, максимальный счёт точно не может быть больше чем «сумма n максимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ минус сумма n минимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$». И оказывается, что такого счёта всегда можно достичь.

Давайте отсортируем все числа $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ и посмотрим: сколько чисел из массива a окажется в первой половине отсортированного массива, пусть x. Тогда чисел из массива b в первой половине всего $n - x$ штук. А во второй половине: $n - x$ чисел из массива a, и x чисел из массива b. Поэтому мы можем образовать n пар: «число из массива a, число из массива b» так, что в каждой паре по одному числу из каждой половины отсортированного массива. И счёт в таком случае, будет в точности равен «сумма n максимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ минус сумма n минимальных чисел среди $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$».

А число способов это сделать будет $x! \cdot (n - x)!$, где $k! = 1 \cdot 2 \cdot \ldots \cdot k$. Так как нужно независимо сопоставить x наименьшим числам из a — x наибольших чисел из b, и независимо $n - x$ наибольшим числам из a — $n - x$ наименьших чисел из b.

При этом при любом другом способе, хотя бы одна из пар (a_i, b_{p_i}) будет содержать пару чисел из одной и той же половины отсортированного массива, и, так как все числа различны, в таком случае счёт будет не максимален.

Асимптотика решения: $O(n \log n)$, из-за сортировки массива.

Задача D. Рекордное удаление

1 группа

Для каждого запроса промоделируем процесс удаления. На каждом этапе будем искать последовательность рекордов и запоминать элементы, которые мы уже удалили. Получаем асимптотику $O(nk)$ на один запрос и $O(qnk)$ итоговую.

2 группа

В данной группе перестановка отсортирована по убыванию. Рассмотрим произвольный запрос. Заметим, что на каждом этапе удаления длина последовательности рекордов не превосходит 1. Если мы совершаем k этапов удаления, то ответ это минимум из k и количества чисел на суффиксе.

3 группа

Сохраним для каждой позиции перестановки все запросы, суффиксы которых начинаются в этой позиции. Будем перебирать индекс перестановки от конца к началу и отвечать сразу на все запросы для одной позиции. Для решения этой группы заведем стек рекордов. При переходе от позиции i к позиции $i-1$ удалим с верхушки стека все элементы, меньшие a_{i-1} и положим a_{i-1} в стек (т.к. a_{i-1} точно окажется в последовательности рекордов на первом этапе удаления). Чтобы узнать длину последовательности рекордов, посмотрим на размер стека. Каждый элемент попадает в стек ровно один раз и будет удален из стека не более одного раза, поэтому итоговая асимптотика $O(n+q)$.

4 группа

Улучшим решение для предыдущей группы. Вместо одного стека будем поддерживать два — в первом будет храниться последовательность рекордов для текущего суффикса, а во втором последовательность рекордов, если совершить 1 этап удаления. Посмотрим на те элементы, которые были удалены из первого стека при переходе от i к $i-1$. Заметим, что эти элементы неизбежно окажутся в последовательности рекордов после первого этапа удаления. Мы добавим их во второй стек, но перед этим нужно понять, какие элементы могут перестать входить в последовательность рекордов после первого этапа удаления. На самом деле, это все те элементы, что не превосходят максимум среди добавляемых элементов. Удалим все эти значения из второго стека и положим новые. Это легко сделать, так как значения в стеках идут в возрастающем порядке. Можно заметить, что после такой процедуры значения в стеках по-прежнему будут идти в возрастающем порядке.

5 группа

Воспользуемся тем, что k в любом запросе не превосходит 20. Обобщим идеи для прошлых групп и заведем сразу 20 стеков. В i-м стеке будет лежать последовательность рекордов после $i-1$ этапа удалений. При добавлении нового элемента обновим первый стек, а дальше будем идти стекам, начиная со второго, и обновлять их через те элементы, которые были выброшены из предыдущего стека. Если на каком-то шаге мы не выбросили ни одного элемента, прекратим этот процесс. Для ответа на запрос посчитаем сумму размеров первых k стеков. Каждый элемент добавится в каждый стек и удалится из каждого стека не более одного раза, поэтому итоговая асимптотика $O(nk+qk)$.

Задача E. Дорожный патруль

Заметим, что оптимальными значениями для ограничения являются числа a_i, поэтому таких значений не более n. Давайте отсортируем машины по убыванию скорости и будем при добавлении очередной машины заново считать ответ.

При $t \geqslant 3000$ заметим, что мы остановим не более $O(\frac{n}{t})$ машин, поэтому мы можем сложить позиции всех машин со скоростью больше данной в set, а следующую, которую нужно остановить, искать с помощью lower bound. Такое решение работает за $O(n\frac{n}{t}\log n)$.

В общем случае, давайте разделим весь массив на блоки по $\sqrt{n}$, для каждого элемента будем пересчитывать первую машину **вне его блока**, которую мы остановим после него. Аналогично будем пересчитывать сумму a_i всех машин между данной и следующей из другого блока, а также число таких машин. Заметим, что при добавлении новой машины мы можем пересчитать весь блок за его размер (т.е. за $O(\sqrt{n})$) проходом справа налево.

Осталось уметь пересчитывать ответ, используя такую структуру. Достаточно поддерживать первую машину, которую мы остановим, после чего, начиная с неё, переходить к машинам в следующих блоках, с помощью того, что мы насчитали. При таком процессе нужно поддерживать число остановленных машин и сумму их скоростей, после чего для данного ограничения скорости можно восстановить ответ. Такое решение работает за $O(n\sqrt{n})$.

Задача F. Бесконечная зима в Карелии

Подзадачи 1–2**.**

В этих подзадачах действовало ограничение $k = 2$. Попробуем для каждого запроса x вычислить наименьшее количество монет a_2, которые можно использовать, чтобы набрать данную сумму.

Заметим, что это количество зависит только от $x \bmod a_1$. А именно мы знаем, что если $x \equiv k \cdot a_2 \pmod{a_1}$, то можно использовать в сумме k монет номиналом a_2, а всю оставшуюся сумму добить монетами номиналом a_1. Рассматривать значения $k \geqslant a_1$ бессмысленно, поэтому мы можем просто перебрать k от 0 до $a_1 - 1$ и для каждого остатка при делении на a_1 вычислить минимальное количество монет номиналом a_2, которое надо набрать в сумму, пусть это количество равно $\min 2[r]$ для $x \equiv r \pmod{a_1}$. Эта часть решения работает за время $O(d)$.

С помощью этого массива легко можно проверить, возможно ли решение в принципе, а также найти решение, которые использует минимальное количество монет номиналом a_2. Разберемся с тем, как выглядят другие решения. Очевидно, что для получения других решений нам понадобится уменьшать количество монет номиналом a_1 и увеличивать количество монет номиналом a_2. Пусть g — наибольший общий делитель a_1 и a_2, тогда мы можем уменьшать количество монет первого номинала на $\frac{a_1}{g}$, а количество монет второго номинала увеличивать на $\frac{a_2}{g}$. Таким образом для подсчета количества решений достаточно вычислить, сколько раз мы можем уменьшать количество монет номинала a_1.

Подзадачи 3–4**.**

Воспользуемся методом динамического программирования. Пусть $\mathrm{dp}[l][x]$ — количество способов набрать x рублей, используя только монеты номиналом $a_1, \ldots, a_l$. Очевидно, что для $l = 0$ есть только одно ненулевое состояние динамики — $\mathrm{dp}[0][0] = 1$. Разберем как выглядит переход, если мы хотим вычислить l-й слой динамики:

- Количество способов набрать сумму x, которые не используют монету a_l равно $\mathrm{dp}[l-1][x]$;
- Количество способов набрать сумму x, которые используют монету a_l равно $\mathrm{dp}[l][x - a_l]$, так как в любом из этих способов можно вычесть одну монету номиналом a_l.

Таким образом $\mathrm{dp}[l][x] = \mathrm{dp}[l-1][x] + \mathrm{dp}[l][x - a_l]$ (если $x < a_l$, то второе слагаемое отсутствует). Эту динамику легко вычислить за время $O(k \cdot \max\{x_i\})$, а затем за $O(1)$ ответить на каждый из запросов.

Подзадача 5**.**

Решение этой подзадачи не ведет к полному решению, но использует популярный в других задачах подход. Дело в том, что динамику можно вычислять с помощью возведения матрицы в степень. Способов это сделать много, но мы приведем самый простой, если вы не знакомы с этой техникой.

$$\mathrm{dp}[x] = \sum_{\emptyset \neq B \subseteq \{a_1, \ldots, a_k\}} (-1)^{|B|-1} \mathrm{dp}\left[x - \sum_{y \in B} y\right]$$

Тут $\mathrm{dp}[x]$ — непосредственно ответ на задачу, количество способов набрать сумму x с помощью монет номиналом $a_1, \ldots, a_k$ ($dp[x] = 0$ для $x < 0$). Объясним почему эта формула верна:

- Каждый набор учтен не менее одного раза в сумме $\mathrm{dp}[x - a_1] + \ldots + \mathrm{dp}[x -_a k]$, мы просто перебираем монету, которая есть в наборе и вычитаем ее.
- Но если набор содержит сразу две различные монеты, то он учтен в этой сумме дважды, поэтому все такие наборы мы опять таки вычитаем $-\sum_{i<j} \mathrm{dp}[x - a_i - a_j]$.
- Но теперь наборы, которые содержат три различные монеты, вообще не учтены в сумме, поэтому их надо прибавить, и так далее.

То, что произошло выше, является классическим объяснением формулы включений-исключений. Теперь вернемся к рекуррентному выражению. Если сгруппировать одинаковые слагаемые, то мы заметим, что формула может иметь вид:

$$\mathrm{dp} = \sum_{j=1}^{dk} \lambda_j \cdot \mathrm{dp}[x-j]$$

Где $\lambda_1, \ldots, \lambda_{dk}$ — какие-то коэффициенты. В этой подзадаче можно было наивно перебрать все подмножества делителей и наивно вычислить эти коэффициенты за время $O(2^k)$. Но теперь после всех этих действий мы, наконец, можем определить переход динамики с помощью матриц:

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \lambda_{dk} & \lambda_{dk-1} & \lambda dk-2 & \cdots & \lambda_1 \end{pmatrix} \cdot \begin{pmatrix} \mathrm{dp}[n-dk+1] \\ \mathrm{dp}[n-dk+2] \\ \vdots \\ \mathrm{dp}[n-1] \\ \mathrm{dp}[n] \end{pmatrix} = \begin{pmatrix} \mathrm{dp}[n-dk+2] \\ \mathrm{dp}[n-dk+3] \\ \vdots \\ \mathrm{dp}[n] \\ \mathrm{dp}[n+1] \end{pmatrix}$$

Так как произведение матриц ассоциативно, то можно и выразить нужное состояние динамики с помощью операции возведения матрицы в степень (символом $*$ обозначены числа, которые нас не интересуют):

$$\begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ \lambda_{dk} & \lambda_{dk-1} & \lambda dk-2 & \cdots & \lambda_1 \end{pmatrix}^N \cdot \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} * \\ * \\ \vdots \\ * \\ \mathrm{dp}[N] \end{pmatrix}$$

В итоге получили решение с асимптотикой $O(q \cdot (dk)^3 \cdot \log x)$, это же решение можно оптимизировать до времени $O((dk)^3 \log C + q \cdot (dk)^2 \cdot \log x)$.

Подзадачи 6–9

Подзадачи 6–7 были нужны, чтобы стимулировать участников на написание полного решения. В этих задачах ясно, что $k \leqslant 10$ и есть мотивация разбивать монетки на группы. При ограничениях этой задачи ясно, что $k \leqslant 72$ (в этом можно убедиться, если непосредственно вычислить количество делителей у всех чисел от 1 до 15 000).

Рассмотрим произвольный набор монет с суммой x. Некоторые монеты одинакового номинала можно сложить в «*пачки*» стоимостью d (если рассматривается i-й номинал, то в пачке будет содержаться $\frac{d}{a_i}$ монет). Таким образом, каждый набор монет однозначно разбивается на две части:

- Монеты, которые образуют пачки. Суммарная стоимость таких монет обязательно делится на d;
- Монеты, которые не попали в пачки. То есть монета a_i взята не более чем $\frac{d}{a_i} - 1$ раз.

Рассмотрим подробнее монеты, которые не попали в пачки. Пусть $\mathrm{dp}[y]$ — количество способов сумму y так, чтобы никакие использованные монеты не могли образовать пачку. Это количество легко вычислить с помощью динамического программирования. Пусть $\mathrm{dp}[l][y]$ — количество способов набрать сумму y используя монеты $a_1, \ldots, a_l$ так, чтобы никакой набор монет одинакового номинала не образовывал пачку.

Переход имеет следующий вид:

$$\mathrm{dp}[l][y] = \mathrm{dp}[l-1][y] + \mathrm{dp}[l-1][y-a_l] + \ldots + \mathrm{dp}[l-1][y-d+a_l]$$

Но вычислять динамику именно по такой формуле долго, поэтому можно сократить вычисления используя следующий трюк:

Понятно, что такую динамику имеет смысл считать только для $0 \leqslant y < k \cdot d$. Таким образом требуемое значение можно вычислить за время $O(k^2 \cdot d)$.

Теперь для ответа на запрос x мы можем перебрать суммарную стоимость монет, которые не образуют пачку x_0 ($x - x_0$ делится на d и $x_0 < k \cdot d$, т.е. есть $\leqslant k$ вариантов значения x_0). Положим $n = \frac{x-x_0}{d}$ — это количество пачек, которые надо набрать, чтобы получить сумму x.

По сути надо вычислить количество массивов $\{y_1 + \ldots + y_k = n\}$, состоящих из целых неотрицательных чисел. Это так, потому что можно считать, что y_i — это количество пачек из монет номиналом a_i. Найти это количество легко с помощью метода шаров и перегородок, оно равно:

$$\frac{(n+1)\cdot(n+2)\cdot\ldots\cdot(n+k-1)}{(k-1)!}$$

Числитель можно вычислить наивно за время $O(k)$, проблемой остается только то, как поделить на $(k-1)!$ по модулю 10^9+7. Так как 10^9+7 — простое число, то обязательно найдется такое число f, что $f \cdot 72! \equiv 1 \pmod{10^9+7}$, такое число f можно найти перебором и вписать это значение как предпосчитанное в код программы.

Теперь просто заметим, что:

$$(k-1)! \cdot (f \cdot k \cdot (k+1) \cdot \ldots \cdot 72) \equiv 1 \pmod{10^9+7}$$

Поэтому просто вычислим $g = f \cdot k(k+1)(k+2)\ldots 72$ и вместо деления на $(k-1)!$ будем домножать выражение на g. Таким образом ответ на задачу равен:

$$\sum_{x_0} \mathrm{dp}[k][x_0] \cdot g \cdot (n+1)(n+2)\ldots(n+k-1), \quad n = \frac{x - x_0}{d}$$

Полученное решение имеет время работы $O((d+q)\cdot k^2)$ и укладывается в ограничения задачи.