Задача A. Грузовики

Данная задача решается с применением жадного алгоритма. Перед его применением массив стоимостей грузовиков надо отсортировать по возрастанию. Т.к. размер массива достаточно большой, то стоит применять алгоритм сортировки с асимптотикой $O(n \cdot \log n)$. Далее нужно найти максимальное k, такое что $\sum_{i=0}^{k} a_i \leqslant M$. Число k и будет ответом задачи.

Задача B. 2025

Существует несколько подходов к решению данной задачи. Приведем два альтернативных решения.

Первый вариант решения

Для решения задачи при $k = 1$ можно число N постепенно увеличивать на 1. После каждого увеличения проверять вхождение цифр 2, 0, 2, 5 (в этом же порядке) в новое число N. К сожалению, уже при $k = 2$ многие тесты не пройдут по времени.

Назовем k повторов чисел 2025 «искомый ряд». Во втором алгоритме будем считать количество t цифр из искомого ряда в записи числа N.

Рассмотрим, например, искомый ряд 2025. Пусть количество цифр в числе N больше четырех (когда число цифр меньше или равно 4 — решение очевидное). Предположим, $t = 0$. В этом случае проверяем число из последних 4-х цифр числа N. Если оно меньше 2025, то 4 последние цифры числа N заменяем на цифры 2, 0, 2, 5. Если число из последних 4-х цифр числа N больше 2025, то к числу N прибавляем 10000 и 4 последние цифры заменяем на 2025. Перейдем к случаю, когда $t > 0$. Рассматриваем число, составленное из $4 - t$ последних цифр числа N и сравниваем его с числом, составленным из $4 - t$ последних цифр числа 2025. Если первое число меньше, то приписываем к нему справа второе число. В противном случае находим число $N1 = N+10\cdot(4-t)$ и рассматриваем количество цифр ряда 2025 в целой части числа $N1/(10\cdot(4-t))$. Пусть это будет число p. Сравниваем числа t и p. Если $p = t$, то ответом будет число $N1$, в котором $4 - t$ последних цифр заменены на $4 - t$ последних цифр числа 2025. Если $p = t + 1$, то ответом будет число $N1$, в котором $4 - t$ последних цифр числа $N1$ заменены соответственно на ‘0’ и $4 - (t + 1)$ последних цифр числа 2025. Если $p = t + 2$, то ответом будет число $N1$, в котором $4 - t$ последних цифр числа $N1$ заменены на ‘00’ и $4 - (t - 2)$ последних цифр числа 2025. Если $p = t - 1$, применяем рекурсию. Ряды вида 20252025, 202520252025 и т.д. рассматриваются аналогично.

Второй вариант решения

Сначала запишем число 2025 k раз подряд. Если это число уже больше числа N, то ответ найден. Иначе, если это не так, то искомое число M должно иметь длину такую же, как и число N или быть на 1 цифру длиннее.

Рассмотрим случай, когда длина искомого числа M равна длине числа N. Понятно, что какой-то префикс числа M должен совпадать с префиксом числа N. Давайте его переберём. Пусть все цифры N_j будут равны M_j для всех $j < i$, где i мы перебираем слева направо. Чтобы M было строго больше N, N_i должно быть больше M_i. Мы можем ещё раз перебрать все цифры, большие N_i за $O(10) = O(1)$. Теперь мы зафиксировали какой-то префикс числа M. Давайте посчитаем, сколько цифр ещё нужно для того, чтобы удовлетворить условие про k чисел 2025. Это делается очень просто — заводим счётчик, и проходимся по префиксу слева направо, увеличивая счётчик каждый раз, когда мы нашли нужную цифру. Например, если префикс равен 25032592, то счётчик будет равен 5, так как этот префикс можно представить как число 20252 длины 5. Теперь мы знаем сколько цифр мы уже использовали, а значит можно посчитать, сколько цифр нам осталось добрать. Пусть в этом примере нам надо 3 раза по 2025. Тогда нужно добрать $3 * 4 - 5 = 7$ цифр. Понятно, что эти недостающие цифры должны быть в самом конце числа M. Все остальные незаполненные цифры должны быть равны нулю. Таким алгоритмом мы получим несколько кандидатов на ответ, среди которых надо вывести наименьшее число. Однако если на суффиксе недостаточно места для всех недостающих цифр, то это значит, что текущий префикс не даст нам кандидата на ответ, и его можно пропустить.

Если после такого алгоритма мы не получили ни одного кандидата на ответ, то это означает, что длина M обязана быть на 1 больше длины N. Тогда в этом случае ответом будет число 100...0020252025...2025.

Описанный алгоритм работает за $O(|N|^2)$, но если поддерживать счётчик чисел 2025 при переборе префикса, и заметить, что нам не надо сравнивать всех кандидатов на ответ, а только вывести число с наибольшим общим префиксом, то мы получим асимптотику $O(|N|)$.

Задача C. Очередная задача о Робин Гуде

В первой подзадаче достаточно заметить, что Шерифу выгоднее взять на хранение деньги наиболее прибыльных особняков. Соответственно, требуется отсортировать массив a в порядке неубывания, после чего мы получаем новый массив, который обозначим через b. В таком случае ответом на подзадачу будет являться $b_1 + b_2 + \cdots + b_{n-k}$.

Во второй подзадаче Робин Гуд может либо поджечь единственный полицейский участок, либо не делать этого. В случае, если он его подожжёт, мы получаем первую подзадачу. Если Робин Гуд не будет сжигать полицейский участок, то обозначим через b_1 и b_2 две отсортированные в порядке неубывания непрерывные подпоследовательности изначального массива a, не содержащие -1, а через $b_{i,j}$ — j-й элемент массива b_i. Соответственно, ответом является:

$$\min_{\max(0,k-|b_2|)\leqslant i\leqslant \min(k,|b_1|)} \max(b_{1,1} + b_{1,2} + \cdots + b_{1,|b_1|-i}, b_{2,1} + b_{2,2} + \cdots + b_{2,|b_2|-(k-i)}).$$

Значение данного выражения можно найти с помощью жадного алгоритма.

Третья подзадача является прямым продолжением второй. Иными словами, теперь у нас массивов не 2, а m. Соответственно, пусть c_i — отсортированный в порядке неубывания массив, состоящий из элементов массивов b_i и b_{i+1}. Допустим, что Робин Гуд поджёг полицейский участок i. Ответом является:

$$\max_{0\leqslant i<m} \min_{\substack{0\leqslant j_l\leqslant |b_l|,\\ j_1+j_2+\cdots+j_m=k}} \max(b_{1,1} + b_{1,2} + \cdots + b_{1,|b_1|-j_1}, b_{2,1} + b_{2,2} + \cdots + b_{2,|b_2|-j_2}, \cdots,$$

$$b_{i-1,1} + b_{i-1,2} + \cdots + b_{i-1,|b_{i-1}|-j_{i-1}}, c_{i,1} + c_{i,2} + \cdots + c_{i,|c_i|-j_i-j_{i+1}}, b_{i+2,1} + b_{i+2,2} + \cdots + b_{i+2,|b_{i+2}|-j_{i+2}}, \cdots,$$

$$b_{m,1} + b_{m,2} + \cdots + b_{m,|b_m|-j_m}).$$

Итоговое значение находится с помощью перебора поджигаемого участка и вышеописанного жадного алгоритма. В случае, если Робин Гуд не поджигал полицейский участок, данную формулу легко адаптировать под такой случай, заменив часть формулы с c_i на части с b_i и b_{i+1}.

Четвёртая и пятая подзадачи требуют иного подхода. Заметим, что Робин Гуд может награбить x или меньше денег, если он может награбить $x-1$ или меньше денег. Также, если Робин Гуд не может награбить x или больше денег, то он не сможет награбить $x+1$ или больше денег. Соответственно, мы можем с помощью бинарного поиска по ответу найти искомое значение. Утверждаем, что ответ лежит внутри полуинтервала $[0, 10^9+1)$ (0 денег он сможет награбить, тогда как 10^9+1 денег точно не сможет). Бинарный поиск дает интервал вида $[x, x+1)$, где ответом является x.

Теперь надо научиться обрабатывать данные в ходе бинарного поиска. Допустим, что сейчас мы перебираем значение x. Сначала нужно вычислить для каждого b_i количество особняков j_i, у которых нужно взять деньги на хранение, чтобы для всех i выполнялось $b_{i,1}+b_{i,2}+\cdots+b_{i,|b_i|-j_i} < x$. Обозначим данное значение через c_i. Если $c_i > k$, то x денег Робин Гуд может награбить. В противном случае ему придётся попробовать поджечь какой-нибудь полицейский участок. Заметим, что после поджога i-о участка изменяются только c_i и c_{i+1}. Для них нужно найти новое количество особняков, у которых следует взять деньги на хранение, после чего снова свериться с k. Если после поджога любого полицейского участка количество особняков, у которых нужно взять деньги на хранение, не превышает k, то Робин Гуд не сможет награбить x или больше денег. Эти подзадачи различаются лишь реализациями. В четвёртой подзадаче можно хранить лишь количество элементов в каждой непрерывной подпоследовательности, тогда как в пятой подзадаче следует хранить исходные подпоследовательности в отсортированном виде.

Задача D. Дежавю

Подзадачи 1 и 2

Если $R - L \leqslant 1$, то необходимо просто вывести либо A_L, либо $A_L^{A_R}$. При наивном возведении в степень получаем асимптотику $O(Q \cdot \max A_i)$, что укладывается в ограничения при $A_i \leqslant 100$. Для больших значений A_i необходимо использовать метод бинарного (быстрого) возведения в степень. Временная сложность составляет $O(Q \cdot \log(\max A_i))$.

Подзадачи 3 и 4

Используя свойство

$$a \equiv b \pmod{m} \Rightarrow a^k \equiv b^k \pmod{m}$$

можно заметить, что достаточно итеративно возводить в степень остатки от предыдущих вычислений. Это приводит к асимптотике $O(Q \cdot N \cdot \max A_i)$ при наивном возведении в степень и $O(Q \cdot N \cdot \log(\max A_i))$ при использовании бинарного возведения в степень.

Подзадача 5

Заметим, что «левостороннюю» башню можно переписать в виде

$$A_L^{A_{L+1} \cdot A_{L+2} \cdot \ldots \cdot A_R}.$$

Если M — простое число, то по малой теореме Ферма для A_L, взаимно простого с M, верно:

$$A_L^n \equiv A_L^{n \pmod{M-1}} \pmod{M}.$$

Если A_L делится на M, то ответ равен 0. Иначе вычисляем показатель степени по модулю $M - 1$ и возводим в степень с помощью бинарного алгоритма. Чтобы эффективно находить произведение на отрезке $[L + 1, R]$, можно использовать дерево отрезков. Итоговая асимптотика — $O(N \log N + Q(\log N + \log(\max A_i)))$.

Подзадача 6

Пусть $M = p^k$. Тогда:

- Если A_L делится на p, то A_L^{32} уже делится на M, поскольку $k \leqslant \log_2(\max A_i) \leqslant \log_2(10^9) < 32$. Следовательно, если произведение чисел на отрезке $[L + 1, R]$ не менее 32, то можно сразу вывести 0. В противном случае можно честно посчитать степень.
- Если A_L не делится на p, то A_L и M взаимно просты, и можно использовать рассуждения из подзадачи 5 с применением теоремы Эйлера вместо малой теоремы Ферма.

Подзадача 7

Лемма. Если $A \equiv B \pmod{M_i}$ для всех i при попарно взаимно простых $M_1, M_2, \ldots, M_k$, то

$$A \equiv B \pmod{M_1 \cdot M_2 \cdot \ldots \cdot M_k}.$$

Доказательство. Из $A \equiv B \pmod{M_i}$ следует, что $A - B$ делится на каждый M_i. Так как модули попарно взаимно просты, то $A - B$ делится и на их произведение. ■

Пусть $M = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot \ldots \cdot p_k^{\alpha_k}$ — разложение M на простые множители. Для каждого простого делителя p_i рассмотрим два случая::

- A_L не делится на p_i. Тогда по малой теореме Ферма

$$A_L^n \equiv A_L^{n \pmod{p_i-1}} \pmod{p_i}.$$

Поскольку функция Эйлера $\varphi(M)$ делится на $p_i - 1$, то верно утверждение

$$A_L^n \equiv A_L^{C\cdot\varphi(M)+(n \pmod{\varphi(M)})} \mod p_i,$$

где C — любое целое число.

- A_L делится на p_i. Так как $\varphi(M) \geqslant \alpha_i$ для любых натуральных чисел, то

$$A_L^n \equiv A_L^{\varphi(M)+r} \equiv 0 \pmod{p_i}$$

при $n \geqslant \varphi(M)$ и $r \geqslant 0$. При $n < \varphi(M)$ можно честно посчитать степень.

Таким образом, если произведение чисел на отрезке $[L+1, R]$ превышает $\varphi(M)$, то:

$$A_L^n \equiv A_L^{\varphi(M)+(n \pmod{\varphi(M)})} \pmod{M}.$$

Задача E. Изменчивые слова

Для решения первых двух подзадач достаточно в явном виде добавлять префиксы к строкам для запросов второго вида. Для запросов первого вида следует в цикле перебрать все строки на отрезке от l до r и посчитать, сколько из них имеют нужный префикс.

Для решения третьей подзадачи можно использовать несколько подходов. Один из них — это построение бора, в каждой вершине которого нужно хранить отсортированный массив индексов строк в массиве, которые имеют префикс, соответствующий вершине бора. На запросы тогда можно отвечать, найдя вершину бора, соответствующую строке запроса. С помощью бинарного поиска можно найти первый элемент $\geqslant l$ и первый элемент $> r$, тогда разность индексов этих элементов и будет количеством строк с нужным префиксом. Итоговая сложность алгоритма — $O(n+L_s+q+L_v)$.

Для полного решения задачи перевернем все строки в массиве и запросах. Тогда запросы первого типа сведутся к нахождению количества строк с заданным суффиксом. А запросы второго типа будут просто добавлять в конец строки суффикс.

Воспользуемся полиномиальными хэшами и посчитаем префиксные хэши для каждой строки массива. При добавлении нового суффикса в строку мы можем за $O(|v|)$ посчитать хэши для новых префиксов. Теперь мы можем в любой момент времени посчитать за $O(1)$ хэш суффикса любой длины.

Далее заметим, что различных длин v в запросах первого типа не больше $O(\sqrt{L_v})$. Действительно, предположив обратное, получим, что $L_v > 1 + 2 + \ldots + 2\sqrt{L_v} = \frac{2\sqrt{L_v}(2\sqrt{L_v}+1)}{2} > 2L_v > L_v$ — противоречие.

Тогда можно завести $O(\sqrt{L_v})$ словарей, задающих соответствие между хэшами суффиксов одной длины, строк в массиве и упорядоченным множеством (например, декартовым деревом по явному ключу) индексов строк, имеющих такой суффикс. В таком случае, ответ на запрос первого типа сводится к нахождению хэша строки и нахождению разности порядков l и $r+1$ (как в подзадаче 3) в словаре, соответствующему его длине. Для запросов второго типа нам нужно убрать индекс из упорядоченных множеств, соответствующих старым суффиксам, и добавить индекс в упорядоченные множества, соответствующие новым суффиксам.

Итоговая сложность алгоритма — $O(L_s + n\sqrt{L_v} + q\sqrt{L_v}\log n)$.