Условия задач, тесты, решения и разбор задач подготовили Даниил Шиндов, Михаил Кондрашин, Михаил Первеев и Никита Голиков.

Ценные замечания по результатам тестирования задач сделали Адам Хромов, Александр Попов, Антон Витюк, Артем Шулятьев, Захар Яковлев и Юрий Скворцов.

Задача 1. Очередная задача про математику

Автор задачи: Сергей Яковлев, разработчик: Михаил Первеев

Для получения частичного балла было достаточно перебрать, сколько операций следует применить к первому числу, а сколько — ко второму числу. Далее будет рассмотрено полное решение.

Для начала заметим, что оптимальной стратегией является использовать ровно k операций. Не умаляя общности будем считать, что $a \leqslant b$, и рассмотрим два случая.

Для начала рассмотрим случай, когда $b - a \geqslant k$. Заметим, что в этом случае выгодно применить все k операций к числу a. Для того, чтобы доказать это, предположим, что мы применили x операций к числу a и $k - x$ операций к числу b. Тогда итоговое произведение равно $(a + x) \cdot (b + k - x) = a \cdot (b + k) + x \cdot (b - a + k - x)$. Так как число $b - a + k - x$ является неотрицательным, выгодно сделать параметр x как можно больше, а его максимальное значение равно k.

Таким образом, в данном случае ответом являются числа $a + k$ и b.

Теперь рассмотрим случай, когда $b - a < k$. Заметим, что в этом случае в оптимальном решении полученные числа c и d отличаются не более, чем на единицу, то есть $0 \leqslant d - c \leqslant 1$. Сразу заметим, что данная ситуация всегда достижима, так как можно сначала $b - a$ раз применить операцию к числу a, сравняв числа, а затем по очереди применять операцию к числам b и a. Для того, чтобы доказать оптимальность такого решения, предположим, что это не так, и в оптимальном ответе $d - c > 1$. Тогда их произведение равно $c \cdot d$. Теперь заметим, что можно увеличить c на единицу и уменьшить d на единицу. В этом случае количество операций не изменится, а разность между числами уменьшится на 2. В то же время произведение будет равно $(c+1) \cdot (d-1) = c \cdot d + (d-c) - 1$. Так как $d - c > 1$, то данное произведение больше, чем $c \cdot d$, а значит ответ улучшился, то есть предположение было не верно.

Таким образом, в данном случае ответом являются числа $b + \lfloor \frac{k-(b-a)}{2} \rfloor$ и $b + \lceil \frac{k-(b-a)}{2} \rceil$.

Асимптотика: $\mathcal{O}(1)$.

Задача 2. Очередная задача про хорошие строки

Автор задачи: Никита Голиков, разработчик: Михаил Кондрашин

Подзадача 1

Для решения данной подзадачи можно обрабатывать элементы массива по очереди и запоминать для каждого числа, сколько раз оно было встречено в массиве. Для этого можно использовать массив cnt, в котором cnt_i равно количеству встреченных вхождений числа i. При рассмотрении очередного элемента массива (пусть этот элемент равен x), к ответу необходимо прибавить $\sum_{i=0}^{999} cnt_i$ для всех таких i, что строка $i + x$ является хорошей, после чего увеличить cnt_x на единицу.

Асимптотика: $\mathcal{O}(n \cdot C)$, где C — максимальное число в массиве.

Подзадача 2

Для решения данной подзадачи можно перебрать каждую пару строк (i, j), после чего проверить, что строка $s_i + s_j$ является хорошей.

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

Подзадача 3

Для решения данной подзадачи заметим, что если строка s_i не является хорошей, то конкатенация s_i с любой другой строкой s_j также не будет являться хорошей строкой. Поэтому достаточно оставить в массиве только хорошие строки.

После этого снова переберем все пары строк (i, j). Строка $s_i + s_j$ является хорошей, если последний символ строки s_i не больше первого символа строки s_j (так как они являются хорошими).

Асимптотика: $\mathcal{O}(n^2 + m)$.

Подзадача 4

Оптимизируем решение третьей подзадачи. Для этого научимся быстро находить для каждой строки s_j количество строк s_i $(i < j)$, таких что $s_i + s_j$ является хорошей строкой. Данное количество равно количеству хороших строк, встреченных раньше, чем строка s_j, у которых последний символ не больше, чем первый символ строки s_i.

Таким образом необходимо прибавить к ответу количество строк, которые заканчиваются на произвольную цифру в диапазоне от первой цифры строки s_j до 9. Для этого будем поддерживать массив, в котором в элементе cnt_i будем хранить количество встреченных хороших строк, которые оканчиваются на цифру i.

Асимптотика: $\mathcal{O}(n + m)$.

Задача 3. Очередная задача про победу над монстрами

Автор задачи: Михаил Кондрашин, разработчик: Даниил Шиндов

Подзадача 1

В этой подзадаче достаточно перебрать все перестановки из $n - 1$ надземелий без учета последнего. Каждая такая перестановка будет описывать порядок посещений надземелий. Для каждой перестановки мы будем перебирать ее элементы, пытаясь победить даркона в соответствующем надземелье. Будем выполнять данный процесс до тех пор, пока не дойдем до надземелья, в котором победить даркона не получится, или пока не получим достаточный уровень силы, чтобы победить последнего даркона.

Асимптотика: $\mathcal{O}(n!)$.

Подзадача 2

Рассмотрим оптимальную стратегию. Она представляет собой некоторое множество надземелий, содержащее последнее надземелье, в котором задан определенный порядок посещения надземелий. Если в этой стратегии упорядочить надземелья по возрастанию уровня силы соответствующих даркон, то такая стратегия все еще будет оптимальной.

Тогда достаточно перебрать все подмножества надземелий, содержащих последнее, отсортируем каждое по уровню силы даркон и выберем из них минимальную по размеру рабочую стратегию.

Асимптотика: $\mathcal{O}(2^n \cdot n \log n)$ или $\mathcal{O}(n \log n + 2^n \cdot n)$.

Подзадача 3

Заметим, что после каждой победы над даркоином уровень силы персонажа увеличивается хотя бы на 1. Так как уровень силы последнего даркона, как и всех остальных, в данной подзадаче не превосходит 200, то нам достаточно победить не более 200 даркон, прежде чем идти в последнее надземелье.

Будем каждый раз выбирать не посещенное ранее надземелье с максимальным уровнем силы даркона, которое мы можем зачистить.

Асимптотика: $\mathcal{O}(n)$ с константой порядка 200.

Подзадача 4

Для полного решения задачи необходимо научиться быстро находить даркона с максимальным уровнем силы, которого можно победить. Для этого отсортируем надземелья по неубыванию уровня силы, после чего будем поддерживать указатель на последнего добавленного даркона в этом массиве, а также упорядоченное множество, в которое мы будем добавлять дарконов, которых мы можем победить.

Сначала добавим в множество все надземелья, дарконы в которых имеют уровень силы меньше нашего. После этого доступный для победы даркон с максимальной силой должен находиться на вершине этой очереди. Добавим его в ответ, увеличим уровень силы персонажа, после чего удалим побежденного даркона из множества. Затем, так как уровень силы персонажа увеличился и мы можем победить больше дарконов, будем двигать указатель в массиве и добавлять дарконов в множество. Будем повторять этот процесс, пока не наберем достаточный уровень силы персонажа.

Асимптотика: $\mathcal{O}(n \log n)$.

Задача 4. 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: $\mathcal{O}(n)$.

Задача 5. Очередная задача про кузнечика

Автор задачи и разработчик: Никита Голиков

Подзадача 1

В этой подзадаче можно перебрать все отрезки, и разобрать все возможные пути кузнечика.

Асимптотика: $\mathcal{O}(1)$, если разобрать все случаи вручную.

Подзадача 2

В этой подзадаче для каждого отрезка можно перебрать все возможные пути (как подмножества индексов), и для каждого обновить ответ.

Асимптотика: $O(2^n \cdot n^3)$.

Подзадача 3

В этой подзадаче можно перебрать все отрезки, и для каждого найти ответ за $\mathcal{O}(n^2)$, применив стандартное динамическое программирование: $dp[i]$ означает максимум из минимальных высот на пути $l \to i$, для перехода нужно перебрать возможную длину прыжка.

Асимптотика: $\mathcal{O}(n^4)$.

Подзадача 4

Для этой подзадачи нужно заметить, что при фиксированной левой границе отрезка динамика из прошлой подзадачи находит ответы для всех правых границ.

Асимптотика: $\mathcal{O}(n^3)$.

Подзадача 5

В этой подзадаче было ограничение $k = 1$, что означает, что для отрезка $[l, r]$ ответ равен минимальному числу на отрезке $[l, r]$. Поэтому для решения подзадачи можно перебрать все левые границы, и далее циклом по правой границе поддерживать текущий минимум, и суммировать ответы.

Асимптотика: $\mathcal{O}(n^2)$.

Подзадача 7

В этой подзадаче было ограничение $k = 1$, но n могло быть большим. По замечанию из прошлой подзадачи, нам нужно просуммировать минимумы на всех подотрезках массива. Это можно сделать следующим образом: переберем позицию i минимума на отрезке, и посчитаем количество отрезков, на которых элемент a_i является самым левым минимумом. Обозначим за l_i позицию предыдущего элемента, большего a_i (или 0, если такого элемента не существует), а за r_i обозначим позицию следующего элемента, не меньшего a_i (или $n+1$, если такого элемента не существует). Тогда отрезки, на которых элемент i является самым левым минимумом, устроены так: их левая граница находится на отрезке $[l_i + 1, i]$, а правая граница на отрезке $[i, r_i - 1]$. Таким образом, если мы нашли все значения l_i и r_i, нужно просуммировать $a_i \cdot (i - l_i) \cdot (r_i - i)$. Найти нужные значения l_i и r_i можно линейным проходом со стеком по массиву (для l_i удаляем все элементы со стека, не большие a_i, для r_i аналогично).

Асимптотика: $\mathcal{O}(n)$.

Общая идея для подзадач 6 и 8

Поймем, как устроен оптимальный маршрут кузнечика из l в r. Если $r - l \leqslant k$, то кузнечик за один ход прыгает в правый конец. Иначе, рассмотрим отрезок $[l+1; l+k]$. Пусть p — позиция максимума на этом отрезке. Тогда заметим, что ответ не больше, чем a_p, так как мы должны посетить хотя бы один столбик на этом отрезке. Тогда оптимально совершить прыжок из l в p, и продолжить путь до столбика r.

Подзадача 6

Воспользовавшись этой идеей, можно написать решение с динамикой для этой подзадачи. Давайте сначала за $O(nk)$ наивно предподсчитаем позиции максимумов на всех подотрезках длины k, обозначим за p_i позицию максимума на отрезке $[i; i+k-1]$. Тогда давайте посчитаем $dp[l][r]$ — оптимальный ответ для пути из l в r. Если $r - l \leqslant k$, то $dp[l][r] = \min(a_l, a_r)$, иначе $dp[l][r] = \min(a_l, dp[p_{i+1}][r])$. Для реализации данного решения можно считать значения динамики по возрастанию длины отрезка $[l, r]$. В конце просуммируем все посчитанные значения динамики.

Асимптотика: $\mathcal{O}(n^2)$.

Подзадача 8

Давайте для начала посчитаем все значения максимумов на отрезках длины k. Для этого можно использовать либо очередь с максимумом, либо «std::multiset». Обозначим за b_i значение максимума на отрезке $[i; i + k - 1]$, либо 10^9, если $i + k - 1 > n$.

Тогда заметим, что ответ для отрезка $[l, r]$ равняется $\min(\min\limits_{i=l}^{r} b_i, a_l, a_r)$. Действительно, ответ не больше a_l и не больше a_r, так как мы обязаны посетить эти столбики, и для любого $l \leqslant i \leqslant r$ ответ не больше b_i, так как либо этот отрезок полностью влезает в наш, тогда мы посещаем хотя бы одну клетку на нем, либо он вылезает за границу, но тогда значение максимума на нем не меньше значения в границе отрезка. С другой стороны, жадный алгоритм посещает только максимумы на отрезках длины k или края отрезка, поэтому ответ в точности такой.

Воспользовавшись этим наблюдением, посчитаем ответ следующим образом. Для каждого i создадим два события: событие «a_i становится активным» и событие «b_i становится активным». Теперь обработаем эти события в порядке убывания соответствующего значения.

После активации очередного значения какие-то отрезки $[l, r]$ становятся активными: в момент активации последнего из значений $a_l, a_r, b_i (l \leqslant i \leqslant r)$. В этот момент ответ для отрезка равен текущему значению события, и нужно обновить ответ количеством таких отрезков, умноженным на текущее значение.

Рассмотрим событие активации значения b_i (события с активацией a_i рассматриваются похожим образом). Давайте найдем ближайшее неактивное значение b_L слева, и ближайшее неактивное значение b_R справа. Для этого можно поддерживать текущие неактивные позиции в «std::set». Теперь

нам подходят отрезки, которые начинаются в $[L+1; i]$ и заканчиваются в $[i; R-1]$, для обеих границ которых уже активны значения массива a. Чтобы найти количество подходящих значений a на отрезках, можно поддерживать дерево Фенвика на сумму, или же поддерживать дерево поиска с возможностью нахождения количества элементов, меньших данного. Например, подойдет «tree» из «gnu pbds», данная структура есть в компиляторе «g++».

Асимптотика: $\mathcal{O}(n \log n)$.