Задача A. Subnautica

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: консольный вывод

Выведите строку согласно условию.

Задача B. Factorio

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: арифметика, целочисленное деление

Подзадача 1. При $xy = 1$ панели вырабатывают одну единицу энергии в час, а суммарная ёмкость аккумуляторов равна nm. Следовательно, ответом будет nm.

Заметим, что значение nm может достигать 10^{18}, поэтому для вычислений необходимо использовать 64-битный тип данных.

Подзадача 2. Поскольку ответ не превосходит 10^5, смоделируем процесс. Будем поддерживать количество накопленной энергии, изначально равное нулю. Пока оно меньше nm, увеличим его на xy, а ответ — на единицу.

Подзадача 3. Поскольку суммарная ёмкость аккумуляторов nm делится на суммарную выработку панелей xy нацело, ответом будет $\frac{nm}{xy}$.

Полное решение. За один час все панели вырабатывают xy единиц энергии, а для полной зарядки всех аккумуляторов требуется nm единиц. Следовательно, ответом будет минимальное целое t, для которого $t \cdot xy \geqslant nm$, то есть

$$t = \left\lceil \frac{nm}{xy} \right\rceil = \left\lfloor \frac{nm + xy - 1}{xy} \right\rfloor.$$

Асимптотика решения: $O(1)$.

Задача C. Barotrauma

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: линейные массивы, жадные алгоритмы

Подзадача 1. При $n = 1$ выравнивать нечего, поэтому ответом будет 0.

При $n = 2$ понизим больший уровень до меньшего. Если $|a_1 - a_2| > k$, люк разрушится уже в начальный момент, и ответом будет -1. Иначе в процессе разница уровней только уменьшается, и ответом будет $|a_1 - a_2|$.

Подзадача 2. Поскольку уровень воды можно только понижать, итоговый уровень не превосходит $\min(a_1, a_2, \ldots, a_n)$. Чем выше итоговый уровень, тем меньше операций потребуется, следовательно, выгодно выровнять все отсеки до уровня $\min(a_1, a_2, \ldots, a_n)$.

В любой момент уровни воды находятся в пределах от 0 до $\max(a_1, a_2, \ldots, a_n) \leqslant k$, поэтому разница уровней в соседних отсеках никогда не превысит k. Следовательно, операции можно выполнять в любом порядке, и ответом будет

$$\sum_{i=1}^{n} a_i - n \cdot \min(a_1, a_2, \ldots, a_n).$$

Асимптотика решения: $O(n)$.

Подзадача 3. Все уровни уже равны, поэтому ответом будет 0.

Подзадача 4. Как и в предыдущей подзадаче, итоговый уровень выгодно сделать равным минимальному элементу массива. Будем моделировать процесс: пока не все уровни равны, найдем текущий максимальный уровень h и по очереди понизим на единицу уровень в каждом отсеке, где он равен h. Перед началом и после каждой операции будем проверять разницу уровней с соседними отсеками. Если какой-то люк разрушается, ответом будет -1.

Асимптотика решения: $O(n \cdot \max a_i)$.

Полное решение. Итоговый уровень выгодно сделать равным $\min(a_1, a_2, \ldots, a_n)$, тогда количество операций равно

$$\sum_{i=1}^{n} a_i - n \cdot \min(a_1, a_2, \ldots, a_n).$$

Осталось понять, когда этого можно добиться, не разрушив ни одного люка. Если в начальный момент найдется i, для которого $|a_i - a_{i+1}| > k$, люк разрушится сразу, и ответом будет -1.

Иначе покажем, что стратегия из предыдущей подзадачи никогда не разрушает люки. Если $k = 0$, то все уровни изначально равны, и операции не нужны. Пусть $k \geqslant 1$. Рассмотрим операцию над отсеком i с максимальным уровнем и его соседом j, тогда $a_i \geqslant a_j$. Если $a_i > a_j$, то, поскольку до операции $a_i - a_j \leqslant k$, после нее разница станет равна $a_i - 1 - a_j$, где $0 \leqslant a_i - 1 - a_j < k$. Если $a_i = a_j$, разница станет равна $1 \leqslant k$. В обоих случаях люк выдержит.

Таким образом, достаточно проверить начальное состояние: если какая-то пара соседних отсеков нарушает ограничение, выведем -1, иначе выведем значение выражения выше.

Асимптотика решения: $O(n)$.

Задача D. Age of Empires

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: стек рекордов

Поскольку все высоты различны, башня j видна с башни i тогда и только тогда, когда $h_j > \max(h_{i+1}, \ldots, h_{j-1})$. Заметим, что высота самой башни i значения не имеет.

Подзадача 1. Поскольку высоты убывают, с башни $i < n$ видна только башня $i+1$, так как она выше всех башен, стоящих за ней. Ответом будет 1 для всех $i < n$ и 0 для $i = n$.

Подзадача 2. При $n = 1$ ответом будет 0. При $n = 2$ между башнями нет других башен, поэтому ответом будет `1 0`.

Подзадача 3. Переберем все пары (i, j), для каждой пары пройдем по всем башням между ними и проверим, есть ли среди них башня выше башни j.

Асимптотика решения: $O(n^3)$.

Подзадача 4. Зафиксируем i и будем перебирать j от $i+1$ до n, поддерживая максимум высот башен строго между i и j. Если h_j больше этого максимума, увеличим ответ для i на единицу, после чего обновим максимум значением h_j.

Асимптотика решения: $O(n^2)$.

Полное решение. Из замечания выше следует, что ответ для башни i равен количеству префиксных максимумов последовательности $h_{i+1}, h_{i+2}, \ldots, h_n$.

Будем перебирать i от n до 1 и поддерживать в стеке все префиксные максимумы последовательности $h_{i+1}, \ldots, h_n$, причем на вершине стека будет находиться ближайшая башня. Тогда высоты в стеке возрастают от вершины ко дну, а ответом для башни i будет текущий размер стека.

После рассмотрения башни i все башни из стека, которые ниже башни i, перестанут быть префиксными максимумами, поскольку башня i закрывает их. Так как такие башни находятся на вершине стека, извлечем из стека все элементы, меньшие h_i, после чего добавим в стек h_i.

Асимптотика решения: $O(n)$.

Задача E. Permutation

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: перестановки, скользящее окно

Подзадача 1. При $k = 1$ любая последовательность из одного индекса является хорошей, и для нее $i_k - i_1 = 0$. Ответом будет 0.

Подзадача 2. При $k = n$ единственной хорошей последовательностью являются все индексы от 1 до n. Ответом будет $n - 1$.

Подзадача 3. Переберем все подмножества индексов размера k. Поскольку все значения перестановки различны, выбранные значения являются последовательными числами тогда и только тогда, когда разность максимального и минимального из них равна $k - 1$. Среди всех подходящих подмножеств выберем минимальное значение $i_k - i_1$.

Асимптотика решения: $O(2^n \cdot n)$.

Подзадача 4. Обозначим через pos_x позицию числа x в перестановке. Поскольку каждое значение встречается ровно один раз, хорошая последовательность однозначно задается числом a: она состоит из индексов $pos_a, pos_{a+1}, \ldots, pos_{a+k-1}$. Тогда

$$i_k - i_1 = \max(pos_a, \ldots, pos_{a+k-1}) - \min(pos_a, \ldots, pos_{a+k-1}).$$

Переберем все a от 1 до $n - k + 1$ и для каждого найдем минимум и максимум за $O(k)$.

Асимптотика решения: $O(nk)$.

Полное решение. Требуется найти минимальную разность максимума и минимума среди всех отрезков длины k массива pos. Будем перебирать a от 1 до $n - k + 1$ и хранить в `set` позиции $pos_a, pos_{a+1}, \ldots, pos_{a+k-1}$. Минимальной и максимальной позициями будут первый и последний элементы множества.

Изначально добавим в множество позиции $pos_1, \ldots, pos_k$. При переходе от a к $a + 1$ удалим из множества pos_a и добавим pos_{a+k}.

Асимптотика решения: $O(n \log n)$.

Задача F. Stardew Valley

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: двоичный поиск по ответу, двумерные массивы, префиксные суммы

Обозначим $h = \left\lceil \frac{k^2}{2} \right\rceil$.

Подзадача 1. При $k = 1$ участок состоит из одной клетки, и медиана равна её твёрдости. Ответом будет минимальный элемент матрицы.

Подзадача 2. Переберём все положения участка, выпишем значения всех k^2 его клеток, отсортируем их и возьмём h-й элемент. Среди найденных медиан выберем минимальную.

Асимптотика решения: $O(n^4 \log n)$.

Подзадача 3. При $k = n$ существует единственное положение участка, совпадающее со всем полем. Отсортируем все n^2 значений и выведем h-е из них.

Асимптотика решения: $O(n^2 \log n)$.

Подзадача 4. Заметим, что медиана участка не превосходит T тогда и только тогда, когда на нём найдётся хотя бы h клеток с твёрдостью не больше T. Следовательно, если существует участок с медианой не больше T, то он подходит и для любого $T' > T$, поэтому ответ можно искать двоичным поиском.

Для фиксированного T построим матрицу b, где $b_{i,j} = 1$, если $a_{i,j} \leqslant T$, и $b_{i,j} = 0$ иначе. Предпросчитаем префиксные суммы в каждой строке матрицы b, тогда количество единиц на участке

можно найти за $O(k)$, просуммировав значения на k отрезках строк. Если хотя бы на одном участке количество единиц не меньше h, то ответ не превосходит T.

Асимптотика решения: $O(n^3 \log A)$, где $A = 10^9$.

Подзадача 5. Поскольку $a_{i,j} \in \{0, 1\}$, ответ равен 0, если существует участок, содержащий хотя бы h нулей, и 1 в противном случае. Построим двумерные префиксные суммы по количеству нулей, тогда количество нулей на каждом участке можно найти за $O(1)$ по формуле суммы на прямоугольнике.

Асимптотика решения: $O(n^2)$.

Полное решение. Объединим идеи двух предыдущих подзадач. Будем искать ответ двоичным поиском, а для проверки фиксированного T построим двумерные префиксные суммы матрицы b. Тогда количество единиц на любом участке вычисляется за $O(1)$, и проверка выполняется за $O(n^2)$.

Асимптотика решения: $O(n^2 \log A)$.

Задача G. Slay the Spire

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: динамическое программирование, жадные алгоритмы, очередь с поддержкой минимума

Заметим, что в любой момент выполняется неравенство $w \leqslant m$: изначально $w = m$, при розыгрыше карты w уменьшается, а m увеличивается, а после восстановления w не превосходит m по определению. Кроме того, если к текущему моменту разыграно t карт, то $m = W + tB$. Обозначим $S = c_1 + c_2 + \ldots + c_n$.

Подзадача 1. При $X = 0$ восстановление не изменяет запас маны, поскольку $\min(w, m) = w$. Следовательно, w только уменьшается, и суммарная стоимость разыгранных карт не превосходит W. Таким образом, требуется выбрать максимальное количество карт суммарной стоимостью не более W. Для этого выгодно брать самые дешевые карты: отсортируем этажи по возрастанию a_i и с каждого этажа возьмем $\min\left(c_i, \left\lfloor \frac{w}{a_i} \right\rfloor\right)$ карт, где w — оставшийся запас маны. Если $a_i = 0$, возьмем все c_i карт.

Асимптотика решения: $O(n \log n)$.

Подзадача 2. Поскольку на одном этаже все карты имеют одинаковую стоимость, важно лишь количество j_i карт, разыгранных на i-м этаже, где $0 \leqslant j_i \leqslant c_i$. Переберем все наборы $(j_1, j_2, \ldots, j_n)$ и для каждого из них смоделируем игру, проверяя, что каждую карту можно разыграть. Количество наборов равно $\prod_{i=1}^{n}(c_i + 1) \leqslant 2^S$.

Асимптотика решения: $O(2^S \cdot n)$.

Подзадача 3. Поскольку вместимость маны однозначно определяется количеством разыгранных карт, состояние героя после прохождения очередного этажа задается парой (t, w). Заметим, что при фиксированном t наибольший запас маны является лучшим состоянием.

Пусть $dp[i][t]$ — максимальный запас маны после прохождения i этажей и восстановления, если всего разыграно t карт, или $-\infty$, если такое состояние недостижимо. Изначально $dp[0][0] = W$.

Рассмотрим переход из состояния $dp[i-1][s] = w$, при котором на i-м этаже разыгрывается $j \leqslant c_i$ карт. Следовательно, переход возможен, если $w \geqslant j \cdot a_i$. После розыгрыша запас маны станет равен $w - j \cdot a_i$, а после восстановления $\min(w - j \cdot a_i + X, W + (s+j)B)$. Обновим этим значением $dp[i][s+j]$. После последнего этажа восстановление не выполняется.

Ответом будет максимальное t, для которого $dp[n][t] \neq -\infty$. Значения динамики могут достигать $W + SB$, поэтому необходимо использовать 64-битный тип данных.

Асимптотика решения: $O(S \cdot (S + n))$.

Подзадача 4. Если все карты имеют одинаковую стоимость a, то выгодно на каждом этаже разыграть максимально возможное количество карт, то есть $\min\left(c_i, \left\lfloor \frac{w}{a} \right\rfloor\right)$, или c_i, если $a = 0$.

Асимптотика решения: $O(n)$.

Полное решение. Возьмем идею из подзадачи 3. Зафиксируем этаж i, обозначим $a = a_i$, $c = c_i$ и $g_s = dp[i-1][s]$. Пусть f_t — максимальный запас маны после розыгрыша карт на i-м этаже до восстановления, если всего разыграно t карт. Тогда

$$f_t = \max_{t-c \leqslant s \leqslant t} (g_s - (t-s)a) = \max_{t-c \leqslant s \leqslant t} (g_s + sa) - ta,$$

Таким образом, требуется найти максимум значений $g_s + sa$ на всех отрезках $[t-c, t]$. Это можно сделать при помощи скользящего окна с поддержкой минимума за $O(S)$. После нахождения максимума обновим состояние динамики: $dp[i][t] = \min(f_t + X, W + tB)$.

Асимптотика решения: $O(nS)$.

Задача H. TREE(3)

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: деревья, комбинаторика, динамическое программирование

Подзадача 1. При $n \leqslant 8$ для каждого k переберём все способы расстановки рекурсивно, на каждом шаге выбирая любую подходящую вершину. Количество способов для одного корня не превосходит $(n-1)!$

Асимптотика решения: $O(n^2 \cdot n!)$.

Подзадача 2. Дерево является бамбуком, поэтому его можно рассматривать как линейный массив. Множество заполненных вершин всегда образует отрезок $[l, r]$, содержащий вершину k, и на каждом шаге очередное число можно записать только в вершину $l-1$ или в вершину $r+1$. Таким образом, способ расстановки однозначно задаётся последовательностью из $n-1$ шагов, на каждом из которых отрезок расширяется влево или вправо.

Слева от вершины k находится $k-1$ вершина, а справа — $n-k$ вершин, поэтому расширений влево ровно $k-1$. Выбрать, какие из $n-1$ шагов являются расширениями влево, можно C_{n-1}^{k-1} способами, что и будет ответом.

Предпросчитаем факториалы и обратные к ним по модулю $M = 10^9 + 7$, тогда каждое значение вычисляется за $O(1)$.

Асимптотика решения: $O(n + \log M)$.

Подзадача 3. Дерево является звёздочкой с центром в вершине 1. Если $k = 1$, остальные вершины можно заполнять в любом порядке, и ответом будет $(n-1)!$. Иначе число 2 обязательно записывается в центр, после чего оставшиеся $n-2$ вершины заполняются в любом порядке, и ответом будет $(n-2)!$.

Асимптотика решения: $O(n)$.

Подзадача 4. Раскраску дерева можно представить в виде последовательности, где i-я вершина имеет цвет i. Множество покрашенных вершин всегда связно и содержит корень, поэтому вершину можно покрасить только после её родителя. Следовательно, фактически нас интересуют последовательности вершин, в которых каждая вершина идёт позже своего родителя. Будем называть такие последовательности корректными.

Обозначим через s_v размер поддерева вершины v, а через f_v — количество корректных последовательностей покраски поддерева v. Рассмотрим значение f_v для произвольной вершины.

Вершина v всегда красится первой, и её место в последовательности фиксируется. Остальные $s_v - 1$ вершин поддерева без учёта ограничений можно поставить после неё $(s_v - 1)!$ способами.

Рассмотрим ребёнка u вершины v. Исключим варианты, в которых последовательности отличаются только перестановкой вершин поддерева u между собой: таких перестановок $s_u!$, поэтому разделим количество последовательностей на $s_u!$. После этого добавим к ответу корректные перестановки вершин поддерева u, умножив на f_u. Между вершинами из разных поддеревьев нет никаких ограничений, поэтому:

$$f_v = (s_v - 1)! \cdot \prod_{j=1}^{r} \frac{f_{u_j}}{s_{u_j}!}.$$

Более того, расписав эту формулу подробнее, можно заметить, что она принимает следующую форму: $f_v = \frac{(s_v-1)!}{\prod\limits_{w \neq v} s_w} = \frac{s_v!}{\prod\limits_{w} s_w}$, где произведение берётся по всем вершинам w поддерева v.

Таким образом, ответом для корня k будет

$$\frac{n!}{\prod\limits_{v=1}^{n} s_v}.$$

Асимптотика решения: $O(n^2 + n \log M)$.

Полное решение. Подвесим дерево за вершину 1, посчитаем размеры поддеревьев s_v и ответ для вершины 1 по формуле из предыдущей подзадачи.

Рассмотрим перенос корня из вершины u в её ребёнка v. Размеры поддеревьев изменятся только у самих вершин u и v: до переноса размер поддерева u равен n, а v — s_v, после переноса размер поддерева v станет равен n, а u — $n - s_v$. Следовательно,

$$ans_v = ans_u \cdot \frac{s_v}{n - s_v}.$$

Обойдём дерево от вершины 1 и вычислим ответы для всех вершин.

Асимптотика решения: $O(n \log p)$.

Задача I. Assassin's Creed

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: строки, бор, суффиксный массив, стек рекордов

Обозначим $N = n \cdot m$ и назовём ценностью подстроки произведение её длины на суммарное количество вхождений.

Подзадача 1. При $m = 1$ каждая запись состоит из одной буквы, поэтому подстроками являются только отдельные буквы, а ценность буквы равна количеству её вхождений. Посчитаем количество каждой буквы и выберем наиболее частую, а при равенстве — первую в алфавитном порядке.

Асимптотика решения: $O(n)$.

Подзадача 2. Переберем все подстроки всех записей и посчитаем количество вхождений каждой из них с помощью `map`. Поскольку `map` хранит строки в лексикографическом порядке, достаточно пройти по нему и выбрать первую подстроку с максимальной ценностью.

Асимптотика решения: $O(nm^3 \log N)$.

Подзадача 3. Добавим все суффиксы каждой записи в бор. Каждой вершине бора соответствует ровно одна различная подстрока. При добавлении суффикса увеличим на единицу счётчик каждой вершины на пути, тогда счётчик вершины будет равен количеству вхождений соответствующей подстроки. Количество вершин бора не превосходит $\frac{nm(m+1)}{2}$.

Обойдем бор в глубину, перебирая переходы в порядке `A`, `C`, `G`, `T`. При таком обходе подстроки посещаются в лексикографическом порядке, поэтому ответом будет первая найденная подстрока с максимальной ценностью.

Асимптотика решения: $O(nm^2)$.

Полное решение. Склеим все записи в одну строку через попарно различные разделители, которые не встречаются в записях, и построим для неё суффиксный массив sa и массив lcp, где

lcp_i — длина наибольшего общего префикса суффиксов sa_i и sa_{i+1}. Поскольку разделители различны, общий префикс двух суффиксов никогда не проходит через разделитель, то есть всегда лежит внутри одной записи.

Каждое вхождение подстроки t — это суффикс, начинающийся с t. Все такие суффиксы стоят в суффиксном массиве подряд и образуют отрезок $[l, r]$, на котором все значения $lcp_l, \ldots, lcp_{r-1}$ не меньше $|t|$. Следовательно, количество вхождений t равно $r - l + 1$.

Верно и обратное: суффиксы $sa_l, \ldots, sa_r$ имеют общий префикс длины $h = \min(lcp_l, \ldots, lcp_{r-1})$, который встречается не менее $r - l + 1$ раз. Чтобы ценность была максимальной, длину подстроки выгодно брать равной h, а отрезок расширять, пока минимум не уменьшается. Поэтому достаточно для каждого i рассмотреть максимальный отрезок, на котором все значения lcp не меньше lcp_i, и посчитать ценность $lcp_i \cdot (r - l + 1)$. Все такие отрезки можно найти с помощью стека рекордов.

Так учитываются все подстроки, встречающиеся хотя бы дважды. Если найденная максимальная ценность меньше m, ответом будет лексикографически минимальная строка. Среди всех кандидатов с максимальной ценностью выберем лексикографически минимального.

Асимптотика решения: $O(N \log N)$.

Задача J. Overflow

Идея: Андрей Коробко
Разбор: Андрей Коробко
Теги: динамическое программирование, Convex Hull Trick

Подзадача 1. При $n = 1$ переберем количество предметов k от 0 до $\left\lfloor \frac{m}{w_1} \right\rfloor$ и выберем максимальное значение $kv_1 - k^2$.

Асимптотика решения: $O(m)$.

Подзадача 2. Если $w_i = 1$ для всех i, то требуется выбрать не более m предметов. Добавление j-го предмета i-го типа увеличивает ценность на

$$\left(jv_i - j^2\right) - \left((j-1)v_i - (j-1)^2\right) = v_i - (2j-1).$$

Эти приращения убывают с ростом j, поэтому выгодно жадно брать предметы с максимальным приращением. Будем хранить в очереди с приоритетом приращение от следующего предмета каждого типа. Не более m раз извлечем максимальное приращение, если оно положительно, прибавим его к ответу и добавим в очередь приращение от следующего предмета того же типа.

Асимптотика решения: $O((n + m) \log n)$.

Подзадача 3. Воспользуемся динамическим программированием. Пусть $dp[i][c]$ — максимальная суммарная ценность при выборе предметов первых i типов суммарным весом не более c. Изначально $dp[0][c] = 0$. Переберем количество k предметов i-го типа:

$$dp[i][c] = \max_{0 \leqslant k \leqslant \lfloor c/w_i \rfloor} \left(dp[i-1][c - kw_i] + kv_i - k^2\right).$$

Ответом будет $dp[n][m]$.

Асимптотика решения: $O(nm^2)$.

Полное решение. Ускорим переход динамики из подзадачи 3. Зафиксируем тип i и обозначим $w = w_i, v = v_i$. При добавлении предметов i-го типа суммарный вес изменяется на величину, кратную w, поэтому переход из $dp[i-1][c']$ в $dp[i][c]$ возможен, только если $c - c'$ делится на w. Следовательно, значения c с разными остатками от деления на w можно обрабатывать независимо. Зафиксируем остаток r и рассмотрим только веса вида $r + sw$. Обозначим $g_s = dp[i-1][r + sw]$. Чтобы перейти от суммарного веса $r + sw$ к суммарному весу $r + tw$, где $s \leqslant t$, нужно взять $t - s$ предметов i-го типа, поэтому

$$dp[i][r + tw] = \max_{0 \leqslant s \leqslant t} \left(g_s + (t-s)v - (t-s)^2\right).$$

$$g_s + (t-s)v - (t-s)^2 = \left(tv - t^2\right) + \left(2s \cdot t + g_s - sv - s^2\right).$$

Первое слагаемое не зависит от s, а второе является линейной функцией от t. Обозначим $k_s = 2s$ и $b_s = g_s - sv - s^2$, тогда

$$dp[i][r + tw] = tv - t^2 + \max_{0 \leqslant s \leqslant t} \left(k_s \cdot t + b_s\right).$$

Таким образом, каждому s соответствует прямая $y = k_s x + b_s$, и для вычисления $dp[i][r + tw]$ нужно найти максимальное значение в точке $x = t$ среди прямых, для которых $s \leqslant t$. Будем перебирать t по возрастанию: на каждом шаге сначала добавим прямую, соответствующую $s = t$, а затем найдем максимум в точке $x = t$. Для поддержки этих операций воспользуемся техникой Convex Hull Trick.

Асимптотика решения: $O(nm)$.