Задача A. Стрельба из пушки

Автор задачи и разработчик: Даниил Голов

Давайте заметим, что прямой выстрел до цели, преодолевающий наикратчайшую дистанцию, и пролетающий над препятствием, идет по вектору, направленному из точки $(0, 0)$ в точку (d, w). Соответственно, если вектор с таким направлением и длиной k имеет X-координату меньше $2d$, то достать цель невозможно.

Если же X-координата такого вектора больше $2d$, то достаточно рассмотреть два случая:

- если $2w \leqslant h$, то прямой выстрел достигнет цели, и искомый угол равен $\tan^{-1}(\frac{w}{d})$;
- если же $2w > h$, то под таким углом снаряд перелетит дом, и необходимо использовать суперспособность NX5 с уходом пучка вниз, чтобы перебросить стену: в таком случае траектория полета пучка будет гипотенузой длины k прямоугольного треугольника с одним из катетов длины $2d$, то есть искомый угол будет составлять $\tan^{-1}\left(\frac{\sqrt{4k^2-4d^2}}{2d}\right)$.

Для полного решения достаточно просто проверить, какой из двух возможных вариантов подходит, и вывести любой подходящий. Во втором случае необходимо и достаточно, чтобы снаряд, запущенный под таким углом, не попал в препятствие. Все вычисления, кроме итогового вычисления угла, могут быть сделаны в целых числах.

Задача B. Иерархия цитадели

Автор задачи: Александр Гордеев, разработчик: Даниил Орешников

Переформулируем задачу: в дереве, в котором все листья расположены на одном и том же уровне, требуется отсортировать эти самые листья по возрастанию, меняя порядок детей у каких-либо вершин.

Заметим следующий факт: если листья в поддереве какой-либо вершины не образуют множество подряд идущих чисел, то отсортировать их не получится. И, наоборот, если поддерево каждой вершины содержит листья с подряд идущими номерами, то есть способ отсортировать их. Действительно, если в каком-то поддереве найдутся i и k, что существует j между ними ($i < j < k$) не из этого поддерева, то эта тройка листьев не сможет быть упорядочена: либо j будет стоять после k, либо до i.

Обратное утверждение следует из алгоритма решения: сделаем `dfs` (на самом деле в рамках данной задачи достаточно просто пройти по внутренним вершинам дерева в порядке убывания их номеров), и для каждой вершины посчитаем, чему равны минимальный и максимальный номера листьев в ее поддереве, проверив, что между этими значениями лежит ровно столько же чисел, сколько листьев в поддереве.

Когда эта информация посчитана для всех u_i — детей вершины v, отсортируем их в порядке возрастания номеров листьев в поддеревьях, и проверим, что для любых двух соседних детей выполняется $\texttt{min\_leaf}(u_i) = \texttt{max\_leaf}(u_{i-1}) + 1$. Только если все такие равенства выполняются, листья поддерева вершины v и поддеревьев всех ее потомков образуют отрезки подряд идущих значений, а подходящий способ сортировки — это перестановка u_i в полученном нами отсортированном порядке.

Для каждой вершины мы сортируем всех ее детей, что дает асимптотику времени работы $\mathcal{O}(n \log n + m)$.

Задача C. Планеты двух измерений

Автор задачи и разработчик: Константин Бац

Рассмотрим случаи.

- Если $n < m$, то единственный оптимальный способ — начать путешествие с планеты из измерения Y, затем полететь на планету из X, затем снова на планету из Y и так далее. Тогда путешествие закончится на планете в изменении Y. Всего получится посетить $2 \cdot n + 1$ планету.

- Если $n = m$, то получится посетить все планеты в обоих измерениях. Ответ в таком случае равен $n + m = 2n$.
- Если $n > m$, то, аналогично первому случаю, можно посетить не больше $2 \cdot m + 1$ планет.

Таким образом, если $n = m$, то ответ $n + m$, иначе $2 \cdot \min(n, m) + 1$.

Задача D. Эффективный двигатель

Автор задачи и разработчик: Мария Жогова

Докажем, что в конце будут активными только те карманные вселенные, которые являются полными квадратами. Посмотрим на произвольное число m и на его разложение на простые:

$$m = p_1^{a_1} \cdot p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}.$$

Известно количество его натуральных делителей, оно равно в точности $(a_1+1)\cdot(a_2+1)\cdot\ldots\cdot(a_k+1)$. Действительно, каждое простое из разложения m входит в его делитель независимо от других со степенью от 0 до соответствующего a_i включительно.

Заметим, что такое произведение нечетно тогда и только тогда, когда все a_i четны. А это равносильно тому, что $\sqrt{m}$ — целое число, то есть m — полный квадрат. А если карманная вселенная номер m активна, это как раз и равносильно тому, что количество делителей m нечетно, ведь изначально все вселенные заморожены, а дальше меняют свое состояние перед каждым полетом, номер которого является делителем m.

Посчитать количество полных квадратов от 1 до n можно за время $\mathcal{O}(1)$, просто вычислив корень из n и округлив его вниз.

Задача E. Инопланетные кальмары

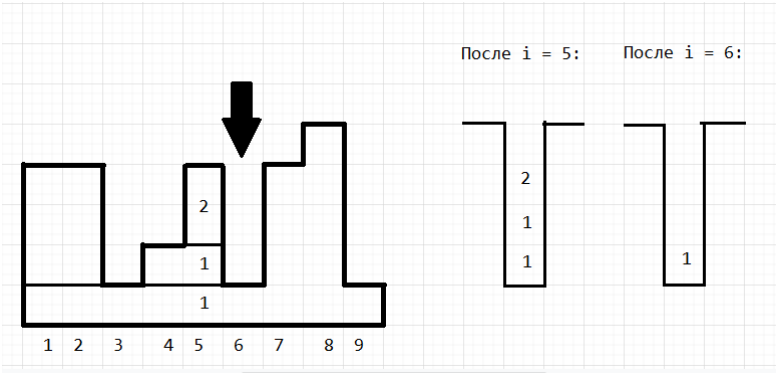
Автор задачи и разработчик: Александр Гордеев

При внимательном рассмотрении условия можно заметить, что эту задачу можно свести к другой — заполнить плоскость, которая была образована кальмарами, минимальным числом прямоугольников.

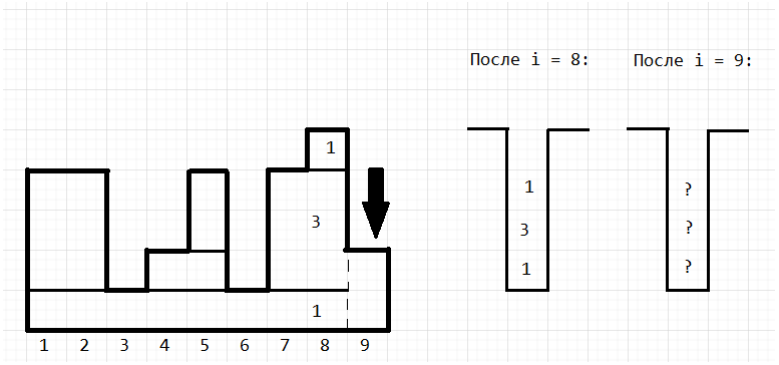
Решим для начала более простую задачу — пусть наши прямоугольники имеют высоту 1. Тогда в начале нужно «открыть» a_1 отрезков, при переходе к a_2 либо открыть ещё $a_2 - a_1$ отрезков, если $a_2 > a_1$ и закрыть $a_1 - a_2$ отрезков в противном случае и новые отрезки не появляются. Тогда ответ на задачу — сумма разностей соседних значений $a_{i+1} - a_i$, если эта разность больше нуля.

Попробуем применить аналогичную логику к этой задаче — в a_1 достаточно открыть один прямоугольник с началом в столбце 1 и изначальной высотой a_1. Конец же этого прямоугольника мы выберем позже. Теперь перейдём на столбец 2. Если $a_2 > a_1$, то достаточно создать ещё один прямоугольник с началом в 2 и высотой $a_2 - a_1$. Но что если $a_1 > a_2$?

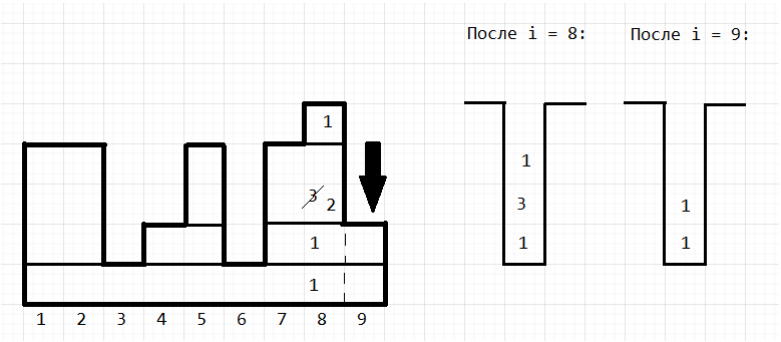
Заведём стек, в который будем добавлять наши открытые, но ещё не закрытые прямоугольники. Тогда при $a_{i+1} > a_i$ мы бы добавили в стек число, равное $a_{i+1} - a_i$. Если $a_{i+1} < a_i$, то нужно как-то уменьшить высоту, на которой мы сейчас находимся. Пользуемся тем, что прямоугольники добавляются в порядке появления, значит числа в стеке будут лежать в том же порядке, что и соответствующие прямоугольнике на гистограмме, и сверху стека будет лежать самый верхний незакрытый прямоугольник. Тогда достаточно убрать столько чисел со стека, чтобы их сумма была равна разнице:



Но возникает проблема — а что делать если сумма верхних элементов не может быть в точности равна a_i?



Если мы уберём прямоугольник высоты 1, то следующий столбец будет высоты $4 \neq 2$, если 1 и 3, то следующий столбец будет высоты 1, что тоже не подойдёт. Поступим так — сверху будем убирать числа пока можем, если же со следующим удаляемым сумма превысит разность соседних столбцов (обозначим непокрытый остаток разницы как $diff$), то тогда достаточно разбить «слишком большой» удаляемый x на два прямоугольника — $diff$ и $x - diff$. Тогда убираем прямоугольник высоты $diff$ и тогда проблема решена.



Так получаем жадный алгоритм решения. В начале добавляем в стек натуральное число a_1. Если $a_i > a_{i-1}$, то добавляем в стек $a_i - a_{i-1}$. Иначе убираем числа с верхушки стека, пока их сумма не будет равна $a_{i-1} - a_i$. Если не получается убрать со стека элементы ровно с этой суммой, то одно из чисел x бьём на два, равные $diff$ и $x - diff$. Ответ — количество добавленных в стек чисел. Важно также заметить, что нули, добавленные в стек, в ответ не идут по очевидным причинам.

Задача F. Артефакты

Авторы задачи и разработчики: Владимир Новиков, Максим Дмитриев

У авторов есть два разных решения, в разборе будет приведено одно из них, альтернативное можно посмотреть в разборе продвинутой версии.

Сразу обозначим за $mask$ описание набора артефактов, которые мы успели собрать. Это будет число от 0 до 3, где 0 — нет собранных артифактов, 1 и 2 — собран первый или второй артифакт, 3 — собраны оба.

Утверждение: оптимальный обход — это сумма ребер, взятых дважды, минус сумма ребер на диаметре нашего пути. Это следует просто из рассмотрения выбранного маршрута: его можно представить как непрерывный простой путь, от которого ответвляются поддеревья, на которых каждое ребро обходится в обе стороны (чтобы обойти все дерево и вернуться на «основной» маршрут).

Будем решать задачу с помощью поиска в ширину. Для каждой вершины создадим 2^k ее копий, отвечающих разным наборам артефактов. Пусть $\texttt{dp}[i][mask]$ — минимальный ответ на задачу, если мы закончили наш путь в вершине i, предварительно посетив $mask$ артефактов. Тогда мы можем запустить поиск в ширину из всех вершин i с $mask = 0$. Теперь осталось научиться обновлять нашу $mask$.

Пусть мы стоим в вершине i с артефактом $a[i]$. Тогда обозначим $new\_mask = mask$ `or` $2^{a[i]}$, то есть маска с добавленным артифактом $a[i]$. Попытаемся обновить ответ для $\texttt{dp}[i][new\_mask]$ через $\texttt{dp}[i][mask]$, после чего присвоим $mask = new\_mask$ (несложно понять, что нам надо выполнять такое присваивание в любом случае). Затем пройдемся по ребрам, исходящим из вершины i, и попробуем обновить ответ для $\texttt{dp}[j][new\_mask]$. Ответом на задачу будет минимум по всем $\texttt{dp}[i][2^k - 1]$. Если же мы не смогли посетить ни одну такую $mask$, то ответ на задачу равен -1. Итоговая асимптотика будет равна $\mathcal{O}(n \cdot 2^k \cdot \log(n \cdot 2^k))$.

Задача G. Незваные гости

Автор задачи: Даниил Голов, разработчик: Константин Бац

Заметим, что Землю посетило существ не меньше, чем существ было на Земле одновременно. Очевидно, что это верно не только для существ в целом, но и для существ в каждой из групп.

Покажем, что такую оценку можно достичь, то есть занумеровать прилетающие и улетающие существа, чтобы различных номеров существ в каждой группе было не больше, чем их было одновременно на Земле. Действительно, рассмотрим момент, когда их было больше всего на Земле, и присвоим каждому уникальный номер. Заметим, что это не нарушает условия на логи того, как они будут улетать с Земли и прилетать на нее. Поскольку существа могут прилетать повторно, то больше номеров, чем мы изначально дали, не понадобится.

Таким образом, если считать, что операция `+` увеличивает некоторый счетчик на один, `-` уменьшает его, то задача сводится к нахождению максимальных значений у n различных счетчиков. Для решения такой задачи достаточно хранить массив из n чисел и явным образом поддерживать число существ на планете в данный момент.

Время работы такого решения — $\mathcal{O}(n + m)$.

Задача H. Халат Рика

Автор задачи: Константин Бац, разработчик: Мария Жогова

Рассмотрим путь раствора с отдельной точки халата, на которую он был нанесён.

По условию, длина этого пути — расстояние от этой вершины до ближайшей дырки. Поскольку вес любого ребра в нашем графе равен единице, то для поиска кратчайшего расстояния до любой дырки в таком графе можно использовать поиск в ширину (`bfs`).

Нам надо найти время, через которое халат высохнет, то есть раствор, нанесённый на каждую из начальных вершин дотечёт до каких-либо дырок. Заметим, что это равно максимальному значению среди минимальных расстояний, найденных выше.

Ограничения задачи не позволяли явно запустить поиск в ширину для каждой из начальных вершин. Однако можно было использовать один `bfs` для поиска всех расстояний сразу. Будем запускать алгоритм не из начальных точек, а из всех дырок в халате одновременно. Перед запуском поиска в ширину добавим в очередь все точки, из которых раствор вытекает, и скажем, что расстояние до них равно 0. Тогда, когда алгоритм закончит свою работу, у нас будут посчитаны минимальные расстояния от ближайшей дырки до каждой вершины графа. Найдем максимальное посчитанное значение среди начальных вершин, это и будет ответом на задачу.

Итого, нам придется запустить один `bfs`, который работает за $\mathcal{O}(n+m)$, и еще несколько раз проитерироваться по массивам. Время работы такого решения — $\mathcal{O}(n+m)$.