Разбор задачи «Акромантулы»

Отсортируем пауков по возрасту. Будем по очереди (по возрастанию возраста) брать пауков и назначать им детей.

Пусть у очередного паука возраст a_i и ограничение на число детей c_i. Тогда рассмотрим пауков возраста меньше a_i (поскольку пауки упорядочены, это префикс, и его можно поддерживать при помощи метода двух указателей). Назначим текущему пауку как можно больше детей из этого префикса, у которых ещё нет родителя (c_i или меньше, если не хватает). Мы можем назначить нужное количество детей наименьшего возраста, тогда пауки с уже назначенными родителями будут образовывать префикс.

Таким образом, будем перебирать текущего паука i, поддерживать x — последнего паука возраста меньше a_i, и y — сколько пауков уже получили родителя. Назначим i-му пауку не более c_i детей, обновим y.

В конце y должно быть равно $n-1$, иначе ответ «NO».

Разбор задачи «Волшебный чемодан»

Откажемся от всех сложных структур. Заметим, что в массиве после всех операций будет $O(k)$ подстрок исходного массива. То есть мы можем представить конечный массив как конкатенацию подстрок из начального массива $a_{i_1} + a_{i_2} + \cdots + a_{i_k}$, где $i_1, i_2, \cdots, i_k$ — перестановка, k — количество подстрок.

После каждой операции будем в явном виде поддерживать массив перестановок i и блоков a. Введем вспомогательную операцию: разделить исходный массив в точке x, если x не является границей какого-либо блока, выполняем операцию за $O(k)$. Рассмотрим как происходит операция первого типа. В начале и в конце каждого отрезка проведем операцию деления на блока, так мы суммарно добавили 4 отрезка или меньше. Теперь мы можем поменять местами блоки на этих отрезках за $O(k)$ потому, что начала и концы отрезков точно находятся на границах блоков. Для запроса второго типа просто проведем операцию разреза.

Для выполнения операции второго типа выполним проход по всем операциям первого, что бы получить массив блоков, в котором после применения любого запроса блоки не будут создаваться, а будут только меняться местами. Вторым проходом по всем операциям будем отвечать на запросы. Теперь отсортируем числа внутри каждого блока и линейным проходом с бинарным поиском ответим на запрос за $O(k \cdot log\, n)$. Числа k не превышает $4 \cdot m$, так как каждая операция добавляет не более 4-ех новых блоков.

Разбор задачи «Волшебные существа»

Рассмотрим очередной отрезок $[a_i, b_i]$. Если правый конец отрезка b_i меньше, чем t, перейдем к следующему отрезку. Далее если левый конец отрезка a_i меньше, чем t, приравняем его t. Затем сдвинем координаты влево на t, для этого вычтем t из a_i и b_i. Теперь нам нужно найти точки, делящиеся на s, то есть $0, s, 2 \cdot s, 3 \cdot s$ и так далее. Для этого найдем к числу a_i ближайшую справа точку, делящуюся на s, по формуле $a_i = a_i + (s - (a_i \mod s))$, и к числу b_i ближайшую слева точку, делящуюся на s, по формуле $b_i = b_i - (b_i \mod s)$. Затем найдем количество точек на отрезке $[a_i, b_i]$ по формуле $ans = \frac{b_i - a_i}{s} + 1$. Прибавим ans к ответу на задачу.

Разбор задачи «Кольцевые дороги»

Заметим, что дуга внешней окружности между соседними дорогами больше соответствующей дуги внутренней окружности. Поэтому, по внешней окружности путь будет проделать от конечной точки до одной из двух ближайших дорог, а вся оставшаяся часть пути будет проходить по внутренней окружности.

Отсортируем по возрастанию углы наклонов дорог. Найдем две ближайшие к конечной точке с каждой стороны дороги с помощью двоичного поиска, выберем из двух возможных ответов минимальный.

Также можно было предподсчитать для каждой точки ближайшие дороги и отказаться от двоичного поиска.

Разбор задачи «За коллективизм!»

Отсортируем помощников по количеству пойманный тварей в порядке неубывания. Несложно видеть, что тогда помощники, которые будут участвовать в отчетности образуют непрерывный отрезок. Если на каком-то шаге мы уже выбрали несколько помощников (которые будут писать отчет), и наибольшее количество пойманных тварей среди них обозначим за a. То на следующем шаге выгодно брать того помощника, который поймал количество тварей b, так чтобы разница между a и b была минимальна. Очевидно, что это именно тот помощник, кто стоит следующим в отсортированном массиве.

То есть теперь нам остается только найти такой отрезок. Пусть суммарное количество набранных тварей на этом отрезке — s, минимальное количество тварей у помощника — c и длина отрезка (то есть количество помощников) — t. Тогда должно выполняться

$$s - c \times t \leqslant k$$

Остается посмотреть кто из помощником может быть началом этого отрезка (просто пробежаться по всем помощникам в отсортированном порядке) и для каждого начала найти наибольшую возможную длину отрезка, например, бинарный поиском. Те, из помощников, кто не попадут в искомый отрезок и будут ответом.

Разбор задачи «Кроссворды»

Будем для удобства называть слова *top*, *bottom*, $left$, $right$ — соответственно верхнее слово, нижнее, левое и правое в расположении.

Для начала переберем, какое слово является *top*, какое *bottom* и т.д. (таких вариантов ровно 4! = 24 штуки). После этого посчитаем, сколько из фиксированных *top*, *bottom*, $left$ и $right$ можно составить кроссвордов:

- Переберем, где $left$, $right$ пересекают *top* — $O(|w|^2)$;
- Переберем сдвиг *bottom* относительно *top* — $O(|w|)$;
- Переберем расстояние между *top*, *bottom* — $O(|w|)$;
- Теперь осталось понять, сколькими различными способами можно подвигать $left$, $right$, чтобы символы на пересечении совпали с символами из *top* и *bottom*;
- Двигать $left$, $right$ можно независимо, поэтому посчитаем для каждой отдельно, а потом перемножим;
- Подсчет для каждой отдельно — $O(|w|)$;
- Итого получаем асимптотика $O(|w|^5)$, что на самом деле является довольно большой оценкой сверху и спокойно укладывается в TL.

Также можно было соптимизировать это решение до более хорошей асимптотики, но этого в задаче не требовалось.

Разбор задачи «Взрывопотам»

Пусть мы зафиксировали циклический сдвиг, и перед нами стоит задача поиска наидлиннейшей подпоследовательности, описанной в условии. Если провести ребро из числа в ближайшее справа к нему, которое больше него, то полученный граф будет лесом деревьев, и в этом лесу нужно найти самый глубокий лист. Построить такое дерево для последовательности можно одним проходом со стеком — при добавлении нового числа нужно снять со стека все числа, которые меньше нового, и провести из них ребра в новое.

Для решения задачи можно было заметить, что если приписать ко входному массиву в конец его копию, то после построения такого леса самый глубокий лист в нем будет ответом на задачу, с учетом всех возможных циклических сдвигов. Чтобы это доказать, можно заметить, что разность

номеров листа и корня не больше n, кроме того, любой путь обязательно будет присутствовать в этом дереве. Таким образом, получили решение, работающее за $O(n)$.

Разбор задачи «Восстановление числа»

Будем решать задачу методов «разделяй и властвуй».

Разобьем число пополам, для младшей половины переберм цифры вместо вопросиков и посчитаем остатки при делении на m. Для старшей половины проделаем то же самое.

Теперь нужно выбрать остаток из старшей половины и из младшей, так чтобы минимизировать результирующий остаток. Пусть мы выбрали остаток из старшей половины, равынй a, из младшей — b, тогда результирующий остаток будет равен $(a \times 10^k + b) \mod m$, где k — количество цифр в младшей половине числа.

Умножим все остатки из большей половины на 10^k по модулю m. Теперь отсортируем остатки в обеих частях. Это можно сделать двумя способами: либо обычной сортировкой, либо используя подход с битовыми масками.

Первый способ не представляет сложности в реализации, но с ним могли возникнуть проблемы со временем исполнения, так как нам нужно отсортировать два массива по 10^7 элементов каждый. Скорость выполнения этой части решения очень зависит от используемого вами языка и компилятора.

Второй подход с битовыми масками представлял следующее: будем хранить в каждой ячейке массива 32-битную маску — какие числа присутствуют в данном массиве. То есть, если число x присутствует в массиве, то в элементе массива $a[\frac{x}{32}]$, в бите под номером $x \mod 32$ будет записана единица, иначе — ноль.

Тогда для меньшей части пометим в этом массиве остатки, которые имеются в левой части числа и просто пройдем по числам от 0 до 10^7 и проверим, что соотвествующий бит равен единице. Размер этого массива будет составлять $\frac{10^7}{32}$, и сортировка произойдет за 10^7 операций.

Для большей части все немного сложнее, так как мы умножили все значения на 10^k и взяли по модулю m и теперь все значения лежат в промежутке от 0 до $m-1$. Заведем массив с битовыми масками, размером $\frac{m}{32}$. Пометим в нем все числа из правой части. Теперь нужно получить их в отсортированном порядке. Просто пройтись от 0 до m будет затратно по времени, поэтому воспользуемся свойством, что массив разрежен, и среди 10^9 значений в нем могут быть только 10^7. Будем проходить только по тем маскам, значения которых неравны нулю. Таких масок будет не более 10^7, и суммарный проход по всех их битам займет не более $32 \times 10^7 + \frac{m}{32}$. Также возможны оптимизации, которые уменьшают это количество до $\frac{m}{32}$.

Вернемся теперь к решению. Пусть мы отсортировали остатки в меньшей и большей половинах, обозначим эти массивы за l_i и r_j соотвественно. Теперь нужно выбрать из этих двух массивов такие остатки l_i и r_j, что $(l_i + r_j) \mod m$ минимально. Это можно сделать двумя указателями, заметив тот факт, что для каждого l_i минимальный остаток могут давать только два значения r: r_0 и такое r_k, что $l_i + r_k \geqslant m$, и $l_i + r_{k-1} < m$. Это следует из того, что для любых i, j: $l_i + r_j < 2 \times m$. То есть просто увеличиваем указатель i, и для каждого i уменьшаем k, пока не встретим первое такое r_{k-1}, что $l_i + r_{k-1} < m$.

Выбрав минимум из всех таких пар мы получим ответ.

Сложность данного решения либо: $\sqrt{n} + \frac{m}{32}$ либо $\log n \times \sqrt{n}$ в зависимости от реализации.

Разбор задачи «Сверкающие плюсы»

Найдем для каждой ячейки с единицей количество единиц слева, справа, сверху и снизу от нее. Рассмотрим подсчет количества единиц сверху, остальные три случая проделываются аналогично. Ответ для каждой ячейки с номером строки i и номером столбца j будем хранить в ячейке массива $up[i][j]$. Во внешнем цикле перебираем столбцы. Во внутреннем строки от второй до n-ой. Если ячейка сверху от текущей равна единице, ответ для текущей ячейки равен ответу в предыдущей плюс один, то есть $up[i][j] = up[i-1][j] + 1$. Иначе ответ для данной ячейки равен нулю.

После подсчета четырех случаев для каждой ячейки найдем минимум k из количества единиц слева, справа, сверху и снизу от нее. Размер плюса с центром в текущей ячейке равен $4 \cdot k + 1$.

Среди всех ячеек найдем ячейку с максимальным размером плюса с центром в ней. Выведем размер максимального плюса и координаты этой ячейки.

Ответа не существует в единственном случае — когда матрица состоит из одних нулей.

Разбор задачи «Тайные комнаты»

В условии дан граф, в котором у каждой вершины одно исходящее ребро.

Несложно заметить, что для того чтобы произвести требуемую в условии операцию, необходимо, чтобы ровно у одной вершины не было входящего ребра. Это понятно из следующего факта: если мы меняем выход у одной вершины, а вершин, у которых нет входящих ребер больше одной, то одна из них точно не будет принадлежать циклу.

Графы с такими ограничениями имеют вид цикла и одного входящего в него пути. И выход нужно поменять у вершины, принадлежащей циклу, в которую входит этот путь.

Отсюда следует решение. Найдем вершину, в которую не входи ни одного ребра. Пройдем n раз по ребрам: мы окажемся в вершине, в которую входит путь. Вернемся назад на одну вершину и изменим у нее выход на стартовую вершину.

Ответ «NO» будет лишь в том случае, если в результате обхода n вершин, мы посетили некоторые вершины дважды (кроме последнего n-го перехода, который должен вернуться в вершину цикла).

Разбор задачи «Новый чемодан»

Разберем несколько случаев:

- Если $n < 7$, нужный прямоугольник составить не получится;
- Если же $n \geqslant 7$, рассмотрим еще несколько подслучаев:
  - $n = 4k$: Составим пары равных сторон как $\{1, 4\}$, $\{2, 3\}$ и $\{5, 8, 9, 12, 13, 16, \ldots\}$, $\{6, 7, 10, 11, 14, 15, \ldots\}$;
  - $n = 4k + 1$: Составить прямоугольник из всех прутиков не получится, потому что их суммарная длина — нечетное число, однако можно составить из всех, кроме прутика длиной 1: $\{2, 5\}$, $\{3, 4\}$ и $\{5, 8, 9, 12, 13, 16, \ldots\}$, $\{6, 7, 10, 11, 14, 15, \ldots\}$;
  - $n = 4k + 2$: Составить прямоугольник из всех прутиков не получится по той же причине, но все еще можно составить из всех прутиков, кроме прутика длины 1: $\{2, 3, 5\}$, $\{4, 6\}$ и $\{7, 10, 11, 14, 15, 18, \ldots\}$, $\{8, 9, 12, 13, 16, 17, \ldots\}$;
  - $n = 4k + 3$: Составим пары равных сторон как $\{1, 2\}$, $\{3\}$ и $\{4, 7, 8, 11, 12, 15, \ldots\}$, $\{5, 6, 9, 10, 13, 14, \ldots\}$.

Соответственно, если $n \geqslant 7$, то при $n = 4k$ и $n = 4k + 3$, прямоугольник можно составить из всех прутиков, а при $n = 4k + 1$ и $n = 4k + 2$ из всех, кроме прутика длины 1.