Задача A. Найдите отличия

Рассмотрим два варианта решения:

Первое решение. Посчитаем количество вхождения каждого элемента в каждый из массивов. Пусть получили два словаря f и h — значение $f[i]$ равно количеству вхождений элемента i в массив a, аналогично значение $h[i]$ равно количеству вхождений элемента i в массив b. Тогда u это такой элемент что $f[u] = h[u] - 1$, а v такой элемент что $f[v] + 1 = h[v]$.

Сложность: $\mathrm{O}(n)$

Второе решение. Отсортируем массивы a и b по неубыванию. Пройдемся по массивам и найдем такой минимальный индекс i, что $a_i \neq b_i$, а также максимальный индекс j. Можно утверждать, что $[u = a_j$ и $v = b_i]$ если $a_i = b_{i+1}$, иначе $[u = a_i$ и $v = b_j]$.

Сложность: $O(n \times logn)$

Задача B. Мне повезет

Заметим, что игра всегда закончится максимум за m раундов, потому что через m раундов величина ставки будет m, а так как игра не закончилась ранее, то у вас от 0 до $m-1$ баллов, следовательно при любом исходе раунда вы закончите игру.

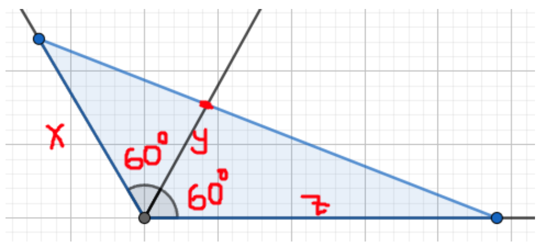
Пусть $dp_{i,j}$ — вероятность получить в игре i баллов на j-м раунде. Изначально $dp_{(n,0)} = 1$, для всех остальных $i \neq n$ выполняется $dp_{(i,0)} = 0$. При новом раунде $dp_{(i,j)}$ пересчитывается как сумма $dp_{(i-j-1,j-1)}/2$ (при $i - j - 1 \geqslant 0$) и $dp_{(i+j-1,j-1)}/2$ (при $i - j - 1 < m$).

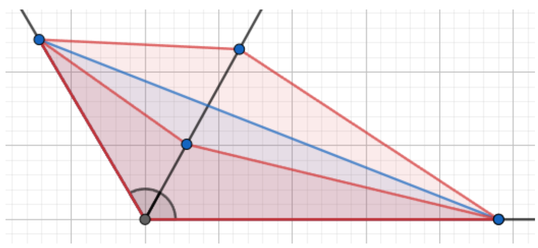
Сложность: $O(nm)$

Задача C. Выпуклость лепестка

Заметим, что "невыпуклость"может проявляться единственным образом: пусть есть некоторая точка X, отмеченная на луче на расстоянии x от точки D, затем в некотором направлении отмечена следующая точка Y на расстоянии y, а затем точка Z на расстоянии z. Проведем отрезок от точки X до точки Z, обозначим точку пересечения отрезка с лучом как P на расстоянии p до точки D. Тогда если $y < p$, то многоугольник невыпуклый.

Существует еще один способ: найдем площадь фигуры, состоящей из двух треугольников XDY и YDZ, и если она меньше площади треугольника XDZ, то многоугольник невыпуклый.





Площадь треугольника XDY равна $1/2 \cdot sin60 \cdot x \cdot y$. Площадь треугольника YDZ равна $1/2 \cdot sin60 \cdot y \cdot z$. Площадь треугольника XDZ равна $1/2 \cdot sin120 \cdot x \cdot z$. Так как $sin60 = sin120$, то нужно сравнить $xy + yz$ и xz.

Сложность: $O(1)$

Задача D. Три фитиля

Решим задачу рекурсией: пусть есть некоторая функция f, которая принимает на вход оставшиеся длины фитилей, их состояния (не горит, горит с одной стороны или с двух) и текущее время с момента первого поджигания.

При первом запуске у нас есть $3^3 - 1$ варианта — поджечь некоторые фитили, возможно с двух сторон. Каждый вызов функции f вызывает новый вызов функции f спустя минимальное время, через которое сгорит какой-то фитиль (мы можем совершить какие-то действия, когда какой-то фитиль полностью сгорает). Если в какой-то момент в каком-то вызове f фитиль сгорел полностью в момент t, то ответ на задачу «YES», иначе «NO».

Оценим сложность: в худшем случае изначально у нас есть $3^3 - 1$ варианта запуска f, затем для каждого запуска есть еще $3^2 - 1$ запуска (когда сгорел одни фитиль, и мы запускаемся со всеми возможными состояниями остальных двух фитилей), соответственно далее у нас есть еще $3^1 - 1$ запусков для оставшегося одного фитиля. Минус 1 мы выполняли для исключения варианта, когда все фитили затушены. Итого $(3^3 - 1) \times (3^2 - 1) \times (3^1 - 1)$, что примерно $3^6 = 729$, что, очевидно, проходит по времени и памяти.

Сложность: $O(3^6)$

Задача E. Операции с массивом

Можно утверждать, что если в конечном массиве стало k нулей, то остальные $n - k$ элементов равны единице, а так как каждое применение операции уменьшает сумму элементов массива на 1, то:

1. Начальная сумма равна $sum(a)$;

2. Конечная сумма равна $n - k$.

Следовательно, нужно применить $sum(a) - (n - k)$ операций.

Сложность: $O(n)$

Задача F. Покрытие вершин дерева

Будем решать задачу с помощью динамического программирования. Пусть ans_i — ответ на задачу в той формулировке, что в приведена в условии, для поддерева с корнем в вершине i (считаем, что никаких особых вершин и вершин в принципе вне этого поддерева не существует).

Посчитаем массив ssw (Sum Subtree Weights), где ssw_i — это сумма всех весов вершин в поддереве вершины i. Еще нам нужно заполнить массив ssv (Subtree Special Vertices), где ssv_i — количество особых вершин в поддереве вершины i.

Далее будем рассматривать вершины в таком порядке, чтобы любая вершина была рассмотрена нами позже, чем любая из ее дочерних вершин — как вариант, можно запустить обход графа в ширину начиная с вершины 1, и запоминать в каком порядке была посещена каждая вершина, а затем использовать обратный порядок.

Пусть текущая вершина u, тогда если u особая вершина, то $ans_u = ssw_u$ — чтобы быстро это определять, можно хранить особые вершины в сете; иначе переберем все дочерние вершины i для вершины u и посчитаем сумму ans_i для всех таких i, что $ssv_i > 0$ или $ans_i < 0$ — это объясняется так: если в поддереве вершины i есть особые вершины, то нам необходимо обязательно покрыть некоторые вершины в поддереве i (это как раз ans_i — тот минимум, который мы можем сделать); если же $ans_i < 0$, то нам выгодно использовать это решение (покрытие каких-то вершин в поддереве i), так как оно уменьшает наш ответ.

Пройдя все вершины таким образом получим ответ на задачу в ans_1.

Сложность: $O(n)$

Задача G. КНБ

Посчитаем сколько элементов равны «камню», «ножницам» и «бумаге» — соответственно числа r, s, p. Переберем каждый элемент (всего три варианта) и проверим сколько минимум нужно сделать ходов чтобы он победил. Рассмотрим «камень» — допустим, в конечном итоге в массиве все элементы равны «камню» (другие элементы — по аналогии):

1. Если все элементы уже равны камню, т.е. $r = n$, то ответ 0;

2. Если $r = 0$, то «камень» не может победить;

3. Если $r \neq 0$ и $r + p = n$, то «камень» не может победить;

4. Иначе «камень» может победить за $s + 2p$ ходов, т.к. «камень бьет ножницы», то ножниц должно быть $n - r = s + p$, а так как «ножницы бьют бумагу», то еще p ходов мы тратим на то, чтобы превратить все элементы в «ножницы»;

Сложность: $O(n)$

Задача H. Таблица умножения

Переберем все числа i такие, что $i \geqslant 1$ и $t = (n-1)/i \geqslant i$ и прибавим к ответу $2t$, учитывая пару $(i; t)$ и $(t; i)$, но тогда все пары в квадратной области $(1 \leqslant x \leqslant z; 1 \leqslant y \leqslant z)$ были учтены дважды, число z равно такому максимальному i, что $i \leqslant (n-1)/i$, что примерно корень из n — вычтем из ответа z^2 и задача решена.

Несложно заметить, что числа i будут перебираться примерно до корня из n.

Сложность: $O(\sqrt{n})$

Задача I. Вставки в очереди

Можно решить задачу с использованием структуры данных «связный список», где элементами являются люди в очереди — например, можно создать класс «человек» с параметрами «имя» (строка), «ссылка на предыдущего человека» (объект) и «ссылка на следующего человека» (объект). Тогда операции каждого типа выполняются за постоянное время, так как мы всего лишь переопределяем ссылки у задействованных людей (их может быть два — если происходит вставка/удаление между двумя людьми, и один — если происходит вставка/удаление в конец или начало очереди). Для того чтобы по имени получить доступ к объекту человека, можно использовать структура «словарь» (или map), где ключом выступает имя, а значением ссылка на объект.

Сложность: $O(q)$

Задача J. Запросы на кувшинах

Из соображений жадности нужно всегда начинать выбор кувшинов с наибольшей вместимостью — тогда мы для заданной суммарной вместимости набора минимизируем количество выбранных кувшинов, что и требуется в запросе.

Отсортируем массив v по невозрастанию и построим массив префиксных сумм p. Теперь p_i — суммарная вместимость кувшинов с 1 по i-ый в отсортированном порядке. Для массива p в силу того, что в v нет отрицательных чисел выполняется $p_i \leqslant p_j$ при любых $i \leqslant j$, то есть массив p монотонно не убывает, а следовательно к нему применителен бинарный поиск. Для каждого i-ого запроса бинарным поиском найдем такое минимальное j, что $d_i \geqslant p_j$ за logn.

Сложность: $O(nlogn + qlogn)$

Задача K. Они чередуются!

Сделаем две переменные $mx = 1$ и $cur = 1$. Пойдем по строке s, начиная со второго символа: если $s_i \neq s_{i-1}$, то выполним $cur = cur + 1$ и $mx = max(mx, cur)$, иначе установим $cur = 1$.

Сложность: $O(n)$

Задача L. Заготовка камня

Оптимальным решением является наем рабочего с минимальным значением a_i, за d дней он может сделать $d//b_i$ блоков камня: если это меньше чем нам осталось, то нанимаем следующего по дешевизне изготовления блока камня рабочего. Вероятно, последний выбранный рабочий за d дней способен изготовить больше камней, чем осталось изготовить, поэтому нанимаем его на изготовление только остатка блоков. Сложность: $O(nlogn)$

Задача M. Чипи-чипи Чапа-чапа

Пусть $t_1 = 0$ и $t_2 = 0$ время, показывающее, на сколько минут Чипи-чипи и Чапа-чапа приползли позже 12:00 соответственно. Посмотрим на описание часов первой улитки, если там слово «hurry», то уменьшим t_1 на указанное далее в строке число, если же слово «late», то увеличим. Аналогичные операции проделаем со второй строкой и t_2.

Затем если $t_1 = t_2$, выводим together; если $t_1 < t_2$, то выводим «Chipy-chipy», а затем $t_2 - t_1$, иначе выводим «Chapa-chapa», а затем $t_1 - t_2$.

Сложность: $O(1)$