Разбор задачи «Эпическая потасовка»

Автор задачи: Алексей Трилис
Подготовка тестов и решений: Михаил Иванов
Автор разбора: Михаил Иванов

Решение 1, медленная симуляция. С помощью цикла промоделируем бой. Пока $h_1 > 0$ и $h_2 > 0$, будем заменять h_1 на $h_1 - a_2$, а h_2 — на $h_2 - a_1$. Как только впервые оказалось, что $\min(h_1, h_2) \leqslant 0$, мы поймём, что на очередном шагу потасовка была завершена. Если $h_2 \leqslant 0$, то мы точно знаем, что победил именно Бандергольф: поскольку он атаковал первым и вывел противника из боя, Бенадрил не сможет совершить свою атаку, так что, даже если на этом шагу оказалось $h_1 \leqslant 0$, в потасовке соответствующего удара Бенадрила не произойдёт. Если же $h_2 > 0$, то $h_1 \leqslant 0$, и, таким образом, победитель — Бенадрил: во время очередной пары ударов он выдержал атаку Бандергольфа, а своей атакой поверг его.

Данное решение требует $\mathcal{O}(k)$ операций, где k — количество атак. В худшем случае — если $h_1 = h_2 = 2 \cdot 10^9$, $a_1 = a_2 = 1$ — всего произошло $4 \cdot 10^9 - 1$ атак. Такое большое количество операций не укладывается в ограничение по времени.

Решение 1.5, улучшенная симуляция. Выберем какое-нибудь натуральное t и запустим предыдущее решение, заменив атаки героев на ta_1 и ta_2. То есть вместо одной пары атак будем за итерацию цикла симулировать сразу t пар атак. (Разумеется, мы вычислим числа ta_1 и ta_2 заранее, а не будем находить их каждый раз — как мы знаем из предыдущего решения, итераций цикла может прозойти очень много, поэтому каждая итерация должна происходить максимально быстро.) Пусть впервые оказалось, что $h_1 \leqslant 0$ или $h_2 \leqslant 0$; тогда отменим последние k пар ударов, то есть заменим h_1 на $h_1 + ka_2$ и h_2 на $h_2 + ka_1$. Теперь оба здоровья стали положительными, и мы применим предыдущее решение уже для нормальных атак a_1 и a_2.

Заметим, что первая стадия решения (с атаками ta_1 и ta_2) потребует примерно в t раз меньше операций, чем первое решение. Вторая же стадия (с атаками a_1 и a_2) потребует не более t итераций, поскольку, если бы мы вместо второй стадии запустили ещё одну итерацию первой стадии, мы бы сделали одно из здоровий неположительным. Таким образом, решение требует $\mathcal{O}(k/t + t)$ операций, где k — количество атак. Эта величина минимальна, когда $t \approx \sqrt{k}$, тогда время работы $\mathcal{O}\left(\sqrt{k}\right)$. Решение жюри, аналогичное данному, проходит ограничение по времени при небольших $t \geqslant 6$.

Это решение имеет неприятный подводный камень: дело в том, что величина ta_1 или ta_2 может не поместиться в 32-разрядный целочисленный тип данных. Можно избежать переполнения различными способами. Например, использовать 64-разрядный тип (что, правда, может негативно сказаться на производительности). Другим вариантом является такой: вместо ta_1 возьмём $\min(h_2, ta_1)$. Действительно, если $ta_1 \leqslant h_2$, решение полностью эквивалентно вышесказанному. Если же $ta_1 > h_2$, то после первой же итерации цикла первой стадии здоровье Бенадрила станет отрицательным, и мы просто для исходных значений здоровья запустим вторую стадию; другими словами, и в этом случае вторая стадия решения будет абсолютно такой же, как если бы атака в первой стадии была равна ta_1, а не $\min(h_2, ta_1)$. Чтобы найти $\min(h_2, ta_1)$ без переполнения, можно поделить h_2 на t и сравнить результат с a_1.

Решение 2, мгновенное. Предположим, что Бандергольфа, имеющего здоровье h_1 и атаку a_1, k раз ударит Бенадрил со здоровьем h_2 и атакой a_2. Тогда здоровье Бандергольфа станет равно $h_1 - ka_2$. Постараемся найти k_1 — наименьшее количество атак, требуемое от Бенадрила, чтобы вывести Бандергольфа из боя. Это наименьшее целое k, такое что $h_1 - ka_2 \leqslant 0$, или $ka_2 \geqslant h_1$, или $k \geqslant h_1/a_2$. Найти наименьшее такое k легко: оно равно результату целочисленного деления $h_1 + a_2 - 1$ на a_2. Таким образом, мы найдём

$$k_1 = \left[\frac{h_1 + a_2 - 1}{a_2}\right]; \qquad k_2 = \left[\frac{h_2 + a_1 - 1}{a_1}\right]$$

и сравним их: если $k_1 \geqslant k_2$, то после $k_2 - 1$ пар ударов и ещё одного удара Бандергольфа у него здоровье всё ещё будет положительным, а Бенадрил падёт. Если же $k_1 < k_2$, то после k_1 пар ударов Бандергольф проиграет Бенадрилу.

Это решение требует обозримого количества арифметических операций и работает за $\mathcal{O}(1)$.

Разбор задачи «Весы и монеты»

Автор задачи:	Аршак Айвазьян
Подготовка тестов и решений:	Павел Глушень
Автор разбора:	Аршак Айвазьян

Назовем монету подлинной, если про неё уже известно, что она не фальшивая. Назовем монету подозрительной, если она может быть как настоящей, так и фальшивой. Таким образом, пока не найдена фальшивая, все монеты делятся на подлинные и подозрительные.

Замечание 1. Если стратегия взвешивания не предполагает смешивания куч из только подлинных и только подозрительных монет, то все подозрительные в каждый момент времени равновероятно фальшивые.

Замечание 2. Если стратегия взвешивания смешивает кучу из подлинных и кучу из подозрительных монет, то существует стратегия, которая так не делает и справляется за такое же количество взвешиваний, что и исходная, или быстрее.

Замечание 3. Пусть n начальное количество монет и k количество чаш фиксированы; x количество подозрительных монет в данный момент времени. Если в каждый момент времени все подозрительные монеты равновероятно фальшивые, то при всех $x \leqslant n - k$ ответ (оставшееся количество взвешиваний в худшем случае при оптимальной стратегии) монотонен по x.

Следствие. Существует оптимальная стратегия, ответ для которой в каждый момент времени, за исключением начального, монотонен по количеству подозрительных монет.

Автор нашёл поистине формальное доказательство всем этим замечаниям, но поля слишком узки для него.

Искать оптимальную стратегию в таком классе (хороших) стратегий достаточно просто. Основная цель — минимизация размера наибольшей кучки.

Любую кучу монет имеет смысл делить максимум на $k + 1$ кучек: k из них можно класть на разные чаши весов, а оставшуюся не класть ни на одну из чаш (оставить на столе).

По принципу Дирихле, при таком разделении найдется хотя бы одна кучка, в которой будет не меньше $\lceil \frac{n}{k+1} \rceil$ монет. Мы получили оценку снизу на количество монет в худшем случае (фальшивая в самой большой кучке).

Предположим, что остаток от деления n на $k + 1$ не равен 2 и не равен $(k + 1) - 2$. (*)

Тогда мы можем доказать, что описанный ниже рекурсивный алгоритм оптимален.

Начнём раскладывать все подозрительные на данный момент монеты по кругу начиная с первой чаши (сначала кладём на первую, потом на вторую, ..., на последнюю, на стол и снова на первую, на вторую, ...). В конце у нас во всех кучках будет по крайней мере $\lfloor \frac{n}{k+1} \rfloor$ монет, причём в некоторых ровно столько, а в остальных $\lceil \frac{n}{k+1} \rceil$. Назовём чаши с меньшим количеством монет «маленькими», а с большим «большими». Если маленьких или больших чаш оказалось две, то добавим одну монету из раннее отсеянных на любую маленькую чашу.

Наш алгоритм отсеивает ненулевое количество монет при каждом взвешивании, поэтому такие есть. Если взвешивание первое, то описанной ситуации не может быть по предположению (*).

Взвешиваем. Если найдётся чаша которая расположена выше или ниже чем чаши с таким же количеством монет, то фальшивая монета на ней, в противном случае она на столе (Заметим, что так как ни маленьких, ни больших чаш не может быть две, то чаша с фальшивой определяется однозначно). Таким образом, количество подозрительных монет уменьшится в худшем случае (когда фальшивая на большой чаше) до $\lceil \frac{n}{k+1} \rceil$. Что в точности совпадает с нижней оценкой полученной из принципа Дирихле. Оптимальность доказана.

Если же остаток от деления n на $k + 1$ равен 2 (или, что симметрично, $(k + 1) - 2$) то нижняя оценка на первом взвешивании недостижима. Не трудно рассмотреть четыре качественно различных конфигурации:

- Чаша и чаша: две лишние (недостающие) монеты оказываются на разных чашах весов; две большие (маленькие) чаши весов, $k-2$ маленьких (больших). В худшем случае одна из этих чаш оказывается выше другой. Тогда обе кучи подозрительные. В худшем случае $2 * \lceil \frac{n}{k+1} \rceil$ ($2 * \lfloor \frac{n}{k+1} \rfloor$) монет.
- Чаша и стол: одна лишняя (недостающая) монета оказывается на весах, а другая на столе; одна большая (маленькая) чаша, k маленьких (больших), и большой стол. В худшем случае фальшивая монета оказывается на особенной чаше или на столе, и мы не получаем совершенно никакой информации. В худшем случае n монет.
- Чаша $\times 2$: две лишние (недостающие) монеты оказываются на одной чаше. В худшем случае фальшивая монета на этой чаше или на столе В худшем случае n монет.
- Стол $\times 2$: две лишние (недостающие) монеты оказываются на столе. В худшем случае фальшивая монета на столе. В худшем случае $\lceil \frac{n}{k+1} \rceil + 1$ монет.

Четвертая конфигурация оказалось самой эффективной. Все предыдущие проигрывали ей из-за утраты однозначности.

Итак, мы построили оптимальный алгоритм. Количество взвешиваний в худшем случае легко считается следующим циклом:

```
r = 0

if (n mod (k+1) == 2) or (n mod (k+1) == (k+1)-2):
    n = ceil(n / (k+1)) + 1
    r += 1

while n > 1:
    n = ceil(n / (k+1))
    r += 1

print(r)
```

Здесь *ceil* — функция, округляющая аргумент вверх.

В этом алгоритме число n делится на $k+1$, пока можно, откуда время работы алгоритма можно оценить как $\mathcal{O}(\log_{k+1} n)$, что есть $\mathcal{O}(\frac{\log n}{\log k})$.

Разбор задачи «Индийская клавиатура»

Запомним для каждой буквы j ее координаты x_j и y_j.

Очевидно, что стоит каждый раз брать прямоугольники, буквами в которых можно напечатать как можно больший последовательный кусок строки. Будем набирать буквы слева направо, поддерживая координаты левого нижнего и правого верхнего углов минимального прямоугольника, в который все набранные буквы помещаются. Если размеры прямоугольника так или иначе превысят $p \cdot q$, необходимо завершить текущий кусок строки и перейти к новому.

Итоговая сложность составит $O(n \cdot m + |s|)$.

Разбор задачи «Киви»

Поймем, как решать задачу для фиксированного k. Пусть $dp_k(u)$ — максимальная глубина k-ичной кучи с корнем в вершине u. Тогда возможны два случая:

- У вершины u менее k детей. Тогда $dp_k(u) = 1$;

- У вершины u хотя бы k детей. Тогда разумно взять среди них те, значения динамики в которых максимальны. Пусть x — k-е в порядке невозрастания значение $dp_k(v_i)$, где v_i — сыновья вершины u. Тогда $dp_k(u) = x + 1$.

Это дает нам возможность посчитать ответ для фиксированного k за $O(n \log n)$ (или даже за $O(n)$, если использовать функцию nth_element). Итоговая сложность— $O(n^2 \log n)$; этого достаточно, чтобы пройти тесты групп 1 и 3. (Тесты группы 2 было достаточно легко обработать отдельно).

Определим $p(k) = dp_k(1)$. Заметим следующие важные факты:

- $p(k) \geqslant p(k+1)$ для всех k;
- $p(k) \leqslant \log_k n$ при $k > 1$ (напрямую следует из формулы для суммы геометрической прогрессии).

Следовательно, функция $p(k)$ для $2 \leqslant k \leqslant n$ представляет собой не более чем $\log_2 n$ отрезков, на которых значения функции совпадают. В силу монотонности правую границу отрезка при фиксированной левой можно найти простым двоичным поиском. Итоговая сложность составит $O(\log^2 n \cdot g(n))$, где $g(n)$ — время, требуемое на нахождение $p(k)$, то есть $O(n \log^3 n)$ с крайне маленькой константой.