

Врата Вавилона

Автор задачи и разработчик: Егор Юлин

Группа 1

Для решения данной подзадачи можно было перебрать стартовое значение силы, оно равно H или $H - h_1$, после чего проэмулировать сражение со Слугой, и повторно перебрать применение артефакта.

Группа 2

Для решения данной подзадачи можно было при фиксированном d перебрать все достижимые значения силы и проэмулировать все сражения со Слугами, полностью перебирая все возможные варианты действий на каждой фазе.

Группы 3, 4, 5, 8

Для решения данных подзадач можно было использовать динамическое программирование. Будем хранить каждую достижимую силу, обозначим достижимость за $\text{dp}[x]$, где данное значение равно 1, если король героев перед битвой может иметь x силы, и 0 в противном случае.

Тогда пересчет можно делать с помощью обхода в глубину, перебирая для одного x различные артефакты: проведем ребро из силы i в силу j , если применением одного артефакта, имея силу i , мы получаем силу ровно j . Для каждого боя будем делать переход «если можно было иметь x силы, то теперь достижима $x - a_i$ ». Для каждой фазы восстановления будем заново запускать обход в глубину всех доступных состояний и обновлять dp .

Итоговое время работы будет $\mathcal{O}(nmH)$, что подходит под данные ограничения. В подгруппах с меньшими ограничениями можно было слегка упростить или менее эффективно реализовать ту же самую логику.

Группы 6, 7

Для решения данных подзадач можно было использовать `std::bitset` для хранения и быстрого пересчета достижимых значений силы за время $\mathcal{O}\left(\frac{nmH}{64}\right)$, что укладывается в ограничения по времени в данных подзадачах. Операция сражения — это просто битовый сдвиг битсета вправо, а применение артефакта — сдвиг влево, плюс после дополнительно учесть действие тех артефактов, остаточная сила которых уходит в минус.

Наблюдение

Заметим, что если хранить достижимую силу не просто как силу, а как недостоющую до максимума силу, то есть хранить $x' = H - x$, то тогда операция применения артефакта эквивалентна следующему переходу: $x' \rightarrow |x' - h_i|$. Это работает и для случая неперевышения H , и для случая, когда артефакт работает сначала в плюс, потом в минус.

Тогда так же заметим, что все переходы будут эквивалентны взятию остатка по модулю $\text{gcd}(h_1, \dots, h_n)$ по теореме Безу, так как все $h_i \leq H$ и после применения артефакта сила не опускается ниже 0 (при $x = H$ и $h_i = H$ нет смысла применять артефакт).

Тогда мы можем считать, что у нас есть ровно один артефакт, сила которого равна $D = \text{gcd}(h_1, \dots, h_n)$. Более того, достижимую силу можно считать по этому модулю, потому что с таким артефактом можно получить любое число от 0 до H с тем же остатком по модулю D .

Группа 9

Если применить выше описанные размышления и алгоритм для подгрупп 3 – 5, мы можем получить время работы $\mathcal{O}(n \log H + mH)$.

Полное решение

Будем использовать наблюдение выше и алгоритм для подгрупп 3 – 5, но теперь достижимость снова будем хранить в `std::bitset`. Тогда при сражении со Слугой необходимо научиться обрабатывать несколько случаев:

- Если $\gcd(h_1, \dots, h_m) + a_i \leq H$, то вся достижимая сила просто циклически сдвинется после сражения со Слугой. Теперь необходимо обновить достижимую силу с учетом фазы отдыха. Так как мы храним достижимую силу в диапазоне $[0; \gcd(h_1, \dots, h_n))$, то для любой силы x и артефакта i верно $x \leq h_i$.

Тогда мы так же можем заменить $x \rightarrow -x \bmod \gcd(h_1, \dots, h_n)$, то есть наложить перевернутый битсет к состояниям достижимости. Данная операция даст нам все достижимые состояния силы после сражения со Слугой, и тогда необходимо будет вывести максимальное значение $H - x$, то есть минимальное x , хранимое в битсете;

- Если $\gcd(h_1, \dots, h_m) + a_i > H$, то несколько крайних значений силы нельзя использовать для битвы, так как они будут не больше, чем a_i , тогда можно занулить граничные значения в битсете, сделав сдвиг битсета на соответствующую величину, после чего провести сражения, как будто $\gcd(h_1, \dots, h_m) + a_i \leq H$.

Тогда время работы данного алгоритма будет равно $\mathcal{O}\left(n \log H + \frac{mH}{64}\right)$, что укладывается в ограничения по времени.