АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА ПОЗНАЙ ЧИСЛОТО

Подзадача 1

С въпрос [0, N) може да определим дали програмата журито е започнало да лъже. С помощта на това наблюдение може да решим подзадача 1. Започваме да питаме за всяко число само за себе си (число y, интервал [y, y + 1)). Ако журито не лъжеше, питаме за числата 0, 1, 2, …, N - 1 и сме готови. Тъй като журито започва да лъже в някой момент, достатъчно е да питаме за [0, N) след всеки въпрос, за да знаем дали отговора е бил истина или лъжа. Общия брой въпроси е 2 * N.
Може да спрем да питаме за [0, N) след като сме сигурни, че журито е започнало да лъже.

Подзадача 2

Можем да използваме подобна идея за подзадача 2. Вместо да търсим линейно върху целия интервал, може да правим двоично търсене с допълнителни въпроси, за да определим дали последния отговор е бил истина или лъжа. В момента, в който започваме отговора на [0, N) е лъжа, трябва да "изхвърлим" последния отговор, защото не знаем дали сме започнали да лъжем на този въпрос или на предния. Общия брой въпроси ще бъде 2 * log N

Следващи подзадачи

Анализ на играта

За да направим прогрес по останалите подзадачи, трябва да анализираме играта по-дълбоко. Нека игнорираме факта, че питаме за циклични интервали за момента и да направим по-общ анализ. Отначало намисленото число е x в множество S = {0, 1, 2, …, N - 1}. Да допуснем, че питаме за множество A. Ако отговора е бил истина, може да заместим S с A и ще бъдем в същата ситуация занапред. Ако отговора е бил лъжа, числото ще е в допълнението на A до S (да го наречем множество B. A U B = S) и всички останали отговори ще бъдат лъжа.
Важно: новата ситуация наподобява началната ситуация!
Нека с G(A, B) да означим ситуацията, в която знаем, че или числото на журито е в множество A и следващите отговори следват правилата от началото на играта, или числото е в множество B и всички следващи отговори ще бъдат лъжа. От тук нататък считаме A и B за непресичащи се множества. Началната ситуация е G(S, 0).
Нека следващия въпрос, който задаваме е множеството $C = A_1 \cup B_1$, където $A = A_1 \cup A_2$ е разбиване на A в две множества, $B = B_1 \cup B_2$ е разбиване на B. Да допуснем, че отговора на множеството C е "да". Ако журито казва истината, тогава намисленото число е в множеството A_1. Ако журито лъже, тогава числото е в $A_2 \cup B_2$. Тоест, влизаме в ситуация $G(A_1, A_2 \cup B_2)$. Ако отговора на журито е "не", с подобни разсъждения стигаме до ситуация $G(A_2, A_1 \cup B_1)$. Ако достигнем ситуация G(A, B), в която броят на числата в A U B е точно един, тогава знаем че това число е отговора на задачата.

Динамично програмиране

Ако позволяваме въпроси с произволни множества (а не задължително с циклични интервали), тогава броят стъпки до отгатването на числото зависи само от големината на множествата A и B, а не от числата в множествата.
Нека с F(a, b) да означим броят въпроси, които са необходими в най-лошия случай да познаем числото ако тръгнем от G(A, B), където a и b са размерите на множествата A и B. Можем да намерим стойността на F() с динамично програмиране.
Стойността F(a, b) е равна на $1 + \max\{ F(a_1, a_2 + b_2), F(a_2, a_1 + b_1) \}$ за някои a_1, a_2, b_1, b_2, такива че $a = a_1 + a_2$, $b = b_1 + b_2$. Ако пробваме всички възможни стойности, за да намерим a_1, a_2, b_1, b_2 които дават най-малка стойност на F(a, b) ще може да изчислим F() с $O(N^4)$ стъпки. Също така, ще знаем размера на множествата A_1, A_2, B_1, B_2, с които да направим въпроса от стъпка G(A, B).

Циклични интервали

Да се върнем към цикличните интервали - ако можем да смятаме F(), може ли да генерираме въпроси $C = A_1 \cup B_1$, с произволен размер на A_1, B_1?
Да видим кои са възможните преходи между състояния на G:
$G(A, B) \rightarrow G(A_1, A_2 \cup B_2)$ или $G(A, B) \rightarrow G(A_2, A_1 \cup B_1)$. Винаги A-множеството е подмножество на A-множеството от предишната стъпка. Нека поддържаме A-множеството в G(A, B) като интервал от числа $[L_A, R_A)$. Ако искаме да генерираме множества A_1, B_1 с размери a_1, b_1, можем да започнем интервала $C = [L_C, R_C)$ от
$L_C = R_A - a_1$ и да продължим да разтягаме интервала “надясно” докато не вземем b_1 елемента от множеството B. Ако стигнем N и все още нямаме достатъчно елементи от B, можем да вземем цикличен интервал, с $R_C < L_C$ (да вземем елементи от началото на [0, N)). С този избор на C ще питаме за елементи, които не са нито в A, нито в B като знаем, че сме в ситуация G(A, B). Но от досегашните отговори знаем, че намисленото число не е измежду тези елементи, така че добавянето им към множеството C няма да повлияе на отговора.

Подзадача 3

Можем да разпишем стойности на F() за малки a, b и ще забележим, че отговора почти винаги идва от $a_1 = a / 2$, $b_1 = b / 2$ (целочислено деление, закръгляне надолу). Ако напишем решение, което генерира съответни множества A_1, B_1 с размер половината на размера на A, B на всяка стъпка, ще получим решение на подзадача 3. Необходимо е да поддържаме G(A, B). Множеството A на всяка стъпка ще е един интервал, а множеството B е обединение на няколко интервала от числа. Броят на интервали в B е по-малък от броят на въпросите, които сме задали досега (48 в тази подзадача), така че всяко представяне на интервали би свършило работа. Също така, трябва да може да генерираме въпроси C, но тъй като броят на интервали в представянето на B е малък, това не е проблем.

Подзадача 4

Можем да смятаме F динамичното програмиране описано по-горе със сложност $O(N^4)$. Това е достатъчно да решим подзадача 4. Тук можем да представяме множествата по какъвто и да е начин, защото имаме по-малко от 100 числа.

Подзадача 5

Нека разгледаме стойностите на F по-внимателно:

b \ a	0	1	2	3	4	5	6	7	8	9
0	ϟ	0	3	5	5	6	6	6	6	7
1	0	2	4	5	5	6	6	6	6	7
2	1	3	4	5	5	6	6	6	6	7
3	2	3	4	5	5	6	6	6	6	7
4	2	3	4	5	5	6	6	6	6	7
5	3	4	4	5	5	6	6	6	6	7
6	3	4	4	5	5	6	6	6	6	7
7	3	4	5	5	5	6	6	6	6	7
8	3	4	5	5	5	6	6	6	6	7
9	4	4	5	5	6	6	6	6	7	7

Table 1

Стойностите във всеки ред и колона растат бавно. Нека компресираме таблицата, като генерираме функция H(a, k) - най-голямата стойност на b, такава че $F(a, b) \leq k$.

Ако $a = u + v$, $b = b_u + b_v$, $F(a, b) \leq 1 + \max\{F(u, v + b_v), F(v, u + b_u)\}$, и равенство се постига за някои u, v, b_u, b_v. Нека $F(a, b) = k$. $F(u, v + b_v) \leq k - 1$, $F(v, u + b_u) \leq k - 1$.

Обратно, ако имаме $H(u, k - 1) = v + b_v$, $H(v, k - 1) = u + b_u$, получаваме

$F(u + v, b_u + b_v) \leq 1 + k - 1 = k$

$H(u + v, k) \leq b_u + b_v = H(u, k - 1) + H(v, k - 1) - (u + v)$

Тоест, получаваме рекурентна зависимост

$H(a, k) = \max\{ H(u, k - 1) + H(v, k - 1) \} - a$, за някои u, v:

$u + v = a$, $u \leq c(v, k - 1)$, $v \leq c(u, k - 1)$.

Да разгледаме стойностите на H в таблица:

k \ a	0	1	2	3	4	5	6	7	8	9
0	1	0	×	×	×	×	×	×	×	×
1	2	0	×	×	×	×	×	×	×	×
2	4	1	×	×	×	×	×	×	×	×
3	8	4	0	×	×	×	×	×	×	×
4	16	11	6	×	×	×	×	×	×	×
5	32	26	20	14	8	×	×	×	×	×
6	64	57	50	43	36	29	22	15	8	×
7	128	120	112	104	96	88	80	72	64	56
8	256	247	238	229	220	211	202	193	184	175
9	512	502	492	482	472	462	452	442	432	422

Table 2

X означава, че F(a, b) > k за всяко b. Можем да видим, че всеки ред k образува аритметична прогресия, която започва от 2^k и намалява с k + 1. Кога аритметичната прогресия спира? Нека $H(m_k, k)$ е последният ненулев елемент на ред k. От рекурентната зависимост за H, виждаме че има $u \leq v$ такива, че
$m_k = u + v$, $v \leq H(u, k - 1)$, и $u \leq H(v, k - 1)$
След опити с малки числа, виждаме, че числата $H(s_{k-1}, k-1)$, за които $s_{k-1} \leq H(s_{k-1}, k-1)$ са важни: полагаме $u = v = s_{k-1}$ и виждаме, че $m_k \geq 2s_{k-1}$. Равенство се постига за $3 \leq k \leq 9$.
Винаги ли се постига равенство?
Ако $v \leq s_{k-1}$, тогава $mk = u + v \leq 2s_{k-1}$. Ако $v > s_{k-1}$, тогава
$m_k = u + v \leq H(v, k-1) + v \leq H(s_{k-1} + 1, k-1) + s_{k-1} + 1$ (тъй като $H(w, k) + w$ е строго намаляваща функция на w - спомнете си, че $H(., k)$ е аритметична прогресия)
$\leq 2s_{k-1} + 1$ (тъй като $s_{k-1} + 1 > s_{k-1}$ ни дава $H(s_{k-1} + 1, k-1) < s_{k-1} + 1$ по дефиниция на s_{k-1}).
Тоест, единствения начин да имаме $m_k > 2s_{k-1}$ е ако
$H(s_{k-1} + 1, k-1) = s_{k-1}$, $u = s_{k-1}$, $v = s_{k-1} + 1$, и $m_k = 2s_{k-1} + 1$.

Напомняме, че $H(a, k) = 2^k - a(k + 1)$, за $a \leq m_k$.
Можем да докажем по индукция точни формули за s и m:

$s_k = 2^k / (k + 2)$
(целочислено деление - следва директно от дефиницията на s_k и аритметичната прогресия на H(a, k)).
$m_k = 2 * s_{k-1}$, ако $s_{k-1} \neq H(s_{k-1} + 1, k-1)$
$m_k = 2 * s_{k-1} + 1$, ако $s_{k-1} = H(s_{k-1} + 1, k-1)$

За ограниченията дадени в задачата, $m_k = 2 * s_{k-1}$ винаги е вярно.
Това дава формулата за F(a, b): най-малкото цяло положително число k, такова че
$a \leq m_k$ и $a(k + 1) + b \leq 2^k$

Как да играем играта?

Като изчислим k = F(a, b), трябва да изчислим разбиване $a = a_1 + a_2$, такова че $H(a_1, k - 1) \geq a_2$ и $H(a_2, k - 1) \geq a_1$. Тогава, ще можем да намерим $b = b_1 + b_2$, такива че $H(a_1, k - 1) \geq a_2 + b_2$ и $H(a_2, k - 1) \geq a_1 + b_1$.
От анализа по-горе следва, че можем да вземем a_1 и a_2 близки до a/2.
Тоест, $a_1 := a/2$ и $a_2 := a - a_1$ работи. Ако a е четно, тогава $a_1 = a_2$ и $b_1 = b / 2$ работи.
Когато a е нечетно, не е трудно да се уверим, че b1 = (b - k + 1) / 2 работи.

Ако заменим изчисляването на F() с описаните резултати, получаваме решението на подзадача 5.

Автори: Николай Белухов, Йордан Чапъров
Решение: Николай Белухов, Йордан Чапъров