AGC #016 Editorial

writer : sugim48

2017 年 6 月 19 日

For International Readers: English editorial starts on page 5.

A: Shrinking

英小文字 c をひとつ固定します．最終的に s が単一の文字 c のみからなるようにしたいとします．すると，各操作では c を優先的に選べばよいです．よって，シミュレーションによって必要な操作回数が求まります．以上より，c を `a` から `z` まで全探索し，操作回数の最小値を求めればよいです．

B: Colorful Hats

すべての猫の色の種類数を A とします．自分と同じ色の猫が他にいないような猫を alone な猫と呼ぶことにします．このとき，猫 i が alone ならば，$a_i = A - 1$ です．また，猫 i が alone でないならば，$a_i = A$ です．よって，a_i の最小値と最大値の差が 2 以上ならば，答えは `No` です．以降は，a_i の最小値と最大値の差が 0 の場合，1 の場合に分けて説明します．

まず，a_i の最小値と最大値の差が 0 の場合，すなわち a_i がすべて同一の場合を説明します．この場合，猫はすべて alone であるか、すべて alone でないかのどちらかです．すべて alone の場合，$A = N$ より $a_i = N - 1$ です．また，すべて alone でない場合，$2A \leq N$ より $2a_i \leq N$ です．よって，$a_i = N - 1$ または $2a_i \leq N$ のどちらかが成り立つならば `Yes` で，そうでないならば `No` です．

次に，a_i の最小値と最大値の差が 1 の場合を説明します．この場合，A は a_i の最大値です．また，$a_i = A - 1$ ならば猫 i は alone で，$a_i = A$ ならば猫 i は alone ではないです．alone な猫の匹数を x，alone でない猫の匹数を y とします．すると，$x < A$ かつ $2(A - x) \leq y$ ならば `Yes` で，そうでないならば `No` です．

C: Subrectangles

H が h の倍数であり，かつ W が w の倍数である場合，答えは明らかに `No` です．この場合，$H \times W$ の長方形は $h \times w$ の部分長方形たちに分割できるので，部分長方形がすべて負ならば，全体の長方形も負になってしまうからです．逆に，H が h の倍数でないか，W が w の倍数でない場合，答えは `Yes` です．以降は，解の構成法を説明します．

W が w の倍数でない場合を説明します．さらに，$H = h = 1$ とします．この場合に解が構成できれば，その解を縦に H 行分繰り返せば，H や h が一般の場合の解となります．解を $a = (a_1, a_2, \ldots, a_W)$ とします．

また，この累積和を $s = (s_0, s_1, \ldots, s_W)$ とします．すなわち，$s_0 = 0$ かつ $s_i = s_{i-1} + a_i$ です．a の代わりに s を構成することにします．解が条件を満たすためには，$s_W > 0$ かつ $s_i > s_{i+w}$ が必要です．W が w の倍数でないので，これは常に構成できます．s が構成できれば，$a_i = s_i - s_{i-1}$ として a を構成できます．

D: XOR Replace

まず，操作をよく観察してみます．$x = a_1 \oplus a_2 \oplus \cdots \oplus a_N$ とします．今，a_i を x に置き換えたとします．すると，全要素の XOR は

$$\begin{aligned} & a_1 \oplus \cdots \oplus a_{i-1} \oplus x \oplus a_{i+1} \oplus \cdots \oplus a_N \\ = & (a_1 \oplus \cdots \oplus a_{i-1} \oplus a_{i+1} \oplus \cdots \oplus a_N) \oplus (a_1 \oplus \cdots \oplus a_N) \\ = & a_i \end{aligned}$$

となります．これは，「全要素の XOR」と a_i を入れ替えたと見ることができます．そこで，仮に $a_0 = a_1 \oplus a_2 \oplus \cdots \oplus a_N$ とすると，操作は「a_0 と a_i $(1 \leq i \leq N)$ を入れ替える」のと等価です．同様に，仮に $b_0 = b_1 \oplus b_2 \oplus \cdots \oplus b_N$ とすると，この問題の目標は，「a_0 と a_i $(1 \leq i \leq N)$ を入れ替える」という操作を繰り返し，$(a_0, a_1, \ldots, a_N)$ を $(b_0, b_1, \ldots, b_N)$ に一致させる，と言い換えることができます．以降は，この言い換えのもとで説明をします．

まず，$(a_0, a_1, \ldots, a_N)$ と $(b_0, b_1, \ldots, b_N)$ が多重集合として一致していなければ，目標は達成不可能です．逆に一致していれば，目標は達成可能です．

(最短とは限らない) 操作列によって，$(a_0, a_1, \ldots, a_N)$ が $(b_0, b_1, \ldots, b_N)$ へ変わったとします．このとき，各 i $(1 \leq i \leq N)$ について，$a_i = x_i^0 \to x_i^1 \to \cdots \to x_i^{(t_i)} = b_i$ と変わっていったとします．ここで，次のようなグラフを考えます．$(a_0, a_1, \ldots, a_N)$ を (多重ではない) 集合としたものを頂点集合とします．また，各 i $(1 \leq i \leq N)$ ごとに，有向辺 $(x_i^0 \to x_i^1)$, $(x_i^1 \to x_i^2)$, ..., $(x_i^{(t_i-1)} \to x_i^{(t_i)})$ を張ります．このグラフは，b_0 を始点，a_0 を終点とするオイラーグラフであることが分かります．逆に，任意の $b_0 \to a_0$ オイラーグラフは，何らかの操作列に対応することが分かります．以上より，最短の操作列を求めるためには，$a_i = x_i^0 \to x_i^1 \to \cdots \to x_i^{(t_i)} = b_i$ をうまく決め，得られるグラフが $b_0 \to a_0$ オイラーグラフであるという条件を満たしつつ，その辺数を最小化すればよいです．

まず，各 i $(1 \leq i \leq N)$ について，$a_i = b_i$ の場合 $a_i = x_i^0 = b_i$ とし，$a_i \neq b_i$ の場合 $a_i = x_i^0 = x_i^1 = b_i$ としてみます．このグラフの辺数は明らかに下界です．よって，このグラフが $b_0 \to a_0$ オイラーグラフならば，この辺数が答えです．このグラフの特徴を観察してみます．$(a_0, a_1, \ldots, a_N)$ と $(b_0, b_1, \ldots, b_N)$ が多重集合として一致していることに注目します．すると，$a_0 = b_0$ の場合，全頂点で (入次数) = (出次数) です．また，$a_0 \neq b_0$ の場合，頂点 b_0 では (入次数) + 1 = (出次数) であり，頂点 a_0 では (入次数) = (出次数) + 1 であり，それ以外の頂点では (入次数) = (出次数) です．よって，$b_0 \to a_0$ オイラーグラフであるための必要十分条件は，始点 b_0，終点 a_0，辺集合がすべて単一の連結成分に属していることです．

下界のグラフが $b_0 \to a_0$ オイラーグラフでない場合を考えます．例えば，$(a_0, a_1, a_2, a_3, a_4) = (15, 1, 2, 4, 8)$, $(b_0, b_1, b_2, b_3, b_4) = (15, 2, 1, 8, 4)$ の場合を考えます．下界のグラフは図 1 です．このグラフは，(始点 / 終点も合わせて) 連結成分が 3 個になっています．そこで，$a_1 = 1 \to 15 \to 2 = b_1$, $a_3 = 4 \to 2 \to 8 = b_3$ と変えてみます．すると，グラフは図 2 へ変わります．このグラフは，始点 b_0，終点 a_0，辺集合がすべて単一の連結成分に属しており，$b_0 \to a_0$ オイラーグラフになっています．このように，下界のグラフの辺数が e，連結成分数が c の場合，答えは $e + c - 1$ となります．

図 1

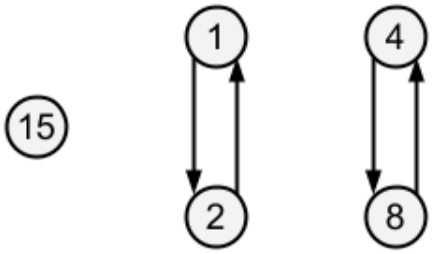
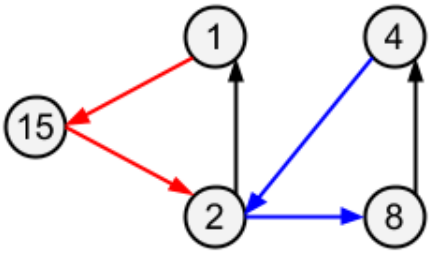


図 2



E: Poor Turkeys

まず，鳥 v が単独で生き残る可能性があるかを判定します．集合 S^i を，男性 1, 2, ..., i が行動を終えた時点で生き残っているべき鳥の集合とします．まず，$S^M = \{v\}$ です．$i = M, M-1, ..., 1$ の順に男性 i の行動を考え，S^i から S^{i-1} を更新していきます．とりあえず，$S^i \supseteq S^{i-1}$ なので，$S^i = S^{i-1}$ とします．$x_i \in S^{i-1}$ かつ $y_i \in S^{i-1}$ の場合，鳥 v は単独で生き残ることはありません．ここで処理を打ち切ります．$x_i \in S^{i-1}$ かつ $y_i \notin S^{i-1}$ の場合，S^i に y_i を追加します．$x_i \notin S^{i-1}$ かつ $y_i \in S^{i-1}$ の場合，S^i に x_i を追加します．$x_i \notin S^{i-1}$ かつ $y_i \notin S^{i-1}$ の場合，何もしません．以上の処理が最後まで終われば，鳥 v は単独で生き残る可能性があると判定できます．具体的には，男性 i は鳥 x_i, y_i のうち，S^i に含まれない方を選んで食べればよいです (以上の処理が最後まで終わったことから，これは常に可能です)．後の処理のため，S^0 を S_v として保存しておきます．

それぞれ単独で生き残る可能性がある鳥 u, v のペアについて，同時に生き残る可能性があるかを判定します．結論から述べると，鳥 u, v のペアは同時に生き残る可能性があるための必要十分条件は，$S_u \cap S_v = \emptyset$ です．もし，$S_u \cap S_v = \emptyset$ ならば，男性 i は鳥 x_i, y_i のうち，S_u^i にも S_v^i にも含まれない方を選んで食べていくことで，鳥 u, v のペアは同時に生き残ります．S_u および S_v の構成により，鳥 x_i, y_i がともに S_u^i にも S_v^i にも含まれることはありえません．逆に，もし $S_u \cap S_v \neq \emptyset$ ならば，S_u および S_v の構成により，鳥 x_i, y_i がともに S_u^i にも S_v^i にも含まれるような男性 i が存在することが示せます．

全体の時間計算量は $O(NM + N^3)$ です．

F : Games on DAG

まず，グラフ G' がひとつ固定されたときに勝敗を判定する方法を考えます．これは各頂点の Grundy 数を計算し，頂点 1, 2 の Grundy 数が等しいか見ることで判定できます．次に，各頂点の Grundy 数がひとつ固定されたときに，それと整合するグラフ G' が何通りあるか数える方法を考えます．今，ある頂点 v を固定します．v を始点とする辺集合の on / off の組合せが何通りか数えます．v の Grundy 数を x とします．頂点集合を Grundy 数ごとに分割し，順に $S_0, S_1, ...$ とします．まず，各 $y = 0, 1, \ldots, x-1$ について，v から S_y への辺集合のうち少なくとも 1 本は on でなければなりません．次に，v から S_x への辺集合はすべて off でなければなりません．最後に，各 $y = x+1, x+2, \ldots$ について，v から S_y への辺集合の on / off は自由です．以上より，v を始点とする辺集合の on / off の組合せが何通りか数えられました．すべての頂点についてこの値の積を計算すれば，G' が何通りあるか数えられます．

Alice が勝つような Grundy 数の組合せを全探索し，各組合せに対して上述の値の総和を計算すれば，答えが求まります．しかし，この方法では TLE になります．そこで，bit DP を用いて高速化します．Grundy 数が 0, 1, ... の順に頂点集合 S_0, S_1, ... を追加していくことを考えます．ただし，頂点 1, 2 は同時に追加されないようにして，Alice が勝つことを保証します．また，頂点集合 T を追加するときに，T 内の頂点を終点とする辺集合の on / off の組合せ数を掛けることにします．既に追加された頂点集合を S とし，今追加しようとしている頂点集合を T とし，$S \cup T$ の補集合を U とします．すると，S から T への辺集合の on / off は自由で，T 内の辺集合はすべて off でなければなりません．また，U から T への辺集合については，各 $v \in U$ に対し，v から T への辺集合のうち少なくとも 1 本は on でなければなりません．以上の遷移において，今の grundy 数は状態として持つ必要がないことに注意してください．

各 $v \in V$, $S \subseteq V$ の組に対して，v から V への辺集合の本数を前計算しておくことで，全体の時間計算量は $\mathrm{O}(N \cdot 3^N)$ となります．

AGC #016 Editorial

writer : sugim48

June 19, 2017

A: Shrinking

Fix a lowercase letter c, and suppose that we want to convert s such that all characters become c.

We simulate the operations. In each operation, we are asked to choose one of t_i and t_{i+1}. If both are c, t'_i must be c. If one of them is c, we should choose c as t'_i. If none of them is c, it doesn't matter which to choose.

This way, we can compute the minimum number of operations for a fixed c. Try all c from `a` to `z`, and the answer is the minimum of those numbers.

B: Colorful Hats

Suppose that there are A colors in total.

We call a cat *alone* if no other cat wears a hat of the same color as this cat. If the cat i is alone, $a_i = A - 1$, and otherwise $a_i = A$. Therefore, the difference between the maximum of a_i and the minimum of a_i must be at most one.

Case I. max = min

There are two cases: all cats are alone or all cats are not alone. In the former case, $a_i = N - 1$, and in the latter case, $2a_i \leq N$. Thus, if one of the above is true, the answer is `Yes`, otherwise the answer is `No`.

Case II. max - min = 1

In this case, A must be the maximum of a_i. Let x be the number of alone cats (the number of $A - 1$), and let y be the number of non-alone cats. The number of colors among alone cats is x, and the number of colors among non-alone cats is between 1 and $y/2$. Thus, if $x+1 \leq A \leq x+y/2$, the answer is `Yes`, otherwise the answer is `No`.

C: Subrectangles

If H is a multiple of h and W is a multiple of w, the answer is `No`. We can divide the entire rectangle into disjoint $h \times w$ subrectangles. The sum in each subrectangle must be negative while the sum of the entire rectangle must be positive, which is obviously impossible.

Otherwise, we can prove that the solution always exists. Without loss of generality, we can assume that H is not a multiple of h.

The following construction works:

- If the row-number (0-based) of a cell is a multiple of h, write a positive number in the cell.
- Otherwise, write a negative number in the cell.

This way, the ratio of positive numbers in an $h \times w$ subrectangle is always $1/h$, and the ratio of positive numbers in the entire rectangle is strictly greater than that. Thus, if we properly choose a positive number and a negative number such that the sum in each subrectangle is only slightly smaller than zero, it satisfies the conditions. For example, we can use $1000(h-1)-1$ for positive cells and -1000 for negative cells.

D: XOR Replace

Let $x = a_1 \oplus a_2 \oplus \cdots \oplus a_N$. If we perform an operation and replace a_i with x, the new XOR of the entire sequence will be the following:

$$\begin{aligned} & a_1 \oplus \cdots \oplus a_{i-1} \oplus x \oplus a_{i+1} \oplus \cdots \oplus a_N \\ = \; & (a_1 \oplus \cdots \oplus a_{i-1} \oplus a_{i+1} \oplus \cdots \oplus a_N) \oplus (a_1 \oplus \cdots \oplus a_N) \\ = \; & a_i \end{aligned}$$

It means that the new XOR of the entire sequence is a_i. Let $a_0 = a_1 \oplus a_2 \oplus \cdots \oplus a_N$. The operation just swaps two elements a_0 and a_i!

Now, the problem can be reduced to the following:

> You are given two sequences $a_0, \ldots, a_N$ and $b_0, \ldots, b_N$. In each operation you can swap the 0-th element and the i-th element for some i. How many operations are required to convert a into b?

In case $a_0, \ldots, a_N$ and $b_0, \ldots, b_N$ are not the same as multisets, it is impossible to do this. Otherwise, this is always possible. As a special case, if the two sequences are the same, the answer is zero. We assume that the two sequences are distinct.

Fix a (not necessarily the shortest) sequence of operations that converts a into b. Let's construct a colored directed graph that (partially) describes this sequence of operations. If we perform an operation between the 0-th element and the i-th element when the 0-th element is x and the i-th element is y,

- Add an edge from vertex x to vertex y of color 0.
- Add an edge from vertex y to vertex x of color i.

If we consider only edges of color i, those edges must form an Eulerian path from vertex a_i to vertex b_i. Since each edge of color 0 corresponds to the reverse of some other edge, edges with nonzero colors must form an Eulerian path from vertex b_0 to vertex a_0. Thus, the graph with only nonzero colored edges will satisfy the following properties:

- The entire graph is an (Eulerian) path from b_0 to a_0.
- This graph can be decomposed into a path from a_1 to b_1, a path from a_2 to b_2, ..., a path from a_N to b_N. Here, if $a_i = b_i$, the i-th path can be empty.

The number of edges in this graph is the number of operations, so we want to find such a graph with minimum possible number of edges.

Now, construct a new graph G. Initially G is empty.

- For each $i (1 \leq i \leq N)$, if $a_i \neq b_i$, add two vertices a_i and b_i to G (in case they are not currently in the graph), and add an edge from a_i to b_i. Note that this graph may contain multiple edges.
- Add two vertices a_0 and b_0 to G, in case they are not added yet (even if $a_0 = b_0$).

It is clear that the number of edges in G gives the lower bound of the number of edges in a desired graph. Furthermore, since we want the graph to be Eulerian, the graph must be connected. In case there are c connected components in G, we need extra $c - 1$ edges to connect them (see the picture below). Therefore, if the number of edges and the connected components in G are e and c, respectively, the answer is $e + c - 1$.

Figure 1:

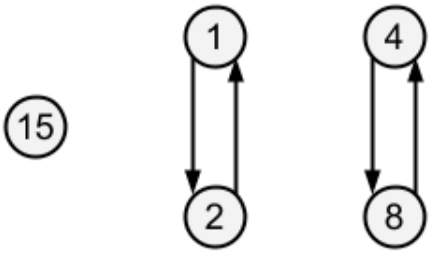
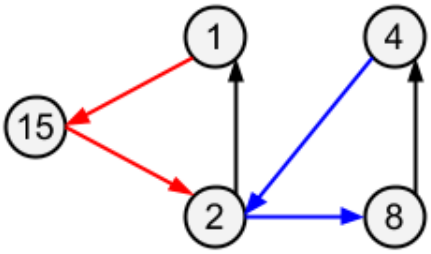


Figure 2:



Strictly speaking, we only gave a lower bound, and we must construct a sequence of length $e + c - 1$ explicitly. It is relatively easy to do this on paper, but it's lengthy to describe the details in words, and we leave details as an exercise for readers. Basically, in each step we can resolve one "mismatch", but for each connected component that doesn't contain a_0 or b_0, we need an extra move in the beginning.

E: Poor Turkeys

First, we generalize the question.

> Let S be a subset of turkeys, and let t be an integer. Is it possible that all turkeys in S survive after t steps?

There are four cases:

- If $x_t \in S$ and $y_t \in S$, we are sure that the answer is "No". Regardless of the state after $t-1$ steps, at least one of x_t and y_t will be eaten after the t-th step.
- If $x_t \in S$ and $y_t \notin S$, all turkeys in S survive after t steps if and only if all turkeys in $S \cup \{y_t\}$ survive after $t-1$ steps.
- If $x_t \notin S$ and $y_t \in S$, all turkeys in S survive after t steps if and only if all turkeys in $S \cup \{x_t\}$ survive after $t-1$ steps.
- If $x_t \notin S$ and $y_t \notin S$, all turkeys in S survive after t steps if and only if all turkeys in S survive after $t-1$ steps.

Let's determine if a single turkey v has a possibility to survive. First, start with a set $S = \{v\}$. In the order $i = M, M-1, ..., 1$, we update the set S, according to the rule we stated above. If we can complete the process without getting "No", the answer is "Yes". Let S_v be the set we get after this process, when we start with $S = \{v\}$. We can compute these sets in $O(NM)$.

Now, let's return to the original problem. How can we check if two turkeys u, v can survive at the same time? If we follow the rules above, we can understand that this is equivalent to the following conditions:

- The turkey u must has a possibility to survive.
- The turkey v must has a possibility to survive.
- $S_u \cap S_v = \emptyset$.

This part requires $O(N)$ time per each pair of turkeys.
The total complexity of this solution is $\mathrm{O}(NM + N^3)$.

F : Games on DAG

When a graph G' is given, we can determine the winner by checking whether the Grundy numbers of two vertices 1 and 2 are the same. In this task, we are asked to count the number of G' where the Grundy number of the two vertices become the same.

$O(Bell(N) \times poly(N))$ solution

We fix a sequence $(g_1, \ldots, g_N)$. How can we count the number of G' such that the Grundy number of the vertex i is g_i?

Let S_i be the set of vertices whose grundy numbers are i. Consider a vertex v, and let $x = g_i$. The Grundy number of v depends on the set of outgoing edges from v, and it must satisfy the fllowing conditions:

- For each $y = 0, 1, \ldots, x-1$, there must be at least one edge from v to S_y.
- There must not be any edges from v to S_x.
- For each $y = x+1, x+2, \ldots$, there is no restriction about the edges from v to S_y.

The number of sets of outgoing edges from v that satisfy the conditions above can be computed easily. By computing this number for all possible sequences $(g_1, \ldots, g_N)$, we can get the answer. This works in $O(Bell(N) \times poly(N))$, where $Bell(N)$ is the N-th Bell Number.

$O(3^N N)$ solution

However, the former solution was (at least was supposed to be) slow, so we want to improve it to an $O(3^N N)$ DP.

Let S be a set of vertices. S contains either both or none of the two vertices 1 and 2. Define $dp[S]$ as the number of ways to choose a subset of edges inside S, such that the Grundy numbers of the vertices 1 and 2 becomes the same.

One possible way is not to add edges at all inside this set. There is one way to do that. In other cases, we divide the set S to two disjoint non-empty sets T and U, such that the Grundy number of each vertex in T is nonzero and the Grundy number of each vertex in U is zero.

- T must contain either both or none of the two vertices 1 and 2.
- We must not add edges inside U.
- The edges from U to T are arbitrary.
- For each vertex in T, there must be at least one edge to U.
- And here is the key observation: the number of ways to choose the set of edges inside T is $dp[T]$!

With proper pre-calculation, this solution works in $\mathrm{O}(N \cdot 3^N)$.