AGC 035 解説

camypaper, chokudai, DEGwer

2019 年 7 月 14 日

For International Readers: English editorial starts on page 7.

A: XOR Circle

両隣のラクダが被っている帽子に書かれた数のビットごとの排他的論理和が自身の被っている帽子に書かれた数と等しいというのは、連続する 3 つの帽子に書かれた数のビットごとの排他的論理和が 0 ということと同値です。ここから円環上で連続する 4 つの数 $x_i, x_{i+1}, x_{i+2}, x_{i+3}$ に着目すると、$x_i \oplus x_{i+1} \oplus x_{i+2} = x_{i+1} \oplus x_{i+2} \oplus x_{i+3} = 0$ より、$x_i = x_{i+3}$ が成立することがわかります。

以上の事実から整理して考えると、条件を満たす被せ方が存在するのは以下の場合に限られます。これは $O(N \log N)$ で判定可能です。

- 全ての帽子に書かれた数が 0
- x が書かれた帽子が $\frac{2N}{3}$ 枚、0 が書かれた数が $\frac{N}{3}$ 枚
- $x \oplus y \oplus z = 0$ が成立するような 3 つの相異なる整数 x, y, z が書かれた帽子がそれぞれ $\frac{N}{3}$ 枚

B: Even Degrees

全ての辺はいずれかの頂点から出るように向き付けられるので、辺の数が奇数なら、全頂点の出次数を偶数にすることは不可能です。

逆に、辺の数が偶数ならそのような向き付けを以下のように構成することができます。まず、適当な根付き全域木を取り、残りの辺を適当に向き付けます。全域木の辺をボトムアップに決めていきます。頂点 v とその親を結ぶ辺の向きを決める際には、v の出次数が偶数になるようにします。このようにすれば、根以外の頂点の出次数は構成のルールより、根の出次数は全体の出次数の合計値が偶数であることより、それぞれ偶数であることが保証されます。

C: Skolem XOR Tree

頂点の番号ではなく頂点に割り当てられた整数のみを考えることにします。

N が 2 の冪の場合、明らかに答えは `No` となります。

それ以外の場合、答えは `Yes` です。例えば N が 3 以上の奇数の場合は以下の図 **??** のようにして構成可能です。

N が偶数の場合もほぼ同様です。先程の構成方法において、1 が書かれた紫色の頂点は $2, 3, \ldots, N-1$ が書かれた頂点と隣接することを利用します。例えば $N = 2018$ の場合、995 と書かれた頂点と 1024 と書かれた頂点につなぐことで $995 \oplus 1 \oplus 1024 = 2018$ と条件を満たすように構成できます。

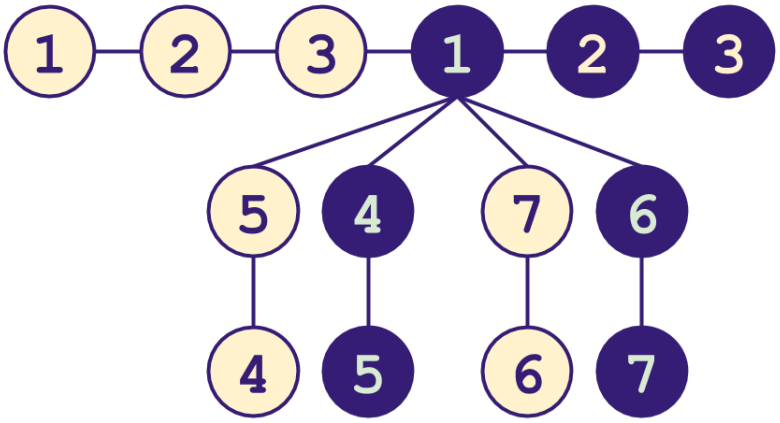

図 1　N=7 の場合

D: Add and Remove

操作を逆順に見ていきましょう。以下、カードを食べる操作の逆操作を「吐き出す」と呼ぶことにします。最初 2 枚のカードがあり、これらに書かれた整数は目的関数値に 1 倍されて足されます。

さて、現在 k 枚のカードがあるとして、これらに書かれた整数は順に目的関数値に $x_1, \ldots, x_k$ 倍され足されるとします。i 番目のカードと $i+1$ 番目のカードの間に吐き出され山に加えられるカードに書かれた整数は $x_i + x_{i+1}$ 倍されて足されることがわかります。こうして、山のどの位置にカードを吐き出して戻したかの情報から、最終的な列の何番目のカードに書かれた整数が何倍されて目的関数値に足されるかを復元することができます。

さて、$\texttt{DP}[l][x_l][r][x_r]$ を、入力で与えられたカード列の l, r 番目のカードが隣接していたタイミングが存在し、順に x_l, x_r 倍されて足されるときの、l 番目から r 番目までのカードに書かれていた整数が目的関数値に足される量の合計の最小値とします。この DP は、$\texttt{DP}[l][x_l][r][x_r] = \min_{l<m<r} \texttt{DP}[l][x_l][m][x_l + x_r] + \texttt{DP}[m][x_l + x_r][r][x_r] + (x_l + x_r)A_m$ として更新していくことができます。

この DP の状態数を見積もりましょう。組 (l, r) の個数は高々 $O(N^2)$ 個です。組 (x_l, x_r) としてありうるものの個数は、以下のように抑えることができます。

l, r 番目のカードの両方が現れるまでカードを吐き出していく操作を考えます。x_l, x_r の値に影響を与えるのは、$l' \leq l < r \leq r'$ を満たすような l', r' 番目のカードの間にカードが吐き出されるような場合だけです。この操作で吐き出されたカードは $l' < t \leq l$ または $r \leq t < r'$ を満たす t 番目のカードであり、x_l, x_r の値に関係するのは l 以下または r 以上のどちらの t が選ばれたかのみです。よって、カードを 1 枚吐き出す操作によって x_l, x_r の値の組の候補は高々 2 倍にしかならず、DP の状態数が全体として $O(2^n poly(n))$ で抑えられることがわかるので、適切な実装によりこの問題を解くことができます。

E: Develop

黒板に現れない整数の集合 S を特徴づけましょう。S の要素を頂点とし、$x, x-2$ の両方が S に含まれるなら x から $x-2$ に、また $x, x+K$ の両方が S に含まれるなら x から $x+K$ にそれぞれ辺を張ったグラフを考えます。S は、こうして作ったグラフにサイクルがないとき、またその時に限り、黒板に現れない整数の集合としてありうるものとなります。これは以下のようにして証明できます。グラフにサイクルが含まれるなら、操作の定義より、サイクルを構成する要素のうちどれかひとつは黒板に残ります。逆に、グラフにサイクルが含まれないなら、そのトポロジカル順に黒板から整数を消していくことができます。

さて、適切な考察により、S から構成されるグラフにサイクルが含まれないことは、S に $a, a-2, ..., b-K, b, b-2, ..., a-K, a$ という形のサイクルが含まれないことと同値であることが証明できます。以下、この特別な形のサイクルを含まないような S を数え上げることを考えます。

整数を小さい順に S に追加していく DP を考えます。整数 a を S に追加できない状況は、ある $b < a, a \neq b (\mod 2)$ が存在し、$a-2, a-4, ..., b-K, b, b-2, ..., a-K$ が S に含まれている状況のみです。このとき、a と偶奇の等しく S に含まれない最後の整数を x、a と偶奇の異なり S に含まれない最後の整数を y とすれば、$x < y$ が成り立ちます (そうでない場合、すでにサイクルが存在します)。

$x < y, a = t = x \neq y (\mod 2), a \leq t$ に対し $\mathtt{DP}[a][x][y][t]$ を、$a-1$ までの整数を S に入れるかどうかを決定し、x, y が上記のように設定され、a と偶奇の等しい整数を連続して t まで選ぶことができるような場合の数とします。$x > y$ に対しても $\mathtt{DP}[a][x][y][t]$ を同様に設定することで、DP を更新していくことができます。時間計算量は $O(N^4)$ ですが、$1/100$ オーダーの定数がつくので高速に動作します。

F: Two Histograms

$k_i+1=j, l_j=i$ なる (i,j) の組が存在すれば、(k_i,l_j) を (k_i+1,l_j-1) で置き換えることにより同じマス目を得ることができます。この操作によって k_i たちの合計は増加するので、できるマス目を変えることなしに、有限回の操作によってそのような組が存在しないように出来ます。こうしてできる $k_1,...,k_N,l_1,...,l_M$ の組を**標準表現**と呼ぶことにしましょう。最初の $k_1,...,k_N,l_1,...,l_M$ によって標準表現は一意に定まることが証明できます。

逆に、標準表現が異なるならばできるマス目も異なることが以下のように証明できます。

異なる標準表現を 2 つ取ってきて $k_1,...,k_N,l_1,...,l_M$ と $k'_1,...,k'_N,l'_1,...,l'_M$ としましょう。これらからできるマス目 A が等しいとして矛盾を導きます。$l_j \neq l'_j$ なる最小の j をとります。$l_j < l'_j$ として一般性を失いません。$j=1$ のとき、$k'_{l'_1}+1=1$ となり矛盾します。$j>1$ のとき、$A_{l'_j j}=1$ より $k_{l'_j} \geq j$ かつ $k'_{l'_j} < j$ ですが、これは $k'_{l'_j} \neq j-1$ と $l_{j-1}=l'_j$ に矛盾します。

さて、あとは標準表現の個数を数えればよいですが、これは $k_i+1=j, l_j=i$ なる (i,j) の組の集合に関する包除原理で線形時間で求めることができ、この問題を解くことができました。

AGC 035 Editorial

camypaper, chokudai, DEGwer

July 14, 2019

A: XOR Circle

The condition "the bitwise XOR of the numbers written on the hats on both adjacent camels is equal to the number on the hat on itself" is equivalent to the following: "the bitwise XOR of the numbers written on the three consecutive hats is 0." Now, let us focus on four consecutive numbers in the circle x_i, x_{i+1}, x_{i+2}, and x_{i+3}. From $x_i \oplus x_{i+1} \oplus x_{i+2} = x_{i+1} \oplus x_{i+2} \oplus x_{i+3} = 0$, it follows that $x_i = x_{i+3}$.

Further analysis based on these facts reveals that the condition can be satisfied only in the following cases, which can be determined in $\mathrm{O}(N \log N)$ time:

- Every hat has 0 written on it.
- There are $\frac{2N}{3}$ hats with x, and $\frac{N}{3}$ hats with 0.
- There are $\frac{N}{3}$ hats with x, $\frac{N}{3}$ hats with y, and $\frac{N}{3}$ hats with z, where x, y and z are distinct integers such that $x \oplus y \oplus z = 0$.

B: Even Degrees

Since every edge will be directed so that it goes out from one of the vertices, it is impossible for every vertex to have an even degree if there are an odd number of edges.

On the other hand, if there are an even number of edges, we can direct the edges so that every vertex has an even degree, as follows. First, we take some rooted spanning tree of the graph, and direct the remaining edges arbitrarily. Now, we will decide the direction for the edges in the spanning tree from bottom to top. When deciding the direction for the edge connecting a vertex v and its parent, we will make sure that the out-degree of v becomes even. This guarantees that every vertex other than the root will have an even degree, which in turn guarantees that the root will have an even degree, since the sum of the out-degrees of all the vertices is even.

C: Skolem XOR Tree

Below, we will refer to the vertices as their weights.

When N is a power of 2, the answer is obviously `No`.

In the other cases, it turns out that the answer is `Yes`.

When N is odd and not less than 3, we can, for example, construct a solution in the following manner:

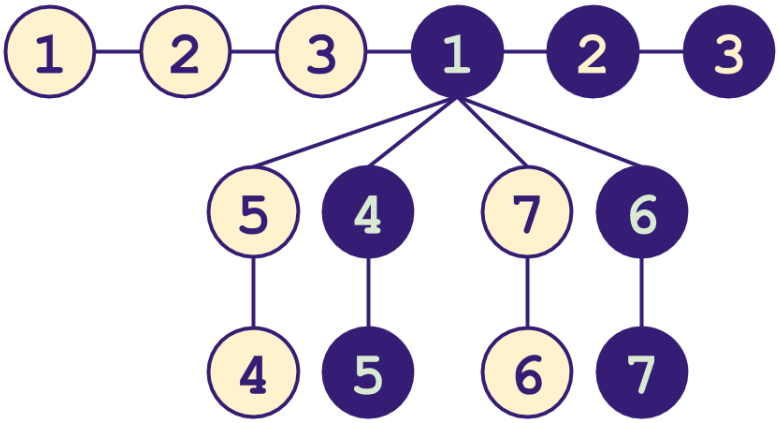

Fig.1 A solution for N=7

We can slightly modify this to handle the case where N is even. We only need to add two vertices with weight N. Let us take advantage of the fact that, in the graph above, the violet vertex with weight 1 is adjacent to the vertices with weight $2, 3, \ldots, N-1$. We can choose two of these vertices and connect them to the vertices with weight N. For example, if $N = 2018$, connect the vertices with weight N to the vertices with weight 995 and 1024 and we have $995 \oplus 1 \oplus 1024 = 2018$.

D: Add and Remove

Let us observe the operations in the reverse order. We say we *vomit* a card when we perform the reverse operation of eating a card. We have initially two cards, whose numbers will be added to the objective function after multiplied by 1.

Assume that now there are k cards remaining, whose numbers will be added to the objective function after multiplied by $x_1, \ldots, x_k$, respectively. We can see that, if we vomit a card between the i-th and $(i+1)$-th cards from the left, the number on it will be added to the objective function after multiplied by $x_i + x_{i+1}$. In this manner, from the information of the positions where we vomit the cards, we can restore the coefficients multiplied to the number written on each card in the objective function.

Let $\texttt{DP}[l][x_l][r][x_r]$ be the minimum possible sum of the contributions of the l-th to r-th cards in the input sequence to the objective function if there is a moment where these two cards are adjacent and they are multiplied by x_l and x_r before added to the objective function. We can update the table as follows: $\texttt{DP}[l][x_l][r][x_r] = \min_{l<m<r} \texttt{DP}[l][x_l][m][x_l + x_r] + \texttt{DP}[m][x_l + x_r][r][x_r] + (x_l + x_r)A_m$.

Let us estimate the number of states in this DP. The number of pairs (l, r) is at most $\mathrm{O}(N^2)$. Now we will estimate the upper bound of the number of pairs (x_l, x_r).

Consider vomiting cards until both the l-th and r-th cards appear. The values of x_l and x_r will be affected only when we vomit a card between the l'-th and r'-th cards such that $l' \leq l < r \leq r'$. Let such a card vomited be the t-th card. Then, $l' < t \leq l$ or $r \leq t < r'$ holds, and which of these two holds is the only factor that matters to the values of x_l and x_r. Thus, the number of possible pairs of the values of x_l and x_r is at most doubled when a card is vomited, so there are at most $\mathrm{O}(2^n poly(n))$ states in this DP, which can solve the problem if properly implemented.

E: Develop

Let us characterize the set S of integers that are not present on the blackboard. Consider a graph where the elements of S are the vertices and an edge is drawn from vertex x to $x-2$ when both x and $x-2$ are contained in S, and an edge is drawn from vertex x to $x+K$ when both x and $x+K$ are contained in S. S can be a set of the integers not present on the blackboard if and only if this graph does not contain a cycle. We can prove it as follows. If the graph contains a cycle, from the definition of the operation, it follows that one of the elements that is a part of the cycle will remain on the blackboard. On the other hand, if the graph does not contain a cycle, we can erase the integers from the blackboard in the topological order of the elements.

Then, we can also prove that the graph generated by S does not contain a cycle if and only if S does not contain a cycle of the form $a, a-2, ..., b-K, b, b-2, ..., a-K, a$, by appropriate consideration. Now, let us count the sets S which does not contain a cycle of this form.

Let us consider an approach by DP where we add integers to S in ascending order. The only situation where we cannot add an integer a to S is the situation where there exists an integer b such that $b < a, a \neq b (\mod 2)$ and $a-2, a-4, ..., b-K, b, b-2, ..., a-K$ are contained in S. Let x be the last integer with the same parity as a and not contained in S, and y be the last integer with the parity different from a and not contained in S. Then, $x < y$ holds (or there is already a cycle).

For integers x, y, a, t such that $x < y, a = t = x \neq y (\mod 2), a \leq t$, let $\texttt{DP}[a][x][y][t]$ be the number of choices for the integers up to $a-1$ whether they are contained in S such that the values of x and y are defined as above and we can choose at most t consecutive integers with the same parity as a. We will also define $\texttt{DP}[a][x][y][t]$ for the case $x > y$ similarly, and now we can update the DP tables. The time complexity of this approach is $\mathrm{O}(N^4)$, with a small constant factor that enables it to work fast enough.

F: Two Histograms

If there exists a pair (i, j) such that $k_i + 1 = j, l_j = i$, we can replace (k_i, l_j) with $(k_i + 1, l_j - 1)$ and still obtain the same grid. This operation increases the sum of k_i, so we can repeat this operation a finite number of times so that there will be no such pair, without changing the resulting grid. Let us call such a sequence obtained, $k_1, ..., k_N, l_1, ..., l_M$, the **regular expression**. We can prove that the regular expression for a fixed initial sequence $k_1, ..., k_N, l_1, ..., l_M$ is unique.

On the other hand, if two initial sequences have different regular expressions, the resulting grid will also be different, which we can prove as follows.

Let the two different regular expressions be $k_1, ..., k_N, l_1, ..., l_M$ and $k'_1, ..., k'_N, l'_1, ..., l'_M$. Assume that we obtain the same grid A from both of them and reach a contradiction. Let j be the minimum index such that $l_j \neq l'_j$. We assume $l_j < l'_j$ without loss of generosity. If $j = 1$, $k'_{l'_1} + 1 = 1$ and a contradiction is reached. If $j > 1$, from $A_{l'_j j} = 1$ it follows that $k_{l'_j} \geq j$ and $k'_{l'_j} < j$, which contradicts with $k'_{l'_j} \neq j - 1$ and $l_{j-1} = l'_j$.

Now, we just have to count the number of regular expressions. We can use the inclusion-exclusion principle on the number of pairs (i, j) such that $k_i + 1 = j, l_j = i$ and count them in linear time, so the problem is solved.