AGC032 解説

writer : many

2019 年 3 月 16 日

For International Readers: English editorial starts on page 8.

A : Colorful Subsequence

同じ文字列でも、異なる位置から作られたものは区別するため、$2^N - 1$ 通りのすべての部分列が区別されることになります。

条件より、同じ文字を 2 度使ってはいけないため、ある文字 c について colorful の条件を壊さないとり方は (c の出現回数 $+ 1$) となります (どれか 1 つを取るケース及び文字 c を一切取らないケース)

すべての文字 c についてのこのとり方の積を求め、空文字列の分の 1 を引いた数が答えとなります。

AGC032 解説

writer : many

平成 31 年 3 月 15 日

B : Reversi

まず、作れる列の連続する部分列であって、すべて同じ色の石からなるような極大なもの最初と最後の石の色は、最初の列から変化していないことがわかります (もし変化していたなら、その石の両側にそれと同じ色の石があるはずです)。逆に、この条件が満たされているような列は、同じ石の色の連続する部分列の両端についてそれぞれ操作を行うことで常に作ることができます。よって、この条件を満たす列を数えればよいです。

DP[i] を、最初から i 個の石たちの条件を満たすような塗り方の個数とします。DP[i] は、i 番目の石の色が $j+1$ 番目の石の色が等しく j 番目の石の色と異なるような j たちすべてに対する DP[j] の和として計算できます (ただし、便宜上 0 番目の石は他のどれとも異なる色で塗られていると考えます)。

このままでは時間計算量が $O(N^2)$ かかってしまい、間に合いません。そこで、各色に対し、$j+1$ 番目の石がその色で塗られており j 番目の石がその色で塗られていないような j に対する DP[j] の和を持っておき、DP 値が新しく計算されるたびにその都度足しこんでいけば、各遷移を $O(1)$ 時間で動くように実装でき、全体として $O(N)$ 時間でこの問題を解くことができます。

AGC032 解説

writer : many

平成 31 年 3 月 15 日

1 C : Differ by 1 Bit

P_i と P_{i+1} はちょうど 1bit だけ異なるという条件から、P_i の立っている bit の数の偶奇がわかります。特に、P_0 と P_{2^N-1} では、立っている bit の個数の偶奇は異なります。よって、A, B の立っている bit の個数の偶奇が等しい場合は、構成は不可能だとわかります。

以下、A, B の立っている bit の個数の偶奇が異なる場合に、必ず構成ができることを示します。帰納法で示します。まず、$N = 1$ のときは明らかに可能です。次に、ある K について、$N = K$ の場合に必ず構成ができると仮定します。ここで、$N = K + 1$ の問題を考えます。A と B で状態が異なる bit が少なくとも 1 つとれます。これが x bit 目だったとします。A, B の x bit 目を取り除いた、$(N-1)$ bit の数を考え、それぞれ A', B' とします。明らかに、A' と B' の立っている bit の個数の偶奇は等しいです。ここで、$(N-1)$ bit の数 C であって、A' 及び B' と立っている bit の数の偶奇が異なるものがとれます。（A' の 1 bit 目を反転させるなどすれば良いです）ここで、帰納法の仮定より、A' から始まって C で終わる長さ 2^{N-1} の順列 Q、及び C で始まって B' で終わる長さ 2^{N-1} の順列 R がとれます。ここで、条件を満たす順列 P を次のように構築できます。

- $i < 2^{N-1}$: P_i の x bit 目は A の x bit 目と同じにする。残りの $(N-1)$ bit は Q_i の対応する bit と同じにする。
- $2^{N-1} \leq i < 2^N$: P_i の x bit 目は B の x bit 目と同じにする。残りの $(N-1)$ bit は $R_{i-2^{N-1}}$ の対応する bit と同じにする。

こうしてできる順列 P が条件を満たすことは簡単に確認できます。

以上で構築が可能であることが示されました。上記の手順をそのまま実装すれば実際に順列を構築できます。実装にもよりますが、計算量は $O(N^2 2^N)$ や $O(N 2^N)$ です。

D : A Sequence of Permutations

順列 p, q に対して、pq でそれらの合成を表すことにします。つまり、pq は i 項目が p_{q_i} であるような順列を指します。この表記を用いると $f(p,q) = qp^{-1}$ となります。

そこで、最初のいくつかの項を書き出してみましょう。すると次のようになります。

- $a_1 = p$
- $a_2 = q$
- $a_3 = qp^{-1}$
- $a_4 = qp^{-1}q^{-1}$
- $a_5 = qp^{-1}q^{-1}pq^{-1}$
- $a_6 = qp^{-1}q^{-1}p^2q^{-1}$
- $a_7 = qp^{-1}q^{-1}pqpq^{-1}$
- $a_8 = qp^{-1}q^{-1}pqp^{-1}qpq^{-1}$
- ...

根気よく実験を続けることが重要です。書き下してみると、中央で同じ項がたくさん消えることが分かります。そこで、$a_i = A_iB_iA_i^{-1}$ という形で書き換えることを考えると、以下のようになります。

- $(A_1, B_1) = (id, p)$
- $(A_2, B_2) = (id, q)$
- $(A_3, B_3) = (id, qp^{-1})$
- $(A_4, B_4) = (q, p^{-1})$
- $(A_5, B_5) = (qp^{-1}, q^{-1})$
- $(A_6, B_6) = (qp^{-1}, q^{-1}p)$
- $(A_7, B_7) = (qp^{-1}q^{-1}p, p)$
- $(A_8, B_8) = (qp^{-1}q^{-1}p, q)$
- ...

$(A_7, B_7) = (qp^{-1}q^{-1}, p)$ とも書けますが、そうすると周期が見えづらくなってしまうので注意しましょう。これを見ると周期 6 を持つことが分かります。実際、(A, p), (A, q) から始めて 6 回進むと $(Aqp^{-1}q^{-1}p, p)$, $(Aqp^{-1}q^{-1}p, q)$ となるので、$qp^{-1}q^{-1}p$ を $\left[\frac{K-1}{6}\right]$ 回合成すれば A_K は求まります。$\{B_i\}$ は周期 6 であるので、適当にシミュレーションをすることで B_K は求まります。よって、この問題を $O(N)$ で解くことができました。

E: Snuke the Phantom Thief

「ちょうど K 個宝石を盗んでください」という条件を足すと一気に見通しがよくなるのがこの問題のキモです。

まずは 1 次元の場合、つまり $t_i = L, R$ の条件のみの場合を考えます。

「盗んだ宝石のうち、x 座標が a_i 以下の宝石が b_i 個以下」は、「盗んだ宝石を x 座標で sort した時、$b_i + 1$ 番目 (1-indexed) の宝石の x 座標が $a_i + 1$ 以上」と同値です。

そして盗む個数を決めると、「盗んだ宝石のうち、x 座標が a_i 以上の宝石が b_i 個以下」は、「盗んだ宝石を x 座標で sort した時、$K - b_i$ 番目の宝石の x 座標が $a_i - 1$ 未満」と同値となります。

要するに盗む宝石の条件は、ある整数 $L_1, L_2, ..., L_K$、$R_1, R_2, ..., R_K$ によって、「盗んだ宝石を x 座標で sort した時、i 番目の宝石の x 座標が L_i 以上 R_i 以下」と表せます。

明らかに $L_1 \leq L_2 \leq ... \leq L_K$, $R_1 \leq R_2 \leq ... \leq R_K$ として良いです。そうでないならば、つまり $L_i > L_{i+1}$ ならば $L_{i+1} = L_i$ として良いからです。

更に実はこの条件は、「各 i について、L_i 以上 R_i 以下から宝石を 1 個盗む」と言い換えられます。(ここの証明いりますか)

ここまで言い換えられれば、割り当て問題、つまり最小費用流に帰着することができます。1(始点) + K(x 座標制約) + N(宝石) + 1(終点) 頂点のグラフを作り、

- 始点から (x 座標制約) の頂点それぞれに重さ 0, 容量 1
- (x 座標制約) の i 番目の頂点からは、x 座標が $L_i \sim R_i$ の (宝石) それぞれに重さ 0, 容量 1
- (宝石) それぞれから終点に重さ=-(宝石の価値), 容量 1

の辺を貼り、このグラフで流量 K 流したものが求める答えです。

2 次元への対応も簡単です。グラフが 2 個できるので、片方をひっくり返して繋げれば良いです、つまり 1(始点) + K(x 座標制約) + N(宝石 x) + N(宝石 y) + K(y 座標制約) + 1(終点) のグラフを作り、

- 始点から (x 座標制約) にそれぞれ重さ 0, 容量 1
- (x 座標制約) の i 番目からは x 座標が $L_i \sim R_i$ の宝石に対応する (宝石 x) の頂点へ重さ 0, 容量 1
- (宝石 x) から (宝石 y) へは 重さ=-(宝石の価値), 容量 1
- 始点から (x 座標制約) にそれぞれ重さ 0, 容量 1
- (y 座標制約) の i 番目へ、 y 座標が $L_i \sim R_i$ の宝石に対応する (宝石 y) の頂点から重さ 0, 容量 1
- (y 座標制約) から終点へそれぞれ重さ 0, 容量 1

の辺を貼り、K 流せば良いです。負の重みの辺は、重みを 10^{15} 足しておき、最後に答えから $K \times 10^15$ 引けば非負の重みと見做すことができます。

計算量は V $= O(N)$, E $= O(N^2)$ のグラフへ容量 $O(N)$ 流すのを $O(N)$ 回行うので、$O(N^4)$ です。最短路を求める時に $O(ElogV)$ のダイクストラ法を使うと $O(N^4 logN)$ となりますが、これでも十分余裕を持って通るように調整したつもりです。

F: Walk on Graph

mod MOD における 2 の位数を ord と置きます。

今頂点 v に居て、ここまで通ってきた辺の本数の総数が mod ord で k であるような状態を (v,k) と書くことにします。

すると、与えられたグラフ上で頂点 s と頂点 t を結ぶ長さ L の辺があったとすると、(s,k) から $(t,k+1)$ へ長さ $2^k \times L$ で行くことが出来る、と思うことができます。

ここで、 (s,k) からスタートして、この辺を往復して頂点 s に戻ることを $ord \times MOD$ 回繰り返すと、再び (s,k) に戻り、かつ経路の長さの mod MOD における値も変化しないことが分かります。

よって、これは $(t,k+1)$ から (s,k) へ長さ $-2^k \times L$ で行くことができるということを示しています。

以下、ある頂点 s から長さ x と長さ y の辺が出ている場合に、ちょうど $3(x-y)$ だけ経路の長さを増やすことができることを示します。長さ x の辺は t 、長さ y の辺は u へのびていることにすると、

- (s,k) , $(t,k+1)$, $(s,k+2)$ と往復するときには $2^k \times x + 2^{k+1} \times x$ だけ経路の長さが増えます。
- (s,k) , $(u,k+1)$, $(s,k+2)$ と往復するときには $2^k \times y + 2^{k+1} \times y$ だけ経路の長さが増え、以上の議論より、これを逆にすると $(s,k+2)$ から (s,k) に $-2^k \times y - 2^{k+1} \times y$ だけ経路の長さを増やしていくことができます。

よって、結局 $3 \cdot 2^k \cdot (x-y)$ だけ経路の長さを増やして (s,k) に戻ってくることが出来ることがわかり、MOD は奇数なので、この動きを繰り返すことで $3(x-y)$ だけ経路の長さを増やすことができることが示せます。

さて、今は頂点 s に繋がる 2 辺について示しましたが、これを繰り返して適用することで、同じ連結成分に属する任意の 2 辺の差分に 3 を掛けた値だけ経路の長さを増やせることが分かります。すると、最終的に元の MOD と、与えられた全ての辺の値が mod M で等しいような最大の M (辺が 1 本しかない場合は $M = MOD$ とします) を取ってきて、mod $\gcd(MOD, 3M)$ で元の問題を解けば良い、ということがわかります。

よって、辺の長さを mod $3M$ で見ることができるので、 $aM + c$ と書くと、全ての辺に対して c は等しく、 a は $0,1,2$ のいずれかになります。以下、長さは全て $pM + q$ の形で表現します。 (p は常に $0,1,2$ のいずれかですが、q は M 以上になり得ます)

ここで、元の問題を後ろから見ることにすると、各クエリを以下のように言い換えることができます:

- 頂点 t から頂点 s までの経路で、長さが mod MOD で r になるようなものが存在するかしないか求めてください。ただし、経路の長さが a の状態で長さ L の辺を通ると経路の長さは $2a + L$ になります。

この問題は以下のように解くことができます:

- まず、今の経路の長さに c を足します。(最初の経路の長さは c であり、最終的に長さが mod MOD で $r+c$ であるような経路を探す問題になります。) (以下、c を足したものを経路の長さと呼びます。)
- こうすることで、現在の経路の長さが $pM+q$ の時に、長さ $aM+c$ の辺を通ることにすると、経路の長さは $(2p+a)M+2q$ になります。
- 最初の経路の長さは $0 \cdot M + c$ であると思うことができます。
- 経路の長さが $pM+q$ の時に長さ $aM+c$ の辺を往復することにすると、経路の長さは $(4p+3a)M+4q$、mod $3M$ で見ると $pM+4q$ になります。よって、 q は自由に 4 倍することができます。
- ここで、(v, p, par) を頂点 v にいて、経路の長さの M の係数が p であり、ここまで通ってきた辺の本数の mod 2 が par であるような状態とおき、互いに到達できるペアを Union-Find 上で繋ぎます。
- すると、$(t, 0, 0)$ と (s, p, par) が同じ連結成分に所属し、$pM + 2^{2N+par} \times c$ と $r+c$ が mod MOD で等しくなるような非負整数 N が存在するかが分かれば良い、ということになります。
- これは、結局 $2^x \times c$ が mod MOD で 何らかの値 num に合同になり、かつ mod 2 が par になるような x が存在するかがわかればよく、これは最初に高々 MOD 回ループを回すことで判定できます。

以上より、この問題を全体で $\mathcal{O}(MOD + (n+m+q)\ \alpha(n))$ 時間で解くことができました。

AGC032 Editorial

writer : many

March 16, 2019

A : Colorful Subsequence

Since we distinguish strings whose contents are the same but made from different positions in the original string, all $2^N - 1$ subsequences will be considered as different.

From the condition, we cannot use the same letter twice. Thus, for a letter c, we have (the number of occurrences of c) + 1 choices. (We either take one of those occurrences, or nothing.)

The answer is the product of these numbers for all letters c, minus 1 for the empty string.

B : Reversi

First, we can see that, in a sequence that can be made, the color of the stones at the both ends of a maximal contiguous subsequence of stones of the same color have not changed. (If the color of such stone has been changed, the two stones adjacent to that stone must both have the same color as that stone.) On the other hand, we can always make a sequence that satisfies above, by performing the operation for both ends of each maximal contiguous subsequence of stones of the same color. Thus, we can count the number of sequences satisfying this condition.

Let DP[i] be the number of ways of coloring the first i stones satisfying the condition. DP[i] can be computed as the sum of DP[j] such that the color of the i-th stone is the same as the $(j+1)$-th stone but different from the j-th stone (for convenience, assume that the 0-th stone is painted in the color different from all other stones).

This is not fast enough, since the time complexity is O(N^2). Now, for each color, let us maintain the sum of DP[j] such that the color of the i-th stone is the same as the $(j+1)$-th stone but different from the j-th stone. Every time we compute one value in the DP table, we add the result to the sum for that color. We can now find one value in the DP table in O(1) time, for a total of O(N) time.

C : Differ by 1 Bit

From the condition that P_i and P_{i+1} differ by exactly 1 bit, we can know the parity of the number of standing bits in P_i. Particularly, P_0 and P_{2^N-1} have different parities of the number of standing bits. Thus, if A and B have the same parity of the number of standing bits, the construction is impossible.

We will now show that, if A and B have different parities of the number of standing bits, the construction is always possible. The proof is done by induction. The construction is obviously possible when $N = 1$. Then, let us assume that the construction is always possible when $N = K$, for some K. Now, let us consider the case when $N = K + 1$. There is at least one bit position we can choose where A and B differ. Let this position be the x-th bit (from the right). Let A', B' be respectively $(N-1)$-bit numbers obtained by removing the x-th bit from A, B. Obviously, A' and B' have the same parity of the number of standing bits. Here, we can take a $(N-1)$-bit number C whose parity of the number of standing bits is different from that of A' and B'. (We can, for example, swap the first bit in A'.) Now, from the inductive assumption, we can take a permutation Q of length 2^{N-1} beginning with A' and ending with C, and a permutation R of length 2^{N-1} beginning with C and ending with B'. Then, we can construct a permutation R that satisfies the condition, as follows:

- $i < 2^{N-1}$: The x-th bit of P_i is the same as the x-th bit of A. The remaining $(N-1)$ bits are the same as the corresponding bits in Q_i.
- $2^{N-1} \leq i < 2^N$: The x-th bit of P_i is the same as the x-th bit of B. The remaining $(N-1)$ bits are the same as the corresponding bits in $R_{i-2^{N-1}}$.

We can easily verify that this permutation P satisfies the condition.

Therefore, it has been proved that the construction is always possible, and we can actually do it by implementing the procedure above. The time complexity is $\mathrm{O}(N^2 2^N)$ or $\mathrm{O}(N 2^N)$, depending on implementation.

D : A Sequence of Permutations

For permutation p and q, let pq denote their composition. That is, pq is a permutation whose i-th element is p_{q_i}. With this notation, $f(p,q) = qp^{-1}$.

Now, let us write out first several elements of our sequence. We get the following:

- $a_1 = p$
- $a_2 = q$
- $a_3 = qp^{-1}$
- $a_4 = qp^{-1}q^{-1}$
- $a_5 = qp^{-1}q^{-1}pq^{-1}$
- $a_6 = qp^{-1}q^{-1}p^2q^{-1}$
- $a_7 = qp^{-1}q^{-1}pqpq^{-1}$
- $a_8 = qp^{-1}q^{-1}pqp^{-1}qpq^{-1}$
- ...

Patience is important in this experiment. From this, we can see the same element disappearing many times. Now, let us rewrite it as $a_i = A_i B_i A_i^{-1}$. We get the following:

- $(A_1, B_1) = (id, p)$
- $(A_2, B_2) = (id, q)$
- $(A_3, B_3) = (id, qp^{-1})$
- $(A_4, B_4) = (q, p^{-1})$
- $(A_5, B_5) = (qp^{-1}, q^{-1})$
- $(A_6, B_6) = (qp^{-1}, q^{-1}p)$
- $(A_7, B_7) = (qp^{-1}q^{-1}p, p)$
- $(A_8, B_8) = (qp^{-1}q^{-1}p, q)$
- ...

Note that we could also say $(A_7, B_7) = (qp^{-1}q^{-1}, p)$, but that would make the period less noticeable. Now, it can be seen that we have a period of 6. In fact, six iterations from (A, p) and (A, q) result in $(Aqp^{-1}q^{-1}p, p)$, $(Aqp^{-1}q^{-1}p, q)$, so we can find A_K as a $\left[\frac{K-1}{6}\right]$-fold product of $qp^{-1}q^{-1}p$. For $\{B_i\}$, it has a period of 6, so we can find B_K by simulation. Thus, the problem is solved in $\mathrm{O}(N)$ time.

E: Snuke the Phantom Thief

The key is to add one more condition: "Snuke must steal exactly K jewels", which makes the problem much more approachable.

First, let us consider the one-dimensional version of the problem, that is, the version where there are only conditions with $t_i = L, R$.

The condition "Snuke can only steal at most b_i jewels whose x-coordinate is a_i or smaller" is equivalent to: "when the stolen jewels are sorted in ascending order of x-coordinate, the x-coordinate of the (b_i+1)-th jewel (1-indexed) must be $a_i + 1$ or larger".

Additionally, if we fix the number of jewels to steal, the condition "Snuke can only steal at most b_i jewels whose x-coordinate is a_i or larger" is equivalent to: "when the stolen jewels are sorted in ascending order of x-coordinate, the x-coordinate of the $(K - b_i)$-th jewel must be smaller than $a_i - 1$".

To sum it up, the conditions on jewels to steal can be represented as "when the stolen jewels are sorted in ascending order of x-coordinate, the x-coordinate of the i-th jewel must be between L_i and R_i (inclusive)" by some integers $L_1, L_2, ..., L_K$ and $R_1, R_2, ..., R_K$.

Obviously, we can assume $L_1 \leq L_2 \leq ... \leq L_K$ and $R_1 \leq R_2 \leq ... \leq R_K$. This is because, otherwise, that is, if $L_i > L_{i+1}$, it can be regarded as $L_{i+1} = L_i$.

In fact, we can further rephrase the condition as "for each i, Snuke must steal one jewel between the x-coordinates L_i and R_i". (The proof is left to the reader.)

Now, we can reduce the problem to the allocation problem, that is, the minimum-cost flow. Let us build a graph with 1(source) + K(constraints on x-coordinate) + N(jewels) + 1(sink) vertices and the following edges:

- From the sink to each of the x-coordinate constraint vertices: an edge of cost 0 and capacity 1
- From the i-th x-coordinate constraint vertex to each jewel vertex with x-coordinate between L_i and R_i: an edge of cost 0 and capacity 1
- From each jewel vertex to the sink: an edge of cost -(the jewel's value) and capacity 1

The answer can be found by sending a flow of K through this graph.

This can be easily extended to the two-dimensional version. Now we have two graphs, but we can flip one of them and connect it to the other.

That is, we build a graph with 1(source) + K(constraints on x-coordinate) + N(jewels x) + N(jewels y) + K(constraints on y-coordinate) + 1(sink) vertices and the following edges:

- From the sink to each of the x-coordinate constraint vertices: an edge of cost 0 and capacity 1
- From the i-th x-coordinate constraint vertex to each "jewel x" vertex with x-coordinate between L_i and R_i: an edge of cost 0 and capacity 1
- From "jewel x" vertex to "jewel y" vertex: an edge of cost -(the jewel's value) and capacity 1
- To the i-th y-coordinate constraint vertex from each "jewel y" vertex with y-coordinate between L_i and R_i: an edge of cost 0 and capacity 1

- From each y-coordinate constraint vertex to the sink: an edge of cost 0 and capacity 1

We can then send a flow of K through this graph. For edges with negative costs, we can add 10^{15} to those costs in advance to make them non-negative and subtract $K \times 10^{15}$ from the answer.

The time complexity is $\mathrm{O}(N^4)$, since we send a flow of $O(N)$ through a graph with $V = \mathrm{O}(N), E = \mathrm{O}(N^2)$, and we repeat it $\mathrm{O}(N)$ times. It becomes $\mathrm{O}(N^4 \log N)$ if you use Dijkstra's algorithm, which takes $\mathrm{O}(E \log V)$ time, to find the shortest paths, but we tried to set the constraints so that even this runs fast enough easily.

F: Walk on Graph

Let's see the path in the reverse order. The query asks the following:

- A person is standing at T with a black integer 0. When a person with a black integer x passes through an edge of length c, the black integer changes to $2x+c \bmod MOD$. Can the person reach S with a black integer R?

Let $[v, x]$ denote the state such that the person is at vertex v and the black integer is x. Let's make various observations:

- Suppose that there is an edge of length c between two vertices a and b. If we start with the state $[a, x]$ and repeat traversing this edge back and forth ($[a, x] \to [b, 2x + c] \to [a, 4x + 3c] \to [b, 8x+7c] \to \cdots$), we will eventually return to the original state $[a, x]$ because the map $x \mapsto 2x+c$ is bijective. This means that $[a, x]$ is reachable from $[b, 2x+c]$. Thus, the reachablity is bidirectional, and we can divide the states into connected components (as an undirected graph). We can answer the queries by checking if the two states $[t, 0]$ and $[s, r]$ are in the same component.
- Suppose that a vertex v is incident to two edges of lengths a and b. For an arbitrary integer x, $[v, x]$ and $[v, 4x + 3a]$ are equivalent (by traversing the edge of length a twice), and also $[v, x]$ and $[v, 4x + 3b]$ are equivalent. This means that we can freely add $3(a - b)$ to the black integer at this vertex.
- Let g be the maximum integer such that g is a divisor of MOD, and all edge lengths are equivalent modulo g. By combining the observation above, we only need to care the value of the black integer modulo $gcd(3g, MOD)$.

Therefore, we can assume that MOD is either g or $3g$, and all edge lengths are equivalent to z modulo g (for some integer z).

Let's define red integer as black integer plus z. Also, let's decrease the lengths of all edges by z. Now, when we pass through an edge of length c, the red integer x changes to $2(x - z) + (c + z) + z = 2x + c$. Thus, we can assume that MOD is either g or $3g$, and all edge lengths are divisible by g.

How to divide the states into connected components in this case? Let's fix a certain vertex as the root. All states reachable from $[root, X]$ can be written of the form $[v, pX + qg]$, where v is some vertex, p is a power of two and q is one of $0, 1, 2$. Notice that now the states $[v, x]$ and $[v, 4x]$ are always equivalent (we can reach $[v, x] \to [u, 2x + c] \to [v, 4x + 3c] = [v, 4x]$ using some edge incident to it). Thus, we can limit the values of p to one of $1, 2$.

We summarize the solution to the problem:

- Let's construct DSU with $6N$ states. Each state represents the state $[v, pX + qg]$, where v is some vertex, and $1 \le p \le 2, 0 \le q \le 2$.
- For each edge (a, b, c) and valid pair (p, q), connect the states $[a, pX + qg]$ and $[b, 2(pX + qg) + c]$. (In case this state doesn't exist among the $6N$ states, convert it using the "four times" rule.)

- To answer queries, we check if $[s, r+z]$ and $[t, z]$ are equivalent. This is "YES" if $[t, X]$ and $[s, pX+qg]$ are equivalent for some (p, q), and $(pZ+qg)/(r+z)$ is a power of four in modulo MOD.

This solution works in $\mathcal{O}(MOD + (n+m+q)\ \alpha(n))$ time.