AtCoder Peterzavodsk Contest 001 解説

japan02(yosupo, sugim, sigma)

2018 年 2 月 4 日

For International Readers: English editorial starts on page ?.

A: Two Integers

Y が X の倍数な場合、 X の倍数はかならず Y の倍数になってしまいます。なのでこの場合は -1 を出力します。

そうでない場合、X を出力すればよいです。

B: Two Arrays

まず，操作をすると，$\sum a_i$ は 2 増え，$\sum b_i$ は 1 増えます。つまり，操作回数は $\sum b_i - \sum a_i$ 回です。

- $a_i > b_i$ の場合，少なくとも b_i を $a_i - b_i$ 回，1 増やす必要があります。
- $a_i < b_i$ の場合，少なくとも a_i を $\lceil (b_i - a_i)/2 \rceil$ 回，2 増やす必要があります。

実はこの 2 つが必要十分です。つまり，この 2 つの式で必要最低限の 1 加算，2 加算を計算し，それが操作回数より多くなければ答えは YES です。

C: Vacant Seat

必ず空席を含むような区間を二分法によって狭めていくことで，$\mathrm{O}(\log N)$ 回のクエリで空席を見つけることができます．必ず空席を含むような区間というのは，次の 2 通りです:

(a) 閉区間 $[l, r]$ であって，$r - l$ が奇数で，席 l, r が同性であるもの．
(b) 閉区間 $[l, r]$ であって，$r - l$ が偶数で，席 l, r が異性であるもの．

(a) に空席が含まれないと仮定すると，席 $l, l + 1, \ldots, r$ には男性と女性が交互に座ることになり，席 l, r は異性となって矛盾します．(b) も同様です．

二分法は次のように行います:

- まず，席 0, $N - 1$ を質問する．いずれかが空席ならば終了．そうでなければ，区間 $[0, N - 1]$ を得る．
- 区間 $[l, r]$ に注目しているとき，席 $m := \lfloor l + r \rfloor / 2$ を質問する．空席ならば終了．そうでなければ，区間 $[l, m - 1]$, $[m + 1, r]$ の一方は (a), (b) のいずれかに該当するので，そちらを新しい区間とする．

D: Forest

コーナーケース: $M = N - 1$ の場合，元から木なので答えは 0 です。

以下では，$M < N - 1$ を仮定します。

まず，深さ優先探索などを使い，入力された森を木に分解します。

この木たちをくっつけて 1 つの巨大な木にするわけですが，この時，以下の 2 つの性質がわかります。

- どの木からも，少なくとも 1 つの頂点は選択される- 連結成分の個数は $N - M$ 個である。つまり，$N - M - 1$ 本の辺が足される。よって，全て合わせて $2(N - M - 1)$ 個の頂点が選ばれる。

2 つめの性質より，$N < 2(N - M - 1)$ ならば答えは 0 です。

ところで，実はこの 2 の性質は必要十分条件です。つまり，頂点から

- 連結成分ごとに，少なくとも 1 つの頂点を選ぶ
- 全て合わせて $2(N - M - 1)$ 個の頂点を選ぶ

の 2 つの条件を選ぶように頂点を選んだ時，また，その時のみ，選んだ頂点のみを使って森を木にすることができます。

実際に，最も選ばれた頂点数が多い 2 の連結成分をマージする，という戦略を繰り返すことで，かならず 1 つの木にすることができます。

よって，まず各連結成分から 1 番小さい値を選び，残ったものすべてから小さい順に，合計 $2(N - M - 1)$ 個になるように選べば，それが答えです。

これは，優先度付きキューや sort などをうまく使えば $O(NlogN)$ で行うことができます。

E: Antennas on Tree

証明は省きますが，問題の条件は次と同値です:

> 木の各頂点 v について，次が成り立つ: 木から v を取り除くと，木が k 個の部分木に分かれるとする．これらのうち $k-1$ 個以上の部分木はそれぞれアンテナを含む．

木のすべての頂点の次数が 2 以下の場合，木はパスの形をしており，いずれかの端にアンテナを置けばよいので，答えは 1 です．以降，木は次数 3 以上の頂点を含むとします．次数 3 以上の頂点 r をひとつ選び，r を根とする根付き木を考えます．すると，上述の条件は次と同値です:

> 根付き木の各頂点 v について，次が成り立つ: v が k 個の子を持つとする．これらをそれぞれ根とする k 個の部分木を考える．これらのうち $k-1$ 個以上の部分木はそれぞれアンテナを含む．

r 以外の各頂点 v について，v の親側の部分木には必ずアンテナが含まれます．なぜならば，r の子を根とする部分木のうちアンテナを含むものは 2 個以上ありますが，これらのうちひとつは v の親側の部分木に完全に含まれるからです．

r を根とする根付き木に対して，上述の条件が成り立つようなアンテナの個数の最小値を求めます．これは DP で可能です．各頂点 v と $x \in \{0,1\}$ について，$dp[v][x]$ を「v を根とする部分木内で，上述の条件が成り立ち，アンテナが存在しない $(x=0)$ / 存在する $(x=1)$ 場合の，アンテナの個数の最小値」と定義し，ボトムアップに計算すればよいです．

F: XOR Tree

まず各頂点 v に対して、v と接続する全ての辺 e に対し a_e を xor した値を b_v と定義します。すると、頂点 u,v を選んでパス上の辺の a_e に x を xor するという操作は、 b の言葉に言い直すと、b_u と b_v に x を xor するという操作になります。また、全ての e に対して a_e が 0 であることと、全ての v に対して b_v が 0 であることは同値です。従って、以下のような問題に置き換えられます。

$b_1, b_2, .., b_N$ が与えられる。i, j, x を選んで b_i と b_j に x を xor するという操作ができる。最小何回の操作で全てを 0 に出来ますか?

新しく以下のようなグラフを考えます。

- 頂点集合は $1, 2, .., N$
- 頂点 u, v を選んで操作した場合、u, v に辺を張る。

操作列の頂点のペアだけ先に決めた時に x を上手く決めて目標を達成できるかを考えます。まず各連結成分ごとに考えても構いません。各連結成分に対し、その頂点集合の b の値の xor が 0 でない場合、これは不可能です。逆に、0 の場合、これは可能であり、操作回数は、(その連結成分の頂点数) -1 になります。

従って、操作回数を最小化するためには、$\{1, 2, .., N\}$ を、できるだけ多くの、b_v の xor が 0 になるような部分集合に分割することができれば良いです。

証明は省略しますがこれは以下のようにできます。

全ての $k \geq 0$ に対し、$b_v = k$ なる v の個数を c_k とします。

まず $b_v = 0$ なる v に対しては単独で取れば良いです。これで c_0 個の部分集合がとれます。

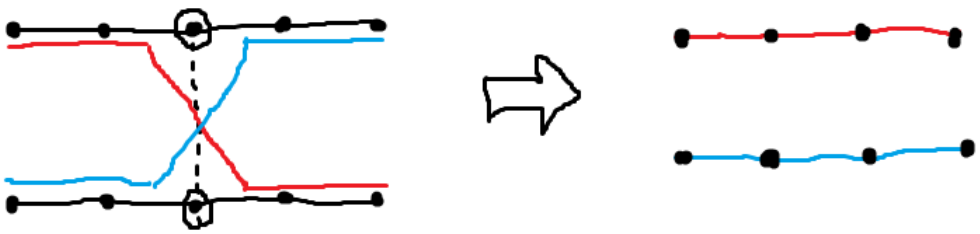
次に、$k > 0$ に対し、$b_v = k$ なる v 同士で出来るだけペアを作ります。これで $\lfloor c_k/2 \rfloor$ 個の連結成分がとれます。

すると最後に残るのは 1 から 15 の値が高々 1 つずつなので、これは bit DP で $O(3^{15})$ で出来ます。

G: Colorful Doors

ドア $2N$ のすぐ右にドア 1 があると考えると，全体が円環となります．このとき，ドア間の区間は全体で $2N$ 個あり，ドア $2N$, 1 間の区間は歩くべき区間です．

- すべての区間が歩くべき区間である場合
  この場合，歩くべき区間はひとつのサイクルになります．
  - (i) $2N$ が $4k$ の場合
    時計回りにドアの色を $1,2,1,2,3,4,3,4,5,6,5,6,\ldots,2N-1,2N,2N-1,2N$ とすればよいです．
  - (ii) $2N$ が $4k+2$ の場合
    証明は省きますが，不可能です．
- 歩くべきでない区間が存在する場合
  この場合，歩くべき区間はいくつかのパスの集合になります．両側を歩くべき区間に挟まれているドアを「内部のドア」と呼ぶことにします．各パスの「長さ」を，そのパスに含まれるの内部のドアの個数と定義します．例えば，$s=$ 010110011 の場合，パスの長さはそれぞれ $0,1,2$ です (末尾に 1 を追加した上で円環にしていることに注意)．
  - (iii) パスの長さの総和が $2k+1$ の場合
    不可能です．なぜならば，内部のドアは別の内部のドアとマッチングさせる必要があるので，内部のドアの総個数は偶数でなければならないからです．
  - (iv) パスの長さの総和が $4k$ の場合
    次のようにして可能です．パスの端どうしを順に繋げていくと，ひとつのサイクルになります．このサイクルは内部のドアを 4 の倍数個含みます．以降はケース (i) と同じです．
  - パスの長さの総和が $4k+2$ の場合
    - (v) 長さ 1 以上のパスが 2 本以上ある場合
      長さ 1 以上のパスを 2 本選び，それらの長さを x,y とします．各パスから内部のドアを 1 個ずつ選び，それらをマッチングさせます．すると，下図のようにパスが組み替えられ，これらの長さの和は $x+y-2$ となります．以降はケース (iv) と同じです．



    - (vi) 長さ 1 以上のパスが 1 本の場合
      長さ $4k+2$ のパスが 1 本と，長さ 0 のパスが何本かあります．これらのパスの端どうしを順に繋げるしかなく，ケース (ii) に帰着されて不可能です．

H: Generalized Insertion Sort

まず，頂点ごとに値の書かれたボールが入ってるとします。

もし葉に正しいボールが入ってたら，その頂点をカットすることが出来ます。これで葉からカットしていって，根まで全部カットする，というのが基本方針です。

ですが，葉を 1 個ずつカットしていてはクエリがとても足りないです。なので，葉パス頂点を，

- 頂点が葉である
- 子が 1 個だけであり，その子が葉パス頂点である

として定義します。

すると，高々 $O(logn)$ 回「木の葉パス頂点をカットする」を繰り返すと，木を全部カットすることが出来ます。これは，k 回のカットに耐えるためには $k-1$ 回のカットに耐える子を 2 個は持たないといけないことから示せます。

よって，$O(n)$ 回で，木の葉パス頂点に正しい値のボールを全部突っ込む，というのが出来ればこの問題は $O(nlogn)$ 回で解けます

これは，葉パス頂点に入れるべきボールを全部赤く塗って，その他を白く塗って，- 根に赤いボールが出てきたら，対応する頂点の葉パスの一番下から頂点を見ていって，「自分より入るべきところが上 || !青いボール」が出てきたら，そこにボールを入れて，ボールを青く塗る- 根に白いボールが出てきたら，赤いボールのうち最も下のものを選んで，そこに入れる

というアルゴリズムで行なえます。

具体的にクエリ回数を見積もると，カット回数が 11 で，各フェーズでは (全体数 + 葉パス頂点の数) でカットできるので，24000 回で抑えることが出来ます。

I: Simple APSP Problem

制約より，かなり黒いマスは「疎」であることがわかり，またこの性質を使うのだろうと想像がつくと思います。

まず，すべてのマスが白い行や列が大半を占めることがわかります。ではここで，i 行目と $i+1$ 行目のマスが全て白いと仮定します。

すると，i 行目側と $i+1$ 行目側から 1 個ずつ白いマスを選んだ時，またその時のみ，最短パスは i 行目と $i+1$ 行目の間を移動する事がわかります。

つまり，i 行目と $i+1$ 行目の間の辺の距離を全て 0 にしたとすると，この問題の答えは (i 行目側の白いマスの個数) $\times$ ($i+1$ 行目側の白いマスの個数) だけ減ることがわかります。

実は，この性質に気づくとこの問題は殆ど解けています。2 行連続で黒いマスが存在しないとき，その間を縮約することが可能なので，列に対しても同様の処理を行うと，この問題は高々 $H, W \leq 2N$ のグリッドに帰着できることがわかるからです。

ただし注意点として，元の問題とは違い，各マスごとに複数のマスが圧縮されていることがあります。ですので係数に気をつける必要がありますが，各マスから愚直に幅優先探索を行うことで，$O(N^4)$ で解くことが可能です。

J: Rectangles

まず A が a で割り切れない場合、B が b で割り切れない場合、C が c で割り切れない場合は条件を満たすものはありません。

とある平面で切れるようなものだけを考えると、どの平面で切れるかで包除をするとその個数は求まります。ただしこれはまだ答えではありません。つまりどんな平面でも切れないようなケースが存在します。

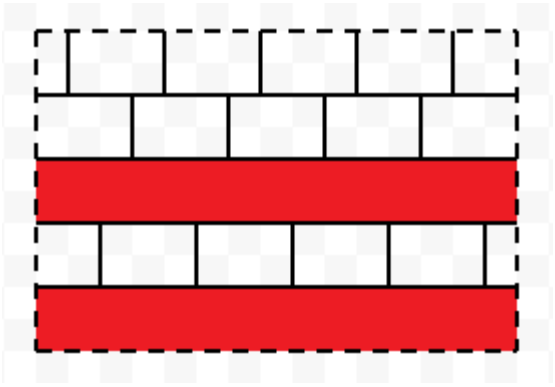
まず条件を満たすトーラス直方体の集合が与えられたとします。このとき、p,q,r を用いて書いたトーラス直方体 $\{((p+i) \mod A, (q+j) \mod B, (r+k) \mod C))\}$ に対して、その各小立方体に値 k を書き込みます。これによって全小立方体に値が書き込まれますが、マス (x,y,z) に書き込まれた値を $v(x,y,z)$ とします。このとき、各 x,y に対して、$v(x,y,0), v(x,y,1), .., v(x,y,C-1)$ という列を考えると、これは $0,1,..,c-1,0,1,..,c-1,..,0,1,..,c-1$ というものをサイクリックに動かしたものになっています。従って、$v(x,y,0)$ の値を決めれば $v(x,y,z)$ の値は全て決まります。なので以降は $v(x,y,0)$ のみを考えることにし、これを $v(x,y)$ とよぶことにします。

全てのペア $0 \leq x < A, 0 \leq y < B$ に対して $v(x,y)$ が定まっている時、この条件をみたすトーラス直方体の集合が何通りあるか考えます。これは、全ての $0 \leq k < C$ に対して、$v(x,y) = k$ をみたす (x,y) の集合を取ってきて、それをサイズ $a \times b$ のトーラス長方形に分割する通り数の積 を C/c 乗したもの となります。

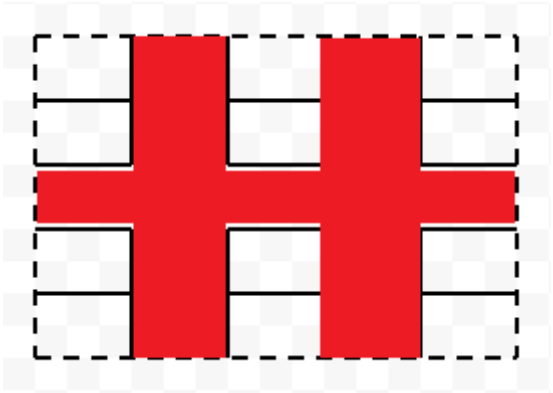
よってまず (x,y) の集合 S が与えられた時に、それをサイズ $a \times b$ のトーラス長方形に分割する通り数をカウントすることを目標にします。

次のようなケースが考えられます。

1. $S = \emptyset$
2. $S = \{(x,y) | 0 \leq x < A, 0 \leq y < B\}$
3. 次のように横にいくつか貫通している場合:



4. 縦に貫通している場合:
5. 縦横両方に貫通している場合:



6. それ以外:

ケース 5 となるような k が存在する場合、そのような k はひとつしか存在しません。このとき縦に i 本, 横に j 本貫通している時の求めたい値は除原理で $O(100^4)$ で求めることが出来ます。

ケース 5 となるような k が存在しない場合を考えます。すると必ずケース $2, 3, 4$ となる k が存在することが示せます。これらの場合はどれも、とある平面で切れるケースとなっています。よって、事前にとある平面で切れるケースの通り数を求めておけばここでは数えないことにすれば良いです。

AtCoder Petrozavodsk Contest 001 Editorial

japan02

2018 年 2 月 3 日

Editorial of G, H, J will be published later.

A: Two Integers

If X is a multiple of Y, the answer is -1: a multiple of X is always a multiple of Y.
Otherwise, X is always a valid answer.

B: Two Arrays

In each operation, the value $\sum b_i - \sum a_i$ decreases by 1. Thus, the total number of operations must be $K = \sum b_i - \sum a_i$.

In case $a_i < b_i$ for some i, you must perform operations on a_i at least $\lceil (b_i - a_i)/2 \rceil$ times. Thus, the sum of $\lceil (b_i - a_i)/2 \rceil$ for all i such that $a_i < b_i$ must be at most K. Otherwise, print "NO".

On the other hand, when this condition is satisfied, we can prove that the answer is "YES". Notice that the order of operations of adding twos and ones doesn't matter. First, add 2 to a_i, $\lceil (b_i - a_i)/2 \rceil$ times. Then, perform remaining "add two" operations on arbitrary elements. At this point, all i satisfies $a_i \geq b_i$, so you can finish operations by performing "add one" operations appropriately.

C: Vacant Seat

Consider the following two types of intervals (we call it "good interval"):

(a) A closed interval $[l, r]$ such that $r - l$ is odd and l, r are filled by people with the same sex.
(b) A closed interval $[l, r]$ such that $r - l$ is even and l, r are filled by people with different sex.

In both cases, we can see that the interval contains at least one empty seat. (Otherwise, males and females must sit alternately in the interval $[l, r]$, and we get a contradiction in both cases.)

We use binary search, and find an empty seat in $O(\log N)$ queries.

- First, perform a query on seat 0. If this is empty, we finish. Otherwise we find a good interval $[0, N]$. (Since the seats are cyclic, we can call seat 0 "seat N").
- Suppose that we have a good interval $[l, r]$. Let $m := \lfloor l + r \rfloor / 2$ and perform a query on seat m. If this is empty, we finish. Otherwise, we can prove that at least one of intervals $[l, m]$, $[m, r]$ will be a good interval by a simple argument about parities.

D: Forest

If $M = N - 1$, the input is a tree, and the answer is 0. Assume that $M < N - 1$.

First, divide the forest into connected components (trees). We want to merge these trees into a single tree by adding edges.

Here, we have the following constraints:

- From each tree, at least one vertex must be chosen (otherwise this tree will be isolated).
- There are $N - M$ trees. Thus, we must add $N - M - 1$ edges. This means that the total number of chosen vertices must be $2(N - M - 1)$.

It turns out that these conditions are sufficient. That is, if a subset of vertices satisfies the conditions above, we can actually make a tree by choosing those vertices. For example, we can connect all trees by repeating the following: choose two components with the biggest number of chosen (and still unused) vertices, and connect them.

The answer is as follows. First, if $N < 2(N - M - 1)$, the answer is "Impossible" (from the second condition). Otherwise, we choose vertices greedily as follows:

- First, from each component, choose a vertex with the smallest cost.
- Then, from remaining vertices, choose vertices from the smallest costs, until we choose $2(N-M-1)$ vertices in total.

These operations can be implemented in $O(NlogN)$ time.

E: Antennas on Tree

Consider two vertices s, t in the tree. When can we distinguish these vertices?

If the distance between them is odd, we can always distinguish them as long as we have at least one antenna (by checking the parity of distance from the antenna). Otherwise, let u be the midpoint between s and t. We see s, t and antennas from u. If the direction of at least one antenna is the same as the direction of s or t, we can distinguish them.

Thus, the condition can be restated as follows:

> For each vertex v, the following holds. Remove v from the tree, and suppose that we get k subtrees. Then, at least $k-1$ of the subtrees must contain antennas.

Now, we'll describe the solution. In case the tree is a path, the answer is one: we can put an antenna on one of the leaves. Otherwise, a vertex r with degree at least 3 exists. Make it the root of the tree.

The condition can be again restated as follows:

> For each vertex v in the rooted tree, the following holds. If v has k children, at least $k-1$ of k subtrees whose roots are children of v must contain antennas.

This is because, if v is not r, v always contain at least one antenna in the "parent direction". (Otherwise, the condition above won't be satisfied for the root r).

Now we can compute the answer by a simple DP. Define $dp[v]$ as the smallest number of vertices we must choose from the subtree rooted at v, when we want to satisfy the conditions above for all vertices in this subtree.

F: XOR Tree

For a vertex v, define b_v as the XOR of all values assigned to edges incident to v. If we perform an operation on the path between two vertices u and v with the value xm we change b_u and b_v to $b_u XOR x$ and $b_v XOR x$, respectively. Also, note that all b_v will be zero if and only if all edges are assigned zeroes. Thus, the problem can be restated as follows:

> You are given a sequence of integers $b_1, b_2, .., b_N$ (here, all integers are up to 15). In each operation, you choose i, j, x, and XOR x into b_i and b_j. How many operations do you need to make all integers zeroes?

Now, consider a graph with N vertices $1, 2, .., N$. Initially, this graph doesn't have edges. Whenever you perform an operation for indices i and j, add an edge between vertices i and j.

After we perform all operations, this graph will have several connected components. When you perform an operation between i and j, the value $b_i XOR b_j$ doesn't change. Thus, the XOR of all values in a single connected component never changes.

Therefore, we want to add minimum possible number of edges to this graph such that in each connected component, the XOR of all b_i is zero. Obviously, each connected component should be a tree if we want to minimize the number of edges. In this case, the total number of edges is N minus the number of connected components.

Thus, we can again restate the problem as follows:

> You are given a sequence of integers $b_1, b_2, .., b_N$ (here, all integers are up to 15). Divide these numbers into disjoint set, such that in each set the XOR of all values is zero. The answer is N minus the maximum possble number of sets we get.

First, we should make sets as follows:

- If we have a zero, we should create a set with this single zero.
- If we have two identical numbers, we should create a set with these two numbers.

After these process, we will have at most 15 elements. Now, we can compute the maximum number of sets in a simple $O(3^k)$ DP, where k is the number of remaining elements.

G: Colorful Doors

H: Generalized Insertion Sort

I: Simple APSP Problem

The black cells will be very sparse, and the vast majority of rows and columns will be entirely white.

Suppose that two adjacent rows, the i-th row and the $i+1$-th row, are entirely white. Let's call the region in the i-th row and above "UP", and the $i+1$-th row and below "DOWN". It's easy to see that the shortest path between two cells crosses an edge (here, we imagine that white cells correspond to vertices and there are edges for each adjacent pair of white cells) between the i-th row and the $i+1$-th row if and only if one of the cells is UP and the other is DOWN.

Thus, if we change the costs of all edges between the i-th row and the $i+1$-th row to zeroes, the answer decreases by the following value:

(The number of white cells in UP region) $\times$ (the number of white cells in DOWN region)

Now, we can "compress" two adjacent empty rows, and we can do the same for columns. By repeating these operations, we can compress the entire grid into a smaller grid with $H, W \leq 2N$, and we can simply solve the problem by BFS in $O(N^4)$.

Note that we should add "weights" to cells after compressions. For example, in the very first compression, if we merge the i-th row and the $i+1$-th row into a single row, the cells of this row should have a weight of 2. Then, in the final grid, the distance between two cells p and q should be added to the answer with the coefficient (the weight of p) $\times$ (the weight of q).

J: Rectangles