ARC 091/ABC 090 解説

Kohei Morita(yosupo)

平成 30 年 3 月 17 日

For International Readers: English editorial starts on page 7.

A: Two Coins

[https://beta.atcoder.jp/contests/abc091/submissions/2218039]

B: Two Colors Card Game

[https://beta.atcoder.jp/contests/abc091/submissions/2218376]

C: 2D Plane 2N Points

青い点のうち，最も x 座標が小さいもの (A とします) に注目します。この点と仲良しペアになれる点が存在した場合，そのような点のうち最も y 座標が小さいもの (B とします) と仲良しペアにしてよいです。

B が A 以外の点とペアになった場合，たとえば B-C と A-D がペアになった場合，これを入れ替えて A-C, B-D というペアに変更することが出来るからです。

よって，これで点の数が 1 点または 2 点少ない問題に帰着できたので，多項式時間で解くことが出来ます。

以上の考察をまとめると，青い点と赤い点を x 座標で sort して，

- 赤い点が出てきたら set に追加
- 青い点が出てきたら set の中で，青い点より y 座標が小さいなかで最も y 座標が大きいものを削除

で解くことが出来ます。

D: Two Sequences

答えの i-bit 目 (bit は 0-indexed とします) だけを求めることを考えます。このような問題を考えると，仮に a_i を固定した場合，

a_i と足して，i-bit 目が 1 になるような b_j は何個ありますか？という問題が解ければ良くなります。

ここで重要な考察として，b_j は i-bit 目を考える上では 2^i で mod を取っておいていいということがあります。

これを使うと，a_i と足して，i-bit 目が 1 になるような数字の区間はたかだか 2 個にできるので，sort をしておくことで二分探索で解くことが出来ます。

E: Both Sides Merger

最終的な答えは a_i をいくつか足したものになります。

ここで，この i の集合を考えます。例えばサンプル 1 なら $a_2 + a_4$ なので，$\{2, 4\}$ です。

この i の集合が与えられた時，最終的な答えがこれに対応するものの総和となるような操作が可能か，という問題を考えます。

まず，条件として，i の集合に含まれる要素の偶奇は全て同じ必要があります。これは，偶奇が違うペアは，どのような操作をしても偶奇が違う (ので，一つの要素にまとめるなど不可能) ということから示せます。実は，これが必要十分です。十分性の証明は実際に構築可能なことから示せます。

これが必要十分であることを仮定すると，偶奇を固定し，0 以上の要素をすべて取ることで，最終的な答えの最大値が求められます。全部の要素が負の場合に注意してください。

構築の方法ですが，端の要素をまず全てカットし，その後

- 両端がどちらも残したい要素
- 両端がどちらも残したくない，かつ自分自身も残したくない要素

のどちらかを必ず取れるので，これを取っていけばかならず要素が少ない良いです。

F: Two Faced Edges

各辺 i に対し，以下の条件を満たすかを調べたいです。

- 1. もし辺 (a_i, b_i) を消したとしても，a_i から b_i に到達可能か？
- 2. b_i から a_i に到達可能か？

この 2 つの条件を両方達成するか，両方達成しない場合，辺の反転で SCC の成分数が変わらないことが容易に示せます。

つまり，この 2 つの条件を $O(NM)$ で調べられるか？がこの問題の本質です。

条件 2 は簡単なので，条件 1 を考えます。

方針としては，a_i を固定して，頂点ごとに $O(M)$ で解きます。まず，a_i から出てくる辺に，適当に連番を振っておきます。

当然 (a_i, b_i) を使わずに a_i から b_i に到達可能ならば，(a_i, b_i) より小さい番号の辺か，大きい番号の辺かを使う必要があります。そしてさらに，a_i から出る辺は 1 本のみ使う場合を考えれば良くなります。

仮にこのうち，小さい番号の辺を使った場合のみに限定して解けることができれば，辺の連番をひっくり返して同じ問題を解くことによりこの問題が解けます。

小さい番号の辺を使った場合のみに限定すれば，辺の番号が小さい順に辺を追加していき，追加する直前に，今まで追加した辺を利用して b_i に到達できるか，を調べられれば良いです。

これは，辺を追加するたびに dfs を走らせ，二度同じ頂点は辿らないようにすれば，まとめて $O(M)$ が達成できます。

A: Two Coins

[https://beta.atcoder.jp/contests/abc091/submissions/2218039]

B: Two Colors Card Game

[https://beta.atcoder.jp/contests/abc091/submissions/2218376]

C: 2D Plane 2N Points

Let A be a blue point with the minimum x-coordinate. If A can be paired with red points, we should choose a red point B such that B can be paired with A, and among all such points the y-coordinate is the maximum possible.

(If B is paired with other point, for example if there are pairs $B-C$ and $A-D$, we can rearrange them because $A-C$ and $B-D$ are also valid pairs).

By doing this, we can reduce the number of points by 1 (if A can't be paired at all) or 2 (if we find a pair $A-B$).

In summary, we first sort all $2N$ points together by x-coordinates, and process points in this order. We also keep a set of points (initially this is empty). Then, for each point, do the following:

- If a red point appears, add its y-coordinate to the set.
- If a blue point appears, let p be its y-coordinate. If all elements in the set are greater than p, do nothing (we can't do anything with this blue point). Otherwise, choose the maximum element q in the set such that $q < p$, remove it, and increase the answer by one (the current blue point and the point that corresponds to q are paired).

D: Two Sequences

Since we can handle each bit independently, let's consider only the k-th bit. We want to count the number of pairs (i, j) such that the k-th bit of $a_i + b_j$ is one. (And if this number is add, we should add 2^k to the answer.)

Let $T = 2^k$. An important observation is that, we are only interested in the values of a_i, b_i in modulo $2T$. Thus, let's replace a_i with $a_i\%(2T)$ and b_i with $b_i\%(2T)$, and assume that $a_i, b_i < 2T$.

Then, there are two cases when the k-th bit of $a_i + b_j$ is one:

- $T \leq a_i + b_j < 2T$
- $3T \leq a_i + b_j < 4T$

Let's sort b in an increasing order. For a fixed i, the set of j that satisfies $T \leq a_i + b_j < 2T$ forms an interval. Thus, we can count the number of such js by binary search (or two-pointers). Similarly, we can handle the second case.

This solution works in $O(NlogNlogMAX)$ time, where MAX is the maximum number that can appear in the input.

E: Both Sides Merger

The answer of this problem will be the sum of a subset of $\{a_i\}$.

For example, consider example 1. In this sample, the answer is $a_2 + a_4$, so the set of indices is $\{2, 4\}$

Suppose that we are given a set of indices, I. Let's check if we can perform operations such that the final integer corresponds to the sum of all elements with indices in I.

It turns out that the following are neccesary and sufficient conditions:

- The parity of all indices in I are the same.
- I is non-empty.

It's easy to prove that these conditions are necessary. Suppose that initially, x is in an odd-indexed position and y is in an even-indexed position. Then, no matter what operations we perform, the parities of the indices of x, y never become the same.

These are also sufficient conditions because we can construct a sequence of operations, as we describe below.

To compute the answer, fix a parity, and take all non-negative elements in the positions with chosen parity. When all elements are negative, you need to be careful not to make it empty.

To construct a sequence of operations, we first remove all unnecessary elements from both ends. Then, we repeat choosing elements while we have more than one elements. We can choose an element x if one of the following holds:

- x is adjacent to two elements (call it y, z), and we want to keep both y and z in the final element.
- x is adjacent to two elements (call it y, z), and we want to remove both y and z from the array.

It's easy to see that such elements always exist.

F: Two Faced Edges

For each edge i, we want to determine the following two things:

- 1. If we remove an edge from a_i to b_i, is b_i still reachable from a_i (by using other edges)?
- 2. Is a_i reachable from b_i?

It's easy to see that the number of SCCs won't change after the reversion of edge i if and only if the answers to the two questions above are the same (i.e., both are yes or both are no). Since the second question can be solved easily in $O(NM)$ by running dfs from all vertices, we will focus on the first question.

Consider a particular vertex x. We will solve the first question for all edges such that $a_i = x$, in $O(M)$ time. Then, we can solve the entire problem in $O(NM)$.

Suppose that there are k edges that goes out of the vertex x, and let's name the destinations $y_1, \ldots, y_k$. We will do the following:

- By running a dfs from y_1, we mark all vertices that are reachable from y_1 without visiting x.
- By running a dfs from y_2, we mark all unmarked vertices that are reachable from y_2 without visiting x.
- By running a dfs from y_3, we mark all unmarked vertices that are reachable from y_3 without visiting x.
- And so on.

Then, for each vertex z, we know the minimum integer $p(z)$ such that z is reachable from x by using an edge $x- > y_{p(z)}$ (and without visiting the vertex x again). Similarly, by running dfs in the reverse order, for each vertex z, we know the maximum integer $q(z)$ such that z is reachable from x by using an edge $x- > y_{q(z)}$ (and without visiting the vertex x again). The question 1 for an edge $x- > y_i$ is "yes" if and only if at least one of $p(y_i) \neq i$ or $q(y_i) \neq i$ holds.

This solution works in $O(NM)$ time.