第 5 回 ドワンゴからの挑戦状 予選 解説
(Editorial for Dwango Programming Contest V Preliminary Round)

ドワンゴ競技プログラミング同好会
https://dwacon5th-prelims.contest.atcoder.jp/

2018/11/24 (土)

For International Readers: English editorial starts on page 6.

A: Thumbnail

a の平均値を X とします。X は整数となるとは限りませんが、NX は平均値の定義より必ず整数となります。これを利用して、$|Na_i - NX|$ が最小となるような i のうち、最小のものを求めることで、精度の問題を考えることなくこの問題を解くことができます。これは $O(N)$ で実行できるので十分高速です。

B: Sum AND Subarrays

$A_1, \ldots A_{N(N+1)/2}$ を、取り得る空でない連続する部分列の美しさすべてから成る数列とします。このとき、以下の条件 1 を満たす最大の非負整数 x が解です。ただし、$\wedge$ はビットごとの論理積を表します。

条件 1. $x \wedge A_i = x$ を満たす i が少なくとも K 個存在する。

これは以下の 2 つの性質が成り立つことから言えます。ただし B は、$A_1, \ldots, A_{N(N+1)/2}$ から K 個選んで取ってきて、それらの美しさのビットごとの論理積を計算した際に取りうる、すべての値からなる集合です。

性質 1. B に含まれる任意の要素 y において、$x \wedge y = x$ を満たす任意の非負整数 x は条件 1 を満たす。
性質 2. 任意の条件 1 を満たす非負整数 x において、$x \wedge y = x$ を満たす B の要素 y は少なくとも 1 つ存在する。

しかし、$x \leq \sum_{i=1}^{N} a_i \leq 10^{12}$ なので、x を全探索する方法ではまだ間に合いません。

取り得る x の最大値を $x_{\max}$ とします。このとき、任意の $x_{\max}$ 以外の取り得る x において、$x_{\max}$ と相異なるビットのうち最上位のビットは必ず 0 です。したがって、変数 x' の初期値を 0 として、$t = m, \ldots, 1$ について順に以下の手続きを行うことで得られる x' が $x_{\max}$ です。ただし、$m = \lceil \log_2 10^{12} \rceil = 40$ とします。

- $(x' + 2^{t-1}) \wedge A_i = (x' + 2^{t-1})$ を満たす i が少なくとも K 個存在するならば、x' に 2^{t-1} を足す。

このとき各ステップにかかる計算量は $N(N+1)/2 = O(N^2)$ です。したがって全体では $mN^2 \leq 4 \times 10^7$ 程度で求められるので十分間に合います。

C: k-DMC

簡単のため，$Q = 1$ の場合の解法を述べます．($k = k_1$ とします．) この問題が $O(N)$ 時間で解ければ十分です。

$S[i] =$'C' となる各 i に対して，$i - k \leq j < l < i, S[j] =$ 'D', $S[l] =$ 'M' となる (j, l) の個数が高速に計算できれば良いです．これには，各 i について区間 $[i - k, i)$ に着目して，その区間内に 'D', 'M', および部分列 ['D', 'M'] が何個現れるかを管理すれば良いです．以下の 2 つを行えば良いです：

- $S[i] =$'D' であれば 'D' の個数を 1 増やし，$S[i] =$'M' であれば 'M' の個数を 1 増やした上で ['D', 'M'] の個数を 'D' の個数だけ増やし，$S[i] =$'C' であれば ['D', 'M'] の個数を集計する．
- $i \geq k - 1$ の場合，$S[i]$ を処理した後で $S[i - k + 1]$ を注目する区間から除外する．これは上と同様にできる．

以上の解法が $O(N)$ 時間で動作することは明らかなので，元の問題が解けました．

D: Square Rotation

この解説では、お気に入りでないぬいぐるみは存在していないものとして説明します。操作の前後で、各ぬいぐるみの $(x_i \bmod D, y_i \bmod D)$ は変化することはありません。さらに、うまく操作順序を選ぶことで位置 (x, y) にあるぬいぐるみは $(x \bmod D, y \bmod D) = (x' \bmod D, y' \bmod D)$ を満たすような任意の (x', y') へ移動させることが可能です (ぬいぐるみの位置が異なる必要はあります)。これは最終的な配置が決まっている際に、はじめに各ぬいぐるみを十分遠くへ動かした後、ただ 1 つのぬいぐるみの位置だけが変化するように操作を繰り返して、(x, y) の辞書順に貪欲に置いていくということで達成可能なことから言えます。これらから、以下の問題が解ければいいことになります。

> 整数 N, D が与えられる。$0 \leq i, j < D$ を満たす (i, j) について $(x \bmod D, y \bmod D) = (i, j)$ なる (x, y) が $A_{i,j}$ 箇所あるように点を配置する必要がある。軸と並行に正方形を配置したとき、全ての点が正方形の中に含まれるようにしたい。これを達成する正方形の辺の長さの最小値を求めよ。

正方形の左下の座標は $(0, 0), (D-1, D-1)$ を対角線に持つような正方形領域内にあるとして構いません。簡単のため $(0, 0)$ にあるものとして説明します。正方形の一辺の長さが $aD + b$ $(0 \leq a, 0 \leq b < D)$ で表される場合について考えます。このとき $0 \leq i, j < D$ を満たすような (i, j) について、$(x \bmod D, y \bmod D) = (i, j)$ を満たす (x, y) のうち正方形領域内に含まれる位置の個数 $B_{i,j}$ は以下のように表せます。

$$B_{i,j} = \begin{cases} (a+1)^2 & (0 \leq i, j \leq b) \\ a^2 & (b < i, j) \\ a(a+1) & (\text{otherwise}) \end{cases}$$

このことから、全ての (i, j) について $A_{i,j} \leq B_{i,j}$ を満たすかどうかを調べることで、一辺の長さが $aD + b$ であるような正方形で全ての点を囲むことが可能かどうか判定できます。正方形の左下の座標の候補は D^2 通りあるため $aD + b$ が固定されている場合については $O(D^4)$ で解くことが可能なことが分かりました。正方形の一辺の長さについて二分探索を行うことで、$O(N + D^4 \log N)$ で実行可能であり、部分点が獲得できます。

上式より、$\max(A_{i,j}) \leq (n+1)^2$ なる最小の n を a として選べばよいことが分かります。これは、一辺の長さが $nD + D - 1$ の正方形であれば必ず全ての点を囲むことが可能なことから明らかです。さらに、$B_{i,j}$ は b の値に依存せず、これらの値が表す領域は図 1 のように 4 つの矩形をなすことが分かります。よって、$A_{i,j} > B_{i,j}$ となるような (i, j) が存在するかどうかは累積和を使ってまとめて判定することができます。この改善により、$O(N + D^2 \log D)$ で実行可能となり満点が獲得できます。答えが変化するタイミングは高々 D 回しかないことを利用すれば、$O(N + D^2)$ で実行可能ですが、この改善はおそらく必要ありません。

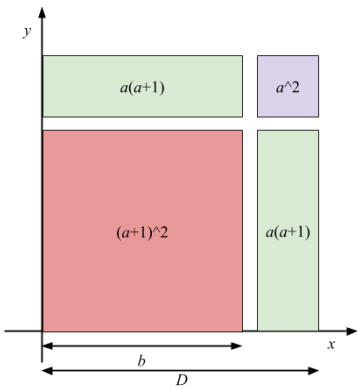

図 1　$B_{i,j}$ の分布

E: Cyclic GCDs

まず，入力 a はソートされているものとしても一般性を失いません．以降 a はソートされているものとして扱います．

サイクル $p_1 \to p_2 \to \cdots \to p_k \to p_1$ に対して，$\min_{i=1}^{k} a_{p_i} = a_{\min_i p_i}$ をこのサイクルの **最小値** と呼ぶことにします．a の左の要素から (つまり小さい要素から) 順番に見ていくことを考えます．$\mathrm{DP}[i][j]$ を，a_i までの要素をチェックした時にサイクルが j 個できている状態における，j 個のサイクルの最小値の積の総和とします．このとき，$\mathrm{DP}[i][j]$ からの遷移は，すでに存在するサイクルのどれかに $i+1$ を追加するか，1 点 $i+1$ からなる新しいサイクルを作る (a_{i+1} 倍) かのどちらかです．(第 1 種スターリング数に類似した遷移になります．) ここで，サイクルの中に新しい要素を追加する遷移の係数を調べましょう．大きさ k のサイクル $p_1 \to p_2 \to \cdots \to p_k \to p_1$ に対して，その中に新しい要素 $i+1$ を追加する方法の総数は，k 個ある隙間のどこに入れるかを選ぶ方法の総数なので，k 通りです．全てのサイクルについてこれらの値を足すと，合計は i になります．サイクルの要素は全て $i+1$ 以下であるため，この遷移により各サイクルの最小値は変わりません．

以上の議論で，遷移 $\mathrm{DP}[i][j] \to \mathrm{DP}[i+1][j]$ (係数 i) と $\mathrm{DP}[i][j] \to \mathrm{DP}[i+1][j+1]$ (係数 a_{i+1}) の 2 種類の遷移があることが分かりました．また DP の初期値は，0 番目まで見たとき (要素を 1 つも見ていないとき) に，サイクルが 0 個存在する順列がただ一つ存在するので，$\mathrm{DP}[0][0] = 1$ です．最終的に求める値は，$b_i = \mathrm{DP}[N][i]$ $(1 \le i \le N)$ の最大公約数 (GCD) です．

ここで，高々 N 次の多項式 $P_i(t)$ $(0 \le i \le N)$ を以下のように定義することにします：

$$P_i(t) := \sum_{j=0}^{N} \mathrm{DP}[i][j] t^j.$$

$\mathrm{DP}[i][j]$ の遷移の係数は j に依存しないため，$0 \le i \le n-1$ を満たす i に対して，

$$P_{i+1}(t) = (a_{i+1}t + i) P_i(t)$$

が成立します．明らかに $P(0) = 1$ であるため，$\mathrm{DP}[N][i]$ は多項式

$$P_N(t) = (a_1 t + 0) \times (a_2 t + 1) \times \cdots \times (a_N t + (N-1))$$

における t^i の係数です．この多項式の係数の GCD が計算できれば，この問題を解くことができます．

ここで，以下の補題が有用です．

補題 1. *P, Q を整数係数多項式とする．このとき，$c(F)$ で 多項式 F の係数の GCD を表すことにすると，$c(PQ) = c(P)c(Q)$ が成立する．*

Proof. 一般性を失わず，$c(P) = c(Q) = 1$ を仮定する．$c(PQ) = 1$ を示せばよい．

背理法を用いる．$c(PQ) \neq 1$ を仮定して矛盾を導く．このときある素数 p が存在して $c(PQ)$ は p の倍数である．$P = r_l t^l + r_{l-1} t^{l-1} + \cdots + r_0, Q = s_k t^k + s_{k-1} t^{k-1} + \cdots + s_0$ とする．$PQ = r_l s_k t^{l+k} + \cdots$ であるため，r_l と s_k のどちらかは p の倍数である．r_l が p の倍数である場合は P として 1 次低い多項式を，s_k が p の倍数である場合は Q として 1 次低い多項式をとり，$c(P) = c(Q) = 1$ で $c(PQ)$ が p の倍数であるような多項式 P, Q を作ることができる．これを繰り返すと P もしくは Q のどちらかが 0 になるため，$c(P) = c(Q) = 1$ と矛盾する． □

この補題を繰り返し用いることにより，$c(P_N(t)) = \prod_{i=0}^{N-1} c(a_{i+1}t + i) = \prod_{i=0}^{N-1} \gcd(a_{i+1}, i)$ が分かるので，$O(N \log N)$ 時間でこの問題を解くことができました．

Editorial for Dwango Programming Contest V Preliminary Round

Dwango Competitive Programming Club

https://dwacon5th-prelims.contest.atcoder.jp/

Saturday, November 24, 2018

A: Thumbnail

Let X denote the average of the representation values. All you have to do is to output the minimum index t that satisfies $|a_t - X| = \min_i |a_i - X|$.

X is not always an integer, so you might care for its precision. In that case, you can calculate the minimum index t that satisfies $|Na_t - NX| = \min_i |Na_i - NX|$. In this way, you do not need to care for the precision since NX is always an integer.

The overall calculation can be done in $O(N)$ time, which is sufficiently fast.

B: Sum AND Subarrays

Let $A = A_1, \ldots, A_{N(N+1)/2}$ be an array of partial sums of all possible contiguous subsequences. We want to choose K of them and maximize their bitwise AND. Let's consider that they are unsigned fixed-size (say, 64-bit) integers.

Intuitively, we want to choose K numbers so that the highest bit of their bitwise AND becomes 1 if possible, because 2^t is larger than $\sum_{i=0}^{t-1} 2^i$. Then we would find that if A has K or more elements with the highest bit of 1, we can remove the others. Let A' be the resulting array ($A' = A$ if there are less than K numbers with the highest bit of 0).

In any way we take K elements from A', the highest bit of their bitwise AND is always the same, because A' consists of only numbers whose highest bit is 1, or A' contains less than K numbers whose highest bit is 1 where we definitely need to choose at least one element of the highest bit of 0. That is, we can safely forget the highest bit and work on the same problem with smaller size. We can apply the same approach to the second highest bit, and so on. Once we iterate for all digits, we can take any K numbers and print their bitwise AND as an answer—the bitwise ANDs are the same in any case. Note that it is always possible to take K elements because each removal process does not make the number of elements less than K.

Determining one bit takes $|A| = O(N^2)$ time. With the assumption that a bit length of A_i is constant, this problem can be solved in $O(N^2)$ time.

C: k-DMC

For simplicity, a solution to $Q = 1$ is described here. It is enough to solve this restricted problem in $O(N)$ time.

For each index i such that $S[i] =$`'C'`, it suffices to find the number of pairs (j, l) that satisfy $i - k \leq j < l < i, S[j] =$ `'D'`, $S[l] =$ `'M'`. To achieve this goal, for each i we consider the interval $[i - k, i)$ and manage the number of occurrences of `'D' 'M'` and `['D', 'M']` (as subsequences) in it. Perform the following two operations for each i:

- Increase the number of `'D'` by 1 if $S[i] =$`'D'`, increase the number of `'M'` by 1 and the number of `['D', 'M']` by the number of `'D'` if $S[i] =$`'M'`, and accumulate the number of `['D', 'M']` if $S[i] =$`'C'`.
- If $i \geq k - 1$, exclude $S[i - k + 1]$ from the interval we consider after processing $S[i]$. This can be done in the same way as the above operation.

Since it is obvious that this procedure works in $O(N)$ time, we can solve the original problem.

D: Square Rotation

In this editorial, we ignore non-black toys of TV-chan. For each toy, $(x_i \bmod D, y_i \bmod D)$ is not changed by the operation. Furthermore, a toy at position (x, y) can be moved to arbitrary position (x', y') such that $(x \bmod D, y \bmod D) = (x' \bmod D, y' \bmod D)$ and no other toys are at the position. This is achieved by moving toys far enough and then arrange toys one by one in the lexicographical order of the target positions (x', y') by performing operations of moving exactly one toy.

With this observation, we find that solving the following problem is sufficient.

> Integers N, D and $A_{i,j}$ $(0 \leq i, j < D)$ are given. We need to arrange N points and an axis-aligned square under the following conditions: (a) the number of points (x, y) such that $(x \bmod D, y \bmod D) = (i, j)$ is $A_{i,j}$ and (b) the axis-aligned square contains all points. Find the minimum side length of such a square.

Without loss of generality, we can assume that the position of the lower left corner of the square in square area with diagonal line $(0, 0), (D-1, D-1)$. To simplify, we consider squares whose lower left corner is at position $(0, 0)$. For a square with side length $aD + b$ $(0 \leq a, 0 \leq b < D)$, the numbers $B_{i,j}$ of points (x, y) such that $(x \bmod D, y \bmod D) = (i, j)$ can be represented as follows:

$$B_{i,j} = \begin{cases} (a+1)^2 & (0 \leq i, j \leq b), \\ a^2 & (b < i, j), \\ a(a+1) & (\text{otherwise}). \end{cases}$$

If $A_{i,j} \leq B_{i,j}$ holds for all (i, j), a square with side length $aD + b$ can enclose all points. Since the number of candidate positions of the lower left corner of the squares is D^2, for fixed $aD + b$, we can solve the problem in $O(D^4)$ time. By applying binary search on the side length of the enclosing square, we can reduce complexity to $O(N + D^4 \log N)$ and obtain partial scores.

From the above equation, we should choose the minimum n such that $\max(A_{i,j}) \leq (n+1)^2$ as a since a square with side length $nD + D - 1$ can always enclose all points. Moreover, $B_{i,j}$ is independent of b and its values form four rectangles as in Figure 1. Therefore, we can determine whether (i, j) such that $A_{i,j} > B_{i,j}$ exists or not at once by cumulative sum technique. This improvement enables us to solve the problem in $O(N + D^2 \log D)$ time and to achieve full scores.

This problem can be solved in $O(N + D^2)$ time, although this optimization may not be required. Instead of performing binary searches for each candidate corner, we can keep the minimum side length and check whether the side length is able to be reduced for each candidate. Since the minimum side length changes at most D times, the time complexity of this solution is $O(N + D^2)$.

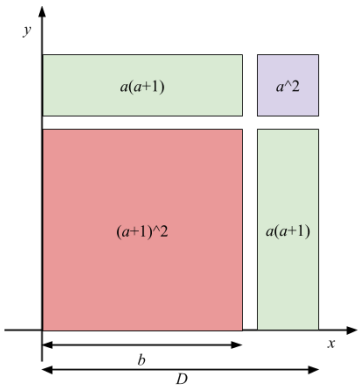

Figure 1 $B_{i,j}$

E: Cyclic GCDs

First, let us sort the input array a in the non-decreasing order. As the order of elements does not matter, this operation will not change the answer. The following argument assumes that a is sorted in the non-decreasing order.

For a cycle $p_1 \to p_2 \to \cdots \to p_k \to p_1$, let us call $\min_{i=1}^{k} a_{p_i} = a_{\min_i p_i}$ the cycle's *minimum value*. Consider checking the elements of a from left to right (hence in the non-decreasing order). Let us define $\mathrm{DP}[i][j]$ by the sum of all products of the cycles' minimum values when elements until a_i are checked and j cycles are formed. Then the transitions from $\mathrm{DP}[i][j]$ are the following two:

- Add $i+1$ to one of the existing j cycles. The order of the existing elements are not changed.
- form a cycle with the single element $i+1$ (coefficient: a_{i+1})

Let us find the coefficients of the first type of transitions. For a cycle $p_1 \to p_2 \to \cdots \to p_k \to p_1$ of size k, there are k ways to insert $i+1$ in the middle of the cycle, since the number of gaps to insert $i+1$ in is k. Summing up these values for all cycles, we obtain the total of i. Because elements of any cycle are less than or equal to $i+1$, this transition does not change the cycles' minimum values.

In the argument above, we found that there are two transitions, namely $\mathrm{DP}[i][j] \to \mathrm{DP}[i+1][j]$ (with coefficient i) and $\mathrm{DP}[i][j] \to \mathrm{DP}[i+1][j+1]$ (with coefficient a_{i+1}). The initial value of DP is $\mathrm{DP}[0][0] = 1$, where zero elements are checked and zero cycles are formed. The value we want is the greatest common divisor (GCD) of values $b_i = \mathrm{DP}[N][i]$ $(1 \le i \le N)$.

Define polynomials $P_i(t)$ $(0 \le i \le N)$ of degree at most N by

$$P_i(t) := \sum_{j=0}^{N} \mathrm{DP}[i][j] t^j.$$

Since the coefficients of transitions from $\mathrm{DP}[i][j]$ do not depend on j, for every i that satisfies $0 \le i \le N-1$, the equality

$$P_{i+1}(t) = (a_{i+1}t + i) P_i(t)$$

holds. The equality $P(0) = 1$ is obvious. Therefore, $\mathrm{DP}[N][i]$ is the coefficient of t^i in

$$P_N(t) = (a_1 t + 0) \times (a_2 t + 1) \times \cdots \times (a_N t + (N-1)).$$

The answer we want is the GCD of these coefficients.

In finding this value, the following lemma is useful.

Lemma 1. *Let P, Q be polynomials with integer coefficients. Let us denote by $c(F)$ the GCD of all coefficients of a polynomial F. Then $c(PQ) = c(P)c(Q)$ holds.*

Proof. Assume $c(P) = c(Q) = 1$ without loss of generality. It is enough to show $c(PQ) = 1$.

Let us use *reductio ad absurdum*. Assume $c(PQ) = 1$ and we are to prove contradiction. Then there exists a prime number p such that $c(PQ)$ is a multiple of p. Let $P = r_l t^l + r_{l-1} t^{l-1} + \cdots + r_0, Q = s_k t^k + s_{k-1} t^{k-1} + \cdots + s_0$. Because $PQ = r_l s_k t^{l+k} + \cdots$, either r_l or s_k is a multiple of p. We can construct another instance of polynomials P, Q where $c(P) = c(Q) = 1$ and $c(PQ)$ is a multiple of p, by lowering P's degree if r_l is a multiple of p or by lowering Q's degree if s_k is a multiple of p. By iterating this process either P or Q becomes 0, which contradicts $c(P) = c(Q) = 1$. □

By iteratively using this lemma, we obtain the equality $c(P_N(t)) = \prod_{i=0}^{N-1} c(a_{i+1}t + i) = \prod_{i=0}^{N-1} \gcd(a_{i+1}, i)$, which gives an $O(N \log N)$-time solution.