AtCoder Grand Contest 030 解説

writer: semiexp, sugim48, DEGwer

2018 年 12 月 29 日

For International Readers: English editorial starts on page 7.

A: Poisonous Cookies

食べることのできる毒入りのクッキーの枚数は、解毒剤入りのクッキーの枚数に 1 を足したもの以下です。すなわち、$A + B + 1 \geq C$ ならすべての毒入りのクッキーを食べることができるので答えは $B + C$ であり、そうでないなら答えは $B + (A + B + 1)$ です。

B: Tree Burning

高橋君は最初に反時計回りに進み始めるとして一般性を失いません (最初に進む向きを両方ためし、出てきた解のうち大きい方を出力すればよいです)。

高橋君が初めて時計回りに進んで以降、高橋君はまだ燃やしていない木に到達するごとに進む向きを変えるとして良いことが以下のように証明できます: もしある木 i を燃やした後進む向きを変えないのならば、その後方にある木を燃やしたときに木 i の座標まで行って木 i を燃やしていたことにすれば、進む距離の合計を増やすことができます。

さて、以上のような行動の方法は、最初に向きを変える位置を決めることで定まるので、$O(N)$ 通りしかありません。最初に向きを変える位置それぞれに対して高橋君の移動距離を求められればこの問題を解くことができ、これは累積和を使うか、隣り合う位置についての移動距離の差分を考えることで $O(N)$ 時間で動くように実装できます。

C: Coloring Torus

具体的な構成を示します．便宜上，色を $0, 1, \ldots, K-1$ で表すことにします．まず，n が偶数のとき，$K = 2n$ に対して次のような構成があります：

- $r \equiv 0 \pmod 2$ のとき，マス (r, c) を色 $(r + c) \bmod n$ で塗る．
- $r \equiv 1 \pmod 2$ のとき，マス (r, c) を色 $n + ((r + c) \bmod n)$ で塗る．

この塗り方が K 色でのよい塗り方であることはすぐに確かめられます．

ここで，任意の i $(0 \leq i \leq n-1)$ について，この塗り方で色 $i + n$ で塗られているマスをすべて色 i に塗り替えても，(K 色のうちすべての色が塗られているという条件を除いて) よい塗り方であることがわかります．このような塗り替えを k $(0 \leq k \leq n)$ 色に対して行って，適宜色の番号を取り替えることで，$2n - k$ 色での塗り方が得られます．よって，K が一般の $2 \leq K \leq 1000$ の場合にも，$n = 2\lceil \frac{K}{4} \rceil$ とすると，このようにして $n \leq 500$ なるよい塗り方を得ることができます．$K = 1$ の場合はこの方法ではうまくいきませんが，この場合の構成は明らかです．

D: Inversion Sum

各操作を指定した 2 つの要素を確率 1/2 で入れ替える操作だと考えることにすれば、求める値は最終的な数列の反転数の期待値の 2^Q 倍です。期待値の線形性より、最終的に $A_i > A_j$ となる確率をすべての $i < j$ について求めて足し合わせれば良いです。

DP$[t][i][j]$ を、最初の t 回の操作をしたときに $A_i > A_j$ となる確率とします。DP$[0][i][j]$ の値はすべて、最初の数列に関する情報から求めることができます。また、DP$[t+1][i][j]$ の値が DP$[t][i][j]$ の値と異なるのは、i または j の一方以上の位置を変える可能性のある操作をする場合だけであり、このような (i, j) の組は $O(N)$ 個です。以上より、同じ配列を使いまわすことで $O(N)$ 時間でひとつの操作に対する DP 配列の更新ができることが分かり、よって $O(N^2 + NQ)$ 時間でこの問題を解くことができました。

E: Less than 3

文字列の 0 と 1 の間に赤い線を，1 と 0 の間に青い線を書くと，線たちは操作によって図 1 のように変化します．これを観察すると，赤い線と青い線が常に交互に並んでいることが分かります．また，各操作によって，端以外では線を距離 1 だけ動かすことができ，端では線の追加・削除ができることが分かります．

端での挙動を扱いやすくするために，両端に無限個の線が赤青交互に重なっていると考えます．すると，線たちは操作によって図 2 のように変化します．これを観察すると，次のことが分かります：

- 赤い線と青い線が常に交互に並んでいる．
- 端以外で線どうしが重なってはならない．
- 隣り合う線どうしの間隔は 2 以下である．
- 各操作によって，線を距離 1 だけ動かすことができる．

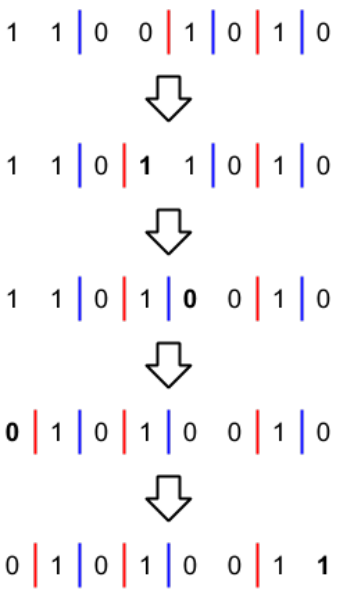

図 1

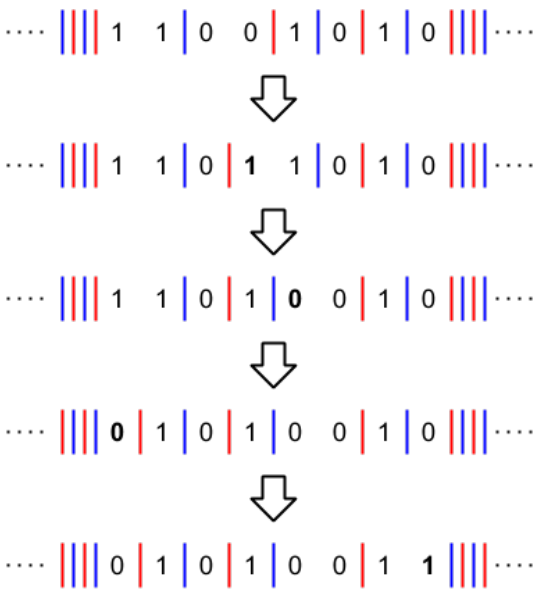

図 2

図 3 のように，s と t の間で線たちの対応が定まっているとします．すると，必要な操作回数の最小値は，対応する線どうしの距離の総和となることが容易に示せます．図 3 の例では，$\cdots+0+0+0+1+0+1+2+3+2+1+0+\cdots=10$ です．線たちの対応の定め方は $\mathrm{O}(N)$ 通りしかありません．なぜならば，あまりに遠くの線どうしを対応させるのは無駄だからです．具体的には，左端の線と右端の線を対応させるような定め方は (ほとんどの場合) 無駄です．(ただし，$s=$ 00, $t=$ 11 などの例外に注意．) よって，線たちの対応を $\mathrm{O}(N)$ 通り全探索することで，$\mathrm{O}(N^2)$ 時間で問題を解くことができます．

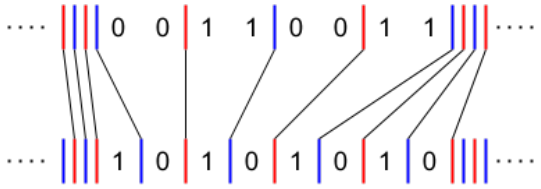

図 3

F: Permutation and Minimum

まず，A_{2i-1} および A_{2i} がともに -1 でないような i は存在しないとして構いません．なぜなら，このような i に対しては B_i は一定なので，数列から A_{2i-1}, A_{2i} を取り除いて，書き込む数も $1, 2, \ldots, 2N$ から A_{2i-1}, A_{2i} を除いた $2N-2$ 個の数としたものに対して場合の数を考えても変わらないためです．

異なる $\{B_i\}$ の個数にのみ興味があるため，一般性を失わず，次の仮定をしてかまいません：

- $A_{2i} = -1$．(さもなくば，A_{2i-1} と A_{2i} を入れ替えてかまいません)
- $A_{2i-1} = A_{2i} = -1$ ならば，A_{2i-1} に書き込まれる整数は A_{2i} に書き込まれる整数より大きい．
- $A_{2i-1} = A_{2i} = -1$ なる i の集合を $S = \{x_1, \ldots, x_k\}$ $(x_1 < \ldots < x_k)$ とする．このとき，$A_{2x_1}, \ldots, A_{2x_k}$ に書き込まれる整数はこの順に降順である．

最後の条件を導入することで，$\{B_i\}$ の個数は変わってしまいますが，最後に $|S|!$ を掛ければ正しい答えが得られます．

整数を $2N, 2N-1, \ldots, 1$ の順に，$\{B_i\}$ 中のどこに現れるか（あるいは，どこにも現れないか）を決定していくことを考えます．整数 n の $\{B_i\}$ 中での出現には次の可能性があります (それぞれに $a(n)$ などの名前をつけておきます)：

- $a(n)$: n は $\{B_i\}$ 中のどこにも現れない．また，$\{A_i\}$ 中にも現れない．
- $a'(n)$: n は $\{B_i\}$ 中のどこにも現れない．一方，$\{A_i\}$ 中に現れる．
- $b(n)$: $A_{2i-1} = A_{2i} = -1$ なる i に対して，B_i に現れる．
- $c_p(n)$: $A_{2i-1} = p$ なる i に対して，B_i に現れる $(p > n)$.
- $d(n)$: $A_{2i-1} \neq -1$ なる i に対して，B_i に現れる．このとき，$A_{2i-1} = n$.

$\{B_i\}$ を，それに対応する各整数の出現の仕方の列で表すことにします．例えば，入力例 1 の $(B_1, B_2, B_3) = (1, 3, 2)$ に対しては，$a(6) \to a(5) \to a(4) \to d(3) \to b(2) \to d(1)$ となります．このとき，異なる $\{B_i\}$ からは異なる列が得られます．よって，正しい $\{B_i\}$ に対応するような異なる列の個数を数えれば答えが得られます．

どのような列が正しい $\{B_i\}$ に対応するかを考えます．まず，$\{B_i\}$ 中に出現する整数は N 個なので，$a(n), a'(n)$ は合わせてちょうど N 回現れる必要があります．次に，n が $\{A_i\}$ 中に現れるなら，n の出現の仕方は $a'(n)$ か $c_p(n)$ のいずれかです．また，n が $\{A_i\}$ 中に現れないなら，n の出現の仕方は $a(n), b(n), d(n)$ のいずれかです．最後に，$b(n), c_p(n), d(n)$ は，それを実現する $\{A_i\}$ への整数の書き方において，n より大きい整数とペアになっている必要があります．すなわち，ある $n' > n$ が存在して，$b(n), d(n)$ は $a(n')$ と，$c_p(n)$ は $a'(p)$ とペアになっている必要があります．(同じ整数が複数のペアに所属することはできません．) 逆に，これらの条件を満たしていれば，そのような列は正しい $\{B_i\}$ に対応することがわかります．

ここで，列が与えられたとき，ペアの構成が可能かどうかは，$n = 2N, 2N-1, \ldots, 1$ の順に，$b(n), c_p(n), d(n)$ が現れたときは今までに現れた，今までにペアになっていない $a(n), a'(p)$ を選んでペアを組むということを行えば良いことがわかります．

関数 $f(n, x, Y)$ を，次の条件を満たす，$2N$ から $n+1$ までの数の出現の列の個数として定めます：

- ペアになっていない $a(i)$ の個数は x.
- $n+1 \leq i \leq n$ であって，ペアになっていない $a'(i)$ の集合は Y に等しい．

$f(n, x, Y)$ で数えられている列に対して n を加えた際の列は次のようになります：

- n が $\{A_i\}$ 中に現れる場合：
  - $a'(n)$ を置いた場合：$f(n-1, x, Y \cup \{n\})$ で数えられる状態．(a)

  - $d(n)$ を置いた場合：$f(n-1, x-1, Y)$ で数えられる状態 ($x \geq 1$ の場合に限る).
- n が $\{A_i\}$ 中に現れない場合：
  - $a(n)$ を置いた場合：$f(n-1, x+1, Y)$ で数えられる状態.
  - $b(n)$ を置いた場合：$f(n-1, x-1, Y)$ で数えられる状態 ($x \geq 1$ の場合に限る).
  - $c_p(n)$ を置いた場合：$f(n-1, x, Y \setminus \{p\})$ で数えられる状態 ($p \in Y$ の場合に限る). (b)

ここで, $g(n, x, y) = \sum_{|Y|=y} f(n, x, Y)$ とおくと, (a) は $(n, x, y) \to (n-1, x, y+1)$ の遷移, (b) は $(n, x, y) \to (n-1, x, y-1)$ への y 個の遷移とみなすことができます. よって, 各 g の値を順次動的計画法で計算することができます. n, x, y は $[0, 2N]$ の値をとるので, これは $O(N^3)$ で計算することができ, $g(0, 0, 0)$ が求める値です.

AtCoder Grand Contest 030 Editorial

writer: semiexp, sugim48, DEGwer

December 29th, 2018

A: Poisonous Cookies

The maximum number of poisonous cookies you can eat is $A + B + 1$. Thus, if $A + B + 1 \geq C$, you can eat all cookies and the answer is $B + C$. Otherwise, the answer is $B + (A + B + 1)$.

B: Tree Burning

For simplicity, suppose that Takahashi's home is tree 0 (and he has just finished burning it), and there are $N - 1$ other trees. Let's see how the path of Takahashi looks like.

Suppose that he first start moving right (we call counter-clockwise "right" and clockwise "left"). (He may start moving left, so don't forget to try both directions and output the maximum.) Then, each time he meets with a tree, he burns it. When he arrives at tree a, he may decide to change the direction and start moving left. Then, when he arrives at tree b for some $b < 0$, he may again decide to change the direction and start moving right.

Here, assume that whenever he moves right, the tree number increases, and whenever he moves left, the tree number decreases. For example, when he moves to the left from tree 0, he will meet tree -1 next (but it means tree $n - 1$.

Then, for example, in case he makes five turnings, his path looks like $0 \to c \to b \to d \to a \to e$, where $a < b < 0 < c < d < e$ and $e - a = N - 1$. Here, we can assume that $b - a = 0 - b = e - d = d - c = 1$ (otherwise we can get a longer path in an obvious way). Thus, once we fix the initial direction and the number of turnings, we can uniquely determine the path. Be careful that depending on the parity of number of turnings, the path looks slightly different.

Therefore, there are only $O(N)$ paths we are interested in. With proper pre-computations of prefix sums, the distance of each path can be computed in $O(1)$, so this solution works in $O(N)$ time.

C: Coloring Torus

The following construction works. For simplicity, we represent colors as $0, 1, \ldots, K-1$.

In case n is even, for $K = 2n$, the following grid is valid:

- If $r \equiv 0 \pmod 2$, color the cell (r, c) with color $(r + c) \bmod n$.
- If $r \equiv 1 \pmod 2$, color the cell (r, c) with color $n + ((r + c) \bmod n)$.

It's easy to verify that this is a valid coloring.

Here, notice that for any i $(0 \leq i \leq n-1)$, even if you repaint all cells of color $i + n$ to color i, the grid is still valid (except for the condition that all colors must be used). If you do this operation for k colors $(0 \leq k \leq n)$, you get a valid grid with $2n - k$ colors.

For general K satisfying $2 \leq K \leq 1000$, the construction above works for $n = 2\lceil \frac{K}{4} \rceil$ (and $n \leq 500$ will be satisfied). The case with $K = 1$ is trivial.

D: Inversion Sum

Instead of computing the sum, let's assume that we choose each operation with probability $1/2$ and compute the expected number of inversions in the final array. The answer is 2^Q times this value.

Define $DP[t][i][j]$ as the probability that after t operations, $A_i > A_j$ holds. We can get the values of $DP[0][i][j]$ from the information of the initial array. Then, if you know the values of $DP[t][i][j]$ for all pairs of (i, j), you can get the values of $DP[t+1][i][j]$ for all pairs of (i, j).

This operation looks like $O(N^2)$ per each t, but notice that the value of $DP[t+1][i][j]$ and $DP[t][i][j]$ differs only when at least one of i or j is involed in the $(t+1)$-th swap. There are only $O(N)$ such pairs. Thus, if you reuse the array and update only necessary parts, this update can be done in $O(N)$ per each t.

Because of the linearity of expected values, the expected number of inversions in the final array is the sum of $DP[Q][i][j]$ for all $i < j$. This solution works in $O(N^2 + NQ)$ time.

E: Less than 3

Suppose that you perform an operation on the i-th character, and this character is not at the end of the string. Then, in order for the operation to be valid, the $(i-1)$-th character and the $(i+1)$-th character must differ (otherwise we get a row of three either before or after the operation). This means that unless the operation happens at the end of the string, the number of runs in the string never changes.

Let's draw a red vertical line between `0` and `1`, and a blue vertical line between `1` and `0`. Figure 1 shows an example of the operation with red and blue lines. The red and blue lines appear alternately, and in each operation one of the lines is moved by a unit distance.

Also, in order to handle operations at the ends, assume that there are infinite number of lines at both ends of the string, and red and blue appear alternately. See figure 2.

Now, in terms of red and blue lines, an opeation can be described as follows:

- Red and blue lines appear alternately. You can never change the relative order of these lines.
- Except for the ends, no two line must appear in the same position (i.e., between the same pair of characters).
- The distance between adjacent lines must be either 1 or 2 (except for the ends).
- In each operation, you can choose a line and move it by a unit distance, while keeping the conditions above.

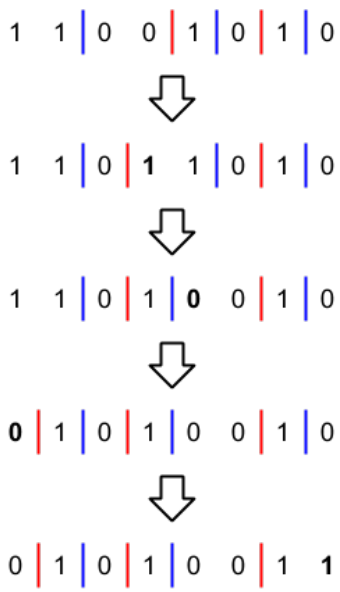

Figure 1:

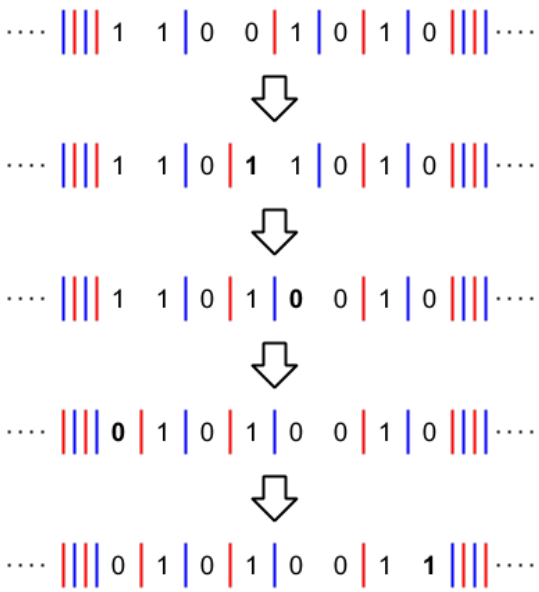

Figure 2:

As in figure 3, suppose that we know which line in s corresponds to which line in t. Clearly, the lower bound of the number of operations required to do this is the sum of distances between each corresponding lines. For example, in the example of 3, $\cdots + 0 + 0 + 0 + 1 + 0 + 1 + 2 + 3 + 2 + 1 + 0 + \cdots = 10$.

It turns out that this lower bound is always achievable. There are three types of lines: moving right, moving left, or staying. Since the distance between two adjacent lines must not exceed 2, a line moving right and a line moving left can't be adjacent. Thus, the lines can be splitted into blocks of "stay, left, ..., left, stay" or "stay, right, ..., right, stay". In both cases it's easy to see that at least one line can be moved to the desired direction.

There only $\mathrm{O}(N)$ ways to decide which lines in s to assign which lines in t. In particular, it's never optimal to assign two or more lines at the left end in s to lines at the right end of t, or vice versa. (Be careful that it's possible that only one such line exists, like $s = \texttt{00}$, $t = \texttt{11}$.)

Therefore, we can try all $\mathrm{O}(N)$ possible ways to assign lines, and this soltuion works in $\mathrm{O}(N^2)$ time in total.

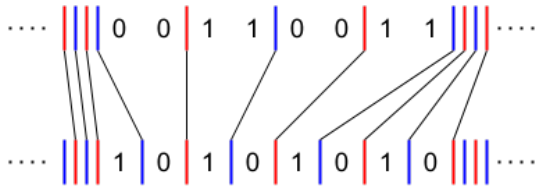

Figure 3:

F: Permutation and Minimum

Suppose there is an index i such that neither A_{2i-1} nor A_{2i} equals -1. Then, B_i is constant regardless of the replacement of A. Hence, we can remove A_{2i-1} and A_{2i} from $\{A_i\}$ and integers A_{2i-1} and A_{2i} from the set of integers to replace $\{A_i\}$, without changing the answer.

Since we are interested only in the number of different sequences $\{B_i\}$, we can make the following assumptions:

- $A_{2i} = -1$ (Otherwise, we can swap A_{2i-1} and A_{2i}).
- If $A_{2i-1} = A_{2i} = -1$ holds, the integer which replaces A_{2i-1} is greater than that which replaces A_{2i}.
- Let $S = \{x_1, \ldots, x_k\}$ $(x_1 < \ldots < x_k)$ be the set of indices i such that $A_{2i-1} = A_{2i} = -1$. Then, the integers which replace $A_{2x_1}, \ldots, A_{2x_k}$ are decreasing in this order.

Please note that, although introducing the last condition changes the number of $\{B_i\}$, we can multiply the number by $|S|!$ to obtain the correct answer.

We decide the position (or absence) of $2N, 2N-1, \ldots, 1$ in $\{B_i\}$, in this order. The following are the possibility of how n appears in $\{B_i\}$ (each possibility is named like $a(n)$):

- $a(n)$: n does not appear either in $\{B_i\}$ or $\{A_i\}$ (before replacing -1).
- $a'(n)$: n does not appear in $\{B_i\}$, but in $\{A_i\}$.
- $b(n)$: n equals B_i, where $A_{2i-1} = A_{2i} = -1$.
- $c_p(n)$: n equals B_i, where $A_{2i-1} = p$ $(p > n)$.
- $d(n)$: n equals B_i, where $A_{2i-1} = n$.

We represent $\{B_i\}$ using the corresponding sequence of appearances of $2N, 2N-1, \ldots, 1$. For example, $(B_1, B_2, B_3) = (1, 3, 2)$ for the Sample Input 1 can be converted to $a(6) \to a(5) \to a(4) \to d(3) \to b(2) \to d(1)$. Here, different $\{B_i\}$ yields different sequence of appearances. Therefore, we can count the number of sequences of appearances which correspond to *correct* $\{B_i\}$ (that is, $\{B_i\}$ which can be found from a valid permutation $\{A_i\}$).

We consider which sequences of appearances correspond to a correct $\{B_i\}$. First, $a(n)$ and $a'(n)$ must appear exactly N times in total, because the length of B is N. Second, if n is in $\{A_i\}$, the appearance of n should be either $a'(n)$ or $c_p(n)$. Similarly, if n is not in $\{A_i\}$, the appearance of n should be either $a(n), b(n)$, or $d(n)$. Finally, in a sequence $\{A_i\}$ which yields $\{B_i\}$, $b(n)$, $c_p(n)$ and $d(n)$ should be paired with an integer greater than n That is, $b(n)$ and $d(n)$ should be paired with $a(n')$ for some $n' > n$, and $c_p(n)$ should be paired with $a'(p)$. Here, an integer cannot belong to more than one pair. On the other hand, if a sequence of appearances satisfies all the conditions above, we can see that it corresponds to a correct $\{B_i\}$.

Given a sequence of apperances, we can check whether there is a valid pairing as follows: for $n = 2N, 2N-1, \ldots, 1$ in this order, if the appearance of n is either $b(n)$, $c_p(n)$, or $d(n)$, we pair it with $a(n')$ or $a'(p)$ which is yet to be paired (the possibility doesn't depend on the choice of n' when there exists multiple $a(n')$ which can be paired with).

We define a function $f(n, x, Y)$ as the number of sequences of appearances *of* $2N, 2N-1, \ldots, n+1$ which satisfies the following conditions:

- The number of unpaired $a(i)$ equals x.
- Y equals the set of $n+1 \leq i \leq n$ such that $a'(i)$ is unpaired.

For a sequence counted by $f(n, x, Y)$, we append an appearance of n at the end of the sequence, and if it is either $b(n)$, $c_p(n)$, or $d(n)$, we pair it with a preceding appearance. The resulting sequences are as follows:

- If n is in $\{A_i\}$:
  - if n appears as $a'(n)$, a sequence counted by $f(n-1, x, Y \cup \{n\})$. (a)
  - if n appears as $d(n)$, a sequence counted by $f(n-1, x-1, Y)$ (only when $x \geq 1$).

- If n is not in $\{A_i\}$:
  - if n appears as $a(n)$, a sequence counted by $f(n-1, x+1, Y)$.
  - if n appears as $b(n)$, a sequence counted by $f(n-1, x-1, Y)$.
  - if n appears as $c_p(n)$, a sequence counted by $f(n-1, x, Y \setminus \{p\})$ (only when $p \in Y$).

Here we let $g(n, x, y) = \sum_{|Y|=y} f(n, x, Y)$. Then, (a) can be seen as a transition from (n, x, y) to $(n-1, x, y+1)$, and (b) can be seen as y distinguishable transitions from (n, x, y) to $(n-1, x, y-1)$. Therefore, each value of g can be computed using dynamic programming. Since n, x and y takes a value in $[0, 2N]$, g can be computed in $O(N^3)$ time. Finally, $g(0, 0, 0)$ is the desired value.