AGC 026 Analysis

sugim48[*] yosupo[†]

2018 年 7 月 16 日

[*] CDEF Author

[†] B Author, Setter, Analysis Writer

A: Colorful Slimes 2

同じ色が奇数匹並んだ区間については，スライムの色を 1 匹飛ばしで適当に違う色に変えればよいです (例: $(2,2,2,2) \to (2,3,2,3)\text{or}(2,1,2,1)\text{or}, ...)$。同じ色が偶数匹並んだ区間についても，スライムの色を 1 匹飛ばしで適当に違う色に変えます (例: $(2,2,2,2,1,...) \to (2,3,2,3,1,...)$)

このように色を変えるのが最適です。つまり，同じ色のスライムが x 匹並んでいたならば，$\lfloor x/2 \rfloor$ 匹の色を変更します。($\lfloor y \rfloor$ で，y の小数以下を切り捨てた値を表すこととします。)

よって解法は，同じ色のスライムが並んでいる区間ごとに列を分割し (例: $(1,1,2,2,2,3) \to (1,1),(2,2,2),(3)$), 各区間ごとに $\lfloor x/2 \rfloor$(x は区間の長さ) を求め，それを足し合わせたものを求めればよいです。計算量は $O(N)$ です。

B: rng_10s

まず，場合分けで以下の 3 つのケースを取り除きます

- $B > A$: 初日でジュースが買えないため，答えは No です
- $(B \leq A), B > D$: 仮に毎日入荷したとしても購入量に追いつけないため，答えは No です
- $(B \leq A, B \leq D), C \geq B$: 買う量がしきい値以下なため、買切る前に必ず入荷が発生し、(入荷の個数は購入量以上なため) 答えは Yes です。

以上より、$B \leq A, B \leq D, C < B$ を仮定してよいです。

次に，個数の遷移を $\bmod B$ で眺めてみます。

- 最初は (個数 $\bmod B$) は $A \bmod B$ 個です。
- すぬけ君が買っても (個数 $\bmod B$) は変化しません。
- 入荷すると (個数 $\bmod B$) は ($\bmod B$ の世界で)$D \bmod B$ 個増えます。

そして、この問題の答えは、(個数 $\bmod B$) が C を超えるタイミングがあるかどうかを判定すればよいことがわかります。このようなタイミングがあれば、その後買えなくなることは簡単にわかり、逆にこのようなタイミングでなければ必ず入荷が発生するため、買いきれません。

以上より、この問題は次のように言い換えられます。

- A からスタートして、D を足していく。個数 $\bmod B$ が C を超えることはあるか？

A からスタートして、D を足していくとき、個数 $\bmod B$ の最大は, $g = \gcd(B, D)$ として $B - g + (A \bmod g)$ となり，これを C と比較すればよいです。個数 $\bmod g$ が常に一定であることを考えるとこれが上界であることは言え、また，$(B/g - 1) \times \mathrm{inv}(D/g, B/g)$ 回 D を足したときにこの上界が達成できます ($\mathrm{inv}(X, Y)$ は，$X \times \mathrm{inv} = 1 \bmod Y$ なる値とします)。

C: String Coloring

文字列を左から N 文字と右から N 文字に分割します。すると，

- 左側で赤く塗られた文字を左から右へ読んだものと，右側で青く塗られた文字を右から左へ読んだものが等しい
- 左側で青く塗られた文字を右から左へ読んだものと，右側で赤く塗られた文字を左から右へ読んだものが等しい

の 2 つが達成された時，またこのときのみに元の問題の条件を達成できることがわかります。

よって，最初に右側の文字列を反転しておき，また右側では赤と青を逆に塗り分けることにすると，条件は以下のようになります。

- 左側で赤く塗られた文字を左から右へ読んだものと，右側で赤く塗られた文字を左から右へ読んだものが等しい
- 左側で青く塗られた文字を右から左へ読んだものと，右側で青く塗られた文字を右から左へ読んだものが等しい

要するに，各色で，左右で塗られた文字たちが全く同じ文字列になるように塗り分ける方法は何通りですか？という問題です。

これは，左右でそれぞれ塗り分け方を全探索し，文字列のペアを key，それを達成する方法を value としたハッシュマップを作れば解くことが出来ます。

ハッシュマップの要素数は高々 2^N なので，$O(N^2 2^N)$ で解くことが出来ます。

D: Histogram Coloring

まずヒストグラムが単一の長方形の場合を考えます。この場合，どのような塗り分け方があるでしょうか？
i 行目より下が全部塗られていて，$i+1$ 行目の塗り分け方を考えているとします。

- (i 行目の色に関わらず，)$i+1$ 行目の色として，i 行目の色を全部反転させたものを塗る
- i 行目の色が赤青赤青..., もしくは青赤青赤... の時，$i+1$ 行目を i 行目の色と全く同じように塗る

この 2 パターンの塗り分け方があることがわかり，またこの場合が全てであることがわかります。

以上の考察を使い，まずヒストグラムを，(最小の値を求め，それを全体から引き，0 の要素で左右に分割する) を繰り返し，長方形からなる木に分解します。

そして木のノードごとに，

- $dp1$:= 長方形とその上の長方形たちの塗り分け方の総数，ただし一番下の列は赤青赤青..., 青赤青赤...,のどちらかの塗り方でないといけない
- $dp2$:= 長方形とその上の長方形たちの塗り分け方の総数

の 2 種類の値を持って，木 DP を行います。

長方形 h の高さを x，子を $c_1, c_2, ..., c_k$，h の幅から子の幅を引いたものを w とすると，遷移式は

- $dp1_h := 2^x \Pi_{i=1..k} dp1_{c_i}$
- $dp2_h := 2^w \Pi_{i=1..k} (dp1_{c_i} + dp2_{c_i}) + (2^x - 2) \Pi_{i=1..k} dp1_{c_i}$

となります。

計算量は木を作るのに $O(N^2)$，その後の木 DP で $O(N \log h)$ です。ただし木は $O(N)$ で作ることも出来ます。

E: Syncronized Subsequence

$S = b_1b_2a_1b_3a_2a_3$ と表記します。

まず，$a = -1, b = 1$ と考え累積和を取り，累積和が 0 になるところで，文字列を分割します。すると，異なる文字列どうしの間に a, b のペアが存在することがないため，取れる部分列というのは文字列ごとに独立になります。もちろん最後にくっつけるため，どのような部分列を文字列ごとに取ればよいのか，というのは非自明なのですが，実は空文字列 or 辞書順最大の文字列の 2 パターンだけ考えればいいことが示せます。

よって，文字列を最初に分割し，独立に元の問題を解けばいいです。

■最初の文字が'a' 仮定より，全ての i について i 番目の a は i 番目の b より先に出現します。

答えの文字列の形は必ず *ababab...* という形になっています。このような形でないならば，必ず a が隣り合ったところが存在するはず (最初の文字が a で最後の文字が b であることを考えると，鳩の巣原理より) ですが，この場合 $...a_ia_{i+1}...b_i...b_{i+1}...$ という位置関係になっているので，$i+1$ を消したほうが必ず良くなるからです。

よって，最大でどのぐらいの長さの *ababab...* を作れるかを考えればよいです。これは，a_1, b_1 間の a と，それに対応する b をすべて消し，$a_1b_1...$ という文字列が出来るのですが，この ... について同じことをやる，というのを繰り返せばよいです。

■最初の文字が'b' 仮定より，全ての i について i 番目の b は i 番目の a より先に出現します。

残す i のうち最小を決めたとします。$1, 2, ..., i-1$ を削除したら，b_i と a_i の間には b しか残りません。この b(と，それに対応する a) は当然全て残すべきです。また，この操作により新たに残すことになった a たちの前に b が存在したら，それも当然全て残すべきです。…というのを繰り返していくと，実は累積和が最初と最後以外 0 にならないという仮定より，i 以降全てを残すのが最適であることがわかります。よって，i を全部決め打ち，最も辞書順が大きい文字列を返せばよいです。

最後の結合ですが，後ろから使うか使わないかを順に決め，DP をすれば $O(N^2)$ で解くことが出来ます。今回は文字列の形が特殊で，greedy でも解くことが可能で，この場合 $O(N)$ になります。

計算量は $O(N^2)$ ですが，実は $O(N)$ で解くことも可能です (ただし，Suffix Array を線形時間で構築できることを要求します)。

F: Manju Game

■N が偶数　N が偶数の場合は証明は省略しますが、偶数番目の sum と奇数番目の sum の max が先手の取れる最大です。

■N が奇数　偶奇が重要なため，数列を $a_1b_1a_2b_2...b_ma_{m+1}$ と表すことにします。

- sugim 君は，もし a を選択するならば，端の a を選択し，$\sum a$ を回収してゲームを終了するのが最善です。これは端以外の a を選択した場合であっても，sigma 君は少なくとも $\sum b$ は回収できてしまうことから言えます。
- sugim 君が b を選択した場合，sigma 君が左右どちらかを選び，選んだほうの a を sigma 君が，b を sugim 君が回収し，そして選ばなかった方の数列で同じゲームをすることになります。

以上の考察より，このゲームは，

- sugim 君は，b を 1 つ選択する，そして sigma 君は左右のどちらかを選択する，選択した方の数列 (と sugim 君が選んだ b) は切り落とし，残った数列で同じことをする。
- sugim 君は b を選択せずゲームを終了することもできる。その時は $\sum_{\text{残った数列}} a - \sum_{\text{残った数列}} b + \sum b$ を最終得点として獲得する。

と言い換えられる事ができます。また，sugim 君は $Z = \sum_{\text{残った数列}} a - \sum_{\text{残った数列}} b$ を最大化するゲームとして考えて良いです。ここで、答えで二分探索を行い，ある X について $sugim$ さんが $Z \geq X$ と出来れば勝ちで、そうでなければ負けという勝ち負けの付くゲームへ変換します。すると，この問題は以下のように解けます

■数列の長さが 1 の場合　その要素が X より大きければ sugim の勝ちです。

■数列の長さが 3 以上の場合　左端の 2 要素 a_1, b_1 に注目します。

もし $a_1 \geq X$ かつ $b_1 \geq a_1$ だとしましょう。この場合，sugim 君が最適に行動すると，絶対に a_1, b_1 は切り落とされた状態でゲームが終了するとしてよいです。なぜならば，a_1, b_1 を切り落とさないままゲームを終了する場合，b_1 を選択し，a_1, b_1 を切り落としてからゲームを終了してもスコアが低くならないからです ($a_1 \geq X$ より，sigma 君は b_1 を選択された時に左しか選べないことに注意して下さい)。

よって，sugim 君は最初に b_1 を選択して a_1, b_1 を切り落としてしまって良く，a_2 以降について同じ問題を解けばよいです。

では次に，逆のパターン，つまり $a_1 < X$ または $b_1 < a_1$ の場合を考えます。

- $a_1 < X$: b_1 を選択して sigma が右を選択すると負けるため，明らかに sugim は b_1 を選択出来ません。
- $(a_1 \geq X), b_1 < a_1$: b_1 を選択すると，sigma 君は左を選択するしかなく，a_1, b_1 が切り落とされます。しかし $b_1 < a_1$ より，最適に行動した時，a_1, b_1 が切り落とされており a_2 が切り落とされていないという状態になることはありません。よって sugim は b_1 を選択する旨味がなく，選択しないものとしてよいです。

よって，sugim 君は結局 b_1 を選択することはありません。つまり，a_1, b_1, a_2 は全部切り落とされるか全部生

き残るかのどちらかなので，$a_2 = a_1 + a_2 - b_1$ とし，a_1, b_1 を切り落とし，同じ問題を解けばよいです。

以上より二分探索後の問題は $O(N)$ で解け，まとめて $O(N \log(\sum a))$ で解けます。

A: Colorful Slimes 2

If multiple slimes of the same color are adjacent, we shuold change the color of all even-indexed slimes. For example, $(2, 2, 2, 2) \rightarrow (2, 3, 2, 3)$ or $(2, 2, 2, 2, 2, 1...) \rightarrow (2, 3, 2, 3, 2, 1, ...)$.

Generally, if there is a run of x slimes of the same color, we should change the color of $floor(x/2)$ slimes.

Thus, we should split the array into runs (for example, $(1, 1, 2, 2, 2, 3) \rightarrow (1, 1), (2, 2, 2), (3)$), and the answer is the sum of $\lfloor length/2 \rfloor$ for each part. This solution works in $O(N)$.

B: rng_10s

There are two types of obvious cases:

- $B > A$: since we can't buy the juice in the first day, the answer is "NO".
- $(B \leq A), B > D$: the number of juice in the evening is monotonously (strictly) decreasing, so the answer is "NO".

Thus, assume that $B \leq A, B \leq D$.

The number of cans of juice change as follows:

- Initially, it starts with $A - B$.
- In case the number is x at some evening, the number in the next evening is $x - B$ if $x > C$, and $x - B + D$ if $x \leq C$.

Thus, for sufficiently large i, we know the following about the number of cans in the i-th evening:

- In modulo D, it is equivalent to $A - iB$.
- It is in the range $[C - B + 1, C - B + D]$.

From these two conditions we can uniquely determine the number of cans, and its minimum is the minimum number x such that

- $x \geq C - B + 1$.
- $x \equiv A \pmod{gcd(B, D)}$.

The answer is "YES" when this x is non-negative.

C: String Coloring

Let's cut the string into two halves (N characters each).

The condition in the statement can be restated as follows:

- The string we get by reading all red letters in the left part from left to right is equal to the string we get by reading all blue letters in the right part from right to left.
- The string we get by reading all blue letters in the left part from right to left is equal to the string we get by reading all red letters in the right part from left to right.

Let's reverse the colors of letters in the right half, and also reverse the order of letters in the right half. Then, this condition beocomes the following:

- The string we get by reading all red letters in the left part from left to right is equal to the string we get by reading all red letters in the right part from left to right.
- The string we get by reading all blue letters in the left part from left to right is equal to the string we get by reading all blue letters in the right part from left to right.

For each half, try all possible colorings (there are 2^N ways), and for each coloring assign a pair of two strings (the red letters from left to right, the blue letters from left to right). We can solve the problem by putting these pairs into a map.

This solution works in $O(N^2 2^N)$ time.

D: Histogram Coloring

Suppose that the histogram is a reactangle. How can we color it in this case?

Suppose that we decided the colors for all cells in row i, and we now want to color cells in row $i+1$ (while satisfying constraints for all 2x2 squares on these two rows). There are two possible ways:

- (Regardless of the colors of the row i), for each j, the color of $(i+1, j)$ is different from the color if (i, j).
- In case row i is either RBRB... or BRBR..., for each j, the color of $(i+1, j)$ is the same as (i, j).

This suggests the following solution. For a histogram H, define the following two values:

- $dp1(H)$:= The number of ways to color this histogram, with the additional constraint that the bottommost row of H must be either RBRB... or BRBR...
- $dp2(H)$:= The number of ways to color this histogram.

Keep these two values and do DP.

Consider a histogram H. We'll decompose it into multiple sub-histograms, compute the values of $dp1$ and $dp2$ for each of the sub-histograms, and then compute $dp1(H)$ and $dp2(H)$.

Let x be the height of the shortest column of H. If we remove the bottom x rows from H, it may be splitted into multiple parts: call them $c_1, c_2, ..., c_k$. Let w be the number of completely removed columns (i.e., the number of columns with height exactly x).

Then, we can compute the values of $dp1(H)$ and $dp2(H)$ using the following recursion:

- $dp1_h := 2^x \Pi_{i=1..k} dp1_{c_i}$
- $dp2_h := 2^w \Pi_{i=1..k} (dp1_{c_i} + dp2_{c_i}) + (2^x - 2) \Pi_{i=1..k} dp1_{c_i}$

This solution works in $O(N \log h)$ time (or $O(N^2)$, depending on the way you decompose the histogram).

E: Syncronized Subsequence

We call the i-th occurrence of 'a' "a_i" (and define b_i similarly).

Let's split the string into as many parts as possible, such that each part contains the same number of 'a's and 'b's. For each i, a_i and b_i belong to the same part.

First, let's consider a single part. There are two cases in a single part:

- For all i, $a_i < b_i$.
- For all i, $a_i > b_i$.

(Otherwise the part can be splitted into multiple parts).

First, assume that the entire string contains a single part.

■In case $a_i < b_i$ for all i The answer will always be of the form $ababab...$.

Otherwise, there must be two consecutive 'a's in the answer (because the answer string starts with 'a', ends with 'b', and contains the same number of 'a's and 'b's). It will look like $...a_i a_j ... b_i ... b_j ...$. However, in this case we can get a better result by erasing both a_j and b_j.

What is the longest possible length of the string of the form $ababab...$ we can get? This can be done greedily: we should choose a_1 and b_1, then we should choose a_i and b_i where i is the minimum index such that a_i is to the right of b_1, and so on.

■In case $a_i > b_i$ for all i Suppose that we choose a_i and b_i, but not a_{i+1} and b_{i+1}, for some i. In this case, we can always improve the result by inserting both a_{i+1} and b_{i+1}. (Notice that these characters appear in the order $...b_i...b_{i+1}...a_i...a_{i+1}...$).

Thus, in the optimal answer, we should choose $a_i, b_i, a_{i+1}, b_{i+1}, ...$ (i.e., all characters indexed with i or greater). We can get the optimal answer by trying all values for i.

What should we do when the string contains multiple parts?

We split the string into parts, and for each part, we get the optimal answer as described above. Then, for each part, we should either choose the optimal answer or choose an empty string. This is because, in case of $a_i > b_i$, all other strings we can get is not a prefix of the optimal string.

Thus, by doing DP from right from left, we can get the optimal string in $O(N^2)$. (It's also possible to do it in $O(N)$ if we use the property of strings we get this way: we should choose a string s if it's greater than or equal to all strings after that.)

This solution works in $O(N^2)$ time ($O(N)$ is also possible with Suffix Arrays).

F: Manju Game

Let's color the boxes black and white. The 1st, 3rd, 5th, ... boxes are black, and the 2nd, 4th, 6th, ... boxes are white. Let B be the sum of values in all black boxes and let W be the sum of values in all white boxes.

■In case N is even If the first player takes the leftmost (rightmost) box, all remaining moves will be forced and the first player's score will be B (W). Thus, the first player can make sure to get at least $max(B, W)$ points.

Suppose that the first player takes a black box in the first turn. Then, if the second player takes its left box, all moves will be forced until all boxes to the left are taken; then, if the second player takes the rightmost box, he can make sure to get at least W points. Similarly, if the first player takes a white box in the first turn, the second player can ensure B points.

Thus, the first player's score will be $max(B, W)$ and the second player's score will be $min(B, W)$ in this case.

■In case N is odd By a similar observation, we get the following:

- The first player can ensure B points.
- In case the first player takes a black box in the first turn, the second player can ensure W points.

Thus, the main question is whether the first player can get more than B points by taking a white box in the first turn. If the first player starts with a white box, the game will proceed as follows:

- Initially, the first player chooses a white box and takes it (call it p).
- The second player chooses "left" or "right".
- For example, if the second player chooses "left", the game proceeds as follows. The first player takes p and all white boxes to the left of p. The second player takes all black boxes to the left of p. Then, they start the same game with all boxes to the right of p, starting from the first player.

In this game, the strategy of the first player can be completely described using a binary tree. The root of the tree corresponds to the white box the first player takes in the first turn. The root's left subtree corresponds to the game in case the second player chooses "right" in his first turn - i.e., the root's left child is the white box the first player takes in his second turn (among all boxes to the left of the root), in case the second player chooses "right" in his first turn. The root doesn't have a left child if the first player wants to start taking black boxes in this case. Similarly, define its right subtree.

Suppose that a binary tree (that describes the first player's strategy) is given. How many points can he ensure with this strategy?

Let $W_1, \ldots, W_k$ be the white boxes that appear in this tree, from left to right. The first player will take all these boxes, no matter what the second player does. These k boxes split the sequence of boxes into $k+1$ parts. For one of these parts, the first player takes all black boxes and the second player takes all white boxes. For all other parts, the first player takes all white boxes and the second player takes all

black boxes. The choice of the "one part" depends on the second player's strategy.

Let's fix a parameter X. The first player can ensure $W + X$ points if he can choose white boxes $W_1, \ldots, W_k$ such that for each of $k+1$ parts we get when we split the sequence with chosen white boxes, (The sum of black boxes in the part) minus (The sum of white boxes in the part) is at least X.

Let's do a binary search on X. We can check if we can choose white boxes as described above in $O(N)$ by a simple DP. This solution works in $O(N \log(\sum a))$ time in total.