AGC040 解説

writer : maroonrk

For International Readers: English editorial starts on page 7.

A : ><

$a_{i-x} < a_{i-x+1} < \cdots < a_i$ のとなっていれば，a_i は x 以上でなくてはなりません．また同様に，$a_i > a_{i+1} > \cdots > a_{i+x}$ のときも，a_i は x 以上でなくてはいけません．

すると，a の各項について，その値は，左に連続する $<$ の個数と右の連続する $>$ の個数の max 以上である必要があるとわかります．逆に，左に連続する $<$ の個数と右の連続する $>$ の個数の max とすれば，条件をみたす数列が得られます．

よって，各項の最小値がわかり，その総和が答えになります．

解答例

B : Two Contests

問題は，区間の集合を 2 つの集合に分割して，それぞれの共通部分の長さの和を最大化することです．以下，区間は全て半開区間だと思って扱います．（つまり，R_i の値を入力 $+1$ だとします）

L_i の最大値を達成する i を p，R_i の最小値を達成する i を q とします．

区間 p と区間 q を同じ集合に含める場合を考えます．区間 p と q を含む集合に，他のどの区間を入れても，この集合の区間の共通部分は減りません．よってこの場合，区間 p,q を含まない方の集合は，区間 p,q 以外でもっとも長い区間 1 個にするのが最適です．（実は，区間 p, q 以外で，という部分を考えなくても正しい答えが出るので，writer 解はそのように実装しています）

区間 p と区間 q を違う集合に含めることを考えます．区間 p と同じ集合に入った区間が，区間 $[l_1, r_1), [l_2, r_2) \cdots$ だとすると，その集合の区間の共通部分の長さは，$min_{(1 \leq i)}\{max(r_i - L_p, 0)\}$ となります．

同様に，区間 q と同じ集合に入った区間が，区間 $[l_1, r_1), [l_2, r_2) \cdots$ だとすると，その集合の区間の共通部分の長さは，$min_{(1 \leq i)}\{max(R_q - l_i, 0)\}$ となります．

よって，次のような問題が解ければ良いです．

> $a_i = max(r_i - L_p, 0), b_i = max(R_q - l_i, 0)$ とする．$\{1, 2, \cdots, N\}$ を 2 つの集合 s, t に分割し，$min_{(i \in s)}\{a_i\} + min_{(i \in t)}\{b_i\}$ を最大化せよ．ただし，$p \in s, q \in t$ とする．（実は $p \in s, q \in t$ を無視しても正しい答えが出るので，writer 解はそのように実装しています）

この問題の解は，i を a_i の昇順（タイブレークは b_i の降順）に並べたあと，先頭の何個かを t に，残りを s に入れるという形です．よって，ソートして前から見ていくことで解くことが出来ます．

以上より，この問題は $O(NlogN)$ 時間で解くことが出来ます．

解答例

C : Neither AB nor BA

奇数番目の文字の ‘A‘,‘B‘ を入れ替えると，行える操作は，‘AA‘，‘BB‘ 以外を選んで消すことになります．この時，s を空にできる必要十分条件は，‘A‘,‘B‘ の個数がどちらも全体の半分以下であることです．必要性は明らかです．十分性は，‘A‘,‘B‘ のうち個数の多い方を優先的に消すことを考えればわかります．

よって，答えは，$3^N -$ ‘A‘ が過半数になる文字列の個数 $-$ ‘B‘ が過半数になる文字列の個数 になります．‘A‘ が過半数になる文字列の個数は，‘A‘ の個数 k $(k > N/2)$ を全通り試せば，あとは簡単なコンビネーションの計算でわかります．

よって，この問題は $O(N)$ で解けます．

解答例

D : Balance Beam

$S = \sum_{i=1}^{N} A_i$ とします.

平均台の並べ方を固定して考えます．明らかに，ある距離 p であって，りんごさんのスタート位置が左端から p 以下ならすぬけくんが勝ち，そうでないならりんごさんが勝つ，というものが存在します.

x 軸に平均台の左端からの距離，y 軸に時間をとって，すぬけくんの移動を 2 次元平面上に折れ線として書いてみます．同様に，りんごさんの動きも折れ線として書くことにします．このとき，りんごさんの折れ線を上下に自由に動して，すぬけくんの折れ線に下側からちょうど触れるように書いてみます．このときのりんごさんの折れ線と，x 軸の交点が，座標 $(p, 0)$ の点になっています.

座標 $(p, 0)$ を含む平均台 k を固定したとします．この時，p の値を最大化することを考えます．そこで，次のような経路 C を考えます.

- 座標 $(p, 0)$ から出発し
- りんごさんの折れ線上を (右上方向へ) すすみ
- すぬけくんの折れ線と触れた地点からすぬけくんの折れ線上をすすみ
- 座標 (N, S) に至る経路

平均台 k より後ろに置かれた平均台 i について，経路 C が平均台 i 上に対応する部分でどれだけ上方向へ移動するか考えます．この値は明らかに $max(A_i, B_i)$ 以下です．そしてこの上界は，達成することができます．具体的には，$A_i \leq B_i$ となる平均台を前に，$A_i > B_i$ となる平均台を後ろに並べれば良いです.

よって，平均台 k を固定したあとは，$max(A_i, B_i)$ の大きいものから貪欲に使うことを考えれば良いです．具体的には，$max(A_i, B_i)$ の大きい順に足していって（ただし $i = k$ は除外），総和が $S - B_k$ 以上になるところを見つければ良いです．ここで，平均台 k に対応する部分では，経路 C は上方向に高々 B_k しか移動しないことに注意する必要があります.

上記のアルゴリズムをすべての k について行うようにするのも容易にできます．よってこの問題は $O(NlogN)$ で解けます.

解答例

E : Prefix Suffix Addition

便利のため，$A_0 = 0, A_{N+1} = 0$ とします．操作 1 だけで数列を作るときの最小の操作回数は，$A_i > A_{i+1}$ となる i の個数になります．必要性は明らかです．十分性は，$A_i > A_{i+1}$ をみたす最小の i について，$c = (A_1, A_2, \cdots, A_i)$ で操作することを考えれば分かります．

同様に，操作 2 だけで数列を作るときの最小の操作回数は，$A_i < A_{i+1}$ となる i の個数になります．

よってこの問題は，次のように言い換えられます．

> 非負整数列 x, y を $A_i = x_i + y_i$ をみたすように作る．「$x_i > x_{i+1}$ なる i の個数」＋「$y_i < y_{i+1}$ なる i の個数」を最小化せよ．

よって，次のような DP が考えられます．

- $dp[i][j]$ = 先頭 i 項を既に決定していて，$x_i = j$ としたときのコスト．

この dp を愚直に行ってはとても間に合いません．しかし，この dp の状態は圧縮できます．まず，どの i についても，$dp[i][j]$ は j に対して広義単調減少であることがわかります．さらに，$dp[i][0] \leq dp[i][A_i] + 2$ であることもわかります．この正当性は，$dp[i][A_i]$ を達成する解を考え，x_i だけ 0 に変更することを考えれば示せます．

よって，$dp[i]$ の情報は，$dp[i][0]$ の値と，$dp[i][j-1] > dp[i][j]$ となる j の値（これは 高々 2 つしかない）さえ覚えておけばよいということになります．

この値の遷移は $O(1)$ で行えます．よってこの問題は全体で $O(N)$ で解けます．

解答例

F : Two Pieces

2 つの駒の座標が 2 以上離れているときのみ，座標の小さい駒を 1 動かすという操作が行えることにします．すると，単に異なる操作列が何個あるか数える問題になります．

駒の状態を，座標の大きい駒の座標 x，2 つの駒の距離 d の組 (x, d) で表すことにします．各操作は，以下のように言い換えられます．

- 操作 1: x, d を 1 増やす．
- 操作 2: d を 1 減らす．$d \geq 2$ の時のみ行える．
- 操作 3: d を 0 にする．

操作 1 の回数は合計でちょうど B 回です．操作 2 の回数 k を全通り試すことにします．

操作 1,2 の順番を固定したとします．このとき，1,2 の順番は，d の値が（初期状態を除いて）常に 1 以上であるようにします．

ここに，$N - B - k$ 個の操作 3 を挿入することを考えます．なお，$N = B + k$ の場合は明らかなので，$N - B - k > 0$ を仮定します．すると，「d の値がこれ以降，現在以下の値にならない」タイミングでのみ操作 3 を行えるとわかります．例えば，操作 1,2 を $(1, 1, 2, 1, 1)$ と行った場合，操作 3 を行えるのは，$0, 3, 4, 5$ 回目の操作のあととなります．それ以外のタイミングで操作 3 を行うと，操作 2 を行うタイミングで $d < 2$ となるためだめです．逆に，上記の条件をみたすタイミングなら，好きな場所に好きな個数操作 3 を挿入できます．操作 3 を挿入し終えたあと，最終的な d の値が $B - A$ になる必要があるので，(操作 1,2 だけを考えたときの)d の値が $A - k$ になったタイミングでは，必ず操作 3 を行う必要があります．そして，残りの $N - B - k - 1$ 個の操作 3 は，d の値が最後に $0, 1, ...A - k$ になったタイミングのうちどこかで行えば良いです．

以上より，k を固定すると，操作 1,2 の順番と，操作 3 をどう挿入するかは独立に考えることが出来ます．操作 1,2 の順番の決め方は，カタラン数と同じように求められます．操作 3 の挿入の仕方は，（区別のつかない）$N - B - k - 1$ このボールを（区別のつく）$A - k + 1$ 個の箱に入れる方法は何通りか，という問題なので，これもコンビネーションの計算で求められます．

よって，この問題は全体で $O(N)$ で解けます．

解答例

AGC040 Editorial

by maroonrk

A : ><

If $a_{i-x} < a_{i-x+1} < \cdots < a_i$, a_i must be at least x. Similarly, if $a_i > a_{i+1} > \cdots > a_{i+x}$, a_i must be at least x.

Thus, for each element of a, its value must be at least max(the number of consecutive '<' to the left, the number of consecutive '>' to the right). On the other hand, if each element of a is max(the number of consecutive '<' to the left, the number of consecutive '>' to the right), the condition is satisfied.

Thus, the answer is the sum of these lower limits.

Sample Code

B : Two Contests

The problem asks us to separate segments into two sets, maximizing the sum of the lengths of the intersections of those individual sets. Below, we consider intervals to be half-open (we add 1 to the given value of R_i).

Let p be the value of i that maximizes L_i, and q be the value of i that minimizes R_i.

Consider when Segment p and Segment q belong to the same set. Adding any other segment to this set will not shorten the intersection of this set. Thus, in this case, it is optimal to make the other set contain just the longest segment except Segment p and q. (Actually, we can omit the part "except Segment p and q" and still obtain the correct answer, so the writer's code does so.)

Now, consider when Segment p and Segment q belong to different sets. When the segments belonging to the set of Segment p are $[l_1, r_1), [l_2, r_2) \cdots$, the length of the intersection of that set is $\min_{(1 \leq i)}\{\max(r_i - L_p, 0)\}$. Similarly, when the segments belonging to the set of Segment q are $[l_1, r_1), [l_2, r_2) \cdots$, the length of the intersection of that set is $\min_{(1 \leq i)}\{\max(R_q - l_i, 0)\}$.

Thus, we want to solve the following problem:

> Let $a_i = \max(r_i - L_p, 0), b_i = \max(R_q - l_i, 0)$. Separate $\{1, 2, \cdots, N\}$ into two sets s and t, maximizing $\min_{(i \in s)}\{a_i\} + \min_{(i \in t)}\{b_i\}$. Here, $p \in s, q \in t$ must hold. (Actually, we can omit the part "$p \in s, q \in t$" and still obtain the correct answer, so the writer's code does so.)

The solution to this problem has the following form. First, sort the indices i in ascending order of a_i (ties are broken in descending order of b_i), then put some number of indices from the beginning to t, and put the others to s. Thus, we can solve it by actually sorting the indices in this way and try all candidates of the solution in this form one by one.

Therefore, the problem can be solved in $O(N \log N)$ time.

Sample Code

C : Neither AB nor BA

For odd positions in s ($s_1, s_3, s_5, \ldots$), let us replace each 'A' with 'B' and vice versa. Now we can erase any two characters except 'AA' and 'BB'. Here, s can be emptied if and only if the numbers of 'A's and 'B's are both at most $N/2$. The necessity is obvious. To see the sufficiency, consider prioritizing erasing 'A' over erasing 'B' when there are more 'A's than 'B's, and vice versa.

Thus, the answer is $3^N -$(the number of strings with more than $N/2$ 'A's)$-$(the number of strings with more than $N/2$ 'B's). We can find the number of strings with more than $N/2$ 'A's by trying all possible counts k ($k > N/2$) of 'A's and simple calculations with binomial coefficients.

Therefore, the problem can be solved in $O(N)$ time.

Sample Code

D : Balance Beam

Let $S = \sum_{i=1}^{N} A_i$.

Let us fix the order in which beams are arranged. Obviously, there exists a distance p such that Snuke wins if and only if the initial position of Ringo is at most p meters to the right of the left end of the long beam.

Let us draw Snuke's motion as a polyline in a two-dimensional coordinate plane. The x- and y-coordinate correspond to the distance from the left end of the long beam and time, respectively. We will also draw a polyline for Ringo's motion. Let us set the vertical position of Ringo's polyline so that it touches Snuke's polyline from the lower side. Now, Ringo's polyline crosses the x-axis at coordinates $(p, 0)$.

Let Beam k be the beam that contains the point $(p, 0)$, and assume that k is fixed. Let us maximize p here. Consider the path C that:

- starts at $(p, 0)$,
- goes along Ringo's polyline (towards upper-left),
- after the intersection of the two polylines, goes along Snuke's polyline,
- and reach (N, S).

For Beam i placed somewhere to the right of Beam k, let us consider how much does the path C go up in the part corresponding to Beam i. This value is obviously at most $\max(A_i, B_i)$, and we can achieve this upper limit, by putting beams such that $A_i \leq B_i$ to the left and beams such that $A_i > B_i$ to the right.

Thus, after fixing Beam k, we can greedy use beams in descending order of $\max(A_i, B_i)$ from left to right. More specifically, take the beams except Beam k one by one in descending order of $\max(A_i, B_i)$, and find out when the sum gets $S - B_k$ or greater. Here, note that the path C only goes up by B_k in the part corresponding to Beam k.

We can easily modify the algorithm above to do the computation for all k. Therefore, the problem can be solved in $O(N \log N)$ time.

Sample Code

E : Prefix Suffix Addition

For convenience, let $A_0 = 0, A_{N+1} = 0$. If we only use Operation 1, the number of operations required to make the sequence is the number of i such that $A_i > A_{i+1}$. The necessity is obvious. To see the sufficiency, consider doing the operation choosing $c = (A_1, A_2, \cdots, A_i)$ for the minimum i such that $A_i > A_{i+1}$.

Similarly, if we only use Operation 2, the number of operations required to make the sequence is the number of i such that $A_i < A_{i+1}$.

Thus, the problem can be rephrased to the following:

> Make two sequences x and y of non-negative integers such that $A_i = x_i + y_i$, minimizing (the number of i such that $x_i > x_{i+1}$) + (the number of i such that $y_i < y_{i+1}$).

This can be solved by the following DP:

- $dp[i][j]$ = The minimum cost when setting the first i elements in x and y so that $x_i = j$.

It is too slow to do it naively, but we can compress the states in this DP. First, for each i, $dp[i][j]$ is non-increasing for increasing j. Additionally, we can see that $dp[i][0] \leq dp[i][A_i] + 2$, which can be validated by modifying a solution achieving $dp[i][A_i]$ so that only x_i is changed to 0.

Thus, the only information we only have to memorize for $dp[i]$ is $dp[i][0]$ and the values of j such that $dp[i][j-1] > dp[i][j]$ (there are at most two such values).

The transition from $dp[i]$ to $dp[i+1]$ can be done in $O(1)$ time, so the whole problem can be solved in $O(N)$ time.

Sample Code

F : Two Pieces

Let us assume that the operation of moving the piece with the smaller coordinate by 1 is only possible when the distance between the two pieces is 2 or greater. Then, we just have to count the number of different sequences of operations.

Let us represent the state of the pieces as (x, d), where x is the greater of the pieces' coordinates, and d is the distance between the two pieces. The possible operations are now rephrased as follows:

- Operation 1: decrement both x and d by 1.
- Operation 2: decrement d by 1. Only possible when $d \geq 2$.
- Operation 3: set d to 0.

We have to do Operation 1 exactly B times. Let us try all candidates for the number of times we do Operation 2 (let this number be k).

Assume that the order in which we do Operation 1 and 2 is fixed. This order should be chosen so that, considering only those operations, d will be always 1 or greater (except at the beginning).

Now, let us insert Operation 3 here $N - B - k$ times. Below, we will assume $N - B - k > 0$, since the case $N = B + k$ is obvious. We can see that Operation 3 can only be inserted when the following holds: "the value of d will never be equal to or less than its current value again." For example, if we do Operation 1 and 2 in the order $(1, 1, 2, 1, 1)$, we can only insert Operation 3 just after the 0-th, 3-rd, 4-th, and 5-th of those operations. If we did Operation 3 at other times, we would have $d < 2$ when we do Operation 2, so it is not allowed. On the other hand, we can insert Operation 3 any number of times to any positions satisfying the condition above. After we finish inserting Operation 3, the final value of d needs to be $B - A$, so we must always insert Operation 3 where the value of d becomes $A - k$ for the last time (considering only Operation 1 and 2). The remaining $N - B - k - 1$ insertions of Operation 3 can be done to any of the positions where the value of d becomes $0, 1, ...A - k$ for the last time.

Thus, when we fix k, we can separately deal with the order in which we do Operation 1 and 2 and the positions to which we insert Operation 3. The number of possible orders in which we do Operation 1 and 2 can be found similarly to how the Catalan numbers are found. The number of possible choices to insert Operation 3 can be found by calculations with binomial coefficients, since it is equivalent to the problem of finding the number of ways in which we put $N - B - k - 1$ indistinguishable balls into $A - k + 1$ distinguishable boxes.

Therefore, the whole problem can be solved in $O(N)$ time.

Sample Code