ARC101-D「Median of Medians」

●「先輩、何見てるんですか？」

■「ちょっと見かけた問題を考えてるんだけど解けなくて。この問題解いたことある？」

●「えっと、『Median of Medians』、ですか。その回は出てなかったみたいです。その問題がどうかしたんですか？」

■「この問題が、黄色コーダーの半分ぐらいが解ける*って言われてたから気になって。僕も考えて見たけど、とっかかりが分からないんだよね」

●「じゃあ考えてみましょうか。」

> * AtCoder（競技プログラミング）の色・ランクと実力評価、問題例 – chokudaiのブログ

●「与えられた配列の、全ての**連続する部分列**について**中央値**を計算して、その**中央値たちのさらに中央値**が何になるか、ですか」

■「元の配列が {10, 30, 20} なら、{10}, {30}, {20}, {10, 30}, {30, 20}, {10, 30, 20} の全部について中央値を計算するんだね。{10, 20} は連続していないから関係ないね」

●「そうですね。中央値がそれぞれ 10, 30, 20, 30, 30, 30 になりますね。**偶数個の場合、真ん中2つのうちの大きい方にする**ことに注意です。だから {10, 30} の中央値が 30 に」

■「それで、中央値たちの中央値が今回は 30 になるんだね」

●「はい。もちろん愚直に計算すると間に合わないですよね」

■「部分列が N*(N+1)/2 個あって、中央値の計算がソートするなら O(NlogN) とかだから、O(N^3*logN) とかになるのか。N≦10^5 だから、到底間に合わないね」

●「うーん、とりあえず中央値の定石を使ってみたいですね」

●「中央値を『小さい方から M/2+1 番目』のように考えるのは、競プロでは筋が悪い場合が多いです」

■「じゃあどう考えるの？」

●「**『中央値が k 以下である』**と『 **k 以下の数が M/2 個より多い**』が同値であることを利用します。M は配列の長さです」

■「えっと、確かに同値みたいだね。中央値が k 以下ってことは k 以下の数が半分より多くあるってことだし、 k 以下の数が半分より多かったら、ソートして真ん中の数は k 以下になるね」

●「さらに、**真の中央値**は、この条件を満たす k の中で最小の数になります」

■「中央値が **k 以下で、かつ k-1 以下ではない**ってことだね」

●「そうすると、**二分探索**が使えます」

■「確かに。OK=とても大きい数, NG=0 とかにしておいて、OK-NG=1 になるまで二分探索で狭めたときの、OKの値が中央値だね」

●「中央値の問題では、この考え方をベースにすると見通しが良くなる場合が多いです。」

■「なるほど、勉強になるね」

●「さて、今回の問題について考えましょう。全ての連続部分列の中央値のうち、k 以下の値がいくつあるのかが計算できれば良いです」

■「つまり、『**中央値が k 以下になるような連続部分列はいくつありますか**』という問題に答えられたらいいんだね」

●「はい。さらに、部分列の中央値についてもさっきの考え方を利用します」

■「**今注目する部分列に含まれる k 以下の数の個数**を数えれば良いんだね」

●「区間についてカウントするので、累積和を使いましょう」

■「 **k 以下の値を 1、k より大きい値を 0 にした数列**を用意して、累積和を計算すれば、引き算だけでカウントできるってことだよね」

●「そうですね。例えば、入力例2 の数列 {a_n}={5, 9, 5, 9, 8, 9, 3, 5, 4, 3} について、k=7 として考えましょう」

■「この場合、{b_n}={1, 0, 1, 0, 0, 0, 1, 1, 1, 1} という数列を考えるんだね」

●「例えば、最初から3つ取り出すと、和は 2 になります。つまり、最初の 3 つの中に 7 以下の数は 2 個あるということです」

■「長さが 3 で、その半分の 1.5（切り捨てると1）より大きいから、最初の 3 つの中央値は 7 以下になるってことか。実際、{5, 9, 5} の中央値は 5 だね」

●「はい。i 番目から j-1 番目まで（半開区間で [i, j) ）の総和は、累積和の配列 {c_n} を考えたときに、c_j − c_i で表されますよね」

$$c_0 = 0$$

$$c_i = \sum_{n=0}^{i-1} b_n$$

とすると、

$$\sum_{n=i}^{j-1} b_n = c_j - c_i$$

■「それが、長さの半分より大きくなればいいから、これが条件かな」

$$c_j - c_i > \frac{j-i}{2}$$

●「そうですね。分母を払って移項して、i ごと、j ごとにまとめるとこうなります」

$$2c_i - i < 2c_j - j$$

■「きれいな式になったね」

●「この 2c_i-i を、新たな配列 {d_n} とします」

$$d_i = 2c_i - i$$

■「なるほど、こうすると、『**区間 [i, j) の部分列の中央値が k 以下**』という条件がこれと同値に

{a_n}	5	9	5	9	8	9	3	5	4	3	
{b_n}	1	0	1	0	0	0	1	1	1	1	
{c_n}	0	1	1	2	2	2	2	3	4	5	6
{d_n}	0	1	0	1	0	−1	−2	−1	0	1	2

■「入力例2 だと、{d_n} が {0, 1, 0, 1, 0, -1, -2, -1, 0, 1, 2} になるから、この条件を満たす (i, j) の組は、i=0 のとき 4 つ、i=1 のとき 1 つで……　えーっと、24通りか。これが、55/2=27.5 よりも大きくないから、中央値たちの中央値は **7 以下ではない**んだね」

●「はい。こうなると、もう解けたようなものですね」

■「もう解けたの？　えっと、『**中央値が k 以下になる連続部分列の個数**』が知りたいんだよね。今の言い換えでこれが『**d_i < d_j を満たす i, j (i < j) の組の個数**』になったのか」

●「良い感じですよ」

■「うーん、どこかで見たことあるような……　そうか、これは**転倒数**か！」

●「その通りです！　**不等号の向きが逆**ですが、**数列を反転**させれば大丈夫ですね」

■「そうだね。数列 {d_n} の順番を逆にすると、 **d_i > d_j (i < j) を満たす i, j の組の個数**になって、これはまさに転倒数だね」

●「転倒数は BIT やセグメント木、もしくはマージソートで計算できますね」

■「そうだね。調べたらたくさん出てくるね」

●「この転倒数が『**中央値が k 以下になる連続部分列の個数**』ですね」

■「その値が、**連続部分列の個数の半分より多ければ**いいんだね」

●「連続部分列の個数は N*(N+1) / 2 なので、条件は **転倒数 > N*(N+1)/4** ですね」

■「そうだね。この条件が真なら、中央値たちの中央値が k 以下になるんだね」

●「あとは二分探索をするだけですね」

■「『中央値が k 以下になる連続部分列の個数』を count(a, k) としておいて、**count(a, k) > N*(N+1)/4 かどうかで場合分けして二分探索**すればいいんだね」

●「ということで、コードはこんな感じですね」

```cpp
    vector<int> c(N + 1);
    vector<int> d(N + 1);

    // b_n を作成
    rep(i, N) {
        if (a[i] <= k) b[i] = 1;
        else           b[i] = 0;

        c[i + 1] = c[i] + b[i];
        d[i + 1] = 2 * c[i + 1] - i;
    }

    // c_n, d_n を計算
    c[0] = 0;
    d[0] = 0;
    repr(i, 1, N + 1){
        c[i] = c[i - 1] + b[i - 1];
        d[i] = 2 * c[i] - i;
    }

    reverse(all(d));         // 一般的な転倒数の定義に合わせるため配列を反転
    return inversion(d);     // 転倒数（求める値）を返す
}

int main(void) {
    cin >> N;
    vector<int> a(N);
    rep(i, N){
        cin >> a[i];
    }

    int ok = INF, ng = 0;
    while (ok - ng > 1) {
        int m = ng + (ok - ng) / 2;
        // Median が m になる連続部分列の個数が、連続部分列の個数の半分より大きいか？
        if (count(a, m) > (ll)N*(N + 1) / 4) {
            ok = m;      // 中央値たちの中央値は m 以下
        }
        else {
            ng = m;      // 中央値たちの中央値は m より大きい
        }
    }
    cout << ok << endl;
}
```

■「**中央値の性質を利用した言い換え**か、これは覚えておくべきだね」

●「中央値が絡む問題では結構使える考え方ですからね」

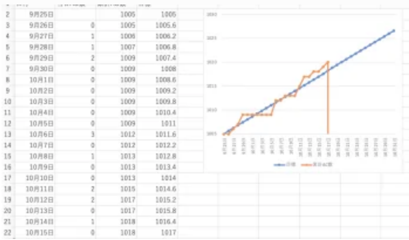
■「この問題は、中央値の言い換えによって見通しが良くなる典型って感じだね。勉強になる」

●「**解説PDF**の方はもうちょっとスマートな考え方みたいですね。そっちも読んでみると勉強になりそうです」

■「これが黄色コーダーの半分が解く問題かぁ。まだまだ勉強しないといけないことが多いなぁ」

漢検準一級を受けました

2019年6月16日
その他



怠惰でもできる精進

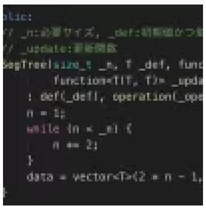
2019年9月25日
競プロ

日経エキシビジョンH「8^kゲーム」

■「日経コンテストのエキシビジョンのH問題って解いた？」
●「『 8^k ゲーム』ですか。その問題な…

2019年2月19日
競プロ

関連記事



セグメントツリーの抽象化（C++）

※注 終盤に「ARC008-D タコヤキオイシクナール」の解法のネタバレが存在します ■「セグメントツリーの抽象化ってしてる？」 …

記事を読む

AtCoder Typical DP Contest – A「コンテスト」その1

▲「ねえ」 ■「ん？　どうした？」 ▲「お兄ちゃんに勧められて "競技プログラミング" をやってるんだけど、"動的計画法"ってい…

記事を読む

OUPC β「Increasing Path」解説

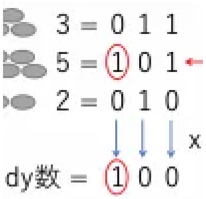
こんにちは、ふるやん（@furuya1223）です。 本日 Hackerrank にて開催された OUPC β に参加いただいた方は、…

記事を読む

DP 練習セット（TDPC編）

こんにちは、ふるやん（@fuurya1223）です。 この記事は DP を練習したい競プロerに向けて作った下記の記事の TDPC 編…

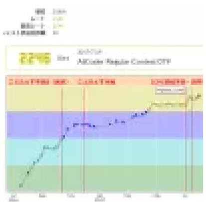
記事を読む



Grundy数（Nim数, Nimber）の理論

■「競プロでのゲーム問題って得意？」 ●「ゲーム問題ですか。ものによりますけど、Grundy 数で解ける問題は得意ですね。先輩は得意で…

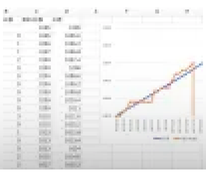
記事を読む



AtCoder黄色になる方法

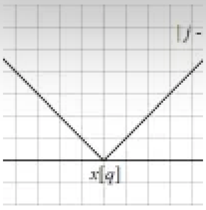
こんばんは。誇大広告ぎみなタイトルでごめんなさい。 正式タイトルは「AtCoder黄色になるまでに私がしたこと」でお願いします。 …

記事を読む



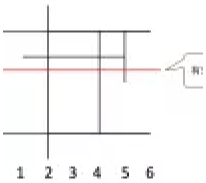
怠惰でもできる精進

こんばんは、ふるやん（@furuya1223）です。 みなさん、怠惰ですか？ 私は怠惰です。 （©TN,K,Re:…



■「うーん、オーダーが落ちないなあ……」 ●「先輩、競プロですか？」 ■「うん。最近、妹に競プロを勧めて、DPを教えてるんだけど...

記事を読む



「Count The Rectangles」 – Educational Codeforces Round 68 E

■「ねえ、この問題の解き方って分かる？」 ●「"Count The Rectangles"……　Educational Codefor...

記事を読む

DP 練習セット（EDPC編）

こんにちは、ふるやん（@fuurya1223）です。 先日、DP （動的計画法）の練習に関してこんなツイートをしました。 DPが...

記事を読む

レトリバのインターンに参加しました

拡張互除法と中国剰余定理（Garnerのアルゴリズム）

コメントをどうぞ

メールアドレスが公開されることはありません。 * が付いている欄は必須項目です

名前 *

次回のコメントで使用するためブラウザーに自分の名前、メールアドレス、サイトを保存する。

コメントを送信