ABC 174 解説

evima, kyopro_friends, ynymxiaolongbao

2020 年 8 月 2 日

For International Readers: English editorial will be published in a few days.

A: Air Conditioner

(解説: evima)

「何をすればいいか全く分からない！」という場合、まずは「practice contest」(`https://atcoder.jp/contests/practice/`) の問題 A「はじめてのあっとこーだー」をお試しください。言語ごとに解答例が掲載されています。

今回の問題に移ります。突然ですが、架空の言語で正解例を示します。[*1]

```
read X as int
if X >= 30:
    write "Yes"
else:
    write "No"
```

これをお使いの言語に合わせて書き換えれば問題を解けます。以下、各行について述べます。

1 行目: 入力された気温 X を整数として読み込みます。「はじめての〜」で整数 a を読み込むのと同様に行えるはずです。

3, 5 行目: 文字列 `Yes` または `No` を出力します。「はじめての〜」で入力された文字列 s を出力するソースコードの `s` を `"Yes"` と書き換えればほとんどの言語で動作するはずです。もし動作しなければ、検索エンジンで「(言語名) 文字列 出力」などと検索してください。(次頁へ続く)

[*1] なぜそのようなことを？というと、AtCoder では数十の言語が使用可能で、それらをできるだけ平等に扱うためです。

2, 4 行目: 読み込んだ整数 X が 30 以上であれば 3 行目、そうでなければ 5 行目を実行させます。言語ごとの if 文の書き方については、検索エンジンで「(言語名) if 文」などと検索してください。整数の大小比較については、`X >= 30` と書けば多くの言語で動作するはずですが、もし動作しなければ検索エンジンで「(言語名) 大小比較」などと検索してください。

B: Distance

(解説: evima)

前問と同様に架空の言語で正解例を示し、各行について述べます。

```
1 read N, D as int
2 ans = 0
3 for i = 1, ..., N:
4     read X, Y as int
5     if X * X + Y * Y <= D * D:
6         ans += 1
7 write ans
```

1 行目: 入力された値 N, D を整数として読み込みます。

2 行目: カウンターとして用いる変数 ans を宣言し、0 で初期化します。

3 行目: 4 行目から 6 行目までを N 回繰り返し実行させます。言語ごとの for 文の書き方については、検索エンジンで「(言語名) for 文」などと検索してください。

4 行目: i 個目の点の座標 X_i, Y_i を整数変数 X, Y に読み込みます。N 個すべての点の座標を配列に保存する手もありますが、この解答例では処理した点の座標は「忘れる」ことにしています。

5, 6 行目: 距離 $\sqrt{X^2 + Y^2}$ が D 以下であるか判定し、そうであれば ans に 1 を加算します。ただし、これら二つの値を直接比較する代わりに両者の二乗同士を比較しています (入力される D の値は 0 以上であることが保証されているため、単に二乗同士を比較しても結果は変わりません)。一般に、計算機上での実数の計算は浮動小数による近似計算であり、その結果には誤差が含まれる[*2]ため、可能な限り避けるべきです。

7 行目: カウンターとして用いた変数 ans の最終的な値を出力します。

[*2] 例えば、コードテストで `print(2**0.5 + 2**0.5 == 18**0.5)` (Python) を実行してみてください

C: Repsept

(解説: evima)

与えられた数列の i 項目は $7 \times (1 + 10 + 10^2 + \ldots + 10^{i-1}) = \frac{7(10^i-1)}{9}$ と書ける (等比数列の和) ため、求めるべきものは $7(10^i - 1)$ が $9K$ で割り切れるような最小の正の整数 i です。

K が 7 の倍数である場合は、代わりに $10^i - 1$ が $\frac{9K}{7}$ で割り切れるかを考えても同じことで、そうでない場合は、代わりに $10^i - 1$ が $9K$ で割り切れるかを考えても同じことです。したがって、整数 L を、K が 7 の倍数であれば $L = \frac{9K}{7}$、そうでなければ $L = 9K$ と定義し、$10^i - 1$ が L で割り切れるような最小の i、すなわち 10^i を L で割った余りが 1 であるような最小の正の整数 i を求めればよいことになります。

L が 2 の倍数であるなら、どの正の整数 i に対しても 10^i は 2 の倍数であるため、L で割った余りが 1 となることはありません。L が 5 の倍数である場合も同様です。このいずれにも該当しない場合、すなわち 10 と L が互いに素である場合、オイラーの定理より $10^{\varphi(L)} \equiv 1 \pmod{L}$ [*3] が成立します。すなわち、$i = 1, 2, \ldots,$ と順に検討していけば遅くとも $i = L(\leq 9000000)$ までには求めるべき整数が見つかる (例えば $i = 10^{100}$ でようやく見つかるといったことはない) ことが保証されているため、あとは実際に $10^1, 10^2, \ldots$ を実際に L で割った余りを計算していけば十分速く解を求められます。

ただし、10^{999982} などといった巨大な値を直接 L で割ろうとするべきではありません。その代わりに、10^i を L で割った余りを 10 倍して L で割れば、10^{i+1} を L で割った余りが求まります。

なお、以上の数学的考察をせずとも、「答えが -1 でないならどうせある程度小さいだろう、でなければこの “枠” には難しすぎる」と予想してしまい、$7, 77, \ldots$ を K で割った余りを上の段落で述べたような方針で計算していくことで答えを求めることも可能ではあります。

[*3] $\varphi(n)$ はオイラーの ϕ 関数、すなわち n と互いに素であるような 1 以上 n 以下の整数の個数です

D: Alter Altar

(解説: evima)

「赤い石の左隣に置かれた白い石」がない状態では、赤い石 A の左隣に石 B があれば B は必ず赤で、B の左隣に石 C があれば C もまた赤で、同様にして A より左にある石は全て赤です。このことから、目標が満たされた状態は次の $N+1$ 通りしか存在しません: `WWW...WWW` (すべて白)、`RWW...WWW` (左端の 1 個のみ赤)、`RRW...WWW` (左端の 2 個のみ赤)、$\cdots$、`RRR...RRW` (左端の $N-1$ 個のみ赤)、`RRR...RRR` (すべて赤)。これをもとに、目標を次のように言い換えます。

新たな目標

操作を行い始める前に、祭壇上のいずれか一箇所を選んで仕切りを置く (つまり、仕切りは石と石の間か、どの石よりも左側か、どの石よりも右側のいずれかに置かれる)。目標は、仕切りより左側の石をすべて赤に、仕切りより右側の石をすべて白にすることである。

仕切りの位置がすでに決まっているとします。すると、操作の性質上、もはや仕切りの左右に赤い石と白い石がそれぞれ何個あるか以外は目標達成と関係しません。仕切りより左側の白い石の個数を W、仕切りより右側の赤い石の個数を R とすると、目標は W と R をともに 0 とすることです。一回の操作で W が 1 より多く減ることはないため、目標達成には少なくとも W 回の操作が必要です。同様に少なくとも R 回の操作が必要であるため、目標達成には少なくとも $\max(W,R)$ [*4] 回の操作が必要です。逆に、次のようにすれば $\max(W,R)$ 回の操作で目標を達成することができます。

- $W \leq R$ の場合: 仕切りより左側の白い石と右側の赤い石の入れ替えを W 回行い、右側に残った $R-W$ 個の赤い石を一個ずつ白くする。合計操作回数: R。
- $W > R$ の場合: 仕切りより左側の白い石と右側の赤い石の入れ替えを R 回行い、左側に残った $W-R$ 個の白い石を一個ずつ赤くする。合計操作回数: W。

よって、仕切りの位置が固定された場合の必要な最小操作回数が $\max(W,R)$ であることがわかりました。あとは、仕切りの位置の候補 $N+1$ 通りをすべて試し、それぞれについて $\max(W,R)$ の値を求めれば、それらのうち最小のものが求めるべき最小操作回数です。

ただし、入力される N の上限はやや大きく、N 個の石すべてを $N+1$ 回見直す時間はありません。これに対処する方法の一つは、あらかじめ白い石の総数と赤い石の総数を数えておき、仕切りを最も左の位置から一つずつ右にずらしていくことです。仕切りが一つ右にずれた際に起こることは、W が 1 増えるか、R が 1 減るかのいずれかであることを用いれば、線形時間で処理が完了します。

[*4] $\max(a,b)$ は a と b のうち小さくない方を表します。例: $\max(3,5) = \max(5,3) = \max(5,5) = 5$

E: Logs

（解説：ynymxiaolongbao）

答えが X 以下であるか？という問題を考えます。この問題は言い換えれば、K 回以内のカットですべての丸太の長さを X 以下にすることができるか？という問題になります。はじめ長さ A_i の丸太を切り分けて長さ X 以下にするためには、$\lceil \frac{A_i}{X} \rceil - 1$ 回切る必要があります。そして、この回数の合計が K 以下であれば答えは Yes、そうでなければ答えは No です。

二分探索を用いて、答えが X 以下であるか？という問題の答えが Yes になる最も小さな整数 X を求め、それを出力すれば良いです。計算量は $O(Nlog(maxA))$ になります。

F: Range Set Query

区間に対するクエリに答えるというと、Range Minimum Query などに用いられる SegmentTree が知られていますが、集合のマージには時間がかかってしまいます。

全体で左から k 番目かそれより左にある玉だけに注目したとき、それぞれの色で最も右にある玉を k についての良い玉と呼ぶことにします。すると、i 番目のクエリに対する答えは $[l_i, r_i]$ にある r_i についての良い玉の個数になります。

クエリを r_i の昇順にソートし、良い玉の集合を管理しながらクエリに答えて行きます。良い玉の集合は以下の二つのデータ構造を用いて管理すると良いです。

- 色 i の良い玉が存在するか、存在するならその場所はどこかを管理し、集合の更新をするための配列
- 良い玉のある場所が 1 、それ以外が 0 であるような数列を管理し、区間和を取ることでクエリに答えるための Fenwick Tree

計算量は $O(N + QlogN)$ です。

解答例

https://atcoder.jp/contests/ghi/submissions/15231679