ABC116

出題、解説:yuma000

2019 年 1 月 19 日

A - Right Triangle

「三角形の面積」=「底辺」×「高さ」÷ 2 です。よって、$|AB| \times |BC| \div 2$ を出力すればよいです。

この問題では、$|CA|$ の値は解答に影響しませんでした。今回は $|CA|$ の値は最後に与えられるため、これを入力として受け取らなくても問題に正答できます。ですが、不都合が生じる場合もあるので (複数のテストケースがある, クエリ形式の問題である等) 慣れないうちは一応全て入力として受け取る癖をつけておくことをおすすめします。

以下に C++ での解答例を示します。

```
#include <iostream>
using namespace std;

int main() {

        int ab, bc, ca;

        cin >> ab >> bc >> ca;

        int area = ab * bc / 2;

        cout << area << endl;

        return 0;
}
```

B - Collatz Problem

実際にシミュレーションして a の要素を順次求めていきましょう。出てきた a_i の値を配列や連想データ構造に保存していって、重複する値を検出することで、この問題に正答することができます。

また、この問題は Collatz problem として知られており、a_1 の値によらず、a はかならず $4 \to 2 \to 1 \to 4$ のループに到達すると予想されています。この予想は 5×10^{260} 以下ならば実証さ

れているため、$a_m = 4, 2, 1$ となる最小の整数 m をシミュレーションで求めて、$m+3$ を出力することでも、この問題に正答できます。

C - Grand Garden

長さ N の数列 h に対する「水やり」の組を $w(h)$ とする。$w(h)$ の取りうる要素数の最小値を $f(h)$ とする。

定理 1

$h_a = \{h_1, h_2, ...h_N\}(h_i \geqq 1)$,
$h_b = \{h_1 - 1, h_2 - 1, ...h_N - 1\}$
に対して、$f(h_a) = f(h_b) + 1$ である。

[証明 1]

$h_i \leqq 1$ より、$w(h_a)$ には、$(1, r_1), (l_2, r_2), (l_3, r_3)..., (l_x, N)$ が含まれる。$(r_k \leqq l_{k+1} (1 \leqq k \leqq N-1)$
$(1, r_1), (l_2, r_2), (l_3, r_3)..., (l_x, N)$ は、$(1, N), (l_2, r_1), (l_3, r_2)..., (l_x, r_{x-1})$ で置き換えられる。つまり、最小要素数の $w(h_a)$ の中に、$(1, N)$ が含まれているものが必ず存在する。$w(h_a)$ に $(1, N)$ を加えたものは、$w(h_a)$ となるので、$f(h_a) = f(h_b) + 1$ が示される。

定理 2

$h_a = \{h_1, h_2, ..., h_{x-1}, h_x = 0, h_{x+1}, ...h_N\}$,
$h_b = \{h_1, h_2, ..., h_{x-1}, \underbrace{0, ...0}_{N-x+1}\}$,
$h_c = \{\underbrace{0, ..., 0}_{x}, h_{x+1}, ...h_N\}$
に対して、$f(h_a) = f(h_b) + f(h_c)$ である。

[証明 2]

$h_x = 0$ より、$w(h_a)$ に含まれる要素 (l, r) に対して、$x < l$ もしくは　$r < x$ のどちらか一方が成り立つ。前者からなる集合は、$w(h_b)$ であり、後者からなる集合は $w(h_c)$ である。よって、$f(h_a) = f(h_b) + f(h_c)$

また、明らかに、以下の定理も成り立ちます。

定理 3

$h_a = \{h_1, h_2, ...h_N, 0\}$,
$h_b = \{h_1, h_2, ..., h_N\}$,
$h_c = \{0, h_1, h_2, ..., h_N\}$
に対して、$f(h_a) = f(h_b) = f(h_c)$ である。

これら三つの定理から、

- 伸ばすべき高さが全ての花に対して 1 以上ならば、全ての花に水をやる。
- ある花 x をもはや伸ばす必要がないならば、花 $1 \sim x-1$ と花 $x+1 \sim N$ についての最小手数を別々に求めて、それらの必要手数を合計する。

この操作を繰り返すことで、この問題に正答することができます。

D - Various Sushi

$p(1 \leqq a \leqq N)$ 種類のネタを選んだ時の「おいしさ基礎点」の最大値を $f(p)$ と定義します。(そのように選べないときは、$-inf$ とします。)
また、$g(p) = f(p) + p * p$ と定義します。$(1 \leqq p \leqq N)$ において、$g(p)$ の最大値を求めればいいことになります。

まず、寿司を「おいしさ」で降順にソートします。ここから貪欲に (「おいしさ」が大きい順に)K 個寿司を選び、その集合を S とします。S の「ネタ」の種類数が x の時、S の「おいしさ基礎点」の和が、$f(x)$ となります。

明らかに $f(i) < f(x)(i < x)$ なので、$g(i) < g(x)(i < x)$ となり、$i < x$ の場合は考えなくても良いことが分かります。

次に、$f(a)$ とそれを構成する寿司の集合の一つ S_a が分かっている場合に、$f(a+1)$ とそれを構成する寿司の集合の一つ S_{a+1} を求めることを考えます。$(x \leqq a \leqq N-1)$

S_{a+1} は、S_a に以下の操作を加えることで構成することができます。

1. 集合 S_a から、取り除いても「ネタ」の種類数が減らない寿司の中で最も「おいしさ」の小さいものを取り除く。
2. 集合 S_a に含まれない寿司の中で、集合 S_a にない「ネタ」を持つ、最も「おいしさ」の大きいものを S_a に加える。

この操作を繰り返すことにより、$f(i)(x < i \leqq N)$ を再帰的に求めることができます。$priorityqueue$ 等の、値の挿入と最小値の取り出しを計算量 $O(log|T|)$(|T| は要素数) で行えるデータ構造を用いることで、一回の操作を計算量 $O(logN)$ で行えます。これを最大 N 回繰り返すので、O(NlogN) で $f(p)(x \leqq p \leqq N)$ を求められます。

後は、$g(p)(x \leqq p \leqq N)$ の最大値を求めることで、この問題に正答することができます。