

ABC 170 解説

evima, gazelle, kort0n, tozangezan, ynymxiaolongbao, yuma000

2020 年 6 月 14 日

For International Readers: English editorial starts on page 7.

A: Five Variables

x_1, \dots, x_5 を標準入力から読み込んだ後は、if 文を 5 つ連ねて 5 つの変数のうちどれが 0 であるかを直接確かめることも、 $15 - x_1 - x_2 - x_3 - x_4 - x_5$ が答えとなることを利用することもできます。それぞれの方針における Python での実装例を示します。

```
1 X = list(map(int, input().split()))
2
3 if X[0] == 0:
4     print(1)
5 elif X[1] == 0:
6     print(2)
7 elif X[2] == 0:
8     print(3)
9 elif X[3] == 0:
10    print(4)
11 else:
12    print(5)
```

```
1 print(15 - sum(list(map(int, input().split()))))
```

B: Crain and Turtle

(原案: yuma000, 準備・解説: evima)

方針 1

発言の前半「庭の動物の総数は X 匹」が正しいと仮定すると、考えられる鶴と亀の数の組合せは「鶴 0 匹、亀 X 匹」「鶴 1 匹、亀 $X-1$ 匹」...「鶴 X 匹、亀 0 匹」の $X+1$ 通りです。これらのそれぞれにおける足の総数を計算し、その結果が一度でも Y 本となれば答えは Yes、一度も Y 本とならなければ答えは No です。素直に実装するなら、以下の Python のコードのように for 文を使うことになるでしょう。

```
1 X, Y = map(int, input().split())
2 ans = 'No'
3 for a in range(X + 1):
4     b = X - a
5     if 2 * a + 4 * b == Y:
6         ans = 'Yes'
7 print(ans)
```

方針 2

答えが Yes であることの必要十分条件は、 Y が $2X$ 以上 $4X$ 以下の偶数であることです。

必要性の証明: a 匹の鶴と b 匹の亀について発言が正しいとすると、 $Y = 2a + 4b = 2(a + 2b)$ であり、 a, b が整数であることから Y は偶数でなければなりません。また、 $a + b = X, 2a + 4b = Y$ から $a = (4X - Y)/2, b = (Y - 2X)/2$ が得られ、 a, b が非負であることから Y は $2X$ 以上 $4X$ 以下でなければなりません。よって示されました。

十分性の証明: Y が $2X$ 以上 $4X$ 以下の偶数であれば、 $(4X - Y)/2, (Y - 2X)/2$ はともに非負整数となり、発言が正しいような鶴と亀の数の組合せとして「鶴 $(4X - Y)/2$ 匹、亀 $(Y - 2X)/2$ 匹」が存在します。よって示されました。

C: Forbidden List

(原案: tozangezan, 準備・解説: evima)

X, p_1, \dots, p_N はいずれも 1 以上 100 以下ですが、答えは 0 または 101 となりうる点に注意を要します。

この点にさえ注意すれば、与えられる値の小ささのため計算を高速化する工夫などは必要なく、「書かれていることを実装しなさい」という通常の Beginner Contest 問題 B 相当の問題です。本問が問題 B として出題されていたなら以下に具体的な実装方法を述べるところですが、問題 C を解かれる方にそのような説明は不要と考え、省略します。

追記 (6/18): 以下に Python での実装例を示します。

```
1 X, N = map(int, input().split())
2 p = list(map(int, input().split()))
3 for d in range(X + 1):
4     for s in [-1, +1]:
5         a = X + s * d
6         if p.count(a) == 0:
7             print(a)
8             exit(0)
```

答えの候補を $X, X-1, X+1, X-2, X+2, \dots$ の順に調べています。count を何度も呼び出している部分に高速化の余地がありますが、この問題の制約下ではその必要はありません。

D: Not Divisible

エラトステネスの篩と同様の処理を、数列 A の要素についてのみ行います。

サイズ A_{max} の bool 配列 dp を用意し、true で初期化します。 $dp[i] = \text{true}$ のとき、 i より小さい i の約数が A に存在しないことを表します。数列の要素を昇順に見ます。 x が数列 A に含まれているとき、 x より大きい x の倍数 y について、 $dp[y]$ を false に変更します。

同じ値について処理を複数回行わないように実装すると、調和級数を用いた解析により、時間計算量が $O(A_{max} \log A_{max} + N \log N)$ であることが分かります。

同じ値が複数存在する場合に、それらを答えに含めないことに注意してください。

E. Smart Infants

ある幼稚園の中で最もレートの高い幼児を、その幼稚園の最強園児と呼ぶことにします。

以下のようなデータを管理することで時系列順に答えを求めてゆくことができます。

- それぞれの園児が所属する幼稚園の番号
- それぞれの幼稚園に所属する幼児のレートの順序付き多重集合（幼稚園一つに対して一つ）
- それぞれの幼稚園の最強園児のレートの順序付き多重集合（全体に対して一つ）

順序付き多重集合は C++ の `multiset` などを用いることで高速に処理することができます。

それぞれのクエリでは、以下のような変更操作を行います。

- 最強園児のレートの集合から、転園する園児の元の幼稚園の元の最強園児のレートを、削除する
- 最強園児のレートの集合に、転園する園児の元の幼稚園の新しい最強園児のレートを、挿入する（園児が一人もいない場合何もしない）
- 最強園児のレートの集合から、転園する園児の新しい幼稚園の元の最強園児のレートを、削除する（園児が一人もいない場合何もしない）
- 最強園児のレートの集合に、転園する園児の新しい幼稚園の新しい最強園児のレートを、挿入する
- 転園する園児の元の幼稚園のレートの集合から、転園する園児のレートを、削除する
- 転園する園児の新しい幼稚園のレートの集合に、転園する園児のレートを、挿入する
- 転園する園児の所属する幼稚園の番号を更新する

以上の変更操作の後、最強園児のレートの集合の中の最も小さい値を出力すれば良いです。計算量は $O(N\log N)$ です。

F: Pond Skater

Dijkstra 法を用います。以下のようなグラフを考えます。
次のような状態量をグラフの頂点と見做します。

- 現在いるマス
- すぬけ君が東西南北のどの方向を向いているか

次のように遷移します。

- 向いている方向に 1 つ進んだマスに蓮がないなら、「向いている方向に 1 進んだマスにいて、向きはそのまま」という状態量に、今の状態量までの最短コストに $1/K$ を足したコストで到達可能とする。
- 「同じマスにいて、向きが 90 度回転した」という状態量に、今の状態量までの最短コストの小数点以下を切り上げたコストで到達可能とする。

計算量は $O(HW \log(HW))$ です。

A: Five Variables

After reading x_1, \dots, x_5 from the standard input, you can write if statements 5 times and check which of 5 variables is 0, or use the fact that $15 - x_1 - x_2 - x_3 - x_4 - x_5$ is the answer.

The following are the sample code in Python for each way.

```
1 X = list(map(int, input().split()))
2
3 if X[0] == 0:
4     print(1)
5 elif X[1] == 0:
6     print(2)
7 elif X[2] == 0:
8     print(3)
9 elif X[3] == 0:
10    print(4)
11 else:
12    print(5)
```

```
1 print(15 - sum(list(map(int, input().split()))))
```

B: Crain and Turtle

(Original writer: yuma000, preparation and editorials: evima)

Course 1

Assuming that the first half description that “there are X animals in total” is true, then there are $X + 1$ possible combinations of the numbers of cranes and turtles: “0 cranes and X turtles”, “1 crane and $X - 1$ turtles”, ..., and “ X cranes and 0 turtles”. Calculate the total number of legs for each situation, and if at least one of the sum is equal to Y , then the answer is Yes, otherwise the answer is No. Naively, for statement can be used in the implementation, like the following Python code.

```
1 X, Y = map(int, input().split())
2 ans = 'No'
3 for a in range(X + 1):
4     b = X - a
5     if 2 * a + 4 * b == Y:
6         ans = 'Yes'
7 print(ans)
```

Course 2

The necessary and sufficient conditions for the answer to be Yes is that Y is an even number between $2X$ and $4X$, inclusive.

Proof of necessity: if the statement is true for a cranes and b turtles, then $Y = 2a + 4b = 2(a + 2b)$, and since a, b is an integer, Y must be an even number. Also, since $a + b = X, 2a + 4b = Y$, we obtain $a = (4X - Y)/2, b = (Y - 2X)/2$, and since a, b are non-negative, Y should be between $2X$ and $4X$, inclusive. Here the proof ends.

Proof of sufficiency: if Y is an even number between $2X$ and $4X$, inclusive, then $(4X - Y)/2, (Y - 2X)/2$ are both a non-negative integer, so there exists a combination of numbers of cranes and turtles: “ $(4X - Y)/2$ cranes and $(Y - 2X)/2$ turtles.” Here the proof ends.

C: Forbidden List

(Original writer: tozangezan, preparation and editorials: evima)

Note that although X, p_1, \dots, p_n are all between 1 in 100, the answer may be 0 or 101;

If you just take care of that point, you don't have to do any optimization because of the small constraints, so it's just as easy as Beginner Contest's Problem B, in which you are only asked to "implement just as described." If this problem was set in Problem B, we would describe the specific way to implement it, but we will omit it since we think such description is unnecessary for those who tries solving Problem C.

P.S. (June 18): Sample implementation in Python follows:

```
1 X, N = map(int, input().split())
2 p = list(map(int, input().split()))
3 for d in range(X + 1):
4     for s in [-1, +1]:
5         a = X + s * d
6         if p.count(a) == 0:
7             print(a)
8             exit(0)
```

It checks candidates for the answer in the order $X, X - 1, X + 1, X - 2, X + 2, \dots$. We can optimize the part where it calls `count` many times, but we have no need to do so under the constraints of this problem.

D: Not Divisible

Perform the same operations to Sieve of Eratosthenes only for the elements of the sequence A . Prepare a bool array dp of size A_{max} , and initialize them with true. $dp[i] = \text{true}$ means that A contains none of the divisors of i less than i . Inspect each element in the increasing order. When x is contained in A , for each multiple y of x larger than x , update $dp[y]$ to false. If you avoid performing operations multiple times for the same value, the total time complexity appears to be $O(A_{max} \log A_{max} + N \log N)$ by the analysis using harmonic series. Be careful not to count the values that appears multiple times.

E. Smart Infants

We will refer to the infant with the highest rating in a kindergarten as the “strongest infant” in the kindergarten.

By managing the following data, you can find the answer in the chronological order.

- The index of Kindergarten to which each infant belongs
- Ordered multiset of ratings of infants who belong to each kindergarten (one multiset for one kindergarten each)
- A multiset of ratings of the strongest infants of all the kindergartens (one common multiset for all of them)

Ordered multisets can be processed fast by using, for instance, multiset in C++.

For each query, perform the following update operations.

- ERASE the rating of the former strongest infant in the former kindergarten of the infant who will transfer FROM the set of ratings of the strongest infants
- INSERT the rating of the new strongest infant in the former kindergarten of the infant who will transfer TO the set of ratings of the strongest infants (or do nothing if no infants are there)
- ERASE the rating of the former strongest infant in the new kindergarten of the infant who will transfer FROM the set of ratings of the strongest infants (or do nothing if no infants are there)
- INSERT the rating of the new strongest infant in the new kindergarten of the infant who will transfer TO the set of the ratings of the strongest infants
- ERASE the rating of the infant who will transfer FROM the set of ratings of the former kindergarten
- INSERT the rating of the infant who will transfer TO the set of ratings of the new kindergarten
- Update the index of kindergarten that the infant who will transfer belongs to

After the update operations above, print the lowest rating of the set of ratings of the strongest infants. The total time complexity is $O(N \log N)$.

F: Pond Skater

We will use Dijkstra's algorithm. Consider the following graph.

We will regard the states determined by the following as vertices.

- The current square
- The direction in which Snuke is facing

It will be updated as follows.

- If there is no leaf in the square one unit ahead, then it can advance to the state where “the current square is the square one unit ahead and the direction remains the same” by consuming $1/K$ cost.
- It can advance to the state where “the current square remains the same and the direction rotates by 90 degrees” by updating the cost to the current cost, rounded up.

The total time complexity is $O(HW \log(HW))$.