ARC 094/ABC 093 解説

DEGwer

2018/04/07

A: abc of ABC

3 つの文字がどの 2 つも相異なるかどうかを調べればよいです。

```
#include<stdio.h>
int main()
{
        char s[10];
        scanf("%s", s);
        if (s[0] != s[1] && s[1] != s[2] && s[2] != s[0]) printf("Yes\n");
        else printf("No\n");
}
```

B: Small and Large Integers

出力すべき範囲を適切に記述し、その範囲の整数をすべて出力すればよいです。A 以上 B 以下の整数が $2K$ 個未満の場合に注意しましょう。

```
#include<stdio.h>
#include<vector>
#include<algorithm>
using namespace std;
int main()
{
        int a, b, k;
        scanf("%d%d%d", &a, &b, &k);
        for (int i = a; i <= min(b, a + k - 1); i++)printf("%d\n", i);
        for (int i = max(b - k + 1, a + k); i <= b; i++)printf("%d\n", i);
}
```

C: Same Integers

3 つの整数の和の偶奇は操作によって不変です。また、整数を減少させることはできないので、3 つの整数が等しくなったならば、その時の値は 3 つの整数の最大値 (以下これを M とする) 以上です。よって、もし $3M$ と最初の 3 つの整数の和の偶奇が等しいなら、3 つの整数が等しくなった時の値は M 以上、そうでなければ $M+1$ 以上です。

逆に、この下限が達成できることは簡単にわかるので、この問題を解くことができました。

D: Wide Flip

1 回目のコンテストで A 位、2 回目のコンテストで B 位を取ることを (A, B) を取ると呼ぶことにします。

高橋君が (A, B) を取ったとします。$A \leq B$ として一般性を失いません。

$A = B$ の場合、高橋君よりスコアが小さい人は 1 回目のコンテストまたは 2 回目のコンテストで $A-1$ 位以内の順位を取っています。よって、そのような人の人数は $2A-2$ 人以下です。逆に、$(1, 2A-1), \cdots, (A-1, A+1), (A+1, A-1), \cdots, (2A-1, 1)$ を取った人が存在する場合、その上限値を達成できます。

$A+1 = B$ の場合、高橋君よりスコアが小さい人は 1 回目のコンテストで $A-1$ 位以内を取るか、2 回目のコンテストで A 位以内を取っています。また、2 回目のコンテスト A 位を取った人のスコアが高橋君より小さくなるためには、その人は 1 回目のコンテストで $A-1$ 位以内を取る必要があります。よって、高橋君よりスコアが小さい人の人数は $2A-2$ 人以下です。逆に、$(1, 2A), \cdots, (A-1, A+2), (A+2, A-1), \cdots, (2A, 1)$ を取った人が存在する場合、その上限値を達成できます。

そうでない場合、$C^2 < AB$ を満たす最大の整数 C を取ります。

$C(C+1) \geq AB$ のとき、高橋君よりスコアが小さい人は 1 回目のコンテストまたは 2 回目のコンテスト

で C 位以内の順位を取っています。1 回目のコンテストで C 位を取った人のスコアが高橋君のスコアより小さくなるためには、その人は 2 回目のコンテストで C 位以内を取らなければなりません。よって、高橋君よりスコアが小さい人の人数は $2C-2$ 人以下です。逆に、$(1, A+B-1), \cdots, (A-1, B+1), (A+1, 2C-A-1), \cdots, (C, C), \cdots, (2C-1, 1)$ を取った人が存在する場合、その上限値を達成できます。

$C^2 < AB$ のとき、高橋君よりスコアが小さい人は 1 回目のコンテストまたは 2 回目のコンテストで C 位以内の順位を取っています。よって、高橋君よりスコアが小さい人の人数は $2C-1$ 人以下です。逆に、$(1, A+B-1), \cdots, (A-1, B+1), (A+1, 2C-A), \cdots, (C, C+1), (C+1, C), \cdots, (2C, 1)$ を取った人が存在する場合、その上限値を達成できます。

以上の場合で尽くされるので、この問題を解くことができました。

E: Tozan and Gezan

とざん君を先手、げざん君を後手と呼ぶことにしましょう。もし数列 A, B が最初から等しいなら、答えは 0 です。そうでない場合、答えは以下の値になることを証明します。

- 数列 A の総和から、$A_i > B_i$ を満たす B_i の中で最小のものを引いた値

一般性を失わずに、$A_i > B_i$ を満たす i の中で B_i を最小にするものが 1 であるとして良いです。まず、先手が操作回数をこの値以上にできることを示します。

先手は以下の戦略にのっとって操作を進めればよいです: A の 1 番目以外の整数をすべて 0 になるまで減らし、そのあと A_i を減らしていく

この操作で上記の値以上の回数の操作が必要になるのは明らかです。

さて、後手が操作回数をこの値以下にできることを示しましょう。後手は以下の戦略にのっとって操作を進めればよいです: $B_i > A_i$ なる i をひとつ選び、B_i を減らす

後手がこの戦略にのっとって行動したとき、ある $A_i > B_i$ なる i が存在し、操作の終了時にも B_i の値が変わっていないことを証明できればよいです。$A_i > B_i$ である間、後手が B_i を減少させることはないので、$A_i > B_i$ であるような i がなくなる直前に先手が操作を行った i について、B_i の値が減らされていることはありません。これが操作の終了まで変化しない B_i の例となっているので、操作回数の上界と下界が一致し、この問題を解くことができました。

F: Normalization

もし S のすべての文字が等しいなら、答えは 1 です。以下、そうでない場合を考えます。

文字 A,B,C を順に整数 $0, 1, 2$ に対応させることにすると、操作によって文字列全体の対応する値の和を 3 で割ったあまりは変化しません。すなわち、文字列全体の対応する値の和を 3 で割ったあまりが S と等しい文字列のみ作ることができます。また、S に 1 回以上の操作を行って作ることのできる文字列には、必ず同じ文字が連続する箇所があります。

さて、作ることのできる可能性のある文字列は上記の条件を満たすものと S 自身に限られました。逆に、これらの文字列をすべて作ることはできるでしょうか？

実は、$|S| \geq 4$ ならこれらの文字列をすべて作ることができることが証明できます。$|S| = 4$ の場合は全通り試せば示すことができ、$|S|$ が 5 以上の場合は先頭の文字を適切に変化させることによって $|S|$ が 1 小さい

場合に帰着できます。

よって、この問題は以下のようにして解くことができます。

- S のすべての文字が等しいなら、1 を出力
- $|S| \leq 3$ なら、全探索を行う
- そうでないなら、 上記の条件を満たす文字列の個数を求め、S がその条件を満たさないならさらに 1 を足した値を答えとする

上記の条件を満たす文字列の個数は簡単な線形時間の DP で求めることができるので、この問題を線形時間で解くことができました。

ARC 094/ABC 093 Editorial

DEGwer

2018/04/07

A: abc of ABC

Check if the characters in s are pairwise distinct.

```
#include<stdio.h>
int main()
{
        char s[10];
        scanf("%s", s);
        if (s[0] != s[1] && s[1] != s[2] && s[2] != s[0]) printf("Yes\n");
        else printf("No\n");
}
```

B: Small and Large Integers

The integers you should print will be two intervals. Make sure to handle cases such that the number of integers between A and B is less than $2K$.

```
#include<stdio.h>
#include<vector>
#include<algorithm>
using namespace std;
int main()
{
        int a, b, k;
        scanf("%d%d%d", &a, &b, &k);
        for (int i = a; i <= min(b, a + k - 1); i++)printf("%d\n", i);
        for (int i = max(b - k + 1, a + k); i <= b; i++)printf("%d\n", i);
}
```

C: Same Integers

Suppose that when you finish the operations, all integers are X. Since the sum of three integers always increases by two in each operation, the total number of operations is $(3X - (A + B + C))/2$. Thus, we want to minimize X.

Let M be the maximum of A, B, C. Since we can never decrease integers, $X \geq M$ must hold. Also, since we can never change the parity of sum of three integers, $3X \equiv A + B + C \pmod 2$ must hold. (It's easy to see that these are sufficient conditions).

Therefore,

- If $3M \equiv A + B + C \pmod 2$, $X = M$.
- Otherwise, $X = M + 1$.

and we should print $(3X - (A + B + C))/2$.

D: Worst Case

Let us denote a participant who was ranked x-th in the first contest and y-th in the second contest as (x, y). Takahashi is (A, B). Without loss of generality, assume that $A \leq B$.

One possible way to solve the problem is to consider the following cases:

- If $A = B$, the answer is at most $2A - 2$ because you must get top $A - 1$ places in at least one contest to beat Takahashi. We can achieve this value by $(1, 2A-1), \cdots, (A-1, A+1), (A+1, A-1), \cdots, (2A-1, 1)$. Thus, the answer is $2A - 2$.
- If $A + 1 = B$, the answer is at most $2A - 2$ because you must get top $A - 1$ places in at least one contest to beat Takahashi. (Note that the A-th place in the first contest is filled by Takahashi, and getting the $A + 1$-th in the first and A-th in the second is not enough.) We can achieve this value by $(1, 2A), \cdots, (A - 1, A + 2), (A + 2, A - 1), \cdots, (2A, 1)$. Thus, the answer is $2A - 2$.
- Otherwise, let C be the maximum integer such that $C^2 < AB$.
  - If $C(C+1) \geq AB$, the answer is at most $2C-2$ because you must get top $C-1$ places in the first contest or top C places in the second contest to beat Takahashi. (Note that one of "the top $C - 1$ places in the first contest" is Takahashi, we shouldn't count him.) We can achieve this value by $(1, A+B-1), \cdots, (A-1, B+1), (A+1, 2C-A-1), \cdots, (C, C), \cdots, (2C-1, 1)$. Thus, the answer is $2C - 2$.
  - If $C(C+1) < AB$, the answer is at most $2C-1$ because you must get top C places in the first contest or top C places in the second contest to beat Takahashi. (Note that one of "the top C places in the first contest" is Takahashi, we shouldn't count him.) We can achieve this value by $(1, A + B - 1), \cdots, (A - 1, B + 1), (A + 1, 2C - A), \cdots, (C, C + 1), (C + 1, C), \cdots, (2C, 1)$. Thus, the answer is $2C - 1$.

Another possible solution is binary search on the answer: for a fixed X, we want to know the following:

- Consider the smallest X positive integers except for A.
- Consider the smallest X positive integers except for B, and reverse the order.
- Check if for all i, the product of the i-th integer in the first list and the i-th integer in the second list is smaller than AB.

To do this, use the fact that the lists are "piecewise" arithmetic sequences (and the number of pieces is constant).

E: Tozan and Gezan

If A, B are the same, the answer is 0. We assume that $A \neq B$.

What is a good strategy for the second player? If $A_i < B_i$ for some i, the second player needs to decrease B_i, regardless of the moves of the first player. He can do this without giving any valuable information for the first player. Thus, the second player should always choose one of such i (it always exists because the sums are the same), and decrease B_i by one.

What is a good strategy for the first player? Intuitively, when $A_i = B_i + 1$, it's not a good idea to decrease A_i by one. Let's make sure not to do this unless the first player is forced to do so.

If both players follow the rules mentioned above, when the game finishes, for all i but one (let's say $i \neq k$), $A_i = B_i = 0$ holds. B_k is unchanged from the beginning.

The first player wants to minimize B_k because the number of steps is (the sum of all elements) minus B_k. Here, $A_k > B_k$ must hold, but among such k's the first player can choose anything (by properly choosing what to do when he is forced to perform an operation for $A_i = B_i + 1$).

It turns out that the answer is the following:

- $X :=$ (The sum of all elements in A) minus (The minimum B_i such that $A_i > B_i$)

Here is a formal proof:

Without loss of generality, assume that the minimum B_i such that $A_i > B_i$ is B_1.

First, we prove that the first player can make sure that the number of steps become at least X. He follows the following strategy: first, decrease $A_2, \ldots, A_N$ to 0, and then decrease A_1. It's clear that by this strategy, the steps will be at least X.

Then, we prove that the second player can make sure that the number of steps become at most X. The second player follows the strategy mentioned above. Consider the last moment when $A_i > B_i$ holds for some i. Then, the first player decreases this A_i, and in the next step the game finishes. For this i, B_i is unchanged during the entire game. Thus, the number of steps will be at most X.

F: Normalization

When can we convert a string S into a string T? We can make the following observations:

- Let's assume the letters A, B, C as integers $0, 1, 2$. By performing the operations, the sum of all integers in the string never changes in modulo 3. Thus, if the sum of all integers in S and the sum of all integers in T are different (in modulo 3), the answer is "no".
- If all letters in S are the same, we can perform any operations. The answer is "no" unless $S = T$.
- If no two adjacent letters in T are the same, T can't be a result of an operation. Thus, the answer is "no" unless $S = T$.

Do we find all "no" cases? It's very difficult to answer this question on paper, so let's write a brute force solution. What we get is the following:

- If $|S| \leq 3$, the behavior is mysterious. From now on, assume that $|S| \geq 4$.
- If $S = T$, "yes". From now on, assume that $S \neq T$.
- If at least one of the three conditions above (the sum of S and T are different in modulo 3, all letters in S are the same, or no two adjacent letters in T are the same), "no". Otherwise, "yes".

(We can prove this by induction on $|S|$: $|S| = 4$ can be verified by brute force, otherwise we can change the first letter properly and decrease $|S|$ by one.)

Thus, we can solve the original problem as follows:

- If all letters in S are the same, print 1.
- If $|S| \leq 3$, compute the answer by brute force.
- Otherwise, count the number of strings T such that
  - $|T| = |S|$
  - $sum(S) = sum(T) \pmod 3$
  - There is at least one pair of adjacent same letters in T.

  (This can be done by a simple DP). Make sure not to forget to add one to the answer in case no two adjacent letters in T are the same.