ABC 155 解説

writer: sheyasutaka, rng_58, beet, drafear

2020 年 2 月 16 日

For International Readers: English editorial starts on page 8.

A: Poor

たとえば $A = B$ かつ $B \neq C$ を C 言語における式で表すと `A == B && B != C` となります.
これを踏まえて，過不足なく場合分けして書くと下の実装例のようになります。
C 言語での実装例は次のとおりです.

```
#include <stdio.h>

int main(void){
        int a, b, c;

        scanf("%d%d%d", &a, &b, &c);

        int ispoor = 0;
        if (a == b && b != c) ispoor = 1;
        if (b == c && c != a) ispoor = 1;
        if (c == a && a != b) ispoor = 1;

        if (ispoor) {
                puts("Yes");
        } else {
                puts("No");
        }

        return 0;
}
```

B: Papers, Please

「偶数であるにもかかわらず，3 でも 5 でも割り切れない」ような整数が書類上にあれば DENIED，さもなくば APPROVED とすればよいです．

C 言語での実装例は次のとおりです．

```
#include <stdio.h>

int main (void) {
        int i;

        int a[100];
        for (i = 0; i < n; i++) {
                scanf("%d", &a[i]);
        }

        for (i = 0; i < n; i++) {
                if (a[i] % 2 == 0 && a[i] % 3 != 0 && a[i] % 5 != 0) {
                        puts("DENIED");
                        return 0;
                }
        }
        puts("APPROVED");

        return 0;
}
```

C: Poll

Python や C# などにおける連想配列を使って，各文字列の出現回数を数えてから，最大値をとるものを昇順に出力すればよいです.

std::map の内部ではキーが昇順になるよう要素がソートされているので，イテレータを .begin() から順に見ることで，キーを昇順に見ることができます.

C++ での実装例は次のとおりです.

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

#include <map>
using std::map;

#include <string>
using std::string;

map<string, int> memo;

int main (void) {
        int n;

        scanf("%lld", &n);
        for (int i = 0; i < n; i++) {
                string s;
                cin >> s;

                memo[s] += 1;
        }


        int maxv = 0;
        for (const auto& x : memo) {
                int v = x.second;
                if (v > maxv) maxv = v;
        }
        for (auto it = memo.begin(); it != memo.end(); it++) {
                if (it->second == maxv) {
                        cout << it->first << endl;
```

```
34                  }
35          }
36
37          return 0;
38  }
```

D: Pairs

積が負のペア，0 のペア，正のペアの各個数は簡単に求まるので，答えが負・0・正のどれになるかはこれでわかります．

答えが負になる場合は，負の数と正の数を 1 つずつ選んで x 以上になるペアがいくつあるかを数えることが尺取り法によって可能なので，二分探索を用いて答えが求まります．

答えが正になる場合も全く同様ですが，同じ要素を 2 回選ぶこと，それを差し引くと各ペアをちょうど 2 回数えることを考慮しましょう．

E: Payment

各桁について，金額のその桁までを見たときの「ピッタリ払うときの最小」「1 余分に払うときの最小」を上の桁から計算すればよいです．

DP の遷移は桁ごとに定数時間で出来るので，桁数に対して線形時間で解けました．

F: Perils in Parallel

簡単のため，爆弾は座標の昇順に番号づけられているとします (ソートすることで帰着できます).

各コードの挙動は「番号 l_j 未満の爆弾を切り替え，さらに番号 r_j 以下の爆弾を切り替える」と言い換えられます.

爆弾 i と $i+1$ のオン・オフが同時に切り替わらないのは「番号 i 以下の爆弾を切り替える」操作をしたときのみなので，これを用いて各操作の回数の偶奇が何でなければならないかがわかります. 逆に，これを満たせば全ての爆弾がオフになります.

さて，グラフに思いを馳せます．各コードを「2 頂点 l_i, r_i を結ぶ辺」と言い換えます ($l_i = r_i$ となるコードは考えるだけ無駄なので結ばなくてよいです)．そして，「i 以下の…」という操作が奇数となる頂点 i に印をつけます.

このとき，各連結成分について，印の数は偶数でなければなりません (どの辺に対応する 2 操作をしても連結成分内の偶奇は変化しない)．逆に，すべての連結成分について印が偶数個であれば，以下のようにして構成できます.

- 全域木を何でもいいので 1 つとる (明示的でなくてもよく，適当に DFS してとれる経路でよい)．以下これを根付き木として考える.
- 各辺について，その辺よりも下の部分木について印の個数が奇数のとき，またその時に限り，その辺に対応するコードを切る.

以上で構成は完了です.

A: Poor

For example, the condition of $A = B$ and $B \neq C$ can be represented as `A == B && B != C` in C Language.

By splitting cases correctly based on this, it will be like the following implementation example. The following is a sample code in C Language.

```c
#include <stdio.h>

int main(void){
        int a, b, c;

        scanf("%d%d%d", &a, &b, &c);

        int ispoor = 0;
        if (a == b && b != c) ispoor = 1;
        if (b == c && c != a) ispoor = 1;
        if (c == a && a != b) ispoor = 1;

        if (ispoor) {
                puts("Yes");
        } else {
                puts("No");
        }

        return 0;
}
```

B: Papers, Please

If there exists a integer such that "although it is an even number, it is not divisible by 3 nor 5," on the document then the answer is `DENIED`, otherwise the answer is `APPROVED`.

The following is a sample code in C Language.

```
#include <stdio.h>

int main (void) {
        int i;

        int a[100];
        for (i = 0; i < n; i++) {
                scanf("%d", &a[i]);
        }

        for (i = 0; i < n; i++) {
                if (a[i] % 2 == 0 && a[i] % 3 != 0 && a[i] % 5 != 0) {
                        puts("DENIED");
                        return 0;
                }
        }
        puts("APPROVED");

        return 0;
}
```

C: Poll

Using an associative arrays in Python or C#, count the appearances of each string, and output the string taking the maximum value in the increasing order.

Inside std::map, the keys are sorted in the increasing order, so by looking at the iterator in order from .begin(), the keys can be iterated in the increasing order.

The following is an implementation example in C++.

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

#include <map>
using std::map;

#include <string>
using std::string;

map<string, int> memo;

int main (void) {
        int n;

        scanf("%lld", &n);
        for (int i = 0; i < n; i++) {
                string s;
                cin >> s;

                memo[s] += 1;
        }


        int maxv = 0;
        for (const auto& x : memo) {
                int v = x.second;
                if (v > maxv) maxv = v;
        }
        for (auto it = memo.begin(); it != memo.end(); it++) {
                if (it->second == maxv) {
                        cout << it->first << endl;
```

```
34                  }
35          }
36
37          return 0;
38  }
```

D: Pairs

The number of pairs such that the products are negative, 0 or positive can be calculated easily, so it is determined whether the answer is negative, 0 or positive.

When the answer is negative, we can count the number of the pairs such that a negative number and a positive number are chosen and the product is more than x by using sliding window, so the answer can be found by binary searching.

The answer can be found likewise when the answer is positive too, but note that you may choose the same element twice, and when they are eliminated each pair is counted twice.

E: Payment

For each digit, calculate "the minimum when paying exactly" and "the minimum when paying one more" when focusing on the prefix of the price until the digit, from the higher to the lower.

Since each DP transition can be performed in a constant time for each digit, the problem has been solved in a linear time of the number of digits.

F: Perils in Parallel

For simplicity, we assume that the bombs are indexed in the increasing order of the coordinates. (It can be achieved by sorting them.)

The behavior of each cord can be reworded to "switch the bombs with the indices less than l_j, and also switch the bombs with the indices less than or equal to r_j."

The activated/deactivated states of bomb i and $i+1$ are switched only when "the bombs with the indices less than or equal to i are switched," so by using the conditions, the parity of the number of each operation is determined. Conversely, if all those parities are satisfied, then all the bombs will be deactivated.

Now, let's think of graphs. Let's regard each code as "an edge connecting between two vertices l_i, r_i" (the cords such that $l_i = r_i$ does not have to be connected because it has no effect). Then, mark vertices i such that the number of operation of "less than or equal to i" is an odd number.

Then, for each connected component, the number of marks has to be even (performing two operations corresponding to any edges does not change the parity in the connected component). Conversely, if the number of marks are even for all the connected components, then the following construction is possible.

- Take an arbitrary spanning tree (not necessarily explicitly; it can be a path of some arbitrary DFS). Hereinafter we will regard this as a rooted tree.
- For each edge, cut the corresponding cord if and only if the number of marks in the subtree under the edge is odd.

Here the construction ends.