

# ABC 155 解説

writer: sheyasutaka, rng\_58, beet, drafear

2020 年 2 月 16 日

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

## 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の実装例  
C 言語での実装例は次のとおりです。

---

```
1 #include <stdio.h>
2
3 int main(void){
4     int a, b, c;
5
6     scanf("%d%d%d", &a, &b, &c);
7
8     int ispoor = 0;
9     if (a == b && b != c) ispoor = 1;
10    if (b == c && c != a) ispoor = 1;
11    if (c == a && a != b) ispoor = 1;
12
13    if (ispoor) {
14        puts("Yes");
15    } else {
16        puts("No");
17    }
18
19    return 0;
20 }
```

---

## 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，さもなければ 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。

---

```
1 #include <stdio.h>
2
3 int main (void) {
4     int i;
5
6     int a[100];
7     for (i = 0; i < n; i++) {
8         scanf("%d", &a[i]);
9     }
10
11     for (i = 0; i < n; i++) {
12         if (a[i] % 2 == 0 && a[i] % 3 != 0 && a[i] % 5 != 0) {
13             puts("DENIED");
14             return 0;
15         }
16     }
17     puts("APPROVED");
18
19     return 0;
20 }
```

---

## 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。

---

```
1 #include <iostream>
2 using std::cin;
3 using std::cout;
4 using std::endl;
5
6 #include <map>
7 using std::map;
8
9 #include <string>
10 using std::string;
11
12 map<string, int> memo;
13
14 int main (void) {
15     int n;
16
17     scanf("%lld", &n);
18     for (int i = 0; i < n; i++) {
19         string s;
20         cin >> s;
21
22         memo[s] += 1;
23     }
24
25
26     int maxv = 0;
27     for (const auto& x : memo) {
28         int v = x.second;
29         if (v > maxv) maxv = v;
30     }
31     for (auto it = memo.begin(); it != memo.end(); it++) {
32         if (it->second == maxv) {
33             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  $l_j$  以下の爆弾を切り替え, 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.