AtCoder Beginner Contest 064 解説

問題作成者: E869120 / square1001

2017/6/10

[[For international readers: English editorial starts at page 5.]]

A 問題 : RGB Cards

3 つの 1 桁の整数 r, g, b をこの順に並べた数は $100r + 10g + b$ となります。

これが 4 の倍数か判定するだけです。「n が 4 の倍数」は「n を 4 で割った余りが 0」であるのと同じなので、多くのプログラミング言語では n % 4 == 0 が true であれば “Yes”, false であれば “No” と出力することで正解が得られます。

サンプルコード (C++)

```
#include <iostream>
using namespace std;
int A, B, C;
int main() {
    cin >> A >> B >> C;
    if((A * 100 + B * 10 + C) % 4 == 0) cout << "YES" << endl;
    else cout << "NO" << endl;
    return 0;
}
```

B 問題：Traveling AtCoDeer Problem

最適な移動方法の一つは、座標の小さい順にプレゼントを配り、寄り道をしないということです。すると、移動距離は（一番右の家の座標）－（一番左の家の座標）となり、これは $\max(a_1, a_2, a_3, \ldots, a_N) - \min(a_1, a_2, a_3, \ldots, a_N)$ と等しいです。

Max と min の値は、条件分岐を使って $O(n)$ で計算することができます。

サンプルコード（C++）

```
#include <iostream>
using namespace std;
int N, A, MIN = 1001, MAX = -1;
int main() {
    cin >> N;
    for(int i = 1; i <= N; i++) {
        cin >> A;
        if(MIN > A) MIN = A;
        if(MAX < A) MAX = A;
    }
    cout << MAX - MIN << endl;
    return 0;
}
```

C 問題：Colorful Leaderboard

まず、全員のレートが 3199 以内であることを考えてみましょう。

灰、茶、緑、水、青、黄、橙、赤色をその順に 0, 1, 2, 3, 4, 5, 6, 7 と番号付けることにします。そのとき、色番号 i の人のレートを r としたとき、$400i \leq r < 400(i+1)$ を満たします。例えば、色番号 5 (=黄色) の人のレートは、400×5=2000 以上 400×6=2400 未満 となります。

次に、「色番号 i の人がいるか」というのを確かめることにします。これは、$400i \leq a_j < 400(i+1)$ を満たすような j が存在するならば色番号 i の人はいます。また、そうでなければいません。

これを $0 \leq i \leq 7$ の範囲で全探索すると、計算量 $O(n)$ で色の種類数を求めることができます。

次に、レート 3200 以上の人が存在することを考えてみましょう。レート 3200 以上の人は、色を自由に設定することが可能なので、当然設定によっては色の種類数は変動してきます。

そこで、レート 3200 以上の人を n 人、レート 3199 以下の人に対する色の種類数を c とします。そのとき、最大値 / 最小値 は次のようになります。

1. 最大値

レート 3200 以上の人 n 人がすべて違う色を設定し、かつどの人も 灰 / 茶 / 緑 / 水 / 青 / 黄 / 橙 / 赤色 を設定している場合、$n+c$ 種類の色ができます。また、変えられる色が n 個しかないので、これが最大であることが分かります。

2. 最小値

a. $c \geq 1$ のとき

レート 3199 以下の人が 1 人以上いるので、このうち 1 人の色を X とすると、レート 3200 以上の人が全員色 X に設定すれば色の種類数は c 種類 となります。

b. $c = 0$ のとき

この場合、全員がレート 3200 以上であるということになります。人が 1 人以上で色の種類数が 0 種類であることはあり得ないので、a. は成り立ちません。また、最適解はこの場合全員が同じ色を選んだ時に色の種類数は 1 種類となります。

よって、この問題は計算量 $O(n)$ で解くことができます。

D 問題：Insertion

文字列 s が括弧列である ⇔ 文字列 s の '(' の個数と ')' の個数が同じかつ任意の接頭辞においても '(' の個数が ')' の個数以上である が成り立つ。

ただし、接頭辞とは s の最初の x 文字 $(0 \leq x \leq |s|)$ のことである。($|s| = (s$の長さ$)$)

d_i を s の最初の i 文字に含まれる '(' の個数 − ')' の個数 とし、$x = \min(d_0, d_1, \ldots, d_n)$ とするとき、'(' や ')' を挿入した時に $d_{|s|} = 0$ かつ $x = 0$ となればよい。

そこで、'(' を 1 つ挿入すると $d_{|s|}$ が 1 増え、x が 0 または 1 増える。')' を 1 つ挿入すると $d_{|s|}$ が 1 減り、x が 0 または 1 減る。そこで、次の 4 つの操作のいずれかをして、$a = 0$ かつ $b = 0$ となるようにすることを考える。 $(a = d_{|s|}, b = x$ に対応している)

a. a を 1 増加させる。
b. a を 1 増加させ、b を 1 増加させる。
c. a を 1 減少させる。
d. a を 1 減少させ、b を 1 減少させる。

まず、$b \leq 0$ の状態から $b = 0$ にしなければならないので、b. の操作を $-b$ 回行うことにする。d. の操作を 1 回やると b. の操作を 1 回してもとに戻り、それ以外に方法はないので無駄になる。そのとき、a の値が (元の a) + b となっており、$a \geq b$ より現在の a は 0 以上になっている。$a = 0$ にするためには c. の操作を (元の a) + b 回行わなければならない。よって、挿入回数が $a - 2b = d_{|s|} - x$ 回未満になることがないことが分かる。

そこで一番左に '(' を $-x$ 個、一番右に ')' を $d_{|s|} - x$ 個挿入すると $d_{|s|} - 2x$ 回の挿入でできるので、先ほどの証明より文字数が最小であることが分かる。また、その中で辞書順最小も達成できる。

よって、この問題は $O(|s|)$ で解くことができた。

AtCoder Beginner Contest 064 Editorial

Problem Setter: E869120 / square1001

2017/6/10

A: RGB Cards

The number that AtCoDeer reads is $100a + 10b + c$. You can calculate with the remainder of $100a + 10b + c$ divided by 4.

C++ code example:

```
#include <iostream>
using namespace std;
int A, B, C;
int main() {
    cin >> A >> B >> C;
    if((A * 100 + B * 10 + C) % 4 == 0) cout << "YES" << endl;
    else cout << "NO" << endl;
    return 0;
}
```

B: Traveling AtCoDeer Problem

One of the optimal way of delivering gifts is to move house in ascending order. In this case, the moving distance is (rightmost house coordinate) − (leftmost one), is equal to $\max(a_1, a_2, \ldots, a_n) - \min(a_1, a_2, \ldots, a_n)$.

C++ code example:

```
#include <iostream>
using namespace std;
int N, A, MIN = 1001, MAX = -1;
int main() {
    cin >> N;
    for(int i = 1; i <= N; i++) {
        cin >> A;
        if(MIN > A) MIN = A;
        if(MAX < A) MAX = A;
    }
    cout << MAX - MIN << endl;
    return 0;
}
```

C: Rating Colors

Let x the number of people who rating is more or equal to 3200, and let c the number of different colors whose rating is less than 3200.

x can calculate with the number of i that satisfies $a_i \geq 3200$, and c is the number of i $(0 \leq i \leq 7)$ that at least one j exists that satisfies $400i \leq a_j < 400(i+1)$, because the color of rating (less than 3200) can identify with the quotient of a_i divided by 400.

Maximum value / Minimum value can calculate as following method:

1. **Maximum value**

   The optimal way is setting distinct colors in people whose rating is more or equal to 3200, and also setting color is different from gray, brown, green, cyan, blue, yellow, orange, and red. The number of different colors will be $x + c$.

2. **Minimum value**
   a. **Case $c \geq 1$**

      In this case, there exist a person X who rating is below 3200. The optimal way is setting same color as person X, for all people whose rating is more or equal to 3200. The number of different colors will be c.

   b. **Case $c = 0$**

      This pattern is tricky. For the constraints $n \geq 1$, if all people select same color, the number of different colors will be 1. This is the optimal way.

D: Insertion

For a string S that consist from '(' and ')', S is a correct bracket sequence if and only if S includes exactly same number of '(' and ')', and the number of '(' is large or equal to the number of ')' for all prefix.

Let d_i = (the number of '(' in i-th prefix) − (the number of ')' in i-th prefix). i-th prefix is the prefix with length i. The objective is to be $d_s = 0$ and $x = 0$, with some insertion of '(' and ')'.

If you insert '(', $d_{|s|}$ increases by 1 and x increases by 0 or 1. If you insert ')', $d_{|s|}$ decreases by 1 and x decreases by 0 or 1. In the constraints $x \leq 0$, $d_{|s|} - x \geq 0$, if you can do all 4 operations freely and independently ($d_{|s|}+= 1$, $d_{|s|}+= 1$ and $x+= 1$, $d_{|s|}-= 1$, $d_{|s|}-= 1$ and $x-= 1$), the optimal way is operating $d_{|s|}+= 1, x+= 1$ for $-x$ times, and operating $d_{|s|}-= 1$ for $d_{|s|} - x$ times, so the minimum number of operation is $d_{|s|} - 2x$ times.

Therefore, the lower bound of number of insertion is $d_{|s|} - 2x$ times, and you can achieve in following way:

1. Insert '(' in the leftmost place $-x$ times.
2. Insert ')' in the rightmost place $d_{|s|} - x$ times.

The total number of insertion is $d_{|s|} - 2x$, and it is same as the lower bound that already proved. So, this is the minimum insertion. And this is also lexicographically smallest.