ABC 165 解説

beet, DEGwer, kort0n, sheyasutaka, tozangezan, ynymxiaolongbao

2020 年 5 月 2 日

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

A: We Love Golf

いろいろな方法がありますが，ここでは B 以下の最大の K の倍数を求め，それが A 以上かどうか確かめる方法を採ってみましょう．

実装はたとえば以下のようになります．

Listing 1　C での実装例

```c
#include <stdio.h>

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

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

        int largest = (b / k) * k;
        if (a <= largest) {
                puts("OK");
        } else {
                puts("NG");
        }

        return 0;
}
```

B: 1%

今持っている金額を P とすると、$P < X$ のあいだ P に $\left\lfloor \frac{P}{100} \right\rfloor$ を足せばよいです。
入力例 2 からもわかるように、 ループ回数は高々 3760 回となり、十分高速です。
64bit 整数を扱う言語ではオーバーフローに注意してください。
($P' = \left\lfloor \frac{P \times 101}{100} \right\rfloor$ のように計算するとオーバーフローする場合があります)
以下は C++ による回答例です。

```
#include <bits/stdc++.h>
using namespace std;
int main(){
  long long X;
  cin >> X;
  long long P = 100, step = 0;
  while(P < X){
    P += P / 100;
    step++;
  }
  cout << step << endl;
  return 0;
}
```

C: Many Requirements

結論から述べると, 問の条件を満たす数列を全探索します.
問の条件を満たす数列は, N 個の仕切りと M 個のボールを並び替えた列と一対一に対応します. 実際, 各列に対して,

$$A_i = i \text{ 番目の仕切りより左側に存在するボールの数}$$

とすれば, 前述の一対一対応が得られます.
N 個の仕切りと M 個のボールを並び替える列の数は, ${}_{N+M}\mathrm{C}_N$ 通り存在します ($N+M$ 個の場所から仕切りを置く N 箇所を選ぶ方法の数え上げに相当).
各数列を深さ優先探索で生成する場合, 各数列の生成に要する時間は $O(N)$ ですから, 全ての数列の生成に要する時間は $O(N\,{}_{N+M}\mathrm{C}_N)$ です.
数列を決めればその数列のスコアは $O(Q)$ で計算出来ます.
以上より, この問題は $O(N\,{}_{N+M}\mathrm{C}_N Q)$ で解けました.

D: Floor Function

$f(x) = floor(Ax/B) - A \times floor(x/B)$ とします。

$f(x+B) = f(x)$ であることは実際に代入することで容易に分かります。なので、$0 \leq x \leq B-1$ の場合のみを考えます。

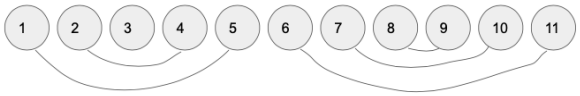
このとき、$f(x) = floor(Ax/B) - A \times floor(x/B) = floor(Ax/B)$ で、$floor(Ax/B)$ は広義単調増加なので、与えられた制約 $(0 \leq x \leq B-1,\ 0 \leq x \leq N)$ の中で最も大きい x について、$f(x)$ が求める最大値です。

よって、答えは $f(min(B-1, N))$ です。

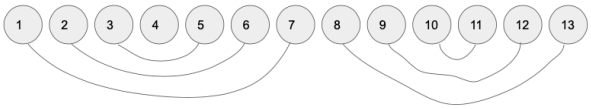
E:Rotation Matching

$N = 2 \times M + 1$ のケースを考えます。すると、以下のような割り振りが条件を満たすことがわかります。ただし、図では、線で結ばれた数字を同じ対戦場に割り振ることを意味します。

M が奇数のとき（具体例は $M = 5$ のとき）



M が偶数のとき（具体例は $M = 6$ のとき）



N が $2 \times M + 1$ より大きい場合でも、このような割り振りが条件を満たします。

F: LIS on Tree

結論から述べると、この問題は最長増加部分列問題を解く動的計画法に巻き戻しのテクニックを適応させることで解くことができます。

はじめに、最長増加部分列問題 (Longest Increasing Subsequence) を解く動的計画法について説明します。LIS には様々なアプローチがありますが、二分探索を用いた動的計画法が最も有名です。dp_i を長さが i である増加部分列における最終要素の最小値（存在しない場合は無限大）として、$dp_1, dp_2, .., dp_N$ を無限大に初期化し、dp_i の値が A_j 以上になる最も小さな i に対して dp_i を A_j の値に更新するということを $j = 1, 2, .., N$ について行っていくといった方法です。

次に、巻き戻しのテクニックについて説明します。ここでの巻き戻しとは、変数の値の更新と質問を次々に処理するという状況の中で、直前の更新をする前の状態に戻すことをいいます。以下のような処理をすることで実現できます。

- ある変数の値を更新したとき、
  - どの変数を更新したか
  - その変数の元の値は何だったか

  という情報を stack に push する。
- 巻き戻しをするとき、stack の top に示された変数をその元の値に変更し、stack を pop する。

最後に、この問題への解法を説明します。頂点 1 を始点として深さ優先探索をし、各頂点に対応する再起関数内では、はじめに dp の値を更新して答えを求めた後、隣接する頂点に対応する再起関数を呼び出し、最後に dp の値を巻き戻すことで、すべての頂点までの答えを正しく求めることができます。計算量は $O(NlogN)$ です。