ABC 161 解説

chokudai, evima, kyopro_friends, latte0119,
namonakiacc, sheyasutaka, tozangezan, ynymxiaolongbao

2020 年 4 月 4 日

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

A: ABC swap

実際にシミュレーションを行えばよいです。以下は C++ における実装例です。

```
#include<iostream>
using namespace std;

int main(){
        int a,b,c;
        cin>>a>>b>>c;
        swap(a,b);
        swap(a,c);
        cout<<a<<"␣"<<b<<"␣"<<c<<endl;
        return 0;
}
```

B: Popular Vote

M 位の商品が総得票数の $\frac{1}{4M}$ 以上の票を獲得していれば YES、そうでなければ NO です。これは得票数順にソートすることで判定できます。

```
N, M = map(int,input().split())
A = list(map(int,input().split()))
A.sort(reverse=True)
S = sum(A)
if A[M-1] >= S / (4*M):
    print("Yes")
else:
    print("No")
```

その他、「総得票数の $\frac{1}{4M}$ 以上の票を獲得している商品が M 個以上あれば YES、そうでなければ NO」として解くこともできます。この場合ソートは不要です。

```
N, M = map(int,input().split())
A = list(map(int,input().split()))
S = sum(A)
cnt = 0
for a in A:
    if a >= S / (4*M):
        cnt += 1
if cnt >= M:
    print("Yes")
else:
    print("No")
```

なお、C++,Java などの言語を利用している場合、除算の挙動に注意してください (整数型変数同士の除算では、結果は整数に切り捨てられます)。

C: Replacing Integer

x が K 以上のとき、操作を行うことで $x-K$ となります。すなわち、N から N/K 回操作を行うことで、整数は N を K で割った余りとなります。

N を K で割った余りを t とします。t に操作を行うと、 $K-t$ となります。$K-t$ に操作を行うと、t に戻るだけです。すなわち K 以下の値として取りうるものは t と $K-t$ のいずれかのみとなります。

よって答えは t と $K-t$ のうち小さい方、すなわち N を K で割った余りと、$K-$ (NをKで割った余り) のうち小さい方です。

D: Lunlun Number

この問題は、Queue というデータ構造を用いることで効率的に解くことができます。まず、空の Queue を 1 つ用意し、$1, 2, ..., 9$ を順に Enqueue します。それから、以下の操作を K 回行います。

- Queue に対して Dequeue を行う。取り出した要素を x とする。
- $x \, mod \, 10 \neq 0$ なら、 $10x - 1$ を Enqueue する。
- $10x$ を Enqueue する。
- $x \, mod \, 10 \neq 9$ なら、 $10x + 1$ を Enqueue する。

K 回目の操作において取り出した数が、 K 番目の Lunlun Number となっています。

E:Yutori

ある期間内に働く日数を最大化するためには、前から貪欲に働く日を決めるのが最適です。したがって前から貪欲に働く日を決めた場合を考えることで「x 回目に働く日は $L[x]$ 日目以降」という配列 L を求めることができます。同様に後ろから貪欲に働く日を決めた場合を考えることで「x 回目に働く日は $R[x]$ 日目以前」という配列 R を求めることができます。i 日目に必ず働くのは、$L[x] = R[x] = i$ となる x が存在するときそのときに限るので、この問題は $O(N)$ で解けました。

F:Division or Substraction

割り算の操作を 1 度もしない場合、操作によって $N \bmod K$ は変化しません。したがって、最終的に 1 になるためには $N \bmod K = 1$ であることが必要十分です。そのような K は $N-1$ の約数のうち、1 でないもの全てであり、その個数は $O(\sqrt{N})$ で求めることができます。

割り算の操作を 1 度以上する場合、そのような K は N の約数です。実際に K で割り切れなくなるまで操作をしたあと、$N \mod K = 1$ になっているかを確かめることにより、各約数毎に $O(\log N)$ で判定することができます。

よって以上により問題が解けました。N および $N-1$ の約数を求める部分がボトルネックとなり、計算量は $O(\sqrt{N})$ となります。