ABC 118 解説

writer: drafear

2019 年 2 月 16 日

A: B +/- A

A が B の約数であるとは、B が A で割り切れることなので、B を A で割った余りが 0 であるかを調べれば判定できます。したがって、C++ で実装した場合下のコードで正答することができます。

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  int A, B; cin >> A >> B;
  if (B % A == 0) {
    cout << A + B << endl;
  }
  else {
    cout << B - A << endl;
  }
}
```

B: Foods Loved by Everyone

各食べ物 i についてそれを好きだと答えた人数をカウントし、c_i とします。すると、$c_i = N$ である i の数が答えとなります。これを C++ で実装すると次のようになります。

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  int N, M; cin >> N >> M;
  vector<int> cnt(M, 0);
  for (int i = 0; i < N; ++i) {
    int K; cin >> K;
    for (int j = 0; j < K; ++j) {
      int A; cin >> A; --A;
      ++cnt[A];
    }
  }
  int ans = 0;
  for (int i = 0; i < M; ++i) {
    if (cnt[i] == N) {
      ++ans;
    }
  }
  cout << ans << endl;
}
```

C: Monsters Battle Royale

$A_1, A_2, ..., A_N$ の最大公約数を g とします。すると、以下で示すように g が答えになります。

入力例 1 において、なぜどのように攻撃を行っても体力 1 にならないのでしょうか。これは、どのモンスターも初期体力が偶数なので、攻撃して体力が変化しても偶数にしか変化しないためです。これを一般化すると、体力 a のモンスターが体力 b $(b > a)$ のモンスターに攻撃したときに攻撃されたモンスターの体力は $b-a$ となりますが、a も b も x の倍数であるなら $b-a$ も x の倍数です。したがって、常に x として g を取れるので、生きているモンスターの体力は必ず g の倍数となり、g 未満になることはありません。

逆に、次のように最後に生き残ったモンスターの体力を g にすることができます。体力 a のモンスターと体力 b のモンスターがお互いの体力が等しくなるまで体力の低いほうが高い方を攻撃し続けるとユークリッドの互除法と同じ遷移になるため最終的な体力はお互いに a と b の最大公約数になります。これを 1 番目のモンスターと 2 番目のモンスター, 2 番目のモンスターと 3 番目のモンスター, ... と順に行うと、最終的に N 番目のモンスターの体力が $A_1, A_2, ..., A_N$ の最大公約数 g になります。最後に N 番目のモンスターが他のモンスターを攻撃し続ければ、最後に生き残ったモンスター (N 番目のモンスター) の体力が g になります。

実装は、ユークリッドの互除法により $A_1, A_2, ..., A_N$ の最大公約数を求めて出力すればよく、これは $O(N + logA_1)$ で動作します。

D: Match Matching

桁数が異なる整数同士では、桁数が多い方が大きいため、まずはちょうど N 本のマッチ棒を使って最大何桁作れるかを求めることを考えます。ちょうど N 本使わないといけないため、貪欲に作ることはできず、動的計画法で解くことになります。

具体的には

$$dp(i) := \text{ちょうど } i \text{ 本のマッチ棒を使って、条件を満たす整数を作るときの最大桁数}$$

と定義し、これを求めます。これが求まると、答えとなる整数の桁数は $dp(N)$ です。これの求め方については後述します。

今度は、桁数が同じ整数同士の大小は辞書順となるため、できるだけ大きい数字を上位の桁に使うようにします。すなわち、一般性を失わずに $A_1 > A_2 > ... > A_M$ とすると、A_1 を最上位の桁に使えるかを調べ、使えるなら使う、使えないなら A_2 を最上位の桁に使えるかを調べ、・・・と試していきます。ここで、整数 k を 1 桁作るのに使うマッチ棒の本数を $num(k)$ とすると、A_i を最上位の桁に使えることを調べるには、$dp(N - num(A_i)) = dp(N) - 1$ であるかを調べればよいです。最上位の桁が決まれば、同様に、上から 2 番目、3 番目と上位の桁から順に決めていきます。

さて、残るは $dp(i)$ の求め方です。これは、次のように求められます。ただし、ちょうど i 本のマッチ棒を使って条件を満たすように整数を作れない場合は $dp(i) = -\infty$ とします。

$dp(i)$ を求める際には、ちょうど i 本のマッチ棒を使って条件を満たす整数の最上位の桁 A_j を全探索します。最上位の桁が A_j だとすると、ちょうど $i - num(A_j)$ 本のマッチ棒を使って作った $dp(i - num(A_j))$ 桁の整数の頭に A_j を付け加えて $dp(i - num(A_j)) + 1$ 桁の整数を作ることができます。すなわち、

$$dp(i) = \max_j (dp_{i-num(A_j)} + 1)$$

が成り立ちます。さらに、$dp(0) = 0$ なので、$i = 1, 2, ...$ と小さい順に求めていくことができます。

このようにして解くことができ、出力桁数が $\frac{N}{2}$ 以下であることに注意すれば[*1]、計算量は $O(NM + MlogM)$ になります。

余談: 0 も作れる場合は少し複雑になりますが同じような方法で解くことができます。

[*1] $111\cdots1$ と出力する場合が最大です