ABC 120 解説

writer: drafear

2019 年 3 月 3 日

A: Favorite Sound

最大 C 回まで聞けるといった制約が無い場合、答えは $\frac{B}{A}$ の小数点以下切り捨てになります。従って、最大 C 回の制約がある場合、この値と C の小さい方が答えになります。以下に C++ による回答例を示します。

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

using namespace std;

int main() {
  int A, B, C; cin >> A >> B >> C;
  int ans = min(B / A, C);
  cout << ans << endl;
}
```

B: K-th Common Divisor

正整数 i が A も B も割り切るなら、$1 \leq i \leq \min(A, B)$ を満たします。したがって、$M = \min(A, B)$ として $i = M, M-1, M-2, ..., 1$ と順に i が A, B を共に割り切るか調べていき、K 番目に見つけた数を出力すれば正答できます。M を $\min(A, B)$ とする代わりに、単に A や B, 100 など M が $\min(A, B)$ 以上であれば正答できます。以下に C++ での実装例を示します。

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

using namespace std;

int main() {
  int A, B, K; cin >> A >> B >> K;
  for (int i = min(A, B); i >= 1; --i) {
    if (A % i == 0 && B % i == 0) {
      --K;
      if (K == 0) {
        cout << i << endl;
        return 0;
      }
    }
  }
  assert(false);
}
```

別解として、A も B も割り切る数 (A と B の公約数) は A と B の最大公約数の約数なので、これを列挙して大きい方から K 番目の数を出力する方法もあります。

C: Unification

文字列 S に 0 も 1 も 1 つ以上残っている場合、どこかで 0 と 1 が隣り合っています。従って、与えられた文字列 S に含まれる 0 の個数と 1 の個数を数え、それぞれ C_0, C_1 とすると、どのような順番で取り除いても $\min(C_0, C_1)$ 回取り除く操作ができます。よって、答え (取り除けるキューブの個数の最大値) は $2 \times \min(C_0, C_1)$ になります。このアルゴリズムは $O(|S|)$ で動作します。

D: Decayed Bridges

島を頂点、橋を辺としたグラフを考えます。グラフから辺を削除していくのは難しいことが多いので、逆に辺が 1 つもない状態から追加していくことを考えます。すなわち、後ろから答えを求めて配列に保存し、逆順に出力します。

i 番目に出力すべき答えを $ans(i)$ とします。辺を追加していくことを考えると、初めは全ての辺がないので、

$$ans(M) = \frac{N(N-1)}{2}$$

です。さらに、各 i $(1 \leq i \leq M-1)$ について、$i+1$ 番目の辺 $e_{i+1} = (a_{i+1}, b_{i+1})$ によって

1. 元々頂点 a_{i+1}, b_{i+1} が連結でなかった (互いに行き来可能でなかった) 場合
   辺 e を加える前のグラフにおいて、頂点 a_{i+1} を含む連結成分の大きさを N_1、頂点 b_{i+1} を含む連結成分の大きさを N_2 とすると、
   $$ans(i) = ans(i+1) - N_1 \times N_2$$
   です。
2. 元々連結だった場合
   不便さに変化はなく、$ans(i) = ans(i+1)$ です。

しかしこれを愚直に実装すると $O(M(N+M))$ かかってしまい、間に合いません。ボトルネックになっているところは、N_1, N_2 を求める部分です。これは、以下の操作が行える、Union-Find と呼ばれるデータ構造を用いることで効率よく求めることができます。

unite(u, v): 頂点 u が属するグループと頂点 v が属するグループを併合し、同じグループにする $(O(\alpha(n)))$
find(v): 頂点 v が属するグループ番号を得る $(O(\alpha(n)))$
size(v): 頂点 v が属するグループと同じグループに属する頂点数を得る $(O(1))$

ここで、n は管理する頂点数、α はアッカーマン関数の逆関数ですが、$O(\alpha(n))$ は $O(1)$ とほとんど変わりません。したがって、全体の計算量は $O(M\alpha(N))$ となり十分間に合います。