ABC 142 解説

beet, drafear, kort0n, ynymxiaolongbao

2019 年 9 月 28 日

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

A: Odds of Oddness

実数 x について,x を超えない最大の整数を $floor\left(x\right)$ と定義します.N 以下の正の奇数は $N - floor\left(\frac{N}{2}\right)$ 個数存在しますから, 答えは $\left(N - floor\left(\frac{N}{2}\right)\right)/N$ です.
ただし, 実装には注意が必要です. 例えば C++ では,N を int 型の変数とした場合 $floor\left(\frac{N}{2}\right)$ は $N/2$ で計算出来ますが,$\left(N - N/2\right)/N$ のように書くと, 二度目の除算も int 型の変数同士の除算となり, 正しい計算結果が得られません.
これを回避する手法はいくつかあります。例えば, 二度目の除算が double 型の変数同士の除算として実行されるように, 型変換を行えば良いです.
より詳しくは, AtCoder Programming Guide for beginners Y-3.01. 数値型 をご参照ください。
C++ による解答例:https://atcoder.jp/contests/abc142/submissions/7731173

B: Roller Coaster

N 個の整数を入力し、そのうち K 以上の整数がいくつあるかを数える問題です。
各整数について K 以上であるかを判定すればいいです。
以下は C++ における実装例です。

```cpp
#include <iostream>
using namespace std;
int main(){
  int N,K;
  cin>>N>>K;

  int cnt=0;
  for(int i=0;i<N;i++){
    int h;
    cin>>h;
    if(h>=K) cnt++;
  }

  cout<<cnt<<endl;
  return 0;
}
```

C: Go to School

与えられた順列に対し、各要素のソート後のインデックスを求める問題です。
$rev[A_i] = i$ となるような配列 rev を求めればいいです。
以下は C++ における実装例です。

```cpp
#include <iostream>
#include <vector>
using namespace std;
int main(){
  int N;
  cin>>N;
  vector<int> A(N);
  for(int i=0;i<N;i++) cin>>A[i];

  vector<int> rev(N);
  for(int i=0;i<N;i++) rev[A[i]-1]=i+1;
  for(int i=0;i<N;i++) cout<<rev[i]<<endl;
  return 0;
}
```

D: Disjoint Set of Common Divisors

x, y が互いに素であるのと、x, y の最大公約数が 1 であることは同値です。さらに、$gcd(x, y)$ を x と y の最大公約数とすれば、$gcd(a, x) \leq gcd(ab, x)$ です。したがって、整数 ab $(a, b > 1)$ を選ぶよりも、a を選んだほうが良いです (注意: ab が公約数ならば a も公約数です)。すなわち、P を最適解とすると、P で選んだ公約数の中に ab $(a, b > 1)$ と表せる整数が存在したとすると、代わりに a を選ぶことができます。

よって、1 または素数のみを選んだ最適解が存在します。さらに、1 または素数である公約数を全て選ぶことができるため、これらを全て選ぶのが最適です。

さて、そのような公約数はどのように探せば良いでしょう。考察も実装も簡単な方法としては、A, B をそれぞれ素因数分解し (時間計算量はそれぞれ $O(\sqrt{A}), O(\sqrt{B})$)、共通の素因数を探す方法があります。素因数の個数はそれぞれ $O(\log A), O(\log B)$ 個なので、共通の素因数を愚直に求めても十分間に合います。1 と共通の素因数を選ぶのが最適なので、共通の素因数の個数に 1 を加えた値が答えになります。

考察をもう少し進めると、x が A と B の公約数であることと、x が $gcd(A, B)$ の約数であることは同値なので、$gcd(A, B)$ を素因数分解し、素因数の個数を数え上げる方法もあります。

いずれの方法でも、時間計算量は $O(\sqrt{A} + \sqrt{B})$ となります。

E: Get Everything

動的計画法により解くことが出来ます.
0 以上 2^N 未満の整数 j が,「1 以上 N 以下の整数 k について, 現在購入している鍵で宝箱 k を開けられることと,j を 2 進数表記した際に $k-1$ 桁目が 1 であることが同値」という状態を表すこととします.
dp テーブルを以下のように定義します.

$$dp_{i,j} = i+1 \text{ 番目以降の鍵は購入せずに状態を } j \text{ にする為に必要な最小費用}$$

1 以上 N 以下の整数 i について,0 以上 2^N 未満の整数 d_i を,「1 以上 N 以下の整数 k について, 鍵 i で宝箱 k を開けられることと,d_i を 2 進数表記した際に $k-1$ 桁目が 1 であることが同値」となるように定めます. このとき, 動的計画法の遷移式は

$$dp_{i,j \text{ or } d_i} = \min\left(dp_{i-1,j \text{ or } d_i}, dp_{i-1,j} + a_i\right)$$

となります. 時間計算量は $O\left(2^N M\right)$ です.
C++ による解答例:https://atcoder.jp/contests/abc142/submissions/7731316

F: Pure

閉路が存在しないとき、条件を満たす誘導部分グラフは存在しません。そうでないとき、以下の手順を実行します。

手順 1 ： 閉路をひとつ探し、閉路に含まれる頂点の誘導部分グラフを残す。

手順 2 ： そのグラフが条件を満たすとき、終了する。

手順 3 ： 閉路に用いられない辺が存在するので、その辺の終点から閉路をたどり、その辺の始点に到達したらやめる。このとき通過した頂点集合の誘導部分グラフを残し、手順 2 に戻る。

手順 3 を行うたび誘導部分グラフの頂点数が必ず減少し、また手順 2, 3 を通して誘導部分グラフは常に閉路を持ちます。閉路を持つ頂点数が 2 のグラフは条件を満たすため、必ず条件を満たす誘導部分グラフが得られます。愚直に実装すれば計算量は $O(N(N+M))$ となり、正答できます。