ABC 148 解説

beet, DEGwer, gazelle, kyopro_friends, sheyasutaka, tozangezan

2019 年 12 月 22 日

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

A: Round One

$6 - A - B$ を計算し，その値を出力すればよいです．

入力および出力の方法は言語によって異なりますが，たとえば C++ であれば cin, cout を使うことで実現できます．

Listing 1　C での実装例

```c
#include <stdio.h>

int main(void){
        int A, B;
        scanf("%d%d", &A, &B); // 入力からA, B を受け取る

        int result = 6 - A - B; // 6-A-B を計算し，変数result に格納する
        printf("%d\n", result); // 出力

        return 0;
}
```

Listing 2　ferNANDo での実装例

```
r a7 a6 a5 a4 a3 a2 a1 a0
r f e r n a nn d o
r b7 b6 b5 b4 b3 b2 b1 b0
d a7 b7
e a7 d
f b7 d
x7 e f
d a6 b6
e a6 d
```

```
f b6 d
x6 e f
d a5 b5
e a5 d
f b5 d
x5 e f
d a4 b4
e a4 d
f b4 d
x4 e f
d a3 b3
e a3 d
f b3 d
x3 e f
d a2 b2
e a2 d
f b2 d
x2 e f
d a1 b1
e a1 d
f b1 d
x1 e f
d a0 b0
e a0 d
f b0 d
x0 e f
x4 x4 x4
x5 x5 x5
x7 x6 x5 x4 x3 x2 x1 x0
```

B: Strings with the Same Length

1 から N までループを回し、i 回目のループでは S の i 文字目を出力する、その後 T の i 文字目を空白を開けずに出力する、ということを繰り返すと、最終的に目的の新しい文字列が出力されます。

以下は C++ による実装例です (include 等は省略しています)。

```
char S[110];
char T[110];
int main(){
        int a;scanf("%d",&a);
        scanf("%s%s",S,T);
        for(int i=0;i<a;i++)printf("%c%c",S[i],T[i]);printf("\n");
}
```

C: Snack

A の正の倍数を小さい順に見ていき、それが B の倍数であるかを確認すればよいです。このとき、$A \times B$ は明らかに両方の倍数となるので、高々 B 回確認すれば十分なことがわかります。

以下は C++ による解答例です。オーバーフローに注意してください。

`https://atcoder.jp/contests/abc148/submissions/9083476`

余談ですが、このような数は A と B の最小公倍数と呼ばれており、$O(\log(\min(A, B)))$ の計算量で求めることも可能です。

D: Brick Break

以下、「すぬけさんが満足するレンガの砕き方」を「いい砕き方」と言います。

まず 1 が書かれたレンガが存在しない場合、答えは明らかに -1 です。

そうでない場合、1 が書かれたレンガの中で最も左にあるものの番号を i とします。このとき、i 番目のレンガを砕く「いい砕き方」に対して、それと同じ個数のレンガが残るような、i 番目のレンガを砕かない「いい砕き方」が存在します。というのも、元の砕き方で残る 1 が書かれたレンガを砕き、代わりに i 番目のレンガを砕かないとしても変わらず「いい砕き方」になるからです。よって、残すレンガの個数が最大の「いい砕き方」を探す上で、i 番目のレンガは砕かないと仮定しても問題ありません。

同様の考え方をすると、i 番目のレンガより右にあるレンガの中に 2 が書かれたものが存在していれば、そのうち最も左にあるものは砕かずに残すと仮定することが可能です。

この貪欲的なアプローチを繰り返していくことで、残すレンガの個数が最大の「いい砕き方」を見つけることができます。

組合せ最適化問題を解く際に、ある要素を選んで損をすることがないので、その要素を選ぶと仮定して探索空間を小さくする、というアイデアは、競技プログラミングにおける貪欲法で典型的です。

E: Double Factorial

$f(N)$ を 10 進法で表記したときに末尾に続く 0 の個数は、$f(N)$ が 2 で割り切れる回数と $f(N)$ が 5 で割り切れる回数の min です。

以下、N の偶奇に分けて考えます

- N が奇数の場合 $\cdots$ $f(N)$ はいくつかの奇数の積なので、2 で割れることはありません。よって答えは常に 0 です。
- N が偶数の場合 $\cdots$ 明らかに、$f(N)$ が 2 で割り切れる回数は 5 で割り切れる回数よりも多いので、5 で割り切れる回数のみを考えます。

$f(N) = N(N-2)(N-4)\cdots 2$ が 5 で割り切れる回数は、$(2, 4, \cdots, N$ のうち 5 で割り切れるものの個数$) + (2, 4, \cdots, N$ のうち 5^2 で割り切れるものの個数$) + (2, 4, \cdots, N$ のうち 5^3 で割り切れるものの個数$) \cdots$ となります。それぞれ、$2, 4, \cdots, N$ のうち 5 で割り切れるものの個数は $floor(N/10)$、$2, 4, \cdots, N$ のうち 5^2 で割り切れるものの個数は $floor(N/50)$、$2, 4, \cdots, N$ のうち 5^3 で割り切れるものの個数は $floor(N/250)$、$\cdots$ なので、$N/10$ から順に、分母が N 以下の間、分母を 5 倍して商を求めるというのを繰り返し、総和を取ればよいです。

F: Playing tag on tree

高橋君が最初の行動で青木君のいる頂点にしか移動できないとき答えは 0 です。そうでないときを考えます。

青木君が最初にいる頂点を根とします。高橋君と青木君はすれ違うことができないので、ゲームが終了するのは高橋君が”行き止まり”に追い詰められたときです (そうでない場合、高橋君は葉の方へ逃げることでゲームを引き延ばせます)。よって、ゲームが終了する直前は「高橋君が葉 x、青木君が x の親の親にいる」という状況になり、その後 2 人が 1 度ずつ行動し、ともに x の親にいる状況でゲームが終了します。

青木君は常に最短距離で高橋君を追い詰めることができるので、青木君の移動回数は高橋君がどの葉で追い詰められたかのみに依存します。したがって高橋君は追い詰められる場所として、「高橋君が青木君より先に到達することができるような葉のうち、青木君から最も遠いもの」を選択するのが最善となります。これは頂点 u, v からの距離をそれぞれ DFS や BFS などによって計算することで求めることができ、計算量は $O(N)$ です。