# ABC 090/ARC 091 解説

DEGwer

2018/03/11

*For International Readers: English editorial starts on page 6.*

## A: Diagonal String

$c_{11}, c_{22}, c_{33}$ を順に出力すればよいです。

```
#include<stdio.h>
int main()
{
	char a[10][10];
	scanf("%s%s%s", a[0], a[1], a[2]);
	printf("%c%c%c\n", a[0][0], a[1][1], a[2][2]);
}
```

## B: Palindromic Numbers

$A$ 以上 $B$ 以下の全ての整数について、(5 桁であることを適宜利用して) 回文数かどうか判定していけばよいです。

```
#include<stdio.h>
int main()
{
	int a, b;
	scanf("%d%d", &a, &b);
	int r = 0;
	for (int i = a; i <= b; i++)
	{
		int s = i % 10, t = i / 10000 % 10;
		int u = i / 10 % 10, v = i / 1000 % 10;
		if (s == t&&u == v)r++;
```

```
        }
        printf("%d\n", r);
}
```

## C: Flip,Flip, and Flip.....

$N \leq M$ として一般性を失いません。偶数回裏返されるカードは全操作の終了後には表を、奇数回裏返されるカードは裏を向くことになるので、各マスのカードが何回裏返されるかを考えることにしましょう。

$N = 1$ かつ $M = 1$ のとき、1 枚だけあるカードは 1 回裏返されるので、答えは 1 です。

$N = 1$ かつ $M \neq 1$ のとき、$M$ 枚のカードのうち両端のカードは 2 回、それ以外のカードは 3 回裏返されるので、答えは $M - 2$ です。

$N \geq 2$ のとき、$N \leq M$ より $M \geq 2$ で、このとき、

- 四隅のカードは 4 回
- それ以外の周上のカードは 6 回
- それ以外のカードは 9 回

裏返されるので、答えは $(N-2)(M-2)$ です。

以上で答えがすべて求まりました。

## D: Remainder Reminder

$b$ を固定して考えましょう。$b = 1, 2, \cdots N$ に対し、$a$ を $b$ で割った余りが $K$ 以上であるような $1 \leq a \leq N$ の個数が高速に求められれば良いです。簡単のため、$a = 0$ も許すことにして、あとで $a = 0$ の場合 (これは簡単に求められます) を引くことにしましょう。

さて、整数 $p, q$ を用いて $N = pb + r (0 \leq r < b)$ という形で $N$ を一意的に表したとき、$a$ を 0 から $N$ まで順に動かせば、$a$ を $b$ で割った余りを順に並べたものは、$0, 1, 2, \cdots b-1$ という列が $p$ 回繰り返され、最後に $0, 1, 2, \cdots r$ という列が付け加わったものになります。

$0, 1, 2, \cdots b-1$ という列が $p$ 回繰り返される部分には条件を満たす $a$ の個数は $p \times \max(0, b-K)$ 個、最後の部分には $\max(0, r-K+1)$ 個あるので、条件を満たす $a$ の個数が $O(1)$ 時間で求められ、$O(N)$ 時間でこの問題を解くことができました。

## E: LISDL

まず、条件を満たす列が存在する条件を考えてみましょう。増加部分列と減少部分列は 2 つ以上の要素を共有できないため、$A + B - 1 \leq N$ が必要です。また、長さ $N$ の列 $P$ と $1 \leq i \leq N$ について、$f(i)$ で $P$ の $i$ 番目の要素を最後の要素とするような $P$ の増加部分列の最大長を表すことにすることにすれば、$f(i)$ の値が等しいような $i$ について $i$ 番目の要素を取ってきて全て順に並べた列は減少列となるので、$f(i)$ が等しいような $P$ の要素の個数は最大で $B$ 個以下であり、よって ($f(i) \leq A$ より) $AB \geq N$ が必要です。

逆に、これらの条件が満たされれば、条件を満たす列を構成することができます。以下、これを示します。

$AB = N$ の場合に、ある最長増加部分列とある最長減少部分列が 1 つの要素を共有するようなものが構成

できれば、あとはそこから (その最長増加部分列または最長減少部分列に含まれる $A+B-1$ 個の要素を残すように) 要素を必要なだけ取り除き、座標圧縮を行うことで条件を満たす列を構成することができます (もちろん、計算量が大きくならないような適切な実装をする必要がありますが、それについては簡単なので割愛します)。よって、以下ではそのような構成のみ考えます。

$N$ 個の要素を $A$ 個ずつの要素を含む $B$ 個のブロックに分け、$i$ 個目のブロックの $j$ 個目の要素を $P_{ij}$ と呼ぶことにしましょう。すなわち、作る列では、要素が $P_{11}, \cdots, P_{1A}, P_{21}, \cdots, P_{2A}, \cdots, P_{B1}, \cdots, P_{BA}$ の順に並ぶようにするとしましょう。このとき、要素を小さい方から順に $P_{B1}, \cdots, P_{11}, P_{B2}, \cdots, P_{12}, \cdots, P_{BA}, \cdots, P_{1A}$ と並ぶようにすれば、条件をすべて満たすことを証明できます。実際、長さ $A$ の増加部分列と長さ $B$ の減少部分列が存在することは容易にわかり、また、

- 増加部分列について、ブロック内でその要素が何番目かを表す値 ($P$ の 2 番目の添え字) は狭義単調増加
- 減少部分列について、その要素が属するブロックの番号 ($P$ の 1 番目の添え字) は狭義単調増加

することから、その長さを超える増加部分列や減少部分列が存在しないことも分かります。よって構成ができ、この問題が解けました。

## F: Strange Nim

各山については独立したゲームとなっているため、各山についてその Grundy 数が求められれば良いです。石 $N$ 個からなり、整数 $K$ の定まった山を山 $(N, K)$ と呼ぶことにします。

このとき、以下が主張できます。

- $N$ が $K$ の倍数のとき、山 $(N, K)$ の Grundy 数は $N/K$
- そうでないとき、山 $(N, K)$ の Grundy 数は山 $(N - \lfloor N/K \rfloor - 1, K)$ の Grundy 数に等しい

これを帰納法で証明しましょう。帰納法の仮定に「山 $(N, K), (N-1, K), \cdots, (N - \lfloor N/K \rfloor, K)$ の Grundy 数はすべて異なる 0 以上 $\lfloor N/K \rfloor$ 以下の整数である」という条件を追加し、帰納法を回します。

$N = 0$ のときは明らかです。$N = t - 1$ で正しいとし、$N = t$ で主張を証明しましょう。

山 $(N, K)$ の Grundy 数は $(N-1, K), (N-2, K), \cdots, (N - \lfloor N/K \rfloor, K)$ の Grundy 数のどれとも異なる最小の値です。$N$ が $K$ の倍数のとき、帰納法の仮定よりこれらはすべて異なる 0 以上 $\lfloor N/K \rfloor - 1$ 以下の整数であり、よって 0 以上 $\lfloor N/K \rfloor - 1$ 以下の全ての整数値をとり、よって山 $(N, K)$ の Grundy 数は $N/K$ です。そうでない場合、これらはすべて異なる 0 以上 $\lfloor N/K \rfloor$ 以下の整数であり、よって 0 以上 $\lfloor N/K \rfloor$ 以下の整数のうちちょうど 1 つだけここに現れないものがあります。山 $(N-1, K)$ の Grundy 数を求めたときにはその Grundy 数への遷移が考慮されていたことから、結局山 $(N, K)$ の Grundy 数は山 $(N - \lfloor N/K \rfloor - 1, K)$ の Grundy 数に等しいことが分かります。また、追加した条件が満たされることは Grundy 数の定義から容易に導かれます。

よって帰納法が回り、上記が示されました。しかし、この通りに計算してもまだ計算量は大きいままです。以下、計算量を落とすことを考えましょう。

この計算量は、遷移をある程度「一気に行う」ことで削減可能です。$N < K^2$ のとき、上記を愚直に実装すれば $N$ の値は $N/K$ ほどしか減少しません。しかし、もし遷移先でも $\lfloor N/K \rfloor$ の値が変化しないのならば、$N$ の減少分も変化しません。よって、$\lfloor N/K \rfloor$ の値が変化しないギリギリまで $N$ の値を減らす操作をまとめ

て行うことで、遷移の回数を減らすことができます。

計算量を見積もりましょう。$N < K^2$ の領域では、遷移をまとめて行えば $\lfloor N/K \rfloor$ の値は定数回の操作で 1 減少するため、$O(\min(N/K, K))$ 時間ですべての遷移を行うことができます。そうでない領域では、$N$ の値は 1 回の操作で必ず $\frac{K-1}{K}$ 倍以下になり、よって $K$ 回の操作でおよそ $\frac{1}{e}$ 倍以下になります。よって必要な遷移の回数は $O(K \log(N/K^2))$ 回であり、この値は $K$ を動かせば $K = \sqrt{N}e^{-1}$ で最大値 $O(K)$ を取ります。$N \geq K^2$ よりこれは $O(\min(N/K, K))$ で、よって各山について $O(\min(N/K, K)) \leq O(\sqrt{N})$ 時間で Grundy 数を求めることができるので、この問題を解くことができました。

# ABC 090/ARC 091 Editorial

DEGwer

2018/03/11

## A: Diagonal String

Print $c_{11}, c_{22}, c_{33}$ in this order.

```
#include<stdio.h>
int main()
{
        char a[10][10];
        scanf("%s%s%s", a[0], a[1], a[2]);
        printf("%c%c%c\n", a[0][0], a[1][1], a[2][2]);
}
```

## B: Palindromic Numbers

For each integer between $A$ and $B$, check if it is a palindrome (possibly by using the fact that it's a 5-digit number).

```
#include<stdio.h>
int main()
{
        int a, b;
        scanf("%d%d", &a, &b);
        int r = 0;
        for (int i = a; i <= b; i++)
        {
                int s = i % 10, t = i / 10000 % 10;
                int u = i / 10 % 10, v = i / 1000 % 10;
                if (s == t&&u == v)r++;
        }
        printf("%d\n", r);
```

}

## C: Flip,Flip, and Flip.....

Without loss of generality, we can assume that $N \leq M$. For each card, let's count the number of times it is flipped: it will be faced down if this number is odd.

In case $N = 1$ and $M = 1$, the only card will be flipped once. The answer is 1.

In case $N = 1$ and $M \neq 1$, the two cards at the end will be flipped twice, and other cards will be flipped three times. The answer is $M - 2$.

In case $N \geq 2$, since $N \leq M$, we have $M \geq 2$, and in this case

- Four cards at the corners will be flipped 4 times.
- Other cards at the edge will be flipped 6 times.
- The cards that are completely surrounded by other cards will be flipped 9 times.

Thus, the answer is $(N - 2)(M - 2)$.

## D: Remainder Reminder

Let's fix $b$. For each $b = 1, 2, \cdots N$, we want to count the number of $a$ that satisfies both $1 \leq a \leq N$ and $a\%b \geq K$. For simplicity, we allow $a = 0$ and assume that $0 \leq a \leq N$ (and subtract the cases with $a = 0$ later).

For a fixed $b$, we want to count the number of terms that are greater than or equal to $K$, in the sequence $0\%b, 1\%b, \cdots, N\%b$. If $N = pb + r (0 \leq r < b)$ (we can represent $N$ in this form uniquely using two integers $p, r$), the sequence starts with $p$ repetitions of $0, 1, 2, \cdots b - 1$, followed by $0, 1, 2, \cdots r$.

In the former part, there are $p \times \max(0, b - K)$ terms that are greater than or equal to $K$. In the latter part, there are $\max(0, r - K + 1)$ terms that are greater than or equal to $K$. Thus, for a fixed $b$, we can compute the number of such $a$s in $O(1)$ time.

This solution works in $O(N)$ time in total.

## E: LISDL

When does a solution exist? First, since a LIS and a LDS can share at most one element in common, $A + B - 1 \leq N$ is necessary. Also, we can prove that $AB \geq N$ as follows. Let $f(i)$ be the length of the longest increasing sequence that ends with the $i$-th element. For a fixed $c$ ($1 \leq c \leq A$), consider a subsequence that contains the $i$-th element if and only if $f(i) = c$. This sequence must be decreasing, and the length must be at most $B$. Thus, we get $AB \geq N$.

On the other hand, when $A + B - 1 \leq N \leq AB$ holds, we can construct a sequence as follows.

First, suppose that $N = AB$. In this case, the sequence $(B-1)A+1, (B-1)A+2, \cdots, BA, (B-2)A+1, (B-2)A+2, \cdots, (B-1)A, \cdots, 1, 2, \cdots, A$ satisfies the conditions. (It consists of $B$ blocks of length $A$, each block is an increasing consecutive sequence).

It's clear that this sequence contains an increasing sequence of length $A$ and a decreasing sequence of length $B$. Also, these are the longest because

- All elements in an increasing sequence must be in the same block.
- No two elements in a decreasing sequence must be in the same block.

When $N < AB$, we first construct a sequence above, keep $A + B - 1$ elements that contain both LIS and LDS, and remove arbitrary $AB - N$ elements from others. Then, the length of LIS and LDS won't change. We compress the sequence while keeping the relative order of elements, and we get a desired sequence. (Of course, make sure not to spend $O(AB)$ to implement this.)

## F: Strange Nim

Since the piles are independent, it is sufficient to compute grundy numbers of all piles. Let $g(N, K)$ be the grundy number of a pile with $N$ stones and a parameter $K$.

If we perform an experiment for a fixed $K$, we notice that $g(N, K) = N/K$ when $N$ is divisible by $K$. Also, we notice that even if we remove the $K$-th, $2K$-th, $3K$-th, $\cdots$ elements from the sequence $g(1, K), g(2, K), \cdots$, the sequence doesn't change. Thus, we get

- If $N$ is a multiple of $K$, $g(N, K) = N/K$
- Otherwise, $g(N, K) = g(N - \lfloor N/K \rfloor - 1, K)$

Once we get the pattern it's straightforward to prove this: just make sure that from each state with grundy number $g$, it's possible to reach states with grundy numbers $0, 1, \ldots, g - 1$, but not $g$.

How can we compute grundy numbers using the fact above? We start with an integer $N$, and while $N$ is not divisible by $K$, we keep replacing $N$ with $N - \lfloor N/K \rfloor - 1$. We are interested in the final value of $N$. However, a straightforward implementation of this will get TL.

To make it faster, notice that if the value of $\lfloor N/K \rfloor$ doesn't change after an operation, we can perform multiple steps at once. More explicitly, if currently $\lfloor N/K \rfloor = d$, we keep decreasing $N$ by $d + 1$ while $N \geq dK$. Thus, instead of performing steps one by one, we can make multiple steps at once until we first get $N < dK$.

What's the time complexity after this improvement?

- Since the value of $\lfloor N/K \rfloor$ decreases in each step, it's $O(N/K)$.
- Since the value of $N$ is multiplied by a factor of at most $\frac{K-1}{K}$ in each step, in every $K$ steps this is multiplied by a factor of approximately $\frac{1}{e}$. Thus, this is $O(K \log N)$.

We should take the better of the two analysis above: it's $O(\min(N/K, K \log N) = O(\sqrt{N \log N})$.