**算法训练 回文数**

时间限制：1.0s   内存限制：256.0MB

问题描述

　　若一个数（首位不为零）从左向右读与从右向左读都一样，我们就将其称之为回文数。  
　　例如：给定一个10进制数56，将56加65（即把56从右向左读），得到121是一个回文数。  
  
　　又如：对于10进制数87：  
　　STEP1：87+78 = 165 STEP2：165+561 = 726  
　　STEP3：726+627 = 1353 STEP4：1353+3531 = 4884  
  
　　在这里的一步是指进行了一次N进制的加法，上例最少用了4步得到回文数4884。  
  
　　写一个程序，给定一个N（2<=N<=10或N=16）进制数M（其中16进制数字为0-9与A-F），求最少经过几步可以得到回文数。  
　　如果在30步以内（包含30步）不可能得到回文数，则输出“Impossible!”

输入格式

　　两行，N与M

输出格式

　　如果能在30步以内得到回文数，输出“STEP=xx”（不含引号），其中xx是步数；否则输出一行”Impossible!”（不含引号）

样例输入

9  
87

样例输出

STEP=6

锦囊1

模拟。

锦囊2

每次对于当前数均使用题设给出的方法模拟出下一个数，再判断是不是回文数。 在模拟的时候，最好使用一个数组来表示数字，使用高精度计算的方法来处理数的加和回文数的判断。

本题的C++参考代码如下：

#include<cstdio>

#include<cstring>

#include<iostream>

using namespace std;

struct bign{

int s[1000];

int n;

int len;

bign(){

memset(s,0,sizeof(s));

n=10;

len=1;

}

bign(int n,char \*m){

int l=strlen(m);

len=l;

memset(s,0,sizeof(s));

this->n=n;

for(int i=l-1,j=0;i>=0;i--){

if(m[i]>='0' && m[i]<='9')

s[j++]=m[i]-'0';

else

s[j++]=m[i]-'A'+10;

}

}

bign operator+(const bign&x)const{

bign ans;

int len=this->len>x.len?this->len:x.len+1;

int n=x.n;

for(int i=0;i<len;i++){

ans.s[i]+=x.s[i]+this->s[i];

ans.s[i+1]=ans.s[i]/n;

ans.s[i]%=n;

}

if(len>1 && ans.s[len-1]==0)len--;

ans.n=n;

ans.len=len;

return ans;

}

bign operator=(const bign&x){

this->len=x.len;

this->n=x.n;

for(int i=0;i<x.len;i++){

this->s[i]=x.s[i];

}

return \*this;

}

bign reversal()const{

bign ans;

ans.len=this->len;

ans.n=this->n;

for(int i=this->len-1,j=0;i>=0;i--){

ans.s[j++]=this->s[i];

}

return ans;

}

bool isPalindromes()const{

int i=this->len-1;

int j=0;

while(i>j){

if(this->s[i]!=this->s[j])break;

i--;

j++;

}

return i<=j;

}

};

ostream&operator<<(ostream&out,const bign&x){

for(int i=x.len-1;i>=0;i--){

if(x.s[i]<10)

out<<x.s[i];

else

out<<x.s[i]-10+'A';

}

return out;

}

int main(){

int n;

char m[1000];

scanf("%d",&n);

getchar();

gets(m);

bign a(n,m);

int i=0;

while(!a.isPalindromes()){

if(i>30)break;

i++;

bign b=a.reversal();

a=a+b;

}

if(i<=30)printf("STEP=%d\n",i);

else printf("Impossible!\n");

return 0;

}

本题的C参考代码如下：

#include<stdio.h>

#include<math.h>

#include <string.h>

#include <stdlib.h>

int p;

char s[1000];

int c[1000];

int reverse(int a[],int n) /\*判断a是否是回文\*/

{

int i,j;

i=0;

j=n-1;

while(i<j&&a[i]==a[j])

{

i++;

j--;

}

return i>=j;

}

int aplus(int a[],int n,int m) /\*计算a=a+a\*/

{

int \*b,i,j,kc;

b=(int \*)malloc(sizeof(int)\*1001);

for(i=0;i<n;i++)

b[n-1-i]=a[i];

kc=0;

for(i=0;i<n;i++)

{

a[i]=a[i]+b[i]+kc;

kc=a[i]/m;

a[i]=a[i]%m;

}

if(kc>0)

a[n++]=kc;

free(b);

return n;

}

int stod(char s[],int a[])

{

int i;

for(i=0;s[i]!='\0';i++)

if(s[i]>='A'&&s[i]<='F')

a[i]=10+s[i]-'A';

else

a[i]=s[i]-'0';

return i;

}

int main()

{

int n,i;

scanf("%d",&p);

scanf("%s",s);

n=stod(s,c);

for(i=0;i<30;i++)

{

n=aplus(c,n,p);

if(reverse(c,n)==1)

{

printf("STEP=%d\n",i+1);

break;

}

}

if(i>=30)

printf("Impossible!\n");

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Main {

private static int n, count;

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

n = Integer.parseInt(br.readLine());

String m = br.readLine();

long a = Long.parseLong(m, n);

long b = Long.parseLong(new StringBuilder(m).reverse().toString(), n);

if (a == b)

System.out.println("STEP=" + 0);

else

func(a, b);

}

private static void func(long a, long b) {

count++;

if (count > 30) {

System.out.println("Impossible!");

return;

}

long sum = a + b;

String str = "";

while (sum >= n) {

long tmp = sum % n;

sum /= n;

if (tmp >= 10)

str = (char) (55 + tmp) + str;

else

str = tmp + str;

}

if (sum >= 10)

str = (char) (55 + sum) + str;

else

str = sum + str;

String reverse = new StringBuilder(str).reverse().toString();

if (!str.equals(reverse)) {

a = Long.parseLong(str, n);

b = Long.parseLong(reverse, n);

func(a, b);

} else {

System.out.println("STEP=" + count);

return;

}

}

}