电子科技大学

University of Electronic Science and Technology of China

第二讲

简单计算题

电子科技大学 汪小平

Fibonacci Again

There are another kind of Fibonacci numbers: F(0) = 7, F(1) = 11, F(n) = F(n-1) + F(n-2) (n ≥ 2).

- **Standard Input**

Input consists of a sequence of lines, each containing an integer n. (n < 1,000,000,000).

- **Standard Output**

Print the word yes if 3 divide evenly into F(n).

Print the word no if not.

- **Samples**

Input	Output
0	no
1	no
2	yes
3	no
4	no
5	no

题目分析

- 需要先求F(n), 然后再判断能否被3整除吗?
- 如果两个数的和能被3整除, 这两个数有什么特点?

$$\text{设} a,b \in Z^+, \text{则} \begin{cases} a = 3 \times p_a + r_a, 0 \le r_a < 3 \\ b = 3 \times p_b + r_b, 0 \le r_b < 3 \end{cases}$$

$$\Rightarrow (a+b)\%3 = (3 \times p_a + 3 \times p_b + r_a + r_b)\%3 = (r_a + r_b)\%3$$

n	0	1	2	3	4	5	6	7	8	9	10	11
F(n)	7	11	18	29	47	76	123	199	322	521	843	1364
余数	1	2	0	2	2	1	0	1	1	2	0	2

- 关于能否被3整除，这两个数一共有多少种组合?

参考程序

```
#include<stdio.h>
int main()
{
    int n;
    while(scanf("%d",&n) == 1)
    {
        if (n%8==2 || n%8==6)
            printf("yes\n");
        else
            printf("no\n");
    }
    return 0;
}
```

Rightmost Digit

Given a positive integer N, you should output the most right digit of N^N.

- **Standard Input**

The input contains several test cases. The first line of the input is a single integer T which is the number of test cases. T test cases follow.Each test case contains a single positive integer N(1<=N<=1,000,000,000).

- **Standard Output**

For each test case, you should output the rightmost digit of N^N.

- **Samples**

Input	Output
2 3 4	7 6

题目分析

- 数据规模怎样? → 1<=N<=1,000,000,000
- 如果先把N^N求出来, 会怎样?
- 基本思路是什么?

	0	1	2	3	4	5	6	7	8	9
0		0	0	0	0	0	0	0	0	0
1		1	1	1	1	1	1	1	1	1
2		2	4	8	6	2	4	8	6	2
3		3	9	7	1	3	9	7	1	3
4		4	6	4	6	4	6	4	6	4
5		5	5	5	5	5	5	5	5	5
6		6	6	6	6	6	6	6	6	6
7		7	9	3	1	7	9	3	1	7
8		8	4	2	6	8	4	2	6	8
9		9	1	9	1	9	1	9	1	9

参考程序

	0	1	2	3	4	5	6	7	8	9
0		0	0	0	0	0	0	0	0	0
1		1	1	1	1	1	1	1	1	1
2		2	4	8	6	2	4	8	6	2
3		3	9	7	1	3	9	7	1	3
4		4	6	4	6	4	6	4	6	4
5		5	5	5	5	5	5	5	5	5
6		6	6	6	6	6	6	6	6	6
7		7	9	3	1	7	9	3	1	7
8		8	4	2	6	8	4	2	6	8
9		9	1	9	1	9	1	9	1	9

```
#include<stdio.h>
int main() {
    int T, i, a, digit;
    int n;
    scanf("%d", &T);
    while(T--) {
        scanf("%d", &n);
        a = n % 10;
        if(a==0 || a==1 || a==5 || a==6 || a==9) digit = a;
        else if(a == 2) {
            if(n%4==0) digit=6;
            else digit=4;
        }
        else if(a==3) {
            if(n%4==1) digit=3;
            else digit=7;
        }
        else if(a==4) digit=6;
        else if(a==7) {
            if(n%4==1) digit=7;
            else digit=3;
        }
        else { //a=8
            if(n%4==0) digit=6;
            else digit=4;
        }
        printf("%d\n",digit);
    }
    return 0;
}
```

偏僻的小路

在电子科大清水河校区的某个偏僻角落里，有一条东西方向的小路，长L米（由西向东位置为0到L），小路上有N个人从t=0秒开始以相同的恒定速率V米/秒前进（面朝西或面朝东）。这条小路太偏僻了，所有人都想尽快离开这条小路。不幸的是，当两个人相遇时，只有男生会给女生让路（视为两人擦肩而过），男生遇上男生、女生遇上女生时，谁也不肯让路，只好都无奈的掉头往回走。

现在HS很好奇，想知道最后一个人离开小路的时间，以及所有人在小路上走的路程的总和，你能编写程序帮助他吗?

Standard Input

第一行包括3个整数，N,L,V,表示小路上的人数、小路的长度、所有人前进的速率 (N<=100, L<= 1000000,V>0)

接下来有N行，每行3个数据，第i行的数据表示第i个人的位置（从0到L的整数）、性别（M或F）、方向（W表示面朝西、E表示面朝东）

当N=L=V=0时，输入结束 。

Standard Output

对于每组输入，输出一行两个小数，表示最后一个人离开的时间以及所有人在小路上走的路程的总和，用一个空格隔开，答案四舍五入保留两位小数 。

Samples

Input	Output
2 4 2 1 M E 3 M W 0 0 0	1.50 6.00

题目分析

参考程序

```
#include <stdio.h>
int N, L, V, pos;
char sex[4], dir[4];
int main() {
  double last, dist;//最后时间与路程
  double len;//当前人的路程
  while (scanf("%d %d %d", &N, &L, &V) == 3) {
        if (N == 0 && L == 0 && V == 0) break;
        last = 0.0, dist = 0.0;//清零
        for (int i = 0; i < N; i++) {
              scanf("%d %s %s", &pos, sex, dir);
              if (dir[0] == 'W')  len = pos;
              else   len = L - pos;
              if (len / V > last) last = len / V;
              dist += len;
        }
        printf("%.2f %.2f\n", last, dist);
  }
  return 0;
}
```

约会

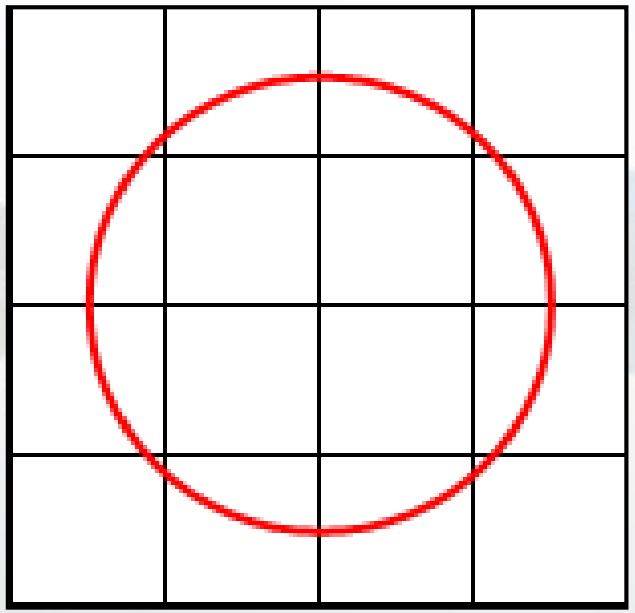
有一天silentsky和lcy同学去教室上自习。silentsky百无聊赖地看着书本，觉得很无聊，看着右手边的lcy认真仔细的在画着她繁重的物理实验报告的图。silentsky无聊地弄着他的脉动瓶子，结果一不小心就把瓶盖弄到了lcy刚画好的坐标纸上，而且冥冥之中仿佛有一双手在安排，瓶盖的中心正好和坐标纸的中心重合了，瓶盖的边缘有水，会弄湿坐标纸的。 lcy很生气，后果很严重。

于是，lcy由此情形想出了一道难题问silentsky,如果他回答正确了。lcy就原谅了silentsky并且答应他星期天去看暮光之城2的请求，不然一切都免谈。然后silentsky就回去面壁思过了，现在silentsky好无助的，希望得到广大编程爱好者的好心帮助。

问题是这样的: lcy现在手上有一张2n * 2n的坐标纸，而silentsky的圆形瓶盖的直径正好有2*n-1大，现在lcy想知道silentsky到底弄湿了多少个坐标纸的格子（坐标纸是由1 * 1的小格子组成的表格）

如果还是有人觉得理解不了焦急的silentsky的意思。干脆silentsky做下翻译，毕竟silentsky还是多了解lcy的O(∩_∩)O~。

问题就是给你一个2n * 2n的正方形格子，分成1 * 1的格子，然后以中心为原点画一个直径为2n - 1的圆，问圆的周线穿过了多少个格子。



- **Standard Input**

含有多组测试数据，每组数据都包含一个正整数n（n <= 1000）。当n = 0的时候结束程序，证明silentky经受住考验了的O(∩_∩)O~

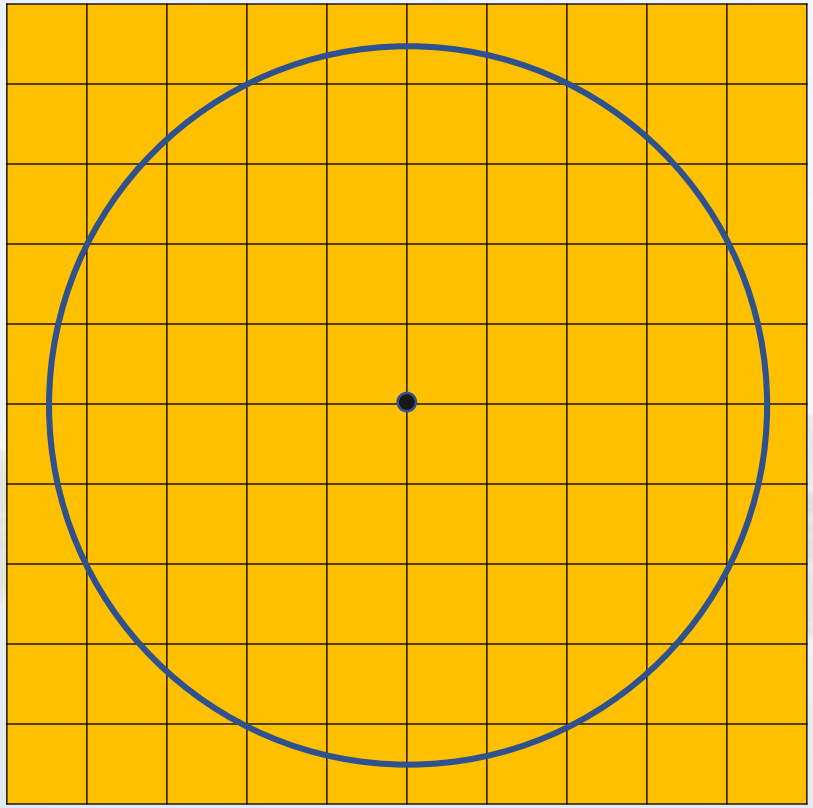
- **Standard Output**

对于每个n，输出被瓶盖边缘的水弄湿了的格子数为多少。

- **Samples**

Input	Output
1 2 0	4 12

题目分析



- 正方形: 2n*2n
- 圆直径: 2n - 1

参考程序

```
#include<stdio.h>
int main()
{
    int n;
    while(scanf("%d", &n) == 1)
    {
        if(n == 0)
            break;
        printf("%d\n", n * 8 - 4);
    }
    return 0;
}
```

木杆上的蚂蚁

在一根细木杆上，有一些速度相同蚂蚁，它们有的往左走，有的往右走，木杆很细，只允许一只蚂蚁通过，所以当两只蚂蚁碰头的时候，它们会掉头继续前进，直到走出边界，掉下木杆。

已知木杆的长度和每只蚂蚁的名字、位置和初始方向，问依次掉下木杆的蚂蚁花费的时间以及它的名字。

- **Standard Input**

输入包含多组测试数据。

第一行包含一个整数T(T <= 20)，代表测试数据组数。

每组测试数据的第一行包含两个整数N L，表示有N只蚂蚁(N <= 100)，木杆长度为L(L <= 1000)。假设蚂蚁每秒前进一个单位距离，掉头转向的时间忽略不计。

以下N行，每行依次为蚂蚁的名字（长度不超过10，仅由英文字母组成），初始位置p（0 < p < L，整数，表示蚂蚁离木杆最左端的距离），初始方向（一个字符，L表示向左，R表示向右），以单个空格分隔，数据保证初始不会有两只蚂蚁在同一个位置。

- **Standard Output**

对于第k组测试数据，首先输出一行为“Case #k:”。

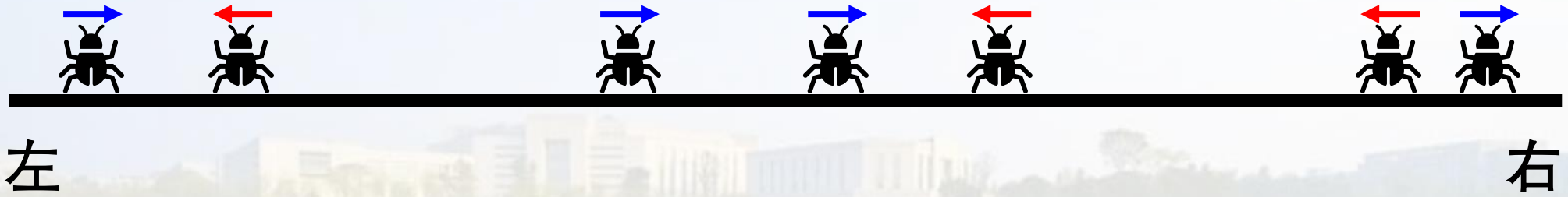
然后输出N行，给出依次掉下木杆的蚂蚁花费的时间以及它的名字，以单个空格分隔。

按照掉下木杆的先后顺序输出，数据保证不会有两支蚂蚁同时掉下木杆。

- **Samples**

Input	Output
2 2 5 GG 1 L MM 3 R 2 5 GG 1 R MM 2 L	Case #1: 1 GG 2 MM Case #2: 2 GG 4 MM

题目分析



参考程序

```
#include <bits/stdc++.h>
#define N 100
using namespace std; //引用名字空间std
struct ant_type{
    int pos;
    char name[11];
} ants[N];
struct event_type{
    int drop_time;
    char dir;
} events[N];
bool cmp_ant(const ant_type& p, const ant_type& q){
    return p.pos < q.pos;
}
bool cmp_event(const event_type& p, const event_type& q){
    return p.drop_time < q.drop_time;
}
```

```
int main(){
    char dir[2];
    int i, k, n, L, R, T;
    scanf("%d", &T);
    for (k = 1; k <= T; k++){
        scanf("%d%d", &n, &L);
        for (i = 0; i < n; i++){
            scanf("%s%d%s", ants[i].name, &ants[i].pos, dir);
            events[i].dir = dir[0];
            events[i].drop_time = (dir[0] == 'L’ ?
                                   ants[i].pos : L - ants[i].pos);
        }
        sort(ants, ants + n, cmp_ant);
        sort(events, events + n, cmp_event);
        printf("Case #%d:\n", k);
```

```
        L = 0;
        R = n - 1;
        for (i = 0; i < n; i++){
            if (events[i].dir == 'L'){
                printf("%d %s\n", events[i].drop_time,
                                                ants[L].name);
                L++;
            }
            else{
                printf("%d %s\n", events[i].drop_time,
                                                ants[R].name);
                R--;
            }
        }
    }
    return 0;
}
```

Flagstone

In the Qingshuihe Campus of UESTC, the most annoy problem to students are the flagstone path on the lawn. The designer seems so stupid that the flagstone path often make students step in the gap. Now a perfect step is wanted in order to not step in any gaps and step on every flagstone. The step length is required to be constant while the length of the flagstone and gap are given different.

The problem is asking you to tell the minimum length of the perfect step. To simplify the question, the foot is considered to be a point and the very beginning is the fore edge of the first flagstone, which also means the first flagstone has already been stepped on.

Standard Input

The first line of the input contains one integer T, which indicate the number of test cases. In each test case, the first line contains an integer N(2<=N<=1e5), indicating the number of flagstone. Following N lines, and each line is the length of one flagstone. And the following N-1 lines are the length of the gaps. All data is integer. All the length will be a positive integer, and the sum of them will fit in a 32bit signed integer.

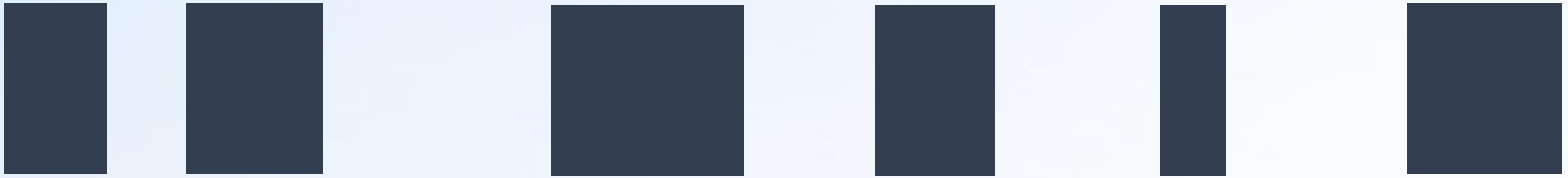
Standard Output

One line for each test case contains only one number indicating the answer. One real number indicating the perfect step length should be accurate to two digits after the radix point. If it is impossible to find out a perfect step, just output “impossible” !

Samples

Input	Output
2 2 10 20 5 3 10 20 5 5 1000	15.00 impossible

题目分析



- 每个石板踏且仅踏一次;
- 对于第 i 块石板, 一定是踏了 i - 1 步到达的;
- 设第 i 块石板的左边界和右边界离起点的距离分别为L和R，可以确定步长必须在区间[L / (i - 1), R / (i - 1)]之内。
- 问题转化为求多个区间的交。如果交集为空，则答案为impossible，否则输出交区间的左界。

参考程序

```
#include<stdio.h>
#define N 100001
int a[N], b[N];
int main()
{
    int i, n, t, p;
    double h, l, s, hh, ll;
    scanf("%d", &t);
    while(t--) {
        scanf("%d", &n);
        for(i = 1; i <= n; i++)
            scanf("%d", &a[i]);
        for(i = 1; i < n; i++)
            scanf("%d", &b[i]);
        s = l = a[1] + b[1];
        h = a[1] + b[1] + a[2];
        for(i = 2; i < n; i++)
        {
            s += a[i] + b[i];
            ll = s / (double)i;
            hh = (s+a[i+1]) / (double)i;
            if(ll > l) l = ll;
            if(hh < h) h = hh;
        }
        if(l <= h) printf("%.2f\n", l);
        else printf("impossible\n");
    }
    return 0;
}
```

输出前m大的数据

给你n个整数，请按从大到小的顺序输出其中前m大的数。

- **Standard Input**

每组测试数据有两行，第一行有两个数n,m(0<n,m<1000000)，第二行包含n个各不相同，且都处于区间[-500000,500000]的整数。

- **Standard Output**

每组测试数据有两行，第一行有两个数n,m(0<n,m<1000000)，第二行包含n个各不相同，且都处于区间[-500000,500000]的整数。

- **Samples**

Input	Output
5 3 3 -35 92 213 -644	213 92 3

题目分析

- 常规的思想是什么? ➔ 排序后遍历
- 常规方法的结果是什么? ➔ TLE
- 数据的特点是什么? ➔ 各不相同, 位于[-500000, 500000]
- 加速的方法是什么? ➔ Hash
- 如果数据可以重复呢? ➔ 处理冲突

参考程序

```
#include <stdio.h>
#include <string.h>
#define N 1000000
int a[N];
int main()
{
    int i,j,n,m,t;
    while(scanf("%d%d",&n,&m)==2)
    {
        memset(a,0,N*sizeof(int));
        for(i=0;i<n;i++)
        {
            scanf("%d",&t);
            a[t+N/2]++; //存储重复次数
        }
    }
        for(t=0,i=N-1;t<m && i>=0;)
        {
            if(a[i]>0)//对应有数
            {
                printf("%d ",i-N/2);//打印数
                t++;  //计数器增加
                if(--a[i]==0)   i--;
            }
            else   i--;
        }
        printf("\n");
    }
    return 0;
}
```