电子科技大学

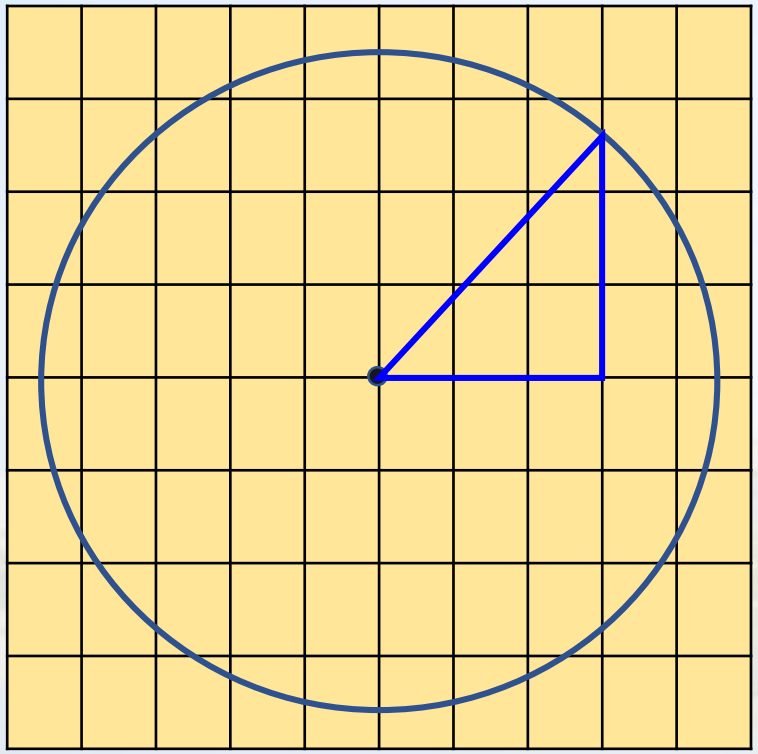
University of Electronic Science and Technology of China

第三讲

简单计算题及模拟

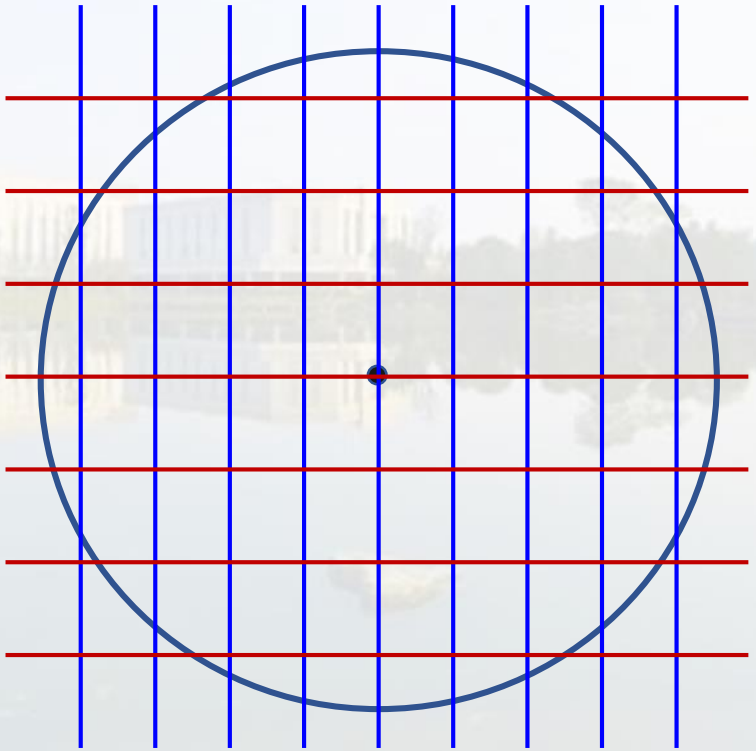
电子科技大学 汪小平

题目分析



- 正方形: 2n*2n
- 圆直径: 2n - 1

- 由于圆直径为奇数, 由左边直角三角形知, 圆不会经过网格交点.
- 圆上的一段对应一个格子. 统计出圆被正方形网格分成的段数, 就可得到圆穿过了多少方格.



- 2*(2n-1)条直线与圆的交点共有8n-4个, 圆被分为的段数为8n-4
- 穿过的方格数目为: 8n-4

Ants Run!

Professor Yang likes to play with ants when he is free. What? Are you asking why he plays with ants instead of others? Ah, because ant is the only non-plant living thing which can be found in Qingshuihe Campus of UESTC apart from human beings.

This time, Professor Yang caught several ants after finishing his lecture for freshmen. At the beginning of the game, he puts N ants around a plate and numbers them in clockwise order. The ants are so obedient that they run clockwise under the guide of Professor Yang on the boundary of the plate which is a circle. When one ant catches up with its previous ant, the game is over. Knowing the speed of ants, Professor Yang wants you to help him to adjust the distance between adjacent ants to make the game last longer.

- **Standard Input**

The first line of the input is T (no more than 10000), which stands for the number of test cases you need to solve.

Each test case begins with “N R”(without quotes) representing the number of ants participating the game is N and the radius of the circle is R cm. The next line lists N integers and the i-th number is the speed (cm/s) of the i-th ant in clockwise direction. All numbers are positive integer not larger than 20.

Standard Output

If the game can last forever, print “Inf” in a single line, otherwise please output the longest time in seconds each game can last, which should be printed accurately rounded to three decimals.

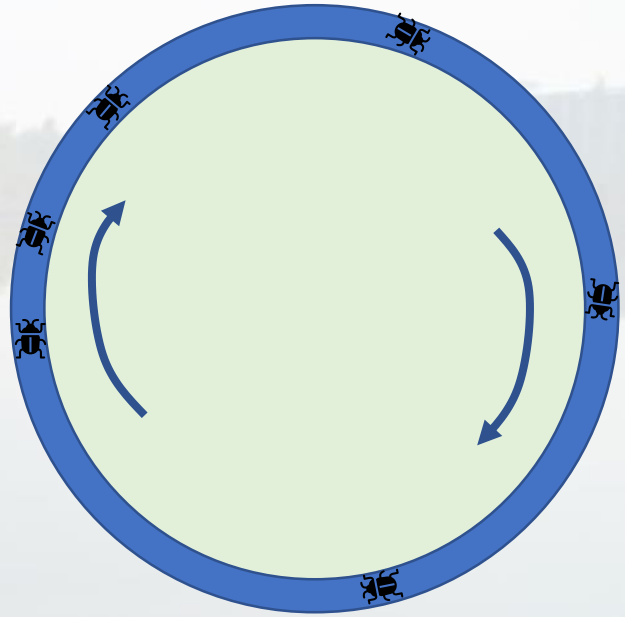
Samples

Input	Output
2 3 1 1 1 1 3 1 3 2 1	Inf 3.142

题目分析

设有n只蚂蚁,速度分别为$v_1,v_2,\cdots,v_n$.考虑$v_1>v_2>\cdots>v_n$.开始时,记第i只蚂蚁距第$i+1$只蚂蚁的距离为$l_i(1\le i\le n-1)$.假设时间为t时,某只蚂蚁追上前面的蚂蚁,这时第i只蚂蚁相对前一只蚂蚁走过的路程为$(v_i-v_{i+1})t$.由于能调整距离,要使游戏能延续更久,则有:

$$\sum_{n=1}^{n-1}(v_i-v_{i+1})t=2\pi r,\ 即t=\frac{2\pi r}{\sum_{n=1}^{n-1}(v_i-v_{i+1})}.$$

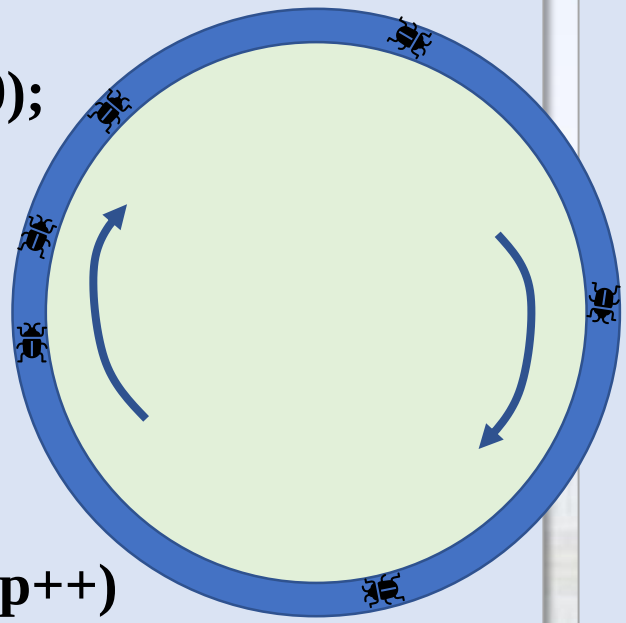


在实际中,当一个蚂蚁不能追上前一只时,初始可以调整两者的距离为0,在上面和式中不计入即可。

- 所有蚂蚁都是相同速度则inf.

参考程序

```
#include <stdio.h>
#include <math.h>
double pi = acos(-1.0);
int v[30];
int main()
{
    int t, p, n, i;
    double r;
    scanf("%d", &t);
    for (p = 1; p <= t; p++)
    {
        scanf("%d", &n);
        scanf("%lf", &r);
        for(i = 1; i <= n; i++)
            scanf("%d", &v[i]);
        v[0] = v[n];
        int s = 0;
        for(i = 1; i <= n; i++)
            if (v[i-1] > v[i])
                s = s + v[i-1]-v[i];
        if (s == 0)
            printf("Inf\n");
        else
            printf("%.3f\n", 2*pi*r / s);
    }
    return 0;
}
```



保护果实

A有一棵果树，他想买一些栅栏把果树围起来。卖栅栏的人有N块栅栏，每块栅栏长度为a_i。但是，买栅栏的人规定如果要买第i块栅栏，那么必须要先买第i-1块栅栏（第一块除外）。同时A是一个不想浪费的人，他想把他买的所有栅栏都用上，并且，让栅栏围成的图形是个多边形。那么，A最少需要买多少块栅栏呢?

- **Standard Input**

一共有两行。

第一行一个数，表示总共的栅栏数N(N≤1,000,000)。

第二行有N个数，第i个数表示第i块栅栏的长度 (ai之和不会超过int的上界)。

- **Standard Output**

输出一个数，表示最少需要的栅栏数，如果无解输出-1.

- **Samples**

Input	Output
3 3 4 5	3

题目分析

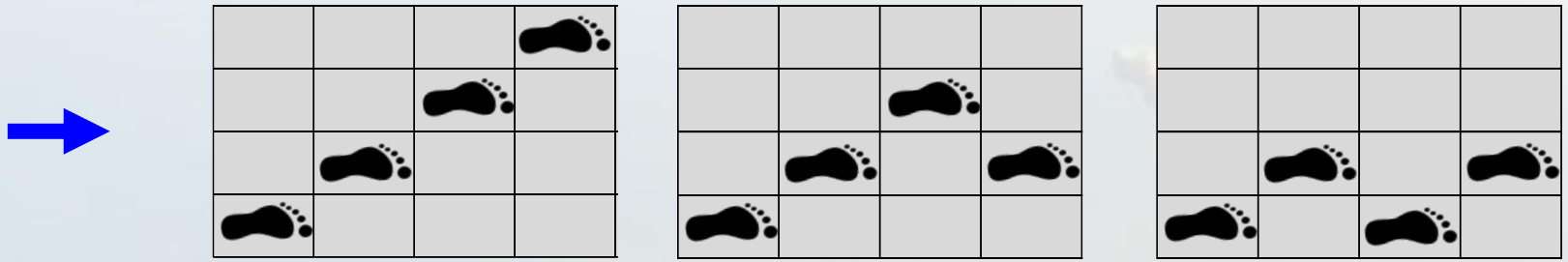
参考程序

```
#include <stdio.h>
int main()
{
    int N, sum = 0, maxLen = 0, t;
    scanf("%d", &N);
    if (N < 3)
    {
        printf("-1\n");
        return 0;
    }
    for (int i = 0; i < N; ++i)
    {
        scanf("%d", &t);
        if (t > maxLen)
        {
            sum += maxLen;
            maxLen = t;
        }
        else
            sum += t;
        if (sum > maxLen && i > 1)
        {
            printf("%d\n", i + 1);
            return 0;
        }
    }
    printf("-1\n");
    return 0;
}
```

Flagstone Walk

There is a long flagstone walk on the way from the dormitory to the main hall. This flagstone walk has N lines and four flagstones arranged in each line. Hongshu always start from rightmost flagstone of the first line. In order to make this walk funny, he would always step onto the left or right flagstone in the next line if there is. For example, if Hongshu stands at the second rightmost flagstone of the third line, he would choose to step to the first or third rightmost one of the forth line. Note that Hongshu has only one choice when he is at the corner of one line.

Because Hongshu has to go to the main hall every day, he wants to know how many different ways to step across the flagstone walk. Can you help him?



- **Standard Input**

The first line of the input is an integer T (T <= 20), which stands for the number of test cases you need to solve. Each case consists of an integer N (1 <= N <= 20) on a single line, which stands for the length of the walk.

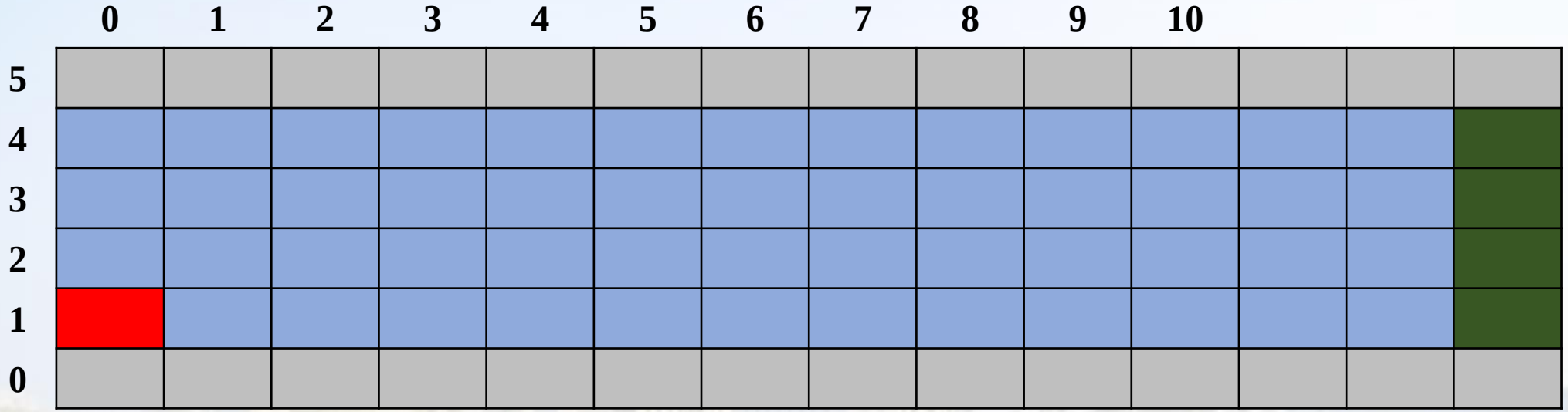
- **Standard Output**

For each case, print the number of ways on a single line.

- **Samples**

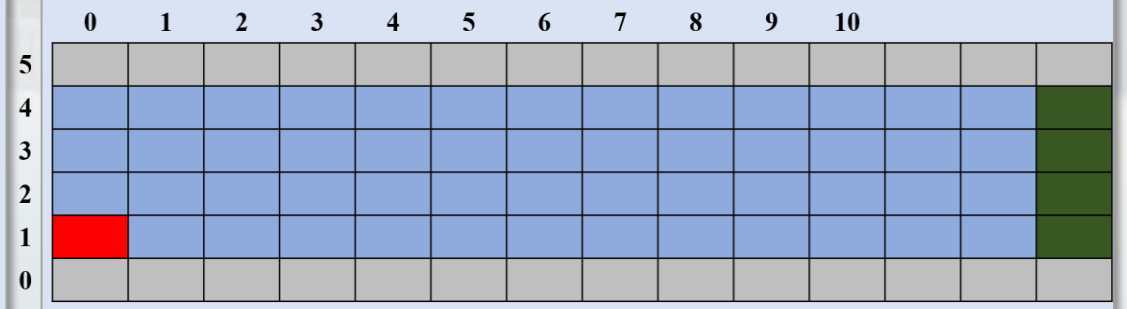
Input	Output
3 2 3 4	1 2 3

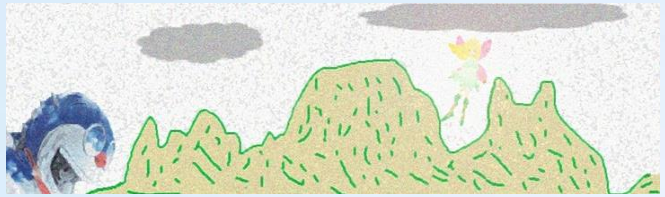
题目分析



参考程序

```
#include <stdio.h>
#include <string.h>
int main()
{
    int i, j, m, n, tot, a[6][21];
    scanf("%d", &n);
    while(n--)
    {
        scanf("%d", &m);
        memset(a, 0, sizeof(a));
        a[1][1] = 1; tot = 0;
        for(i = 2; i <= m; i++)
            for(j = 1; j <= 4; j++)
                a[j][i] = a[j-1][i-1] + a[j+1][i-1];
        for(i = 1; i <= 4; i++)
            tot += a[i][m];
        printf("%d\n", tot);
    }
    return 0;
}
```





Fly Through

The home of flower fairies is being devastated by a monster called Littlefatcat. They have to leave the place where their generations lived with no other choices. CC, the greatest investigator of flower fairies, found a paradise in the west and will lead all the flower fairies there. This paradise is full of flowers and safe from attack of Littlefatcat. However, there are lots of huge rocks on the way to the paradise.

Flower fairies are of different levels which are determined by their power. Fairies of higher level will fly higher. A fairy will persist in flying at the height corresponding to his or her level for honor and self-respect. Also, rocks on the way have specific height. When the route of a fairy hit a rock, the fairy will have to use magic to fly through the rock.

Being aware of the height of all rocks and the specific height each fairy can fly at, CC want to know how many rocks each fairy will fly through.

Standard Input

The first line will be “N M”(without quotes), representing for the numbers of rocks and fairies. The following line will give N numbers, giving the height of the rocks from the east to west. Then M lines followed. On the i-th line, a number representing the specific height of fairy numbered i can fly at will be given. All numbers are positive and not bigger than 100000.

Standard Output

Please output the number of rocks each fairy has to fly through in order to get to the paradise in a single line.

Samples

Input	Output
5 3 1 2 3 4 5 2 4 6	4 2 0

题目分析

- 本题相当于问在一组数中有多少个是大于等于h的。
- 看看数据范围: All numbers are positive and not bigger than 100000. 时间限制: 1000ms.
- 方法：先排序，再找到插入位置
- 简单的排序和查找方法可能会导致Time Limit Exceed，时间复杂度应该控制在O(nlogn), 查找用二分法比较合适。
- 可以考虑用STL。
- 还有其它方法吗?

参考程序

```
#include <bits/stdc++.h>
using namespace std;
int data[100005];
int main() {
    int n, m;
    while (scanf("%d%d", &n, &m) == 2)  {
        for (int i = 0; i < n; ++i) scanf("%d", data+i);
        sort(data, data+n); //STL 快排
        while (m--)  {
            int v;scanf("%d", &v);
            int where = lower_bound(data, data+n, v) - data;
            printf(“%d\n”, n - where);
        }
    }
    return 0;
}
```

The lower_bound() algorithm compares a supplied value to elements in a sorted sequence and returns the first position in the container at which value can be inserted without violating the container's ordering.

参考程序

```
#include <stdio.h>
#include <string.h>
int a, b[100001];
int main()
{
    int n, m, i, k;
    while(scanf(“%d%d”,&n,&m) == 2)
    {
        memset(b, 0, 10001*sizeof(int));
        for (i = 1; i <= n; i++)
        {
            scanf(“%d”, &a]);
            b[a]]++; //把数作为下标,计数
        }
        for(i = 100000; i >= 1; i--)
            b[i] += b[i+1];//累加得>=b[i]个数
        for (i = 1; i <= m; i++)
        {
            scanf(“%d”, &k);
            printf("%d\n", b[k]);
        }
    }
    return 0;
}
```

红帽自动机

红人无数的红帽侠决定金盆洗手啦！

由于红帽侠后继无人，按照祖祖辈辈的祖训，红帽侠要把位置传给这个王国里红帽最少的那个人。

但是，红帽侠曾记得那个被红的晚上，而那个人，还在王国里潇潇洒洒。

“当然是选择 不 原谅他啊！”

于是红帽侠搬出了祖辈流传的神器：红帽自动机——对着一个人喊一声，“你将加冕为王”，除了这个人以外的所有人，都被戴上一顶红帽。

红帽侠决定金盆洗手前，再干一票！

不为别的，就为了让当初曾经红了他的人成为这个王国内唯一的红帽最多的人！

“屏幕前的你，如果不想被我戴上红帽的话，就帮我算算，我最少需要喊多少次吧。这条咸鱼还没熟，我要再烤烤。”

- **Standard Input**

第一行两个整数n,x (2≤n≤100000,1≤x≤n)，表示这个王国有n个人，红帽侠希望第x个人红帽最多。

第二行包括n个整数，用空格隔开，第i个整数g_i表示第i个人头上有g_i顶红帽。(0≤ g_i ≤10000)

- **Standard Output**

输出一个整数a，表示红帽侠最少需要喊a次“你将加冕为王”。

- **Samples**

Input	Output
5 3 1 1 3 4 4	4
4 2 0 3 2 1	0

题目分析

- 红帽侠的目标让第x个人帽子最多. 假设他是对着第k个人喊的, 那么结果是给除了第k人以外的所有人加戴一顶帽子. 这个操作可以分解成两步:
  - ➔ 给所有人戴一顶帽子
  - ➔ 给第k人摘一顶帽子
- 第一个步骤不会影响帽子的相对数量关系, 因此可以当作只发生了第二步. 因此若是要让x帽子最多, 只需要对着所有帽子数大于等于x的人不停的喊, 起到他们的帽子比x少就可以了.

参考程序

```
#include <stdio.h>
int main()
{
    int n, x, num[100010], ans = 0;
    scanf("%d%d", &n, &x);
    for(int i = 1; i <= n; i++)
        scanf("%d", &num[i]);
    for(int i = 1; i <= n; i++)
    {
        if (i == x) continue;
        if (num[i] >= num[x])
            ans += num[i] - num[x] + 1;
    }
    printf("%d\n", ans);
    return 0;
}
```

模拟问题

模拟的定义是将原本真实的系统、流程或事物，用一定的方法进行接近或是完全的重现。

而在程序竞赛里，模拟是解决问题的方法之一，它指按照题目的要求，一步步用代码实现，最终得出题目的答案。

模拟的代码相对来说会偏长，而且一旦出现错误比较难检查，所以更要求我们在第一次编码一定要仔细。

摘花生

鲁宾逊先生有一只宠物猴，名叫多多。这天，他们两个正沿着乡间小路散步，突然发现路边的告示牌上贴着一张小小的纸条：“欢迎免费品尝我种的花生！——熊字”。

鲁宾逊先生和多多都很开心，因为花生正是他们的最爱。在告示牌背后，路边真的有一块花生田，花生植株整齐地排列成矩形网格（如右图）。

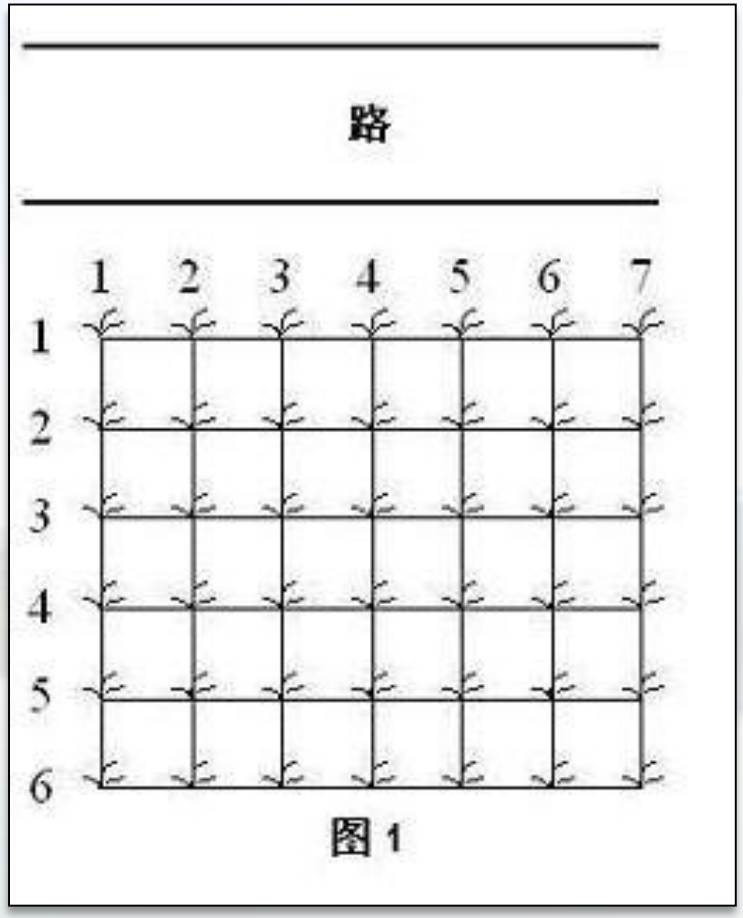


图 1

有经验的多多一眼就能看出，每棵花生植株下的花生有多少。为了训练多多的算术，鲁宾逊先生说：“你先找出花生最多的植株，去采摘它的花生；然后再找出剩下的植株里花生最多的，去采摘它的花生；依此类推，不过你一定要在我限定的时间内回到路边。”

我们假定多多在每个单位时间内，可以做下列四件事情中的一件：

(1) 从路边跳到最靠近路边（即第一行）的某棵花生植株；

(2) 从一棵植株跳到前后左右与之相邻的另一棵植株；

(3) 采摘一棵植株下的花生；

(4) 从最靠近路边（即第一行）的某棵花生植株跳回路边。

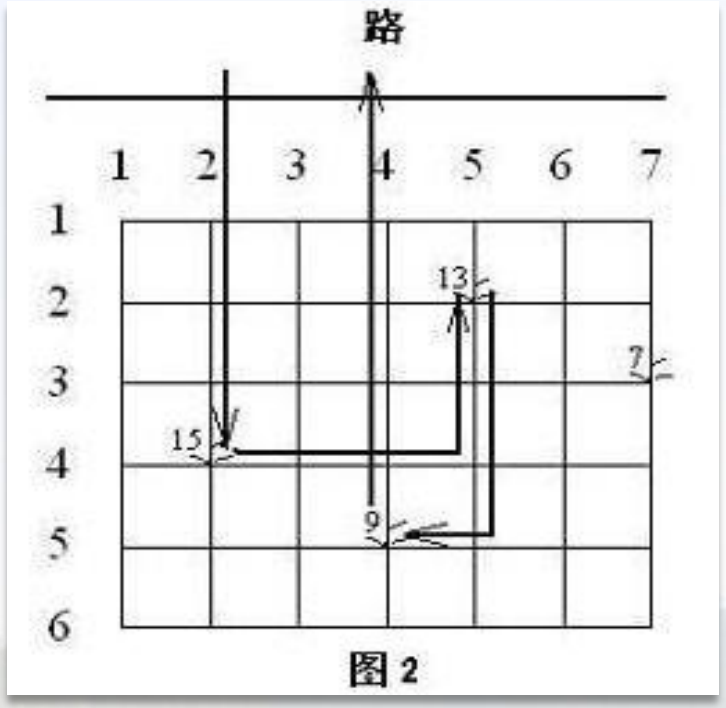


图 2

现在给定一块花生田的大小和花生的分布，请问在限定时间内，多多最多可以采到多少个花生？注意可能只有部分植株下面长有花生，假设这些植株下的花生个数各不相同。

例如在图2所示的花生田里，只有位于(2, 5), (3, 7), (4, 2), (5, 4)的植株下长有花生，个数分别为13, 7, 15, 9。沿着图示的路线，多多在21个单位时间内，最多可以采到37个花生。

Standard Input

输入的第一行包括一个整数T，表示数据组数。

每组输入的第一行包括三个整数，M, N和K，用空格隔开；表示花生田的大小为M * N（1 <= M, N <= 50），多多采花生的限定时间为K（0 <= K <= 1000）个单位时间。接下来的M行，每行包括N个非负整数，也用空格隔开；第i + 1行的第j个整数P_{ij}（0 <= P_{ij} <= 500）表示花生田里植株(i, j)下花生的数目，0表示该植株下没有花生。

Standard Output

输出包括T行，每一行只包含一个整数，即在限定时间内，多多最多可以采到花生的个数。

Samples

Input	Output
1 6 7 21 0 0 0 0 0 0 0 0 0 0 0 13 0 0 0 0 0 0 0 0 7 0 15 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0	37

题目分析

- 找规律得到一个以花生矩阵作为自变量的公式来解决这个问题，是不现实的。
- 直接模拟过程。每次要采下一株花生之前，先计算一下剩下的时间够不够走到那株花生、采摘，并从那株花生走回到路上。如果时间够，则走过去采摘；如果时间不够，则采摘活动到此结束。
- 设二维数组aField存放花生地的信息。然而，用aField[0][0]还是aField[1][1]对应花生地的左上角是值得思考一下的。因为从地里到路上还需要1个单位时间，题目中的坐标又都是从1开始。所以若aField[1][1]对应花生地的左上角，则从aField[i][j]点回到路上所需时间就是i，这样更为方便。

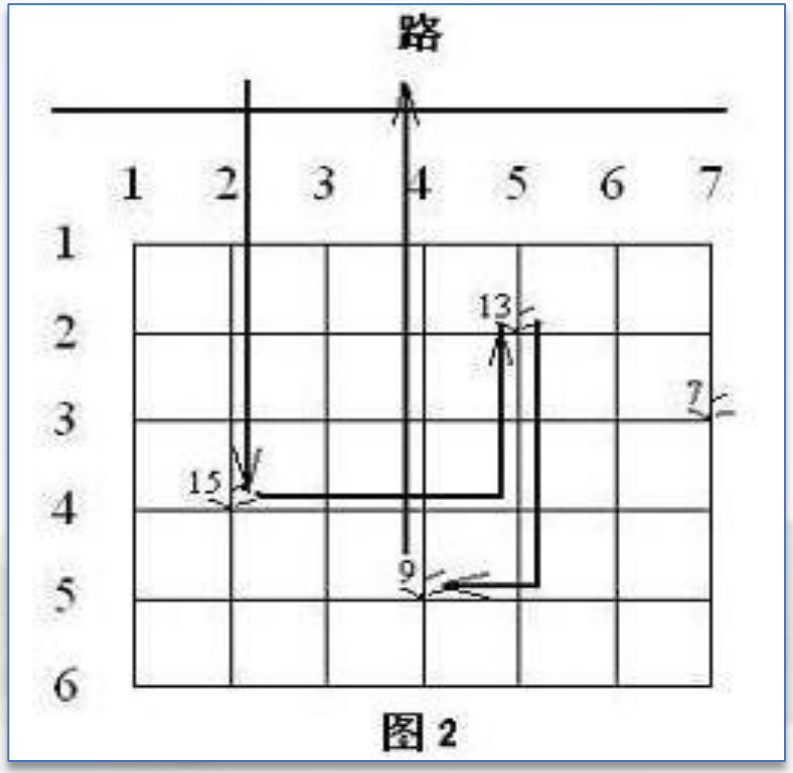


图 2

参考程序

```
#include <stdio.h>
#include<stdlib.h>
#define MAX_NUM 55
int T, M, N, K;
int aField[MAX_NUM][MAX_NUM];
int main() {
   scanf("%d", &T);
   for(int t=0; t<T; t++) {
      scanf("%d%d％d", &M, &N, &K);
      for(int m=1; m<=M; m++)
         for(int n=1; n<=N; n++)
            scanf("%d",&afield[m][n]);
      int nTotalPeanuts=0;  //摘到的花生总数
      int nTotalTime=0;     //已经花去的时间
      int nCuri=0,nCurj;    //当前位置坐标
      while(nTotalTime<K) {
         int nMax=0, nMaxi, nMaxj;
```

```
        for(int i=1; i<=M; i++)
           for(int j=1; j<=N; j++)
              if(nMax<aFieId[i][j]) {
                 nMax=aFieId[i][j];
                 nMaxi=i;   nMaxj=j;
              }
        if(nMax == 0) break;  //地里已经没有花生了
        if(nCuri == 0) nCurj=nMaxj; //如果当前位置是在路上
        if(nTotalTime+nMaxi+1+abs(nMaxi-nCuri)+abs(nMaxj-nCurj) <=K) {
           nTotalTime+=1+abs(nMaxi-nCuri)+abs(nMaxj-nCurj);
           nCuri=nMaxi;  nCurj=nMaxj;
           nTotaiPeanuts+=aField[nMaxi][nMaxj];   aField[nMaxi][nMaxj]=0;    //摘走花生
        }
        else    break;
     }
     printf("%d\n", nTotalPeanuts);
  }
  return 0;
}
```

显示器

你的一个朋友买了一台电脑。他以前只用过计算器，因为电脑的显示器上显示的数字的样子和计算器是不一样，所以当他使用电脑的时候会比较郁闷。为了帮助他，你决定写一个程序把在电脑上的数字显示得像计算器上一样。

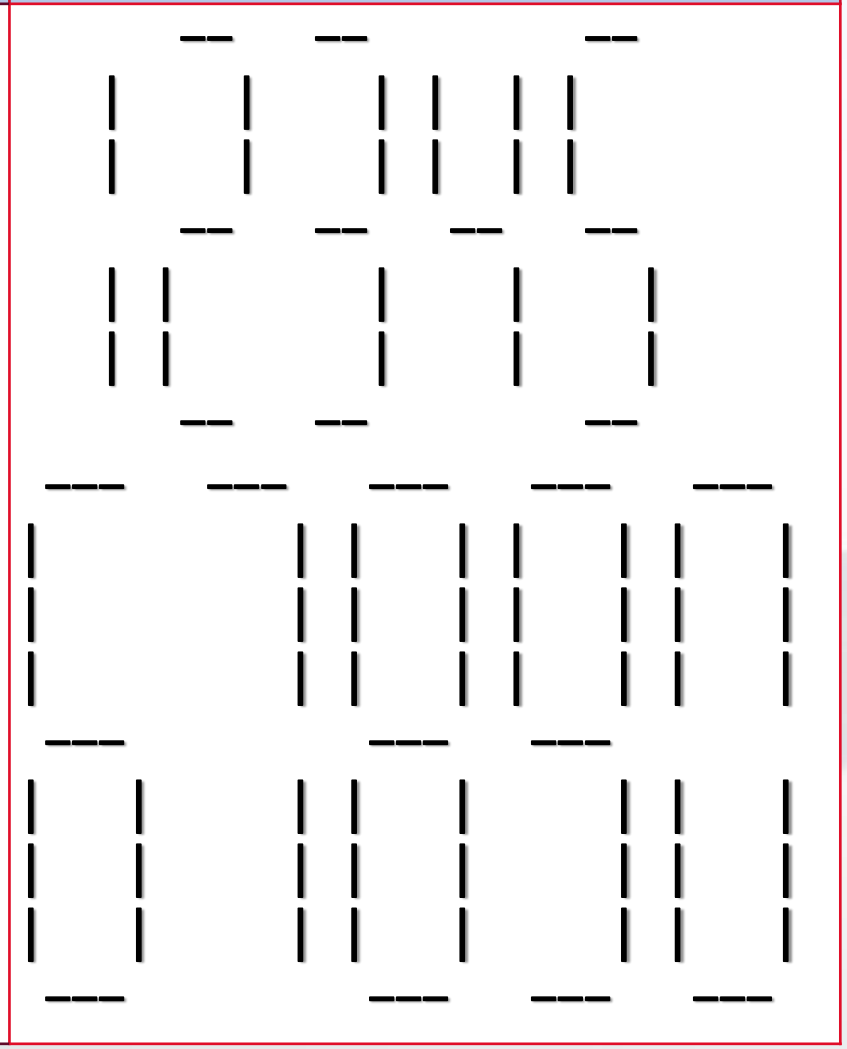
- **Standard Input**

输入包括若干行，每行表示一个要显示的数。每行有两个整数s和n (1 <= s <= 10, 0 <=n<= 99999999)，这里n 是要显示的数，s是要显示的数的尺寸。如果某行输入包括两个0，表示输入结束。这行不需要处理。

- **Standard Output**

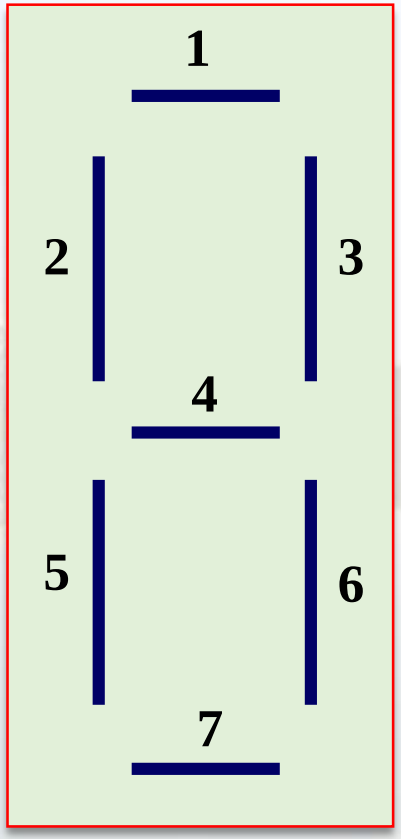
显示的方式是：用s 个'-'表示一个水平线段，用s 个'|'表示一个垂直线段。这种情况下，每一个数字需要占用s+2 列和2s+3 行。另外，在两个数字之间要输出一个空白的列。在输出完每一个数之后，输出一个空白的行。注意：输出中空白的地方都要用空格来填充。

- **Samples**

Input	Output
2 12345 3 67890 0 0	

题目分析

- 一个计算器上的数字显示单元，可以看作由以下编号从1 到7 的7 个笔画组成：



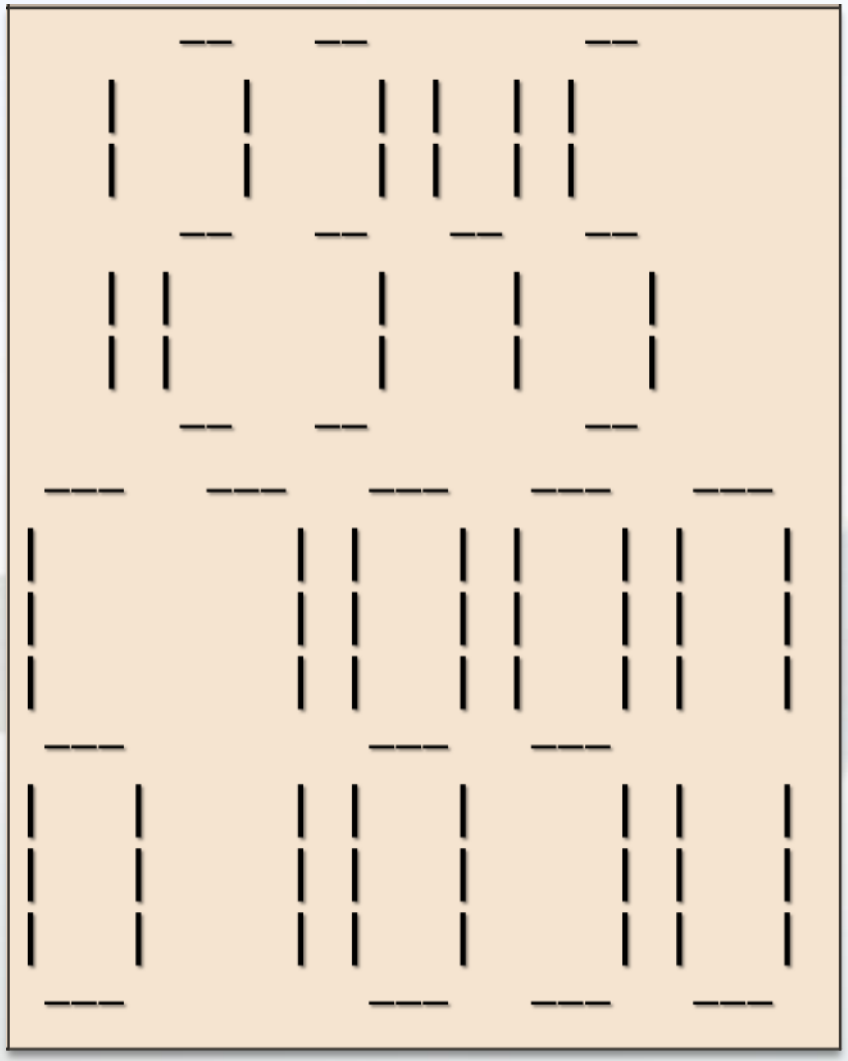
- 用一个字符数组记录所有数字在某笔划的情况，例如对第1笔划，定义字符数组：

  char n1[11] = {"- -- -----"};

  其中，n1[i](i = 0……9) 代表笔画1 是否被数字i 覆盖。

- 又如对于竖向的笔画2，由字符'|' 组成，则记录其覆盖情况的数组如下：

  char n2[11] = {"| ||| ||"};



参考程序

```
#include <stdio.h>
#include <string.h>
char n1[11]={"- -- -----"};//笔画1 被数字0,2,3,5,6,7,8,9 覆盖
char n2[11]={"|   ||| ||"};//笔画2 被数字0,4,5,6,8,9 覆盖
char n3[11]={"|||||  |||"};//笔画3 被数字0,1,2,3,4,7,8,9 覆盖
char n4[11]={"  ----- --"};//笔画4 被数字2,3,4,5,6,8,9 覆盖
char n5[11]={"| |   | | "};//笔画5 被数字0,2,6,8覆盖
char n6[11]={"|| |||||||"};//笔画6 被数字0,1,3,4,5,6,7,8,9 覆盖
char n7[11]={"- -- -- --"};//笔画7 被数字0,2,3,5,6,8,9 覆盖
int main()
{
   int s;
   char szNumber[20];
   int nDigit, nLength, i, j, k;
   while(1)
   {
      scanf( "%d%s", &s, szNumber);
```

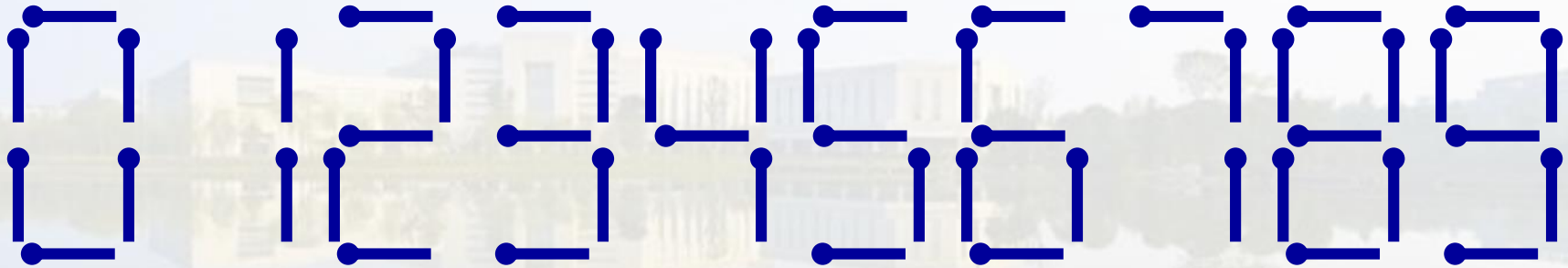
```
if(s == 0)
    break;
nLength = strlen(szNumber);
for(i = 0; i < nLength; i++)
{  //输出所有数字的笔画1
    nDigit = szNumber[i] - '0';
    printf(" ");
    for(j = 0; j < s; j++) //笔画由s 个字符组成
        printf("%c", n1[nDigit]);
    printf("  ");//两个空格
}
printf("\n");
for(i = 0; i < s; i++)
{ //输出所有数字的笔画2 和笔画3
    for(j = 0; j < nLength; j++)
    {
        nDigit = szNumber[j] - '0';
        printf("%c", n2[nDigit]);
```

```
        for(k = 0; k < s; k++)
            printf(" "); //两笔画之间的空格
        printf("%c ", n3[nDigit]);//有一空格
    }
    printf("\n");
}
for(i = 0; i < nLength; i++)
{  //输出所有数字的笔画4
    printf(" ");
    nDigit = szNumber[i] - '0';
    for(j = 0; j < s; j++)
        printf("%c", n4[nDigit]);
    printf("  ");//两个空格
}
printf("\n");
for(i = 0; i < s; i++)
{  //输出所有数字的笔画5 和笔画6
```

```
        for(j = 0; j < nLength; j++)
        {
            nDigit = szNumber[j] - '0';    printf("%c", n5[nDigit]);
            for(k = 0; k < s; k++)    printf( " "); //笔画5 和笔画6 之间的空格
            printf("%c ", n6[nDigit]);//有一个空格
        }
        printf("\n");
    }
    for(i = 0; i < nLength; i++)
    { //输出所有数字的笔画7
        printf(" ");
        nDigit = szNumber[i] - '0';
        for(j = 0; j < s; j++)      printf("%c", n7[nDigit]);
        printf("  ");//两个空格
    }
    printf("\n");     printf("\n");
  }
  return 0;
}
```

火柴棍拼等式

给你n根火柴棍, 你可以拼出多少个形如”A+B=C”的等式? 等式中的A、B、C是用火柴棍拼出的整数 (若该数非零, 则最高位不能是0). 用火柴棍拼数字0~9的拼法如下图所示.



- **Standard Input**

  只有一个整数n, n<=24.

- **Standard Output**

  输出能拼成的不同等式的数目.

  注意:

  (1)加号与等号各自需要两根火柴棍.

  (2)如果A!=B, 则A+B=C与B+A=C视为不同的等式 (A、B、C>=0).

  (3)N根火柴棍必须全部用上.

- **Samples**

Input	Output
18	9

题目分析

- 题目重要信息:
  - → +与=需要用4根火柴, n <= 24;
  - → 如果A!=B, 则A+B=C与B+A=C视为不同的等式(A、B、C>=0);
  - → N根火柴棍必须全部用上;
  - → 每个数字至少用两根火柴.
- 记住每个数字所用的火柴数目: int a[10] = {6,2,5,5,4,5,6,3,7,6};
- 预处理得到0~2000这些整数需要的火柴数目, 直接枚举判断即可.

参考程序

```
#include <iostream>
using namespace std;
int main()
{
    int k, n, ans = 0, temp;
    int a[10] = {6,2,5,5,4,5,6,3,7,6}, num[2001];
    cin >> n;
    for(int i = 0; i <= 2000; i++) {
        k = i;
        temp = 0;
        do
        {
            temp += a[k%10];
            k /= 10;
        }while(k > 0);
        num[i] = temp;
    }
    for(int i = 0; i <= 999; i++)
    {
        for(int j = 0; j <= 999; j++)
        {
            if(num[i]+num[j] >= n)  continue;
            else if(num[i]+num[j]+num[i+j]+4 == n)
                ans++;
        }
    }
    cout << ans;
    return 0;
}
```

A + B = C