ACM算法与程序设计

第三讲

简单题及模拟

数学科学学院：汪小平
wxiaoping325@163.com

Ants Run!

http://acm.uestc.edu.cn/ShowProblem.aspx?ProblemID=1161

- **Description**

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.

- **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.

- **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.

- **Sample Input**

  2
  3 1
  1 1 1
  3 1
  3 2 1

- **Sample Output**

  Inf
  3.142

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$t = \dfrac{2\pi r}{\sum_{n=1}^{n-1}(v_i - v_{i+1})}$。

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
- 如何求Pi：acos(-1.0)

```
#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;
}
```

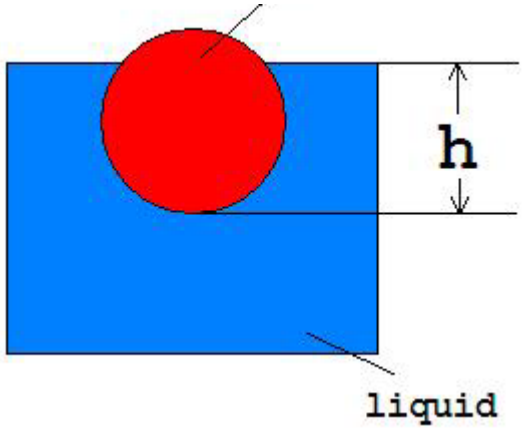
Archimedes

http://acm.uestc.edu.cn/ShowProblem.aspx?ProblemID=1162

- **Description**

Mr. Wang has got a ball made of a certain kind of metal (assume the ball is uniform density). He wants to find out what the metal is by calculating the density of the ball. Mr. Wang follows the story of Archimedes and crown. By putting the ball in the liquid (assume the liquid is deep enough). Mr. Wang can measure the height h indicated in the image.

And the density of the liquid is d, the radius of the ball is R. Now Mr. Wang asks you to simply tell the density of the ball. You can assume the density of the ball is no greater than the liquid.h measures the height from the bottom of the ball to the liquid surface.



- **Input**

  The input has many test cases. The first line is an integer T indicating the number of test cases. Following T lines there are three real numbers h, R, d.(0 < h <= 100, 0 < R <= 50,0 < d <= 100)

- **Output**

  For each test case, output one line with the answer, the density of the ball. round to 0.01. Please refer to the sample output.

- **Sample Input**

  2
  9 9 1
  9 8 1

- **Sample Output**

  0.50
  0.59

$$M=\rho_{球}\times V_{球}=d\times V_{液}$$

$$M=\frac{4}{3}\pi R^3\rho_{球}\qquad V_{液}=\int_0^h\pi[R^2-(R-x)^2]dx$$

整理得：$$\rho_{球}=\frac{d(3Rh^2-h^3)}{4R^3},(h\le 2R)$$

容易遗漏的是若小球不是漂浮，而是悬浮在水中（造成Wrong Answer的主要原因）

$$\rho_{球}=d \text{ 或 令} h=2R,(h\ge 2R)$$

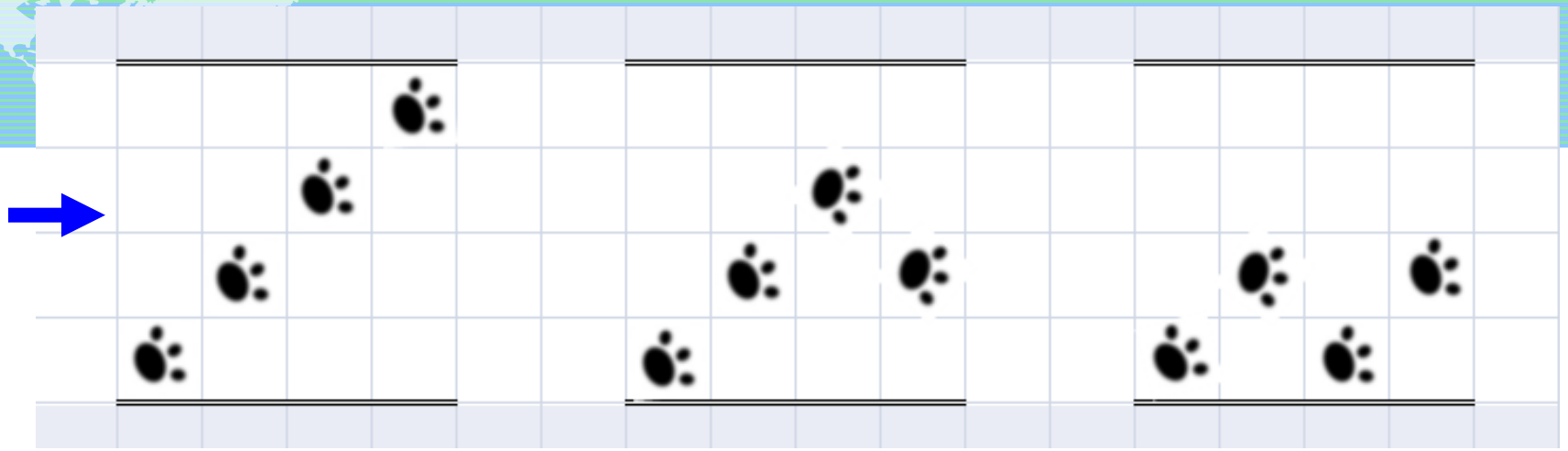
```
#include <stdio.h>
#include <iostream>
double r;
int main()
{
    int t,p;
    double h,d,v;
    scanf("%d",&t);
    for (p=1;p<=t;p++)
    {
        scanf("%lf%lf%lf",&h,&r,&d);
        if (h>2*r) h=2*r;
        v=3*r*h*h-h*h*h;
        printf("%.2lf\n",d*v/(4*r*r*r));
    }
    return 0;
 }
```

Flagstone Walk

http://acm.uestc.edu.cn/ShowProblem.aspx?ProblemID=1165

- **Description**

  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?

- **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.

- **Output**

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

- **Sample Input**

```
3
2
3
4
```

- **Sample Output**

```
1
2
3
```

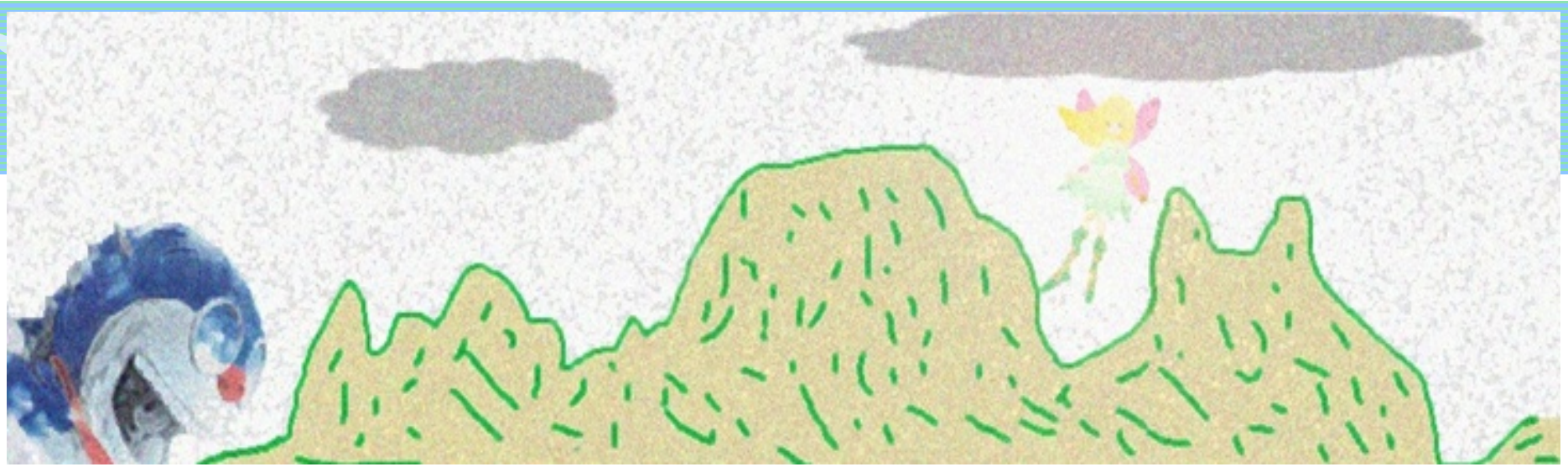
```
#include <stdio.h>
#include <string.h>
int main()
{
    int i, j, m, n, tot, a[6][21];
    scanf("%d", &n);
    while(n--)
    {
        scanf("%d", &m);
        memset(a, 0, sizeof(a));
        a[1][1] = 1; tot = 0;
        for(i = 2; i <= m; i++)
            for(j = 1; j <= 4; j++)
           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

http://acm.uestc.edu.cn/ShowProblem.aspx?ProblemID=1166

- **Description**

  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.**

- **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.

- **Output**

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

- **Sample Input**

5 3
1 2 3 4 5
2
4
6

- **Sample Output**

4
2
0

- 这个问题相当于问在一组数中有多少个是大于等于i的。
- All numbers are positive and not bigger than 100000.
- Time Limit:1000ms.
- 普通的方法可能会导致Time Limit Exceed，时间复杂度应该控制在O(nlogn)。
- 方法：先排序，再二分法找插入位置
- STL。。。
- 看看专业队员的编程

```
#include <iostream>
#include <sstream>
#include <iomanip>
#include <vector>
#include <deque>
#include <list>
#include <set>
#include <map>
#include <stack>
#include <queue>
#include <bitset>
#include <string>
#include <numeric>
#include <algorithm>
#include <functional>
#include <iterator>
#include <cstdio>
#include <cstring>
#include <cmath>
#include <cstdlib>
#include <cctype>
#include <complex>
#include <ctime>
```

```
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); //
        }
```

- 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.

```
#include <stdio.h>
#include <string.h>
int a[100001],b[100001];
int main()
{
   int n,m,i,j;
   while(scanf("%d%d",&n,&m)==2)
   {
        memset(b,0,10001*sizeof(int));
        for (i=1;i<=n;i++)
        {
               scanf("%d",&a[i]);
               b[a[i]]++; //把数作为下标，然后进行计数
        }
        for (i=99999;i>=1;i--)
               b[i]+=b[i+1];//找出大于等于b[i]的个数
        for (i=1;i<=m;i++)
        {
               scanf("%d",&j);
               printf("%d\n",b[j]);
        }
   }
   return 0;
}
```

模　拟

- 现实中的有些问题难以找到公式或规律来解决。只能按照一定步骤不停地做下去，最后才能得到答案。这样的问题，用计算机来解决十分合适，只要能让计算机模拟人在解决问题时的行为即可。这一类的问题可以称之为“模拟题”。

摘花生

http://poj.grids.cn/practice/2950

•问题描述

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（如图1）。

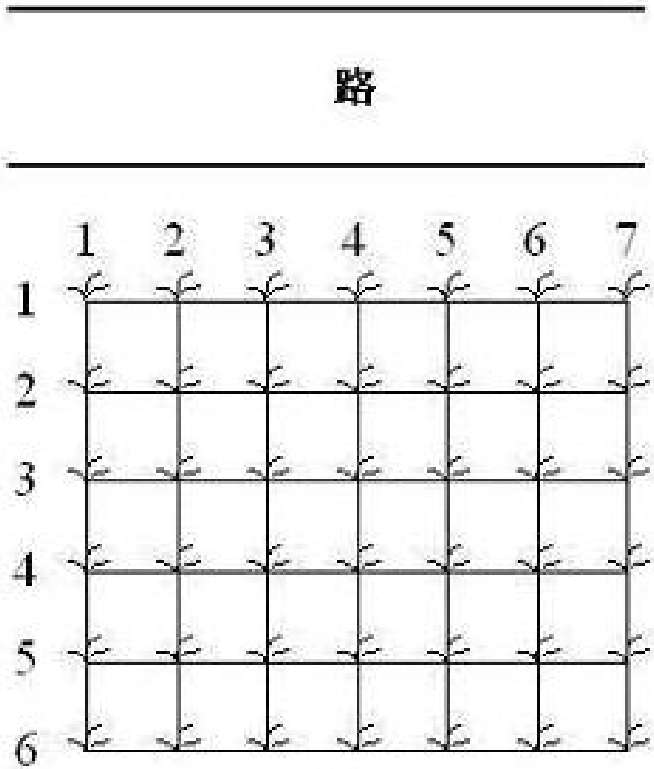


图 1

- 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。

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。

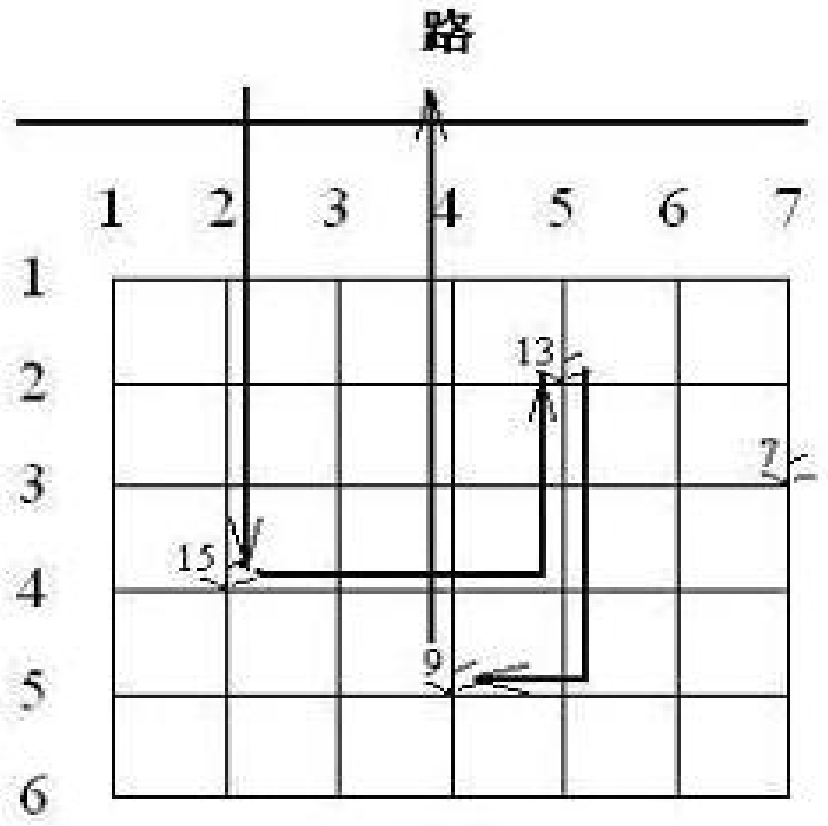


图 2

• 输入格式

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。

• 输出要求

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。

- **样例输入**

```
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
```

- **样例输出**

```
37
```

解题思路

- 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。
- 结果只能是做了才知道。即走进花生地．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。如果时问够．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，这样更为方便和自然，不易出错。

```
#include <stdio.h>
#include<stdlib.h>
#include<memory.h>
#include<math.h>
int T, M, N, K;
#define MAX_NUM 55
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);
                //花生地的左上角对应aField[1][1]，路的纵坐标是0
                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，
                //nCuri代表纵坐标，开始是在路上，所以初值为0
```

```
while(nTotalTime<K)
{  //如果还有时间
        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)    //地里已经没有花生了
              break;
        if(nCuri = = 0)
             nCurj=nMaxj; //如果当前位置是在路上，
        //那么应走到横坐标nMaxj处，再进人花生地
```

```
        // 下一行看剩余时间是否足够走到(nMaxi，nMaxj)处，
        //摘取花生，并回到路上
         if(nTotaITime+nMaxi+1+abs(nMaxi-nCuri)
               +abs(nMaxj-nCurj) <=K)
        {   // 下一行加上走到新位置，以及摘花生的时间
            nTotalTime+=1+abs(nMaxi-nCuri)+abs(nMaxj-
nCurj);
            nCuri=nMaxi;
            nCurj=nMaxj;  //走到新的位置
            nTotaiPeanuts+=aField[nMaxi][nMaxj];
            aField[nMaxi][nMaxj]=0;     //摘走花生
          }
          else
               break;
       }
     printf("%d\n", nTotalPeanuts);
    }
    return 0;
}
```

显示器

1、链接地址

http://poj.grids.cn/practice/2745

2、问题描述

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。

- **输入数据**

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。

- **输出要求**

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。

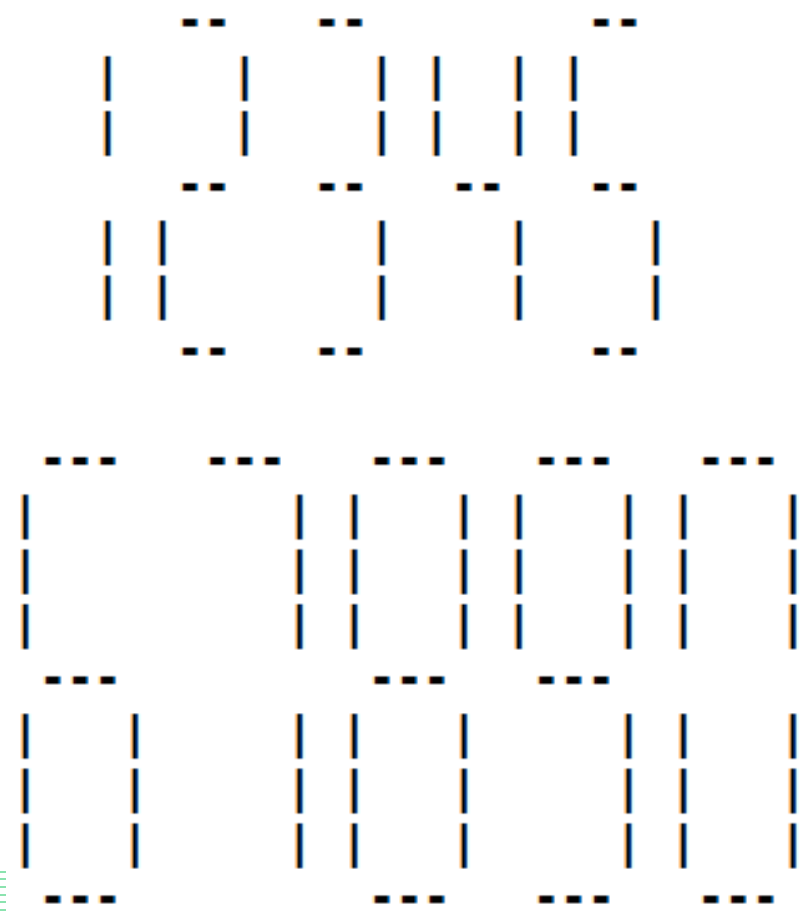
- 输入样例

2 12345

3 67890

0 0

- 输出样例



3、解题思路

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：

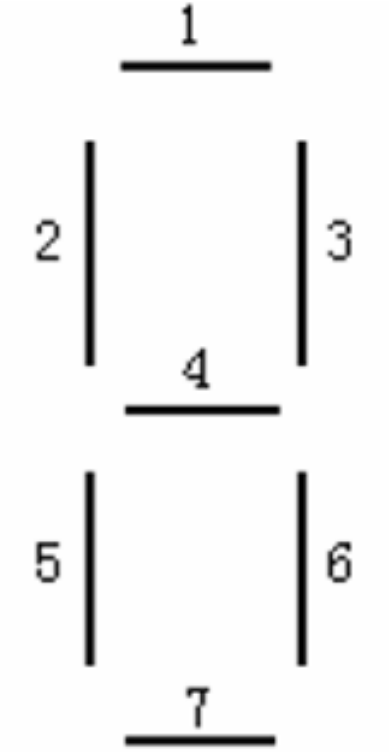


图 1 显示单元的笔画

解题思路

那么，我们可以说，数字8 覆盖了所有的笔画，数字7覆盖笔画1、3 和6，而数字1覆盖笔画3、6。注意，每个笔画都是由s 个’ -‘或s 个’ |’组成。

输出时，先输出第1 行，即整数n 中所有数字里的笔画1，然后输出第2 行到第s+1 行，即所有数字的笔画2和笔画3，接下来是第s+2 行，即所有数字的笔画4，再接下来是第s+3行到2×s+2 行，，就是所有数字的笔画5 和笔画6，最后的第2×s+3 行，是所有数字的笔画7。如果某个数字d 没有覆盖某个笔画m (m = 1…7)，那么，输出数字d 的笔画m 的时候，就应该都输出空格；如果覆盖了笔画m，则输出s 个’ -‘或s 个’ |’，这取决于笔画m 是横的还是竖的。

解题思路

由上思路，解决这道题目的关键，就在于如何记录每个数字都覆盖了哪些笔画。实际上，如果我们记录的是每个笔画都被哪些数字覆盖，则程序实现起来更为容易。一个笔画被哪些数字所覆盖，可以用一个数组来记录，比如记录笔画1 覆盖情况的数组如下：

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

其中，n1[i](i = 0……9) 代表笔画1 是否被数字i 覆盖。如果是，则n1[i] 为'-'，如果否，则n1[i]为空格。上面的数组的值体现了笔画1 被数字0, 2, 3, 5, 6, 7, 8, 9 覆盖。

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] = {"|   ||| ||"};
```

该数组的值体现了笔画2 被数字0, 4, 5, 6, 8, 9 覆盖。

4、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(void)
{
    int s;
    char szNumber[20];
    int nDigit , nLength, i , j , k;
```

4、参考程序

```
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++) //一个笔画由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(" "); //笔画2 和笔画3 之间的空格
        printf("%c ", n3[nDigit]);//有一个空格
    }
    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");
```

4、参考程序

```
    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;
}
```

5、实现技巧

一个笔画被哪些数字所覆盖，最直接的想法是用整型数组来记录，比如：

int n1[10] = {1, 0, 1, 1, 0, 1, 1, 1, 1, 1 };

表示笔画1 的被覆盖情况。可是与其在数字i 的笔画1 所处的位置进行输出的时候，根据n1[i]的值决定输出空格还是'-''，还不如直接用下面的char 类型数组来表示覆盖情况：

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

这样，在数字i 的笔画1 所处的位置进行输出的时候，只要输出s 个n1[i]就行了。这是一个很好的思路，它提醒我们，以后在编程时设置一些标志的时候，要考虑一下是否可以直接用更有意义的东西将0，1 这样的标志代替。