**编程题整理**

**目录：**

**[零、写在最前面](#_Toc7750)** [2](#_Toc7750)

**[一、简单字符、数字处理：](#_Toc4838)** [3](#_Toc4838)

**[1.soj 1623 大写转换为小写](#_Toc24616)** [3](#_Toc24616)

**[2.soj 1624 去掉空格](#_Toc21805)** [3](#_Toc21805)

**[3.soj 2517 回文数](#_Toc1328)** [4](#_Toc1328)

**[4.soj 3907 蛇形填数字](#_Toc6054)** [6](#_Toc6054)

**[5.soj 1974 大数取余](#_Toc18870)** [7](#_Toc18870)

**[6.soj 3915](#_Toc17504)** [8](#_Toc17504)

**[二、几个经典问题](#_Toc31296)** [9](#_Toc31296)

**[7.约瑟夫环 soj 1026](#_Toc7607)** [9](#_Toc7607)

**[8.八皇后 soj 1033](#_Toc32765)** [11](#_Toc32765)

**[9.汉诺塔](#_Toc24404)** [13](#_Toc24404)

**[10.排序（归并排序、快速排序、冒泡排序）](#_Toc10824)** [13](#_Toc10824)

**[三、简单dp](#_Toc23605)** [16](#_Toc23605)

**[11.soj 1114 数字三角](#_Toc25857)** [16](#_Toc25857)

**[12.soj 4292](#_Toc10766)** [17](#_Toc10766)

**[四、最小生成树](#_Toc19205)** [18](#_Toc19205)

**[12.soj 2198 prime](#_Toc11383)** [18](#_Toc11383)

**[五、迷宫地图](#_Toc15627)** [20](#_Toc15627)

**[14.马踏棋盘 soj2518](#_Toc9925)** [20](#_Toc9925)

**[六、二分 模版](#_Toc10406)** [24](#_Toc10406)

**[七、最短路](#_Toc29063)** [24](#_Toc29063)

**[八、几道例题：](#_Toc18508)** [25](#_Toc18508)

**[1.soj 3918](#_Toc6388)** [25](#_Toc6388)

**[2.soj 3926](#_Toc11400)** [26](#_Toc11400)

**[3.soj 2666 分解n！](#_Toc5077)** [26](#_Toc5077)

**[参考答案：](#_Toc8130)** [26](#_Toc8130)

**零、写在最前面**

首先，这些题目是根据历年复试的题目整理出来，复试过程中可能的考点。第一、二部分涉及简单的编程思想和基础的算法，最为关键，后面几个部分都是简单的算法，可能会出题。

这里选的例题都是scu online judg（http://acm.scu.edu.cn/soj/index.action）上的，因为考虑到出题的老师可能是四川大学集训队的教练左劼老师。

最后，说一下答题的小trick，因为川大不上机去考编程（不知道今年会不会改），所以答题时候一定先写一下自己的解题思路，写的代码尽量写点注释。即使不会也要把自己的想法写上。

**一、简单字符、数字处理：**

**1.soj 1623 大写转换为小写**

读入一些字符串，将其中的小写字母转成大写字母（其他字符不变）。

输入示例

Hello

ICPC2004

12345abcde

输出示例

HELLO

ICPC2004

12345ABCDE

示例代码：

#include <stdio.h>  
#include <stdlib.h>  
**int** main ()  
{  
    **int** ch;  
    **while**((ch=getchar( )) !=EOF)  
    {  
        **if**(ch>='a'&& ch<='z')  
        {  
            putchar(ch-'a'+'A');  
        }  
        **else**  
        {  
            putchar(ch);  
        }  
    }  
      
}

**2.soj 1624 去掉空格**

读入一些字符串，将其中的空格去掉。

输入示例

Hello World

1 2 3

Nice to meet you

abc

输出示例

HelloWorld

123

Nicetomeetyou

abc

示例代码：

#include<stdio.h>  
  
**void** fun(**char** \*str)  
{  
   **int** i,j;  
   **for**(i=0,j=0;str[i]!='\0';i++)  
   {  
    **if**(str[i]!=' ')  
    {   
        str[j]=str[i];  
        j++;  
    }  
   }  
   str[j]='\0';  
}  
  
**int** main(**void**)  
{  
  **char** str1[81];  
  **while**(gets(str1)!=NULL)  
  {  
      fun(str1);   
      puts(str1);  
  }  
 }

**3.soj 2517 回文数**

Description

Farmer John has been branding the cows with a serial number ever since he started the farm. The new cow fad is 'RADAR' brands, so-called because they read the same forwards and backwards (they are palindromic). All the cows want their daughters branded in the new RADAR style.

Each mother wants her daughter's brand to be derived from her own non-RADAR brand by summing the mother's brand and its reverse. Sometimes (e.g., 12 + 21 = 33) this yields a RADAR palindrome right away. Sometimes the process must be repeated several times until a RADAR brand emerges. Consider the brand '87' that requires four steps to convert to a RADAR brand:

**Brand Reverse Sum**

**Step 1: 87 + 78 = 165**

**Step 2: 165 + 561 = 726**

**Step 3: 726 + 627 = 1353**

**Step 4: 1353 + 3531 = 4884**

**Given the mother's brand (a positive integer), determine the number of steps and ultimate RADAR brand that results from applying the procedure above. No answer will be greater than two billion.**

题目大意是输入n，若不是回文数，则将n反转和n求和得到新的n，再判断n是否为回文数，问几次这样的操作可以使得最开始的n变成回文数。

示例输入

87

示例输出

4 4884

示例代码：

#include <stdio.h>  
**int** reverse(**int** x);  
**int** judge(**int** a);  
**int** main()  
{  
    **int** a,b,nun;  
  
    **while**(scanf("%d",&a)==1)  
    {  
        nun=0;  
        **while**(judge(a))  
        {          
            b=reverse(a);  
            a=a+b;  
            nun++;  
        }  
        printf("%d %d\n",nun,a);  
    }  
      
}  
  
  
**int** reverse(**int** x)  
{  
    **int** a=0;  
    **while**(x!=0)  
    {  
        a=x%10+a\*10;  
        x=x/10;  
    }  
    **return** a;  
}  
  
**int** judge(**int** a)  
{  
    **int** b;  
    b=reverse(a);  
    **if**(a-b==0)  
    {  
        **return** 0;  
    }  
    **else** **return** 1;  
}

**4.soj 3907 蛇形填数字**

Oh how the cows love to walk in their square pasture with sides of length N (1 <= N <= 750) and which is partitioned into N\*N squares.They enjoy the sights, the smells, and the general ambience of the

grass and trees.

Bessie has decided to take the cows on the longest possible walk from the upper left corner to the center (or near the center when N is even) of the pasture, passing through each and every square along the way after starting.

She has decided to create the obvious clockwise spiral route (example

below) for this evening's stroll. Write a program to create a map for her that shows the order of squares she should visit.

By way of example, for pastures of size N=3 and N=4, here are the

routes Bessie should use:

1 2 3 1 2 3 4

8 9 4 12 13 14 5

7 6 5 11 16 15 6

10 9 8 7

大意给你个n按照如上图所示的格式输出n\*n的矩阵。

示例代码：

#include<stdio.h>  
#include<string.h>  
**int** a[750][750];  
**int** main()  
{  
    **int** n,x,y,k;  
      
    **while**(scanf("%d",&n)==1)  
    {  
        memset(a,0,**sizeof**(a));  
        a[0][0]=1;  
        k=1;  
        x=y=0;  
        **while**(k<n\*n)  
        {  
            **while**(y+1<n&&a[x][y+1]==0)  a[x][++y]=++k;  
            **while**(x+1<n&&a[x+1][y]==0)  a[++x][y]=++k;  
            **while**(y-1>=0&&a[x][y-1]==0)  a[x][--y]=++k;  
            **while**(x-1>=0&&a[x-1][y]==0)  a[--x][y]=++k;  
        }  
        **for**(x=0;x<n;x++)  
        {  
            **for**(y=0;y<n-1;y++)  
            {  
                printf("%d ",a[x][y]);  
            }  
            printf("%d\n",a[x][y]);  
        }  
  
    }  
}

**5.soj 1974 大数取余**

Problem

9 is a lucky number in the chinese traditional culture. This problem gives you an integer number, whose absolute value is not larger than 10^1000000, and your task is calculating the integer mod 9.

题目大意是给你一个大数，问其mod9的结果

示例代码:

#include<stdio.h>  
#include<string.h>  
**char** str[1000001];  
**int** main()  
{  
    **int** len,i,ans,cas;  
    scanf("%d",&cas);  
    getchar();  
    **while**(cas--)  
    {  
        gets(str);  
        len=strlen(str);  
        **if**(str[0]=='-')  
        {  
            ans=0;  
            **for**(i=1;i<len;i++)  
                ans= ( ans\*10+str[i]-'0')%9;  
            ans=9-ans;  
        }  
        **else**  
        {  
            ans=0;  
            **for**(i=0;i<len;i++)  
                ans= (ans\*10+str[i]-'0' )%9;  
        }  
        printf("%d\n",ans);  
    }  
}

**6.soj 3915**

**只有4和7两个数组成的序列，从小到大排列，给你n代表第n个数要你输出这个数。**

Digits 4 and 7 are lucky, while all others are unlucky. An integer is lucky if it contains only lucky digits in decimal notation. We would like to know the K-th lucky positive integer.

**题目大意只用4和7来组成数字，问第n大的是多少。**

示例输入：

1

2

3

示例输出：

4

7

44

题解：这题有个小trick，将4看成0，7看成1。4、7、44、47、74、77、444、447……

4 0

7 1

44 00

47 01 再在每个数前加上1。

4 10

7 11

44 100

47 101 发现正好是二进制的2、3、4、5……

示例代码：

#include<stdio.h>  
**int** main()  
{  
    **int** k,i,j,a[100];  
    **while**(scanf("%d",&k)!=EOF)  
    {  
        k+=1;  
        i=0;  
        **while**(1)  
        {  
            a[i]=k%2;  
            i++;  
            k/=2;  
            **if**(k==0) **break**;  
        }  
        **for**(j=i-2;j>=0;--j)  
        {  
            **if**(a[j]==1)  
                printf("7");  
            **else**  
                printf("4");  
        }  
        printf("\n");  
    }  
}

**二、几个经典问题**

**7.约瑟夫环 soj 1026**

n个人（编号0~(n-1）），从0开始报数，报到（m-1）的退出，剩下的人继续从0开始报数。求胜利者的编号。这是经典的约瑟夫问题。

1026这题是编号 1到n-1的n-1个人，求m使得最后1活下来。

示例输入：

3

4

5

6

7

8

9

10

11

12

示例输出：

2

5

2

4

3

11

2

3

8

16

题解：

我们知道第一个人（编号一定是（m-1） mod n) 出列之后，剩下的n-1个人组成了一个新的[约瑟夫环](http://baike.baidu.com/view/717633.htm" \t "_blank)（以编号为k=m mod n的人开始）：

k k+1 k+2 ... n-2,n-1,0,1,2,... k-2

并且从k开始编为0。

现在我们把他们的编号做一下转换：

k --> 0

k+1 --> 1

k+2 --> 2

...

...

k-2 --> n-2

变换后就完完全全成为了（n-1）个人报数的子问题，假如我们知道这个子问题的解：例如x是最终的胜利者，那么根据上面这个表把这个x变回去不刚好就是n个人情况的解吗？！！变回去的公式很简单，相信大家都可以推出来：x'=(x+k) mod n

如何知道（n-1）个人报数的问题的解？对，只要知道（n-2）个人的解就行了。（n-2）个人的解呢？当然是先求（n-3）的情况 ---- 这显然就是一个倒推问题！好了，思路出来了，下面写递推公式：

令f表示i个人玩游戏报m退出最后胜利者的编号，最后的结果自然是f[n]

递推公式

f[1]=0;

f=(f+m) mod i; (i>1）

有了这个公式，我们要做的就是从1-n顺序算出f的数值，最后结果是f[n]。因为实际生活中编号总是从1开始，我们输出f[n]+1

由于是逐级递推，不需要保存每个f，程序也是异常简单：

#include <stdio.h>

int main()

{

int n,m,i,s=0;

printf ("N M = "); scanf("%d%d",&n,&m);

for (i=2; i<=n; i++) s=(s+m)%i;

printf ("The winner is %d\n",s+1）;

}

而本题是生还者和报数的人已知，求报数的个数。于是我从2开始循环往后，看看报完后是不是我要的剩下了即可。

示例代码：

#include<stdio.h>  
**int** main()  
{  
    **int** n,i,j,s,flag;  
    **while**(scanf("%d",&n)&&n!=0)  
    {  
        **for**(i=2;;i++)  
        {  
            s=0;  
            **for** (j=2; j<=n-1; j++)   
            {  
                s=(s+i)%j;  
            }  
            **if**(s==0)  
            {  
                    printf("%d\n",i);  
                    flag=1;  
                    **break**;  
            }  
        }  
    }  
}

**8.八皇后 soj 1033**

The Sultan of Nubia has no children, so she has decided that the country will be split into up to k separate parts on her death and each part will be inherited by whoever performs best at some test. It is possible for any individual to inherit more than one or indeed all of the portions. To ensure that only highly intelligent people eventually become her successors, the Sultan has devised an ingenious test. In a large hall filled with the splash of fountains and the delicate scent of incense have been placed k chessboards. Each chessboard has numbers in the range 1 to 99 written on each square and is supplied with 8 jewelled chess queens. The task facing each potential successor is to place the 8 queens on the chess board in such a way that no queen threatens another one, and so that the numbers on the squares thus selected sum to a number at least as high as one already chosen by the Sultan. (For those unfamiliar with the rules of chess, this implies that each row and column of the board contains exactly one queen, and each diagonal contains no more than one.)

Write a program that will read in the number and details of the chessboards and determine the highest scores possible for each board under these conditions. (You know that the Sultan is both a good chess player and a good mathematician and you suspect that her score is the best attainable.)

题目大意：8皇后问题，只不过每个点都加上了值，让你求出其中最大的那种情况。我是直接放摆皇后然后求sum，判断max=max>sum?max:sum。同样的搜索。

示例输入：

1

1 2 3 4 5 6 7 8

9 10 11 12 13 14 15 16

17 18 19 20 21 22 23 24

25 26 27 28 29 30 31 32

33 34 35 36 37 38 39 40

41 42 43 44 45 46 47 48

48 50 51 52 53 54 55 56

57 58 59 60 61 62 63 64

示例输出：

260

示例代码：

#include<stdio.h>  
#include<string.h>  
**void** search (**int** cur);  
  
**int** a[8][8],max;  
**int** p[8];  
**int** main()  
{  
    **int** cas,i,j;  
    scanf("%d",&cas);  
    **while**(cas--)  
    {  
        **for**(i=0;i<8;i++)  
            **for**(j=0;j<8;j++)  
                scanf("%d",&a[i][j]);  
      
            max=0;  
            search(0);  
  
  
            printf("%5d\n",max);  
    }  
}  
  
**void** search (**int** cur)  
{  
    **int** i,j,sum,ok;  
    **if**(cur==8)  
    {  
        sum=0;  
        **for**(i=0;i<8;i++)  
            sum+= a[i][p[i]];  
        max=sum>max?sum:max;  
    }  
    **else**  
    {  
        **for**(i=0;i<8;i++)  
        {  
            ok=1;  
            p[cur]=i;  
            **for**(j=0;j<cur;j++)  
                **if**( p[cur] ==p[j] || p[cur]-p[j] == cur-j || p[cur]-p[j] == j-cur)  
                {  
                    ok=0;  
                    **break**;  
                }  
                **if**(ok==1)  
                    search(cur+1);  
        }  
    }  
}

**9.汉诺塔**

**递归问题**

#include<stdio.h>

void move(int n,char a,char b,char c)

{

if(n==1)

printf("\t%c->%c\n",a,c); //当n只有1个的时候直接从a移动到c

else

{

move(n-1,a,c,b); //第n-1个要从a通过c移动到b

printf("\t%c->%c\n",a,c);

move(n-1,b,a,c); //n-1个移动过来之后b变开始盘，b通过a移动到c

}

}

main()

{

int n;

printf("请输入要移动的块数：");

scanf("%d",&n);

move(n,'a','b','c');

}

**10.排序（归并排序、快速排序、冒泡排序）**

**归并排序**

**///归并排序模板。记得和快排一块对着看。**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<iostream>**

**#include<string.h>**

**#include<cstring>**

**#include<string>**

**#include<math.h>**

**#include<algorithm>**

**#define LL long long**

**#define inf 0x3f3f3f3f**

**#define mod 1e9+7**

**const int maxn=1e5+5;**

**using namespace std;**

**int temp[maxn];**

**int num=0;///统计逆序对数的。**

**void Merge(int a[],int left ,int mid,int right)**

**{**

**int i=left,j=mid+1,n=0,length=right-left;///i开始为左半部分最左边，j为右半部分最左边。temp数组是从下标0开始存数。**

**while(i<=mid&&j<=right){**

**if(a[i]>a[j]){///左边比右边大。**

**temp[n++]=a[j++];**

**num+=mid-i+1;///从i到mid都是比a[j]大。**

**}**

**else{**

**temp[n++]=a[i++];**

**}**

**}**

**if(i>mid){///这里说明的是左边全部填满了(因为前面的判断条件是i<=mid，那就是天右边了。**

**while(j<=right){**

**temp[n++]=a[j++];**

**}**

**}**

**else{**

**while(i<=mid){**

**temp[n++]=a[i++];**

**}**

**}**

**for(int k=0;k<=length;k++){///最后赋值到原数组必须要有的。**

**a[left+k]=temp[k];**

**}**

**}**

**void mergesort(int a[],int left,int right)**

**{**

**if(left<right){**

**int mid=(left+right)/2;**

**mergesort(a,left,mid);**

**mergesort(a,mid+1,right);**

**Merge(a,left,mid,right);**

**}**

**}**

**int main()**

**{**

**int number[30] = {23,3,26,24,5,1,12,21,29,15,17,10,7,22,6,20,19,11,2,4,9,25,13,27,14,18,28,8,16,30};**

**mergesort(number,0,30-1);///初始化调用也要注意额。**

**for(int i=0;i<30;i++){**

**printf("%d ",number[i]);**

**}**

**}**

**快速排序**

**///快排模板，我的那种真的简单额。**

**#include<stdio.h>**

**#include<stdlib.h>**

**#include<iostream>**

**#include<string.h>**

**#include<cstring>**

**#include<string>**

**#include<math.h>**

**#include<algorithm>**

**#define LL long long**

**#define inf 0x3f3f3f3f**

**#define mod 1e9+7**

**#include<vector>**

**using namespace std;**

**int Partition(int a[], int left, int right)///7行代码**

**{**

**int i = left-1;///初始化一定要赋值好。**

**for(int j=left;j<=right-1;j++){**

**if(a[j] < a[right]){///把right这个作为轴了。**

**i++;///这个i坐标左边的值就是比a[right]小的。**

**swap(a[i], a[j]);///必须交换一下。**

**}**

**}**

**swap(a[i+1], a[right]);///最后把i+1和right交换，这样轴就是i+1了必须是保证i+1上当初就是作为标杆的a[right]啊。**

**return i+1;**

**}**

**void Qsort(int a[], int left, int right)**

**{**

**if(left<right){**

**int q = Partition(a, left, right);**

**Qsort(a, left, q-1);**

**Qsort(a, q+1, right);**

**}**

**}**

**int main()**

**{**

**Int number[30] = {23,3,26,24,5,1,12,21,29,15,17,10,7,22,6,20,19,11,2,4,9,25,13,27,14,18,28,8,16,30};**

**Qsort(number,0,30-1);///c初始化调用也要注意额。**

**for(int i=0;i<30;i++){**

**printf("%d ",number[i]);**

**}**

**}**

**三、简单dp**

**11.soj 1114 数字三角**

**下图是个数字三角，请编写一个程序计算从顶部至底部某处一条路径，使得该路径所经过的数字总和最大。**

**7**

**3 8**

**8 1 0**

**2 7 4 4**

**1． 每一步可沿左斜线向下或右斜线向下走；**

**2． 1<=三角形行数<=100**

**3． 三角形中的数字为整数 0，1，……，99。**

**4． 如果有多种情况结果都最大，任意输出一种即可。**

**dp[i][j]表示到第i行j列的位置的最大值，点dp(i,j)的状态由dp(i+1,j)和dp(i+1,j+1)转移而来，状态转移方程就是dp[i][j]=a[i][j]+max(dp[i+1][j],dp[i+1][j+1]) 。**

**示例代码：**

#include <stdio.h>  
  
**int** a[102][102],n,d[102][102];  
**int** main()  
{  
    **int** i,j;  
    scanf ("%d",&n);  
      
    **for** (i=1;i<=n;i++)  
          **for** (j=1;j<=i;j++)  
             scanf("%d",&a[i][j]);  
  **for** (j=1;j<=n;j++) d[n][j]=a[n][j];  
  
  **for** (i=n-1;i>=1;i--)  
      **for** (j=1;j<=i;j++)  
      {  
          d[i][j]=a[i][j]+(d[i+1][j] >d[i+1][j+1]?d[i+1][j]:d[i+1][j+1]);  
            
      }  
      printf ("%d\n%d",d[1][1],a[1][1]);  
      **for**(i=2;i<=n;i++)  
      {  
          j=1;  
          **if**(d[i][j]>=d[i][j+1])  
          {printf(" %d",a[i][j]);j=j;}  
          **else**  
          {printf(" %d",a[i][j+1]);j=j+1;}  
            
      }  
      printf("\n");  
**return** 0;  
}

**12.soj 4292**

**One can transform xx into x+dx+d if dd is divisor of xx. Find out the minimum number of steps to transform aa into bb.**

**题目大意，有个操作是，给定x，每次可以进行x+d的操作，d是x的约数。找出最小的步数使得a变到b。**

**dp[i]表示a到数字i要的最小步数，转台转移方程为：(j是i的约数)**

**if(i+j<=b)**

**dp[i+j]=min(dp[i+j],dp[i]+1);**

**if(i+i/j<=b)**

**dp[i+i/j]=min(dp[i+i/j],dp[i]+1);**

**示例代码：**  
#include<iostream>  
#include<stdio.h>  
#include<math.h>  
#include<string.h>  
**using** **namespace** std;  
  
**int** min(**int** a,**int** b)  
{  
    **return** a<b?a:b;  
}  
  
**int** main()  
{  
    **int** a,b,i,j,dp[100005];  
    **while**(scanf("%d %d",&a,&b)==2)  
    {  
        **if**(a>b)  
        {  
            printf("-1\n");  
            **continue**;  
        }  
        memset(dp,1,**sizeof**(dp));  
        dp[a]=0;  
        **for**(i=a;i<=b;i++)  
            **for**(j=1;j\*j<=b;j++)  
            {  
                **if**(i%j==0)  
                {  
                    **if**(i+j<=b)  
                    dp[i+j]=min(dp[i+j],dp[i]+1);  
                    **if**(i+i/j<=b)  
                    dp[i+i/j]=min(dp[i+i/j],dp[i]+1);  
                }  
            }  
        **if**(dp[b]>=100000)  
            printf("-1\n");  
        **else**  
            printf("%d\n",dp[b]);  
    }  
}

**四、最小生成树**

**12.soj 2198 prime**

**The island nation of Flatopia is perfectly flat. Unfortunately, Flatopia has no public highways. So the traffic is difficult in Flatopia. The Flatopian government is aware of this problem. They're planning to build some highways so that it will be possible to drive between any pair of towns without leaving the highway system.**

**Flatopian towns are numbered from 1 to N. Each highway connects exactly two towns. All highways follow straight lines. All highways can be used in both directions. Highways can freely cross each other, but a driver can only switch between highways at a town that is located at the end of both highways.**

**The Flatopian government wants to minimize the length of the longest highway to be built. However, they want to guarantee that every town is highway-reachable from every other town.**

**题目大意是，N个村庄，给你N个村庄修路的代价，问使得N个村庄联通的最小代价。**

**示例输入：**

3

0 990 692

990 0 179

692 179 0

**示例输出：**

692

最小生成树两个解法prime和kruskal，一个从点出发，一个从最短的边出发，根据实际情况来定。

示例代码：

#include<cstdio>  
#define MAX 65537  
**int** i,j,n,u,T,max,tp,map[501][501],d[501];  
**bool** s[501];  
**void** Prim()  
{  
    **for**(i=1;i<=n;i++)  
    {  
        d[i]=map[1][i];  
        s[i]=**false**;  
    }  
    s[1]=**true**;  
    **for**(j=1;j<=n;j++)  
    {  
        tp=MAX,u=1;  
        **for**(i=1;i<=n;i++)  
        {  
            **if**(!s[i]&&d[i]<tp)  
                tp=d[i],u=i;  
        }  
        s[u]=**true**;  
        **for**(i=1;i<=n;i++)  
        {  
            **if**(!s[i]&&map[u][i]<d[i])  
                d[i]=map[u][i];  
        }  
    }  
}  
**int** main()  
{  
    scanf("%d",&T);  
    **while**(T--)  
    {  
        scanf("%d",&n);  
        **for**(i=1;i<=n;i++)  
            **for**(j=1;j<=n;j++)  
            scanf("%d",&map[i][j]);  
        Prim();  
        **for**(i=1,max=0;i<=n;i++)  
            **if**(d[i]>max)max=d[i];  
        printf("%d\n",max);  
    }  
    **return** 0;  
}

**五、迷宫地图**

**14.马踏棋盘 soj2518**

**One day while grazing in particularly deep hay, Betsy discovered four magic cow shoes! She donned them and found that they enabled her to jump around the pasture which, of course, is subdivided into a convenient grid with R rows (1 <= R <= 50) and C columns (1 <= C <= 50). The shoes enabled her to jump two different ways: both like a chess game knight and in another pattern she'd never seen before. She noticed that she jumped like a knight on her first, third, fifth, and odd-numbered moves while she jumped in the new way on the second, fourth, sixth, and even-numbered moves. Here is a map of the possible move patterns she discovered:**

**Knight (Odd moves) Other (Even moves)**

**. . K . K . . . . . O . . .**

**. K . . . K . . . . . . . .**

**. . . B . . . . O . B . O .**

**. K . . . K . . . . . . . .**

**. . K . K . . . . . O . . .**

**When Betsy starts at the 'B', depending on whether her next move is an odd or even one, she can jump to any one of the 'K's or 'O's.**

**Realizing she can now move about the pasture much more quickly, Betsy wonders how long it will take her to jump using the magic shoes all the way over to the Milky Way candy bar Farmer John accidentally dropped on his recent visit to the cows.**

**Given the size of the field along with the locations of Betsy and the candy bar, determine the minimum number of magic shoe jumps required for Betsy to land on the square with the candy bar. She is not allowed to jump outside the pasture but is sure that it is always possible to get to the candy bar.**

**Input**

**The input file contains multiple test cases, for each test case:**

**\* Line 1: Two space-separated integers: respectively R and C.**

**\* Line 2: Two space-separated integers: respectively the row and column of Betsy's starting location.**

**\* Line 3: Two space-separated integers: respectively the row and column of the candy bar.**

**示例输入：**

4 5

4 1

4 3

**示例输出：**

3

**给定r\*c的格子，起点和终点，进行跳跃，当时奇数跳跃时候，采用一种跳跃方式，当时偶数跳跃的时候采用另外一种跳跃方式。**

**搜索bfs**

**示例代码：**

#include<iostream>  
#include<queue>  
#include<cstring>  
#include<stdio.h>  
**using** **namespace** std;  
  
**int** endx,endy,startx,starty,r,c,k,step;  
**int** visit[2550][2];  
**int** visit1[2550][2];  
**int** visit2[55][55];  
**int** m,Sum;  
**int** oddx[8]={-2, -1, 1, 2, 2, 1, -1, -2};  
**int** oddy[8]={1, 2, 2, 1, -1, -2, -2, -1};  
**int** evenx[4]={2, 0, -2, 0};  
**int** eveny[4]={0, 2, 0, -2};  
  
**int** judge(**int** x,**int** y)  
{  
    **if**((x==endx) && (y==endy))  
        **return** 1;  
    **if**( x<1 || y<1 || x>r || y>c )  
        **return** 0;  
    **return** 3;  
}  
  
**int** odd(**int** x,**int** y)  
{  
    **int** i;  
    **int** tx,ty;  
    **for**(i=0;i<8;i++)  
    {  
        tx=x+oddx[i];  
        ty=y+oddy[i];  
        **if**(visit2[tx][ty]==1)  
            **continue**;  
        **if**(judge(tx,ty)==1)  
            **return** 1;  
        **else** **if**(judge(tx,ty)==0)  
            **continue**;  
        visit1[m][0]=tx;  
        visit1[m++][1]=ty;  
        visit2[tx][ty]=1;  
        Sum++;  
    }  
      
    **return** 0;  
}  
  
**int** even(**int** x,**int** y)  
{  
    **int** i;  
    **int** tx,ty;  
    **for**(i=0;i<4;i++)  
    {  
        tx=x+evenx[i];  
        ty=y+eveny[i];  
        **if**(visit2[tx][ty]==1)  
            **continue**;  
        **if**(judge(tx,ty)==1)  
            **return** 1;  
        **else** **if**(judge(tx,ty)==0)  
            **continue**;  
        visit1[m][0]=tx;  
        visit1[m++][1]=ty;  
        visit2[tx][ty]=1;  
        Sum++;  
    }  
      
    **return** 0;  
}  
  
**int** main()  
{  
    **int** sum,flag,i;  
    **while**(scanf("%d %d",&r,&c)==2)  
    {  
        cin>>visit[0][0]>>visit[0][1]>>endx>>endy;  
        **for**(i=0;i<55;i++)  
            memset(visit2[i],0,**sizeof**(visit2[0]));  
        step=1;  
        sum=1;  
        Sum=0;  
        m=0;  
        flag=0;  
        **if**((visit[0][0]==endx)&&(visit[0][1]==endy))  
            cout<<0<<endl;  
        **else**  
        {  
            **while**(flag!=1)  
            {  
                **if**(Sum!=0)  
                    sum=Sum;  
                Sum=0;  
                  
                **for**(i=0;i<sum;i++)  
                {  
                    startx=visit[i][0];  
                    starty=visit[i][1];  
                    **if**(step%2==1)  
                        flag=odd(startx,starty);  
                    **else**  
                        flag=even(startx,starty);  
                    **if**(flag==1)  
                        **break**;  
                }  
                step++;  
                **for**(i=0;i<55;i++)  
                    memset(visit2[i],0,**sizeof**(visit2[0]));  
                **if**(flag==1)  
                    **break**;  
                **for**(i=0;i<m;i++)  
                {  
                    visit[i][0]=visit1[i][0];  
                    visit[i][1]=visit1[i][1];  
                }  
                **if**(m==0)  
                    **break**;  
                m=0;  
            }  
            cout<<step-1<<endl;  
        }  
    }  
}

**六、二分 模版**

**int binary\_search(int \*array, int length, int key) {**

**int start = 0, end = length - 1;**

**while(end >= start) {**

**int middle = start + (end - start) / 2;**

**int tmp = array[middle];**

**if(tmp < key) start = middle + 1;**

**else if (tmp > key) end = middle - 1;**

**else return middle;**

**}**

**return -1;**

**}**

**七、最短路**

**Floyd (O(n^3))模版 dp(i,j)表示i到j的最短距离，初始化别忘了。**

**for(k=0;k<n;k++)**

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

**for(j=0;j<n;j++)**

**if(dp[i][j]>(dp[i][k]+dp[k][j]))**

**dp[i][j]=dp[i][k]+dp[k][j];**

**Dijkstra 单源最短路 得到的dist数组里存的是从vo开始到各个点的最短距离。**

**const int MAXINT = 32767;**

**const int MAXNUM = 10;**

**int dist[MAXNUM];**

**int prev[MAXNUM];**

**int A[MAXUNM][MAXNUM];**

**void Dijkstra(int v0)**

**{**

**bool S[MAXNUM]; // 判断是否已存入该点到S集合中**

**int n=MAXNUM;**

**for(int i=1; i<=n; ++i)**

**{**

**dist[i] = A[v0][i];**

**S[i] = false; // 初始都未用过该点**

**if(dist[i] == MAXINT)**

**prev[i] = -1;**

**else**

**prev[i] = v0;**

**}**

**dist[v0] = 0;**

**S[v0] = true;**

**for(int i=2; i<=n; i++)**

**{**

**int mindist = MAXINT;**

**int u = v0; 　　 // 找出当前未使用的点j的dist[j]最小值**

**for(int j=1; j<=n; ++j)**

**if((!S[j]) && dist[j]<mindist)**

**{**

**u = j; // u保存当前邻接点中距离最小的点的号码**

**mindist = dist[j];**

**}**

**S[u] = true;**

**for(int j=1; j<=n; j++)**

**if((!S[j]) && A[u][j]<MAXINT)**

**{**

**if(dist[u] + A[u][j] < dist[j]) //在通过新加入的u点路径找到离v0点更短的路径**

**{**

**dist[j] = dist[u] + A[u][j]; //更新dist**

**prev[j] = u; //记录前驱顶点**

**}**

**}**

**}**

**}**

**八、几道例题：**

**1.soj 3918**

**There is a snail on the ground. It wants to climb to the top of a wooden pole with the height of V meters, measuring from the ground level. In one day it can climb A meters upwards, however during each night it sleeps, sliding B meters back down. Determine the number of days it needs to climb to the Top.**

**（题意一个小时候经常做到的问题，假设青蛙每天向上爬a，掉下来b，高度为h。）**

**示例输入：**

2 1 5

5 1 6

100 99 1000000000

**示例输出:**

4

2

999999901

**2.soj 3926**

**One day, Jiameier is tidying up the room,and find some coins. Then she throws the coin to play.Suddenly,she thinks of a problem ,that if throw n times coin ,how many situations of no-continuous up of the coin. Hey,Let's solve the problem.**

**Input： n (1<=n<=1000) in a line, indicate the times of throwing coins.**

**（题意是将一个硬币抛n次，求没有连续两次向上的次数。）**

**示例输入：**

**1**

**2**

**3**

**示例输出：**

**2**

**3**

**5**

**3.soj 2666 分解n！**

**给你一个数 n (1 < n <= 1000000) ,求 n! (n的阶乘)的质因数分解形式，质因数分解形式为**

**n=p1^m1\*p2^m2\*p3^m3……**

**\* 这里 p1 < p2 < p3 < …… 为质数**

**\* 如果 mi = 1, 则 ^ mi 就不需要输出**

**示例输入：**

**6**

**7**

**示例输出：**

6=2^4\*3^2\*5

7=2^4\*3^2\*5\*7

**参考答案：**

**1.**

//只要设出经过x天出来即可列出方程(a-b)\*x+a>=h 化简得到:x>=(h-b)/a-b

#include<stdio.h>  
**int** main()  
{  
    **int** a,b,h,t1,t2;  
    **while**(scanf("%d%d%d",&a,&b,&h)==3)  
    {  
        t1=(h-b);  
        t2=(a-b);  
        **if**(t1%t2!=0)  
            printf("%d\n",t1/t2+1);  
        **else**  
            printf("%d\n",t1/t2);  
    }  
}

**2.**

/\*

其实是斐波那契数列,先来证明为啥那么是斐波那契。

假设第n次的时候答案是a(n)，n+1次为a(n+1)，假设n次的时候有a次向上b次向下，那么n+1时候就有a+b次向下，b次向上。n+2就有a+b向上，a+2b向下，所以a(n+2)=a(n)+a(n+1)

\*/

#include <iostream>  
#include <math.h>  
**using** **namespace** std;  
**int** F[1005];  
  
**int** main()  
{  
  
    **int** t,n;  
    cin>>t;  
    **while**(t--)  
    {  
        cin>>n;  
        F[0]=2;  
        F[1]=3;  
        **for** (**int** i=2;i<=n;i++)  
        {  
            F[i]=(F[(i-1)]+F[(i-2)]);  
        }  
        cout<<F[(n-1)]<<endl;  
          
    }  
  
    **return** 0;  
}

\*备注\*这题原题oj上的n范围是（1，1e9）很大，所以用快速幂解决的。

附代码：

#include<iostream>  
**using** **namespace** std;  
#include<stdio.h>  
#include<stdlib.h>  
#include<string.h>  
**typedef** **struct** node  
{  
    **int** a[3][3];  
}nodes;  
nodes mul(nodes x,nodes y)  
{  
    **int** i,j,k;  
    nodes temp;  
    temp.a[1][1]=0;  
    temp.a[1][2]=0;  
    temp.a[2][1]=0;  
    temp.a[2][2]=0;  
    **for**(i=1;i<3;i++)  
    {  
        **for**(j=1;j<3;j++)  
        {  
            **for**(k=1;k<3;k++)  
            {  
                temp.a[i][j]=(temp.a[i][j]+x.a[i][k]\*y.a[k][j])%10000;  
            }  
        }  
    }  
    **return** temp;  
}  
nodes powmul(nodes x,**int** num)  
{  
    nodes temp;  
    **if**(num==1)  
    {  
        **return** x;  
    }  
    **if**(num==2)  
    {  
        **return** mul(x,x);  
    }  
    **if**(num%2==0)  
    {  
        temp=powmul(x,num/2);  
        temp=mul(temp,temp);  
    }  
    **else**  
    {  
        temp=powmul(x,num/2);  
        temp=mul(temp,temp);  
        temp=mul(temp,x);  
    }  
    **return** temp;  
}  
**int** main(**void**)  
{  
    **int** t,n,j;  
    nodes temp;  
    scanf("%d",&t);  
    **while**(t--)  
    {  
        scanf("%d",&n);  
        **if**(n==1)  
        {  
            printf("2\n");  
            **continue**;  
        }  
        **if**(n==2)  
        {  
            printf("3\n");  
            **continue**;  
        }  
        temp.a[1][1]=0;  
        temp.a[1][2]=1;  
        temp.a[2][1]=1;  
        temp.a[2][2]=1;  
        temp=powmul(temp,n-1);  
        j=(2\*temp.a[1][1]+3\*temp.a[1][2])%10000;  
        printf("%d\n",j);  
    }  
    **return** 0;  
}

**3.**

//先打一张1000000以内的素数表，然后如果n%prime[i]==0则计算出prime[i]的次数用num数组记录输出

#include<iostream>  
#include<stdio.h>  
#include<math.h>  
  
**using** **namespace** std;  
  
  
**int** prime[1000005],num[1000005];  
  
  
**int** main()  
 {  
     **int** j,i,n,m,sum;

*//打印素数表*  
     **for**(i=2;i<=1000;i++)  
     {  
         **for**(j=2;i\*j<=1000000;j++)  
             prime[i\*j]=1;  
     }  
     **int** p=0;  
     **for**(i=2;i<=1000000;i++)  
     {  
         **if**(prime[i]==0)  
             prime[p++]=i;  
     }  
     prime[p]=0;  
*//    cout<<k<<endl;  
//    for(i=0;i<100;i++)  
//        cout<<prime[k-1]<<endl;*    **while**(scanf("%d",&n)==1 && n)  
     {  
         printf("%d=",n);  
         **for**(i=0; prime[i]<=n && prime[i]!=0 ;i++)  
         {  
             m=n;  
             sum=0;  
             **for**(j=0;m>0;)  
             {  
                 m/=prime[i] ;  
                 sum+=m;  
             }  
             num[i]=sum;  
               
        }  
           
         **if**(num[0]==1)  
             printf("%d",prime[0]);  
         **else**  
             printf("%d^%d",prime[0],num[0]);  
  
        **for**(j=1;j<i;j++)  
         {  
             **if**(num[j]==1)  
                 printf("\*%d",prime[j]);  
             **else**  
                 printf("\*%d^%d",prime[j],num[j]);  
         }  
         printf("\n");  
  
     }  
 }