**KMP算法总结**

**1.查找两个字符串匹配位置**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define N 100

void cal\_next( char \* str, int \* next, int len ){

int i, j;

next[0] = -1;

for( i = 1; i < len; i++ )

{

j = next[ i - 1 ];

while( str[ j + 1 ] != str[ i ] && ( j >= 0 ) )

{

j = next[ j ];

}

if( str[ i ] == str[ j + 1 ] )

{

next[ i ] = j + 1;

}

else

{

next[ i ] = -1;

}

}} **//计算next数组**

int KMP( char \* str, int slen, char \* ptr, int plen, int \* next ){

int s\_i = 0, p\_i = 0;

while( s\_i < slen && p\_i < plen )

{

if( str[ s\_i ] == ptr[ p\_i ] )

{

s\_i++;

p\_i++;

}

else

{

if( p\_i == 0 )

{

s\_i++;

}

else

{

p\_i = next[ p\_i - 1 ] + 1;

}

}

}

return ( p\_i == plen ) ? ( s\_i - plen ) : -1;}**//KMP算法**

int main(){

char str[ N ] = {0};

char ptr[ N ] = {0};

int slen, plen;

int next[ N ];

while( scanf( "%s%s", str, ptr ) )

{

slen = strlen( str );

plen = strlen( ptr );

cal\_next( ptr, next, plen );

printf( "%d\n", KMP( str, slen, ptr, plen, next ) );

}

return 0;

}**//主函数**

1. **下面给出一段计算next[]的代码：**

这样求出来的next数组其实是从下标1开始的，因为下标0之前是个空串，下标1则对应着M串的第0个字符。我们设next[0]=-1，仅仅是个标志而已，没有什么特殊的含义。

#include<string.h>

#include <iostream>

using namespace std;

void compute\_next(const string& pattern)

{

const int pattern\_length = pattern.size();

int \*next\_function = new int[pattern\_length];

int index;

next\_function[0] = -1;

for(int i=1;i<pattern\_length;++i)

{

index = next\_function[i-1];

//store previous fail position k to index;

while(index>=0 && pattern[i]!=pattern[index+1])

{

index = next\_function[index];

}

if(pattern[i]==pattern[index+1])

{

next\_function[i] = index + 1;

}

else

{

next\_function[i] = -1;

}

}

for(int i=0;i<pattern\_length;++i)

{

cout<<next\_function[i]<<endl;

}

delete[] next\_function;

}

int main()

{

string pattern = "abaabcaba";

compute\_next(pattern);

return 0;

}

**运行结果为：**

-1

-1

0

0

1

-1

0

1

2

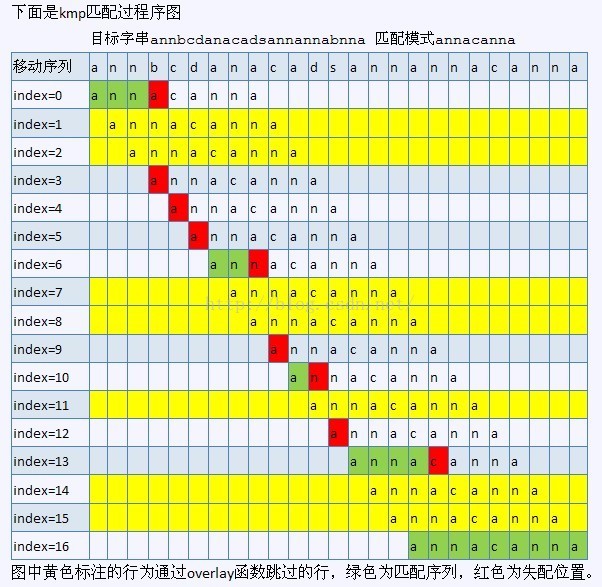
1. **改进型KMP算法:**
2. 若引入f(j)作为媒介，对 f(j) 和 next(j) 重新定义如下：

* f(j)是满足pattern[1...k - 1] = pattern[(j - (k - 1))...j -1](k < j)的k中，k的最大值
* next[j]是所有满足pattern[1...k - 1] = pattern[(j - (k - 1))...j -1](k < j)，且pattern[k] != pattern[j]的k中，k的最大值

根据定义，f(j)与next[j]的有如下递推公式：

* 如果 pattern[j] != pattern[f(j)]，next[j] = f(j);
* 如果 pattern[j] = pattern[f(j)]，next[j] = next[f(j)];

可以看出，本篇介绍的next(i) 其实就是f(j), 并不是最优跳转表。通过f(j)可以进一步计算最优的跳转表，最优跳转表对有多个重复字符的pattern[]，会表现出非常高的性能！



**B，Prefix Substring，给两个串a,b，问多少个a的子串是b的前缀。**

**去长春那天刚好讲了KMP。用next数组的性质水过。**

#include <iostream>

#include <stdio.h>

#include <string.h>

using namespace std;

typedef long long LL;

const int maxn = 1e5 + 5;

char a[maxn], b[maxn];

int next[maxn], f[maxn];

void prekmp(char \*b, int \*next)

{

memset(next,-1,sizeof next);

memset(f,0,sizeof f);

int j = -1;

for (int i=1; b[i]; i++)

{

while (j!=-1 && b[i]!=b[j+1])

j = next[j];

if (b[i] == b[j+1]) j++;

next[i] = j;

}

for (int i=0; b[i]; i++)

{

int tmp = i;

while (tmp != -1)

{

f[i]++;

tmp = next[tmp];

}

}

}

LL kmp(char \*a, char \*b, int \*next)

{

int j = -1;

LL ret = 0;

for (int i=0; a[i]; i++)

{

while (j!=-1 && a[i]!=b[j+1])

j = next[j];

if (a[i] == b[j+1]) ret += f[++j];

}

return ret;

}

int main()

{

scanf("%s%s",a,b);

prekmp(b, next);

printf("%lld\n",kmp(a,b,next));

return 0;

}

**1715: 字符串匹配**

Time Limit: 0 Sec  Memory Limit: 128 MB  
Submit: 27  Solved: 5  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1715)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1715)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1715)]

**Description**

给你两个串A，B，可以得到从A的任意位开始的子串和B匹配的长度。 给定K个询问，对于每个询问给定一个x，求出匹配长度恰为x的位置有多少个。 N，M，K< =200000

**Input**

第一行三个数  N，M，K，表示A的长度、B的长度和询问数。 第二行为串A。 第三行为串B。 接下来K行，每行1个数X。

**Output**

对于每个询问输出一个数。

**Sample Input**

6 2 2

aabcde

ab

0

2

**Sample Output**

4

1

**HINT**

**【分析】**

**想法是，用KMP计算出cnt[i]表示可以达到的长度大于等于i的匹配的个数。**

**那么长度为i的匹配的个数就是cnt[i] – cnt[i + 1]。因为KMP只能求到某个位置的最大匹配。**

**记a的每个位置的最大匹配长度是x。初始化cnt[i]为i这个最长匹配长度的频率。**

**如果a[i]所能达到的最大匹配是x，那么以a[i]为结尾长度为ne[x]的字符串也是一个匹配。因为ne[x] <= x。所以要在cnt[ne[x]]上加上cnt[x]。**

#include <stdio.h>

#define MAXN 200001

char a[MAXN],b[MAXN];

int ne[MAXN],cnt[MAXN];

int m,n,k,x,t;

int main() {

scanf("%d%d%d",&n,&m,&t);

scanf("%s%s",a,b);

for (int i = n;i > 0;--i)

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

for (int i = m;i > 0;--i)

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

int k = 0;

for (int i = 2;i <= m;++i)

{

while ((k > 0) && (b[k + 1] != b[i]))

k = ne[k];

if (b[k + 1] == b[i])

++k;

ne[i] = k;

}

k = 0;

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

{

while ((k > 0) && (b[k + 1] != a[i]))

k = ne[k];

if (b[k + 1] == a[i])

++k;

++cnt[k];

}

for(int i = n;i > 0;--i)

cnt[ne[i]] += cnt[i];

for(int i = 1;i <= t;++i) {

scanf("%d",&x);

printf("%d\n",cnt[x] - cnt[x + 1]);

}

return 0;

}

B-View Code

<http://blog.csdn.net/yearn520/article/details/6729426>

<http://blog.csdn.net/wangbaochu/article/details/50687160>

<http://blog.csdn.net/joylnwang/article/details/6778316#comments>

<http://www.tuicool.com/articles/e2Qbyyf>

-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

<https://www.wxz.name/2016/07/27/acmac/>

<http://www.cnblogs.com/zjutlitao/p/4337775.html>

<http://www.cnblogs.com/zichi/p/5173917.html>

<http://m.blog.csdn.net/article/details?id=47448599>

# Chessboard

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 1055    Accepted Submission(s): 447**

**Problem Description**

Consider the problem of tiling an n×n chessboard by polyomino pieces that are k×1 in size; Every one of the k pieces of each polyomino tile must align exactly with one of the chessboard squares. Your task is to figure out the maximum number of chessboard squares tiled.

**Input**

There are multiple test cases in the input file.  
First line contain the number of cases T (T≤10000).   
In the next T lines contain T cases , Each case has two integers n and k. (1≤n,k≤100)

**Output**

Print the maximum number of chessboard squares tiled.

**Sample Input**

2

6 3

5 3

**Sample Output**

36

24

**用 k × 1 的小矩形覆盖一个 n × n 的正方形棋盘，问正方形棋盘最多能被覆盖多少。**

**规律就是：如果n<k，肯定不行。**

**定义mod=n%k;**

**如果(mod<=k/2),结果为：n\*n-mod\*mod；**

**否则结果为：n\*n-(k-mod)\*(k-mod)；**

#include<stdio.h>

int main()

{

int t,n,k;

scanf("%d",&t);

while(t--)

{

scanf("%d%d",&n,&k);

if(n<k){printf("0\n");continue;}

int mod=n%k;

if(mod<=k/2)printf("%d\n",n\*n-mod\*mod);

else printf("%d\n",n\*n-(k-mod)\*(k-mod));

}

return 0;

}

**费马小定理**（Fermat Theory）是数论中的一个重要定理，其内容为： 假如p是[质数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=67850&ss_c=ssc.citiao.link" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)，且Gcd(a,p)=1，那么 a(p-1) ≡1（mod p）。即：假如a是整数，p是质数，且a,p互质(即两者只有一个[公约数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7531927&ss_c=ssc.citiao.link" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)1)，那么a的(p-1)[次方](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=73782511&ss_c=ssc.citiao.link" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)除以p的[余数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=795886&ss_c=ssc.citiao.link" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)恒等于1。该定理是1636年皮埃尔·德·费马发现的。

**所以我们要做的就是把这一题与费马小定理联系起来！**

**f(x)=5\*x^13+13\*x^5+k\*a\*x转化成f(x)=(5\*x^12+13\*x^4+k\*a)\*x**

**1.当x=65的倍数时就行了；**

**2.当x=5的倍数时则需要（5\*x^12+k\*a）是13的倍数，由费马小定理可知，因为x是5的倍数所以（x^(13-1)%13==1）,所以（5\*x^12%13==5），所以只需要做到k\*a%13==8即可！**

**3.同理可得当x=13的倍数时，只需要做到（13\*x^4+k\*a)是5的倍数由费马小定理可知，因为x是5的倍数所以（x^(5-1)%5==1）,所以（13\*x^(5-1)%5==3），所以只需要做到k\*a%5==2即可**

**4.当x不是上面的特殊数时，则需要f(x)=(5\*x^12+13\*x^4+k\*a)\*x被65整除，也就是需要同时满足上面那两个条件！**

# Ignatius's puzzle

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)Total Problem Description**

Ignatius is poor at math,he falls across a puzzle problem,so he has no choice but to appeal to Eddy. this problem describes that:f(x)=5\*x^13+13\*x^5+k\*a\*x,input a nonegative integer k(k<10000),to find the minimal nonegative integer a,make the arbitrary integer x ,65|f(x)if  
no exists that a,then print "no".

**Input**

The input contains several test cases. Each test case consists of a nonegative integer k, More details in the Sample Input.

**Output**

The output contains a string "no",if you can't find a,or you should output a line contains the a.More details in the Sample Output.

**Sample Input**

11

100

9999

**Sample Output**

22

no

43

#include <stdio.h>

int main()

{

int k,flag;

while (scanf("%d",&k)!=EOF)

{

flag=0;

for (int i=1;i<=65;i++)

{

if (i\*k%13==8&&i\*k%5==2)

{

flag=i;

break;

}

}

if (flag)

{

printf("%d\n",flag);

}

else

{

printf("no\n");

}

}

}

大数相加模板

#include<iostream>

#include<string>

using namespace std;

int main()

{

int n,i;

scanf("%d",&n);

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

{

string sum(string s1,string s2);

string s1,s2;

cin>>s1>>s2;

printf("Case %d:\n",i);

cout<<s1<<" + "<<s2<<" = "<<sum(s1,s2);

printf("\n\n");

}

return 0;

}

string sum(string s1,string s2)

{

if(s1.length()<s2.length())

{

string temp=s1;

s1=s2;

s2=temp;

}

for(int i=s1.length()-1,j=s2.length ()-1;i>=0;i--,j--)

{

s1[i]=char(s1[i]+(j>=0?s2[j]-'0':0));

if(s1[i]-'0'>=10)

{

s1[i]=char((s1[i]-'0')%10+'0');

if(i)

s1[i-1]++;

else

s1='1'+s1;

}

}

return s1;

}

大数阶乘

代码如下：

#include<iostream>

#include<cmath>

using namespace std;

int main()

{

void factorial(int n) ;

int n;

cin>>n;

factorial(n);

return 0;

}

void factorial(int n)

{

long a[10000];

int i,j,l,c,m=0,w;

a[0]=1;

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

{

c=0;

for(j=0;j<=m;j++)

{

a[j]=a[j]\*i+c;

c=a[j]/10000;

a[j]=a[j]%10000;

}

if(c>0)

{

m++;

a[m]=c;

}

}

cout<<a[m];

for(i=m-1;i>=0;i--)

cout<<a[i];

cout<<endl;

}

**[Jerry Lee’s Stones(动态规划（最长不重复子串）+Hash)](http://blog.csdn.net/baidu_23955875/article/details/47206805)**

#include<stdio.h>

#include<string.h>

**int** str[10000000];

**int** max(**int** a,**int** b)

{

**if**(a<b)

**return** b;

**else**

**return** a;

}

**int** solve(**int** \*str,**int** n)

{

**int** last=0,glo=0,loc=0;

**int** Hash[10000000]={0};

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

    {

**int** k = str[i]-0;

         Hash[k] ++;

**if**(Hash[k] == 2)

         {

            glo = max(glo, loc);

**for**(; last < i && str[last] != str[i]; last++)

                Hash[str[last]-0]--;

            loc = i - last;

            Hash[k]--;

            last++;

         }

**else**

         loc ++;

    }

**return** max(glo, loc);

}

**int** main()

{

**int** t,n,i,a;

**scanf**("%d",&t);

**while**(t--)

    {

**scanf**("%d",&n);

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

**scanf**("%d",&str[i]);

**printf**("%d\n",solve(str,n));

    }

}

**康托展开：**

X=an\*(n-1)!+an-1\*(n-2)!+...+ai\*(i-1)!+...+a2\*1!+a1\*0!

ai为整数，并且0<=ai<i(1<=i<=n)

应用实例：

{1,2,3,4,...,n}的排列总共有n!种，将它们从小到大排序，怎样知道其中一种排列是有序序列中的第几个？

如 {1,2,3} 按从小到大排列一共6个：123 132 213 231 312 321。想知道321是{1,2,3}中第几个大的数。

这样考虑：第一位是3，小于3的数有1、2 。所以有2\*2!个。再看小于第二位，小于2的数只有一个就是1 ，所以有1\*1!=1 所以小于32

的{1,2,3}排列数有2\*2!+1\*1!=5个。所以321是第6个大的数。2\*2!+1\*1!是康托展开。

再举个例子：1324是{1,2,3,4}排列数中第几个大的数：第一位是1小于1的数没有，是0个，0\*3!，第二位是3小于3的数有1和2，但1已经在第一位了，所以只有一个数2，1\*2! 。第三位是2小于2的数是1，但1在第一位，所以有0个数，0\*1!，所以比1324小的排列有0\*3!+1\*2!+0\*1!=2个，1324是第三个大数。

int fac[] = {1,1,2,6,24,120,720,5040,40320}; //i的阶乘为fac[i]

/\* 康托展开.

{1...n}的全排列由小到大有序，s[]为第几个数 \*/

int KT(int n, int s[])

{

int i, j, t, sum;

sum = 0;

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

{

t = 0;

for (j=i+1; j<n; j++)

if (s[j] < s[i])

t++;

sum += t\*fac[n-i-1];

}

return sum+1;

}

康托展开的逆运算：

{1,2,3,4,5}的全排列已经从小到大排序，要找出第16个数：

1. 首先用16-1得到15

2. 用15去除4! 得到0余15

3. 用15去除3! 得到2余3

4. 用3去除2! 得到1余1

5. 用1去除1! 得到1余0

有0个数比它小的数是1

所以第一位是1

有2个数比它小的数是3，但1已经在之前出现过了所以是4

有1个数比它小的数是2，但1已经在之前出现过了所以是3

有1个数比它小的数是2，但1,3,4都出现过了所以是5

最后一个数只能是2

所以这个数是14352

**/\* 康托展开的逆运算.**

**{1...n}的全排列，中的第k个数为s[] \*/**

**void invKT(int n, int k, int s[])**

**{**

**int i, j, t, vst[8]={0};**

**k--;**

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

**{**

**t = k/fac[n-i-1];**

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

**if (!vst[j])**

**{**

**if (t == 0) break;**

**t--;**

**}**

**s[i] = j;**

**vst[j] = 1;**

**k %= fac[n-i-1];**

**}**

**}**

**Power of Cryptography**

**2 16**

**3 27**

**7 4357186184021382204544**

**求b的1/a次幂是多少：**

**#include<cstdio>**

**#include<cmath>**

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**double a,b;**

**while(scanf("%lf%lf",&a,&b)!=EOF)**

**{**

**printf("%.0lf\n",pow(b,1.0/a));**

**// cout<<pow(b,1.0/a)<<endl;**

**}**

**}**

**凯莱定理：**

**定理的另一种表述**

**过n个有标志顶点的[树](http://baike.baidu.com/view/143352.htm" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)的数目等于n^(n-2)。**

**[3]****定理的理解**

此定理说明用n-1条边将n个一致的顶点连接起来的**[连通图](http://baike.baidu.com/view/3148644.htm" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)**的个数为**n^(n-2)**，也可以这样[理解](http://baike.baidu.com/view/58766.htm" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)，将n个城市连接起来的树状公路网络有**n^(n-2)**种[方案](http://baike.baidu.com/view/556358.htm" \t "http://blog.csdn.net/baidu_23955875/article/details/_blank)。所谓树状，指的是用n-1条边将n个顶点构成一个连通图。当然，建造一个树状的公路网络将n个城市连接起来，应求其中长度最短、造价最省的一种，或效益最大的一种。Cayley定理只是说明可能方案的数目。

星际之门（一）

时间限制：3000 ms  |  内存限制：65535 KB

难度：3

描述

公元3000年，子虚帝国统领着N个星系，原先它们是靠近光束飞船来进行旅行的，近来，X博士发明了星际之门，它利用虫洞技术，一条虫洞可以连通任意的两个星系，使人们不必再待待便可立刻到达目的地。

帝国皇帝认为这种发明很给力，决定用星际之门把自己统治的各个星系连结在一起。

可以证明，修建N-1条虫洞就可以把这N个星系连结起来。

现在，问题来了，皇帝想知道有多少种修建方案可以把这N个星系用N-1条虫洞连结起来？

输入

第一行输入一个整数T,表示测试数据的组数(T<=100)  
每组测试数据只有一行，该行只有一个整数N，表示有N个星系。(2<=N<=1000000)

输出

对于每组测试数据输出一个整数，表示满足题意的修建的方案的个数。输出结果可能很大，请输出修建方案数对10003取余之后的结果。

样例输入

2

3

4

样例输出

3

16

**#include<cstdio>**

**#include<iostream>**

**#include<cmath>**

**#define ll long long**

**using namespace std;**

**ll pow\_mod(ll a,ll n,ll m)**

**{**

**ll ans = 1;**

**a = a % m;**

**while(n>0)**

**{**

**if(n%2 ==1)**

**ans=(ans\*a)%m;**

**n=n/2;**

**a=(a\*a) % m;**

**}**

**return ans;**

**}**

**int main()**

**{**

**long long n;**

**int t;**

**cin>>t;**

**while(t--)**

**{**

**cin>>n;**

**cout<<pow\_mod(n,n-2,10003)<<endl;**

**}**

**}**

# [Just a Numble（求小数的第N位数）](http://blog.csdn.net/baidu_23955875/article/details/42236681)

# Just a Numble

**Time Limit: 3000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 2410    Accepted Submission(s): 1160**

**Problem Description**

Now give you two integers n m, you just tell me the m-th number after radix point in 1/n,for example n=4,the first numble after point is 2,the second is 5,and all 0 followed

**Input**

Each line of input will contain a pair of integers for n and m(1<=n<=10^7,1<=m<=10^5)

**Output**

For each line of input, your program should print a numble on a line,according to the above rules

**Sample Input**

4 2

5 7

123 123

**Sample Output**

5

0

8

**#include<stdio.h>**

**int main()**

**{**

**int n,m,i,j,sum;**

**int k; while(scanf("%d%d",&n,&m)!=EOF)**

**{**

**sum=1;**

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

**{**

**sum=sum\*10;//这个题的意思是让每一位都X10，比如求得第一位**

**k=sum/n;//k即为第一位小数**

**sum=sum%n;//求得余数，还得让余数继续除以n也很好理解！**

**}**

**printf("%d\n",k%10);**

**}**

**}**

**[异或详解](http://blog.csdn.net/baidu_23955875/article/details/42124685)**

**异或门符号'^',在编程中的使用，有的时候能够让你的程序更加精炼简捷，  
尤其在C++的acm程序应用中，可以避免许多麻烦！！！它的几个简单作用  
讲解如下：  
它的简单规则是相同为0，不同为1，例如  
int a=3=011（2进制，在计算机中的存储形式）；  
int b=6=110（2进制，在计算机中的存储形式）；  
int c=a^b=101=5;  
切忌，在异或门的计算中都是应用的2进制！！  
任何数和0异或都等于它本身;两个相同的数异或后的结果是0；  
1）实现两个数的交换  
 　a=a^b; 　　  
   b=a^b;//b=a^b^b=a^0=a; 　　  
   a=a^b;//a=a^a^b=0^b=b;  
应用例子如下：  
#include<iostream>  
using namespace std;  
int main()  
{  
 int a=3;  
 int b=4;  
 a=a^b;  
 b=a^b;  
 a=a^b;  
 cout<<a<<' '<<b<<endl;  
 return 0;  
}  
2在一排数中找到独一无二的一个数  
例子如下：  
他的思路是;  
只要有相同的数那么他们异或后的结果就是0，最后就剩下0和那个独一无二的数相异或，就等于  
那个独一无二的数了！！哈哈！！例子，如下：  
#include<iostream>  
using namespace std;  
int main()  
{  
    int n,i,x,y;  
    while(scanf("%d",&n)!=EOF&&n)  
    {  
           x=0;  
           while(n--)  
           {  
               cin>>y;  
                x^=y;      
           }          
           printf("%d\n",x);              
    }  
    return 0;  
}  
随便输入几个数，找到其中没有和他相同的那个数！！哈！  
3）判断两个数是否相等  
if(a^b==0)  
printf("a=b");  
else   
printf("a!=b");**

**auto函数：**

#include <stdlib.h>

#include <stdio.h>

int main(void)

{

    int n;

    char \*str = "12345.67";

    n = atoi(str);

    printf("int=%d\n",n);

    return 0;

}

输出：

int = 12345

|  |
| --- |
| //vs2013里调用printf函数请使用预处理命令#define \_CRT\_SECURE\_NO\_WARNINGS  #include <stdlib.h>  #include <stdio.h>    int main()  {      char a[] = "-100";      char b[] = "123";      int c;      c = atoi(a) + atoi(b);      printf("c=%d\n", c);      return 0;  } |

执行结果：

c = 23

**[RMQ (Range Minimum/Maximum Query)算法](http://blog.csdn.net/baidu_23955875/article/details/44901357)**

本文章已收录于：

RMQ（Range Minimum/Maximum Query），即区间最值查询，是指这样一个问题：对于长度为n的数列A，回答若干询问RMQ（A,i,j）(i,j<=n)，返回数列A中下标在i，j之间的最小/大值。这两个问题是在实际应用中经常遇到的问题，下面介绍一下解决这两种问题的比较高效的算法。当然，该问题也可以用线段树（也叫区间树）解决，算法复杂度为：O(N)~O(logN)，这里我们暂不介绍。

**2.RMQ算法**

对于该问题，最容易想到的解决方案是遍历，复杂度是O(n)。但当数据量非常大且查询很频繁时，该算法无法在有效的时间内查询出正解。

本节介绍了一种比较高效的在线算法（ST算法）解决这个问题。所谓在线算法，是指用户每输入一个查询便马上处理一个查询。该算法一般用较长的时间做预处理，待信息充足以后便可以用较少的时间回答每个查询。ST（Sparse Table）算法是一个非常有名的在线处理RMQ问题的算法，它可以在O(nlogn)时间内进行预处理，然后在O(1)时间内回答每个查询。

**（一）首先是预处理，用动态规划（DP）解决。**

设A[i]是要求区间最值的数列，F[i, j]表示从第i个数起连续2^j个数中的最大值。（DP的状态）

例如：

A数列为：3 2 4 5 6 8 1 2 9 7

F[1，0]表示第1个数起，长度为2^0=1的最大值，其实就是3这个数。同理 F[1,1] = max(3,2) = 3, F[1，2]=max(3,2,4,5) = 5，F[1，3] = max(3,2,4,5,6,8,1,2) = 8;

并且我们可以容易的看出F[i,0]就等于A[i]。（DP的初始值）

这样，DP的状态、初值都已经有了，剩下的就是状态转移方程。

我们把F[i，j]平均分成两段（因为f[i，j]一定是偶数个数字），从 i 到i + 2 ^ (j - 1) - 1为一段，i + 2 ^ (j - 1)到i + 2 ^ j - 1为一段(长度都为2 ^ (j - 1))。用上例说明，当i=1，j=3时就是3,2,4,5 和 6,8,1,2这两段。F[i，j]就是这两段各自最大值中的最大值。于是我们得到了状态转移方程F[i, j]=max（F[i，j-1], F[i + 2^(j-1)，j-1]）。

代码如下：

**[cpp]** [view plain](http://blog.csdn.net/niushuai666/article/details/6624672" \t "_blank" \o "view plain)[copy](http://blog.csdn.net/niushuai666/article/details/6624672" \t "_blank" \o "copy)

1. **void** RMQ(**int** num) //预处理->O(nlogn)
2. {
3. **for**(**int** j = 1; j < 20; ++j)
4. **for**(**int** i = 1; i <= num; ++i)
5. **if**(i + (1 << j) - 1 <= num)
6. {
7. maxsum[i][j] = max(maxsum[i][j - 1], maxsum[i + (1 << (j - 1))][j - 1]);
8. minsum[i][j] = min(minsum[i][j - 1], minsum[i + (1 << (j - 1))][j - 1]);
9. }
10. }

[http://static.blog.csdn.net/images/save_snippets.png](javascript:;)

这里我们需要注意的是循环的顺序，我们发现外层是j，内层所i，这是为什么呢？可以是i在外，j在内吗？

答案是不可以。因为我们需要理解这个状态转移方程的意义。

状态转移方程的含义是：先更新所有长度为F[i,0]即1个元素，然后通过2个1个元素的最值，获得所有长度为F[i,1]即2个元素的最值，然后再通过2个2个元素的最值，获得所有长度为F[i,2]即4个元素的最值，以此类推更新所有长度的最值。

而如果是i在外，j在内的话，我们更新的顺序就是F[1,0],F[1,1],F[1,2],F[1,3],表示更新从1开始1个元素，2个元素，4个元素，8个元素（A[0],A[1],....A[7]）的最值，这里F[1,3] = max(max(A[0],A[1],A[2],A[3]),max(A[4],A[5],A[6],A[7]))的值，但是我们根本没有计算max(A[0],A[1],A[2],A[3])和max(A[4],A[5],A[6],A[7])，所以这样的方法肯定是错误的。

为了避免这样的错误，一定要好好理解这个状态转移方程所代表的含义。

**（二）然后是查询。**

假如我们需要查询的区间为(i,j)，那么我们需要找到覆盖这个闭区间(左边界取i，右边界取j)的最小幂（可以重复，比如查询5，6，7，8，9，我们可以查询5678和6789）。

因为这个区间的长度为j - i + 1,所以我们可以取k=log2( j - i + 1)，则有：RMQ(A, i, j)=max{F[i , k], F[ j - 2 ^ k + 1, k]}。

举例说明，要求区间[2，8]的最大值，k = log2（8 - 2 + 1）= 2，即求max(F[2, 2]，F[8 - 2 ^ 2 + 1, 2]) = max(F[2, 2]，F[5, 2])；

在这里我们也需要注意一个地方，就是<<运算符和+-运算符的优先级。

比如这个表达式：5 - 1 << 2是多少？

答案是：4 \* 2 \* 2 = 16。所以我们要写成5 - (1 << 2)才是5-1 \* 2 \* 2 = 1。

## 1735: 小明的账目

Time Limit: 10 Sec  Memory Limit: 1280 MB  
Submit: 46  Solved: 6  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1735)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1735)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1735)]

## Description

小明是一个聪明能干的人。他工作的10年中，为了让账目清楚，每天都要记k次账。小明的上司将每次的账目按1，2，3…编号，然后不定时的问小明问题，问题是这样的：在a到b号账中最少的一笔是多少？上司总是一次问多个问题。请你帮小明来回答。

## Input

输入中第一行有两个数m,n表示有m(m< =100000)笔账,n表示有n个问题，n< =100000。第二行为m个数,分别是账目的钱数后面n行分别是n个问题,每行有2个数字说明开始结束的账目编号。

## Output

输出为每个问题的答案。具体查看样例。

## Sample Input

10 3

1 2 3 4 5 6 7 8 9 10

2 7

3 9

1 10

## Sample Output

2 3 1

## HINT

**#include<stdio.h>**

**#include<string.h>**

**#include<string>**

**#include<stack>**

**#include<queue>**

**#include<math.h>**

**#include<limits.h>**

**#include<iostream>**

**#include<algorithm>**

**using namespace std;**

**const int N=100002;**

**int minsum[20][N];**

**void RMQ(int num)//RMQ算法**

**{**

**for(int i=1;i!=20;++i)//动态规划思想预处理**

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

**if(j+(1<<i)-1<=num)**

**minsum[i][j]=min(minsum[i-1][j],minsum[i-1][j+(1<<i>>1)]);**

**}**

**int main()**

**{**

**int m,n,i,l,r;**

**scanf("%d%d",&m,&n);**

**for(i=1;i<=m;i++)**

**scanf("%d",&minsum[0][i]);**

**RMQ(m);**

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

**{**

**scanf("%d%d",&l,&r);**

**int k=(int)(log(r-l+1.0)/log(2.0));//找到被2的n次方覆盖的区间**

**int minres=min(minsum[k][l],minsum[k][r-(1<<k)+1]);**

**if(i==n-1)**

**printf("%d\n",minres);**

**else**

**printf("%d ",minres);**

**}**

**return 0;**

**}**

士兵杀敌（三）   
时间限制：2000 ms | 内存限制：65535 KB   
难度：5   
描述   
南将军统率着N个士兵，士兵分别编号为1~N,南将军经常爱拿某一段编号内杀敌数最高的人与杀敌数最低的人进行比较，计算出两个人的杀敌数差值，用这种方法一方面能鼓舞杀敌数高的人，另一方面也算是批评杀敌数低的人，起到了很好的效果。

所以，南将军经常问军师小工第i号士兵到第j号士兵中，杀敌数最高的人与杀敌数最低的人之间军功差值是多少。

现在，请你写一个程序，帮小工回答南将军每次的询问吧。

注意，南将军可能询问很多次。

输入   
只有一组测试数据   
第一行是两个整数N,Q，其中N表示士兵的总数。Q表示南将军询问的次数。(1<N<=100000,1<Q<=1000000)   
随后的一行有N个整数Vi(0<=Vi<100000000)，分别表示每个人的杀敌数。   
再之后的Q行，每行有两个正正数m,n，表示南将军询问的是第m号士兵到第n号士兵。   
输出   
对于每次询问，输出第m号士兵到第n号士兵之间所有士兵杀敌数的最大值与最小值的差。   
样例输入   
5 2   
1 2 6 9 3   
1 2   
2 4   
样例输出   
1   
7

RMQ算法的水题，当然也可以用线段树，等会贴出来RMQ算法的总结，我要先去上（shui）课（jiao）了。

代码贴上，解析晚上写

#include<stdio.h>

#include<string.h>

#include<string>

#include<stack>

#include<queue>

#include<math.h>

#include<limits.h>

#include<iostream>

#include<algorithm>

using namespace std;

const int N=100010;

int maxsum[20][N],minsum[20][N];

void RMQ(int num)//RMQ算法

{

for(int i=1;i!=20;++i)//动态规划思想预处理

for(int j=1;j<=num;++j)

if(j+(1<<i)-1<=num)

{//位运算用多了会不会很难懂

maxsum[i][j]=max(maxsum[i-1][j],maxsum[i-1][j+(1<<i>>1)]);

minsum[i][j]=min(minsum[i-1][j],minsum[i-1][j+(1<<i>>1)]);

}

}

int main()

{

int num,query\_num;//士兵数量，询问次数

int m,n;//询问区间

scanf("%d%d",&num,&query\_num);

for(int i=1;i<=num;++i) //输入信息处理

{

scanf("%d",&maxsum[0][i]);

minsum[0][i]=maxsum[0][i];

}

RMQ(num);

while(query\_num--)//直接查询，此时时间复杂度为O(1)!!!

{

scanf("%d%d",&m,&n);

int k=(int)(log(n-m+1.0)/log(2.0));//找到被2的n次方覆盖的区间

int maxres=max(maxsum[k][m],maxsum[k][n-(1<<k)+1]);

int minres=min(minsum[k][m],minsum[k][n-(1<<k)+1]);

printf("%d\n", maxres-minres);

}

return 0;

}

**线段树**

**主要应用**

(1)：区间最值查询问题（见模板1）同 **RMQ**

(2)：连续区间修改或者单节点更新的动态查询问题 （见模板2）

(3)：多维空间的动态查询 （见模板3）

**Problem Description（模板2**连续区间修改或者单节点更新的动态查询问题**）**

C国的死对头A国这段时间正在进行军事演习，所以C国间谍头子Derek和他手下Tidy又开始忙乎了。A国在海岸线沿直线布置了N个工兵营 地,Derek和Tidy的任务就是要监视这些工兵营地的活动情况。由于采取了某种先进的监测手段，所以每个工兵营地的人数C国都掌握的一清二楚,每个工 兵营地的人数都有可能发生变动，可能增加或减少若干人手,但这些都逃不过C国的监视。  
中央情报局要研究敌人究竟演习什么战术,所以Tidy要随时向Derek汇报某一段连续的工兵营地一共有多少人,例如Derek问:“Tidy,马上汇报 第3个营地到第10个营地共有多少人!”Tidy就要马上开始计算这一段的总人数并汇报。但敌兵营地的人数经常变动，而Derek每次询问的段都不一样， 所以Tidy不得不每次都一个一个营地的去数，很快就精疲力尽了，Derek对Tidy的计算速度越来越不满:"你个死肥仔，算得这么慢，我炒你鱿 鱼!”Tidy想：“你自己来算算看，这可真是一项累人的工作!我恨不得你炒我鱿鱼呢!”无奈之下，Tidy只好打电话向计算机专家 Windbreaker求救,Windbreaker说：“死肥仔，叫你平时做多点acm题和看多点算法书，现在尝到苦果了吧!”Tidy说："我知错 了。。。"但Windbreaker已经挂掉电话了。Tidy很苦恼，这么算他真的会崩溃的，聪明的读者，你能写个程序帮他完成这项工作吗？不过如果你的 程序效率不够高的话，Tidy还是会受到Derek的责骂的.

**Input**

第一行一个整数T，表示有T组数据。  
每组数据第一行一个正整数N（N<=50000）,表示敌人有N个工兵营地，接下来有N个正整数,第i个正整数ai代表第i个工兵营地里开始时有ai个人（1<=ai<=50）。  
接下来每行有一条命令，命令有4种形式：  
(1) Add i j,i和j为正整数,表示第i个营地增加j个人（j不超过30）  
(2)Sub i j ,i和j为正整数,表示第i个营地减少j个人（j不超过30）;  
(3)Query i j ,i和j为正整数,i<=j，表示询问第i到第j个营地的总人数;  
(4)End 表示结束，这条命令在每组数据最后出现;  
每组数据最多有40000条命令

**Output**

对第i组数据,首先输出“Case i:”和回车,  
对于每个Query询问，输出一个整数并回车,表示询问的段中的总人数,这个数保持在int以内。

**Sample Input**

1

10

1 2 3 4 5 6 7 8 9 10

Query 1 3

Add 3 6

Query 2 7

Sub 10 2

Add 6 3

Query 3 10

End

**Sample Output**

Case 1:

6

33

59

思路：这道题目考查的是对区间的一个动态操作，所以需要用到数据结构中的线段树，本题的难点便在于线段树的构建。

#include"iostream"

#include"cstdlib"

using namespace std;

#define maxn 50010//指定所要构造的线段树节点的最大值

struct Node

{

int sum;

int l, r;

}node[maxn\*4];

int num[maxn];

int ans;

void build\_tree(int rt, int ll, int rr)

{

if (ll == rr)

{

node[rt].l = ll;

node[rt].r = rr;

node[rt].sum = num[ll];

return;

}

node[rt].l = ll;

node[rt].r = rr;

int mid = (ll + rr) / 2;

build\_tree(rt \* 2, ll, mid);

build\_tree(rt \* 2 + 1, mid + 1, rr);

node[rt].sum = node[rt \* 2].sum + node[rt \* 2 + 1].sum;//sum值更新到父亲节点

}

void update(int rt, int pos, int value)

{

if (node[rt].l == pos&&node[rt].r == pos)

{

node[rt].sum += value;

return;

}

int mid = (node[rt].l + node[rt].r) / 2;

if (pos <= mid)

{

update(rt \* 2, pos, value);

}

else

{

update(rt \* 2 + 1, pos, value);

}

node[rt].sum = node[rt \* 2].sum + node[rt \* 2 + 1].sum;

}

void query(int rt, int l, int r)

{

int mid = (node[rt].l + node[rt].r) / 2;

if (node[rt].l == l&&node[rt].r == r)

{

ans += node[rt].sum;

}

else if (l > mid){

query(rt \* 2 + 1, l, r);

}

else if (mid >= r){

query(rt \* 2, l, r);

}

else{

query(rt \* 2, l, mid);

query(rt \* 2 + 1, mid + 1, r);

}

}

int main()

{

int t,n;

int u = 0;

cin >> t;

char str[12];

while (t--)

{

cin >> n;

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

{

cin >> num[i];

}

build\_tree(1, 1, n);

cout << "Case " << ++u <<":"<< endl;

while (cin>>str)

{

int x, y;

cin >> x >> y;

if (str[0] == 'E'){

break;

}

if (str[0] == 'Q')

{

ans = 0;

query(1, x, y);

cout << ans << endl;

}

else if (str[0] == 'A')

{

update(1, x, y);

}

else if (str[0] == 'S')

{

update(1,x,-y);

}

}

}

return 0;

}

**字典树**

#include <iostream>

using namespace std;

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

typedef struct Trie{

int v;

Trie \*next[26];

}Trie;

Trie root;

void createTrie(char \*str)

{

int len = strlen(str);

Trie \*p = &root, \*q;

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

{

int id = str[i]-'a';

if(p->next[id] == NULL)

{

q = (Trie \*)malloc(sizeof(root));

q->v = 1;

for(int j=0; j<26; ++j)

q->next[j] = NULL;

p->next[id] = q;

p = p->next[id];

}

else

{

p->next[id]->v++;

p = p->next[id];

}

}

}

int findTrie(char \*str)

{

int len = strlen(str);

Trie \*p = &root;

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

{

int id = str[i]-'a';

p = p->next[id];

if(p == NULL)

return 0;

}

return p->v;

}

int main()

{

//freopen("input.txt", "r", stdin);

char str[15];

int i;

for(i=0; i<26; ++i)

root.next[i] = NULL;

while(gets(str) && str[0]!='\0')

createTrie(str);

memset(str, 0, sizeof(str));

while(scanf("%s", str) != EOF)

{

int ans = findTrie(str);

printf("%d\n", ans);

}

return 0;

}

**模板：**

**#define MAX 26//字符集大小**

**typedef struct TrieNode**

**{**

**int nCount;//记录该字符出现次数**

**struct TrieNode\* next[MAX];**

**}TrieNode;**

**TrieNode Memory[1000000];**

**int allocp=0;**

**/\*初始化\*/**

**void InitTrieRoot(TrieNode\* \*pRoot)**

**{**

**\*pRoot=NULL;**

**}**

**/\*创建新结点\*/**

**TrieNode\* CreateTrieNode()**

**{**

**int i;**

**TrieNode \*p;**

**p=&Memory[allocp++];**

**p->nCount=1;**

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

**{**

**p->next[i]=NULL;**

**}**

**return p;**

**}**

**/\*插入\*/**

**void InsertTrie(TrieNode\* \*pRoot,char \*s)**

**{**

**inti,k;**

**TrieNode\*p;**

**if(!(p=\*pRoot))**

**{**

**p=\*pRoot=CreateTrieNode();**

**}**

**i=0;**

**while(s[i])**

**{**

**k=s[i++]-'a';//确定branch**

**if(!p->next[k])**

**p->next[k]=CreateTrieNode();**

**p->next[k]->nCount++;**

**p=p->next[k];**

**}**

**}**

**//查找**

**int SearchTrie(TrieNode\* \*pRoot,char \*s)**

**{**

**TrieNode \*p;**

**int i,k;**

**if(!(p=\*pRoot))**

**{**

**return0;**

**}**

**i=0;**

**while(s[i])**

**{**

**k=s[i++]-'a';**

**if(p->next[k]==NULL) return 0;**

**p=p->next[k];**

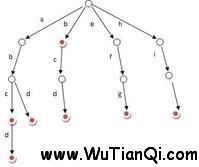
**}**

**return p->nCount;**

**}**

## [字典树（讲解+模版）](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/1833717.html)

又称**单词查找树**，[Trie树](http://www.wutianqi.com/?p=1359)，是一种树形结构，是一种哈希树的变种。典型应用是用于统计，排序和保存大量的字符串（但不仅限于字符串），所以经常被搜索引擎系统用于文本词频统计。它的优点是：利用字符串的公共前缀来节约存储空间，最大限度地减少无谓的字符串比较，查询效率比哈希表高。



字典树与字典很相似,当你要查一个单词是不是在字典树中,首先看单词的第一个字母是不是在字典的第一层,如果不在,说明字典树里没有该单词,如果在 就在该字母的孩子节点里找是不是有单词的第二个字母,没有说明没有该单词,有的话用同样的方法继续查找.字典树不仅可以用来储存字母,也可以储存数字等其 它数据。

**Trie的数据结构定义：**

[IMG_257](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

#define MAX 26  
typedef struct Trie     
{     
    Trie \*next[MAX];     
    int v;   //根据需要变化  
};     
   
Trie \*root;

[IMG_258](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

next是表示每层有多少种类的数，如果只是小写字母，则26即可，若改为大小写字母，则是52，若再加上数字，则是62了，这里根据题意来确定。  
v可以表示一个字典树到此有多少相同前缀的数目，这里根据需要应当学会自由变化。

**Trie的查找（最主要的操作）：**  
(1) 每次从根结点开始一次搜索；  
(2) 取得要查找关键词的第一个字母，并根据该字母选择对应的子树并转到该子树继续进行检索； 　　(3) 在相应的子树上，取得要查找关键词的第二个字母,并进一步选择对应的子树进行检索。 　　  
(4) 迭代过程…… 　　  
(5) 在某个结点处，关键词的所有字母已被取出，则读取附在该结点上的信息，即完成查找。

这里给出生成字典树和查找的**模版**：  
生成字典树：

[IMG_259](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

void createTrie(char \*str)  
{  
    int len = strlen(str);  
    Trie \*p = root, \*q;  
    for(int i=0; i<len; ++i)  
    {  
        int id = str[i]-'0';  
        if(p->next[id] == NULL)  
        {  
            q = (Trie \*)malloc(sizeof(Trie));  
            q->v = 1;    //初始v==1  
            for(int j=0; j<MAX; ++j)  
                q->next[j] = NULL;  
            p->next[id] = q;  
            p = p->next[id];  
        }  
        else  
        {  
            p->next[id]->v++;  
            p = p->next[id];  
        }  
    }  
    p->v = -1;   //若为结尾，则将v改成-1表示  
}

[IMG_260](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

接下来是查找的过程了：

[IMG_261](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

int findTrie(char \*str)  
{  
    int len = strlen(str);  
    Trie \*p = root;  
    for(int i=0; i<len; ++i)  
    {  
        int id = str[i]-'0';  
        p = p->next[id];  
        if(p == NULL)   //若为空集，表示不存以此为前缀的串  
            return 0;  
        if(p->v == -1)   //字符集中已有串是此串的前缀  
            return -1;  
    }  
    return -1;   //此串是字符集中某串的前缀  
}

[IMG_262](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

对于上述动态字典树，有时会超内存，比如 [HDOJ 1671 Phone List](http://acm.hdu.edu.cn/showproblem.php?pid=1671)，这是就要记得释放空间了：

[IMG_263](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)

int dealTrie(Trie\* T)  
{  
    int i;  
    if(T==NULL)  
        return 0;  
    for(i=0;i<MAX;i++)  
    {  
        if(T->next[i]!=NULL)  
            deal(T->next[i]);  
    }  
    free(T);  
    return 0;  
}

## [IMG_264](http://www.cnblogs.com/tanky_woo/archive/2010/09/24/javascript:void(0);)1645: IMMEDIATE DECODABILITY

时间限制: 1 Sec  内存限制: 64 MB  
提交: 1  解决: 1  
[[提交](http://cst2.jlnu.edu.cn/oj/submitpage.php?id=1645)][[状态](http://cst2.jlnu.edu.cn/oj/problemstatus.php?id=1645)][[讨论版](http://cst2.jlnu.edu.cn/oj/bbs.php?pid=1645)]

## 题目描述

An encoding of a set of symbols is said to be immediately decodable if no code for one symbol is the prefix of a code for another symbol. We will assume for this problem that all codes are in binary, that no two codes within a set of codes are the same, that each code has at least one bit and no more than ten bits, and that each set has at least two codes and no more than eight. Examples: Assume an alphabet that has symbols {A, B, C, D} The following code is immediately decodable: A:01 B:10 C:0010 D:0000 but this one is not: A:01 B:10 C:010 D:0000 (Note that A is a prefix of C)

## 输入

Write a program that accepts as input a series of groups of records from standard input. Each record in a group contains a collection of zeroes and ones representing a binary code for a different symbol. Each group is followed by a single separator record containing a single 9; the separator records are not part of the group. Each group is independent of other groups; the codes in one group are not related to codes in any other group (that is, each group is to be processed independently).

## 输出

For each group, your program should determine whether the codes in that group are immediately decodable, and should print a single output line giving the group number and stating whether the group is, or is not, immediately decodable.

## 样例输入

01

10

0010

0000

9

01

10

010

0000

9

## 样例输出

Set 1 is immediately decodable

Set 2 is not immediately decodable

## 提示

## 来源

#include<cstdio>

#include<cstring>

#include<cstdlib>

#include<iostream>

using namespace std;

char word[15][20];

struct Tree

{

int num;//判断此节点是否为空的（就是一棵树的结束点）

Tree \*next[2];//这个是这棵树的子节点

Tree()

{

num=0;

for(int i=0;i<2;i++)

{

next[i]=NULL;

}

}//这是一棵树

}\*root;//建立一个树根

void insert(Tree \*p,char \*s)

{

int i=0;

while(s[i])//当这个字符串的某一个元素不为空的时候

{

int x=s[i]-'0';//看这个元素在哪里？

if(p->next[x]==NULL)//然后看这棵树有没有这个元素如果没有

{

p->next[x]=new Tree();//就在这棵树的此节点重新建立一棵子树

}

p=p->next[x];//p指向他的子树，这一点你就可以知道了所有字符串的元素原来他们都是主仆关系没有相同的等级关系！

p->num++;

i++;//继续遍历

}

}//插入字符

int find(Tree \*p,char \*s)

{

int i=0,ans=0;//寻找树的元素，ans是结果！

while(s[i])//当他不为空的时候就是字符串还有的时候

{

int n=s[i]-'0';//找到他在这个树的节点

if(p->next[n])//如果节点不为空

{

p=p->next[n];//p就指向该节点

ans=p->num;//结果应该为1

i++;

}

else

{

return 0;

}

}

return ans;

}

int deltree(Tree\* T)

{

int i;

if(T==NULL)

return 0;

for(i=0;i<2;i++)

{

if(T->next[i]!=NULL)

deltree(T->next[i]);

}

free(T);

return 0;

}

int main()

{

int times=1,i=0,j,success=1;

char s[20];

root=new Tree();

while(cin>>s)

{

if(strcmp(s,"9")==0)

{

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

{

// cout<<find(root,word[j])<<endl;

if(find(root,word[j])>1)

{

success=0;

break;

}

}

if(success)

{

printf("Set %d is immediately decodable\n",times++);

}

else

{

printf("Set %d is not immediately decodable\n",times++);

}

deltree(root);

root=new Tree();

success=1;

i=0;

}

else

{

strcpy(word[i],s);

insert(root,word[i]);

i++;

}

}

}

**最长公共子串（LCS）**

#include<iostream>

#include<cstring>

#include<vector>

using namespace std;

//str1为横向，str2这纵向

const string LCS(const string& str1,const string& str2){

int xlen=str1.size(); //横向长度

vector<int> tmp(xlen); //保存矩阵的上一行

vector<int> arr(tmp); //当前行

int ylen=str2.size(); //纵向长度

int maxele=0; //矩阵元素中的最大值

int pos=0; //矩阵元素最大值出现在第几列

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

string s=str2.substr(i,1);

arr.assign(xlen,0); //数组清0

for(int j=0;j<xlen;j++){

if(str1.compare(j,1,s)==0){

if(j==0)

arr[j]=1;

else

arr[j]=tmp[j-1]+1;

if(arr[j]>maxele){

maxele=arr[j];

pos=j;

}

}

}

tmp.assign(arr.begin(),arr.end());

}

string res=str1.substr(pos-maxele+1,maxele);

return res;

}

int main(){

string str1("21232523311324");

string str2("312123223445");

string lcs=LCS(str1,str2);

cout<<lcs<<endl;

return 0;

# }

# [【模板】各种欧几里得](http://blog.csdn.net/libin56842/article/details/9364231)

int gcd(int n,int m)//n>m

{

//最大公约数

int r;

while(m)

{

r = n%m;

n = m;

m = r;

}

return n;

}

int kgcd(int a,int b)

{

if(!a) return b;

if(!b) return a;

if(!(a&1) && !(b&1))

return kgcd(a>>1,b>>1)<<1;

else if(!(b&1)) return kgcd(a,b>>1);

else if(!(a&1)) return kgcd(a>>1,b);

else return kgcd(abs(a-b),min(a,b));

}

//扩展欧几里得

//求方程ax+by+c = 0有多少整数解

int extgcd(int a,int b,int &x,int &y)

{

if(!b)

{

x=1;

y=0;

return a;

}

int d = extgcd(b,a%b,x,y);

int t = x;

x=y;

y=t-a/b\*y;

return d;

}

**数据结构**

**1.优先队列**

书：189

**AC自动机**

**拓扑排序**

# 确定比赛名次

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 22765    Accepted Submission(s): 9231**

Problem Description

有 N个比赛队（1<=N<=500），编号依次为1，2，3，。。。。，N进行比赛，比赛结束后，裁判委员会要将所有参赛队伍从前往后依次排 名，但现在裁判委员会不能直接获得每个队的比赛成绩，只知道每场比赛的结果，即P1赢P2，用P1，P2表示，排名时P1在P2之前。现在请你编程序确定 排名。

Input

输入有若干组，每组中的第一行为二个数N（1<=N<=500），M；其中N表示队伍的个数，M表示接着有M行的输入数据。接下来的M行数据中，每行也有两个整数P1，P2表示即P1队赢了P2队。

Output

给出一个符合要求的排名。输出时队伍号之间有空格，最后一名后面没有空格。  
  
其他说明：符合条件的排名可能不是唯一的，此时要求输出时编号小的队伍在前；输入数据保证是正确的，即输入数据确保一定能有一个符合要求的排名。

Sample Input

4 3

1 2

2 3

4 3

Sample Output

1 2 4 3

#include <iostream>

using namespace std;

int map[502][502], indegree[502], m, n, pur[502];

**void topsort()**

**{**

**int i, j, k=1;**

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

**{**

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

**{**

**if(indegree[j]==0)**

**{**

**indegree[j]--;**

**pur[k++] = j;**

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

**if(map[j][x])**

**indegree[x]--;**

**break;**

**}**

**if(j>n)**

**{**

**cout<<"存在环"<<endl;**

**return ;**

**}**

**}**

**}**

**}**

void main()

{

int i, j;

while(cin>>n>>m)

{

memset(map,0,sizeof(map));

memset(indegree,0,sizeof(indegree));

int a, b;

for(i=1; i<=m; i++)

{

cin>>a>>b;

if(!map[a][b])

{

map[a][b] = 1;

indegree[b]++;

}

}

topsort();

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

if(i!=n)

cout<<pur[i]<<" ";

else

cout<<pur[i]<<endl;

}

}

---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**组合数**

|  |
| --- |
| Recipient |
|  |
| 题目描述 快递小哥每天都辛苦的送快递，今天他需要送N份快递给N个收件人，第i份快递需要送给第i个收件人。请问其中发生恰好K个送错了的情况数是多少？ 输入 存在多样例。每行输入两个整数N和K，1≤N≤1000,0≤K≤N。如果两个都为0，则表示输入结束，这个样例不需要处理。 输出 每行输出一个样例的结果，因为数值会比较大，所有结果需要对109+7取模。 样例输入 1 1  2 1  3 2  1000 1000  0 0 样例输出 0  0  3  37043040  解释：  组合数和错排这里的解法都是用到的递推**组合数：**递推公式是C(n,m)=C(n-1,m-1)+C(n,m-1);因为对于m个物体中的一个物体，要么被选，要么不被选，两种可能，所以如果被选了，就只需要从剩下的m-1个物体中选n-1个即C(n-1,m-1),  如果没有被选，就需要从剩下的m-1个物体中选n个即C(n,m-1),然后相加，即C(n,m)=C(n-1,m-1)+C(n,m-1)。**错排公式：**递推公式为f(n)=(n-1)\*(f(n-1)+f(n-2));因为对于n个物体错排，如果第一个物体错排的是第k个物体(n-1)种选择，那么分析第k个物体，如果第k个物体的位置错排的是第1个物体那么剩下的n-2个物体再进行错排就可以了即f(n-2)  如果第k个物体的位置错排的不是第1个物体而是w那么就相当于把第k个物体扔了直接再第一个物体下面排w即f(n-1)然后相加。即f(n)=(n-1)\*(f(n-1)+f(n-2))。  代码如下 |

#include<iostream>

#include<cstdio>

#include<cstring>

using namespace std;

const int mod =1000000007;

long long c[1005][1005],f[1005];

void init1()

{

for(int i=0;i<1005;i++){

c[i][i]=c[0][i]=1;

}

for(int i=1;i<1005;i++){

for(int j=i+1;j<1005;j++){

c[i][j]=(c[i-1][j-1]+c[i][j-1])%mod;

}

}

}

void init2(){

f[0]=1;

f[1]=0;

f[2]=1;

for(int i=3;i<1005;i++){

f[i]=((i-1)\*(f[i-2]+f[i-1]))%mod;

}

}

int main(){

int n,k;

init1();

init2();

while(~scanf("%d%d",&n,&k),n+k){;

printf("%I64d\n",(c[n-k][n]\*f[k])%mod);

}

return 0;

}

**组合数：**

**1.卡特兰数**

令h(0)=1,h(1)=1，catalan数满足递推式[1]  ：

**h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0) (n>=2)**

例如：h(2)=h(0)\*h(1)+h(1)\*h(0)=1\*1+1\*1=2

h(3)=h(0)\*h(2)+h(1)\*h(1)+h(2)\*h(0)=1\*2+1\*1+2\*1=5

另类递推式[2]  ：

**h(n)=h(n-1)\*(4\*n-2)/(n+1);**

递推关系的解为：

**h(n)=C(2n,n)/(n+1) (n=0,1,2,...)**

递推关系的另类解为：

**h(n)=c(2n,n)-c(2n,n-1)(n=0,1,2,...)**

**卡特兰数程序：**

#include<stdio.h>

#define n 20

main()

{

long long a[20]={0,0,0,0,0,0,0,0,0,0};

int i,j;

a[0]=1;

a[1]=1;

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

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

if(i>j)

a[i]+=a[j]\*a[i-j-1];

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

printf("%lld\n",a[i]);

}

**序列：1 1 2 5 14 42 132 429**

**1430 4862 16796**

**Problem 2020 组合**

### Time Limit: 1000 mSec    Memory Limit : 32768 KB

## IMG_257Problem Description

给出组合数C(n,m), 表示从n个元素中选出m个元素的方案数。例如C(5,2) = 10, C(4,2) = 6.可是当n,m比较大的时候，C(n,m)很大！于是xiaobo希望你输出 C(n,m) mod p的值！

## IMG_258Input

输入数据第一行是一个正整数T,表示数据组数 (T <= 100) 接下来是T组数据，每组数据有3个正整数 n, m, p (1 <= m <= n <= 10^9, m <= 10^4, m < p < 10^9, p是素数)

## IMG_259Output

对于每组数据，输出一个正整数，表示C(n,m) mod p的结果。

## IMG_260Sample Input

2

5 2 3

5 2 61

## IMG_261Sample Output

1

10

## IMG_262Source

FOJ有奖月赛-2011年04月（校赛热身赛）

#include <iostream>

#include <string.h>

#include <stdio.h>

using namespace std;

typedef long long LL;

LL n,m,p;

LL quick\_mod(LL a, LL b)

{

LL ans = 1;

a %= p;

while(b)

{

if(b & 1)

{

ans = ans \* a % p;

b--;

}

b >>= 1;

a = a \* a % p;

}

return ans;

}

LL C(LL n, LL m)

{

if(m > n) return 0;

LL ans = 1;

for(int i=1; i<=m; i++)

{

LL a = (n + i - m) % p;

LL b = i % p;

ans = ans \* (a \* quick\_mod(b, p-2) % p) % p;

}

return ans;

}

LL Lucas(LL n, LL m)

{

if(m == 0) return 1;

return C(n % p, m % p) \* Lucas(n / p, m / p) % p;

}

int main()

{

int T;

scanf("%d", &T);

while(T--)

{

scanf("%I64d%I64d%I64d", &n, &m, &p);

printf("%I64d\n", Lucas(n,m));

}

return 0;

}

**组合数**

|  |
| --- |
| Recipient |
|  |
| 题目描述 快递小哥每天都辛苦的送快递，今天他需要送N份快递给N个收件人，第i份快递需要送给第i个收件人。请问其中发生恰好K个送错了的情况数是多少？ 输入 存在多样例。每行输入两个整数N和K，1≤N≤1000,0≤K≤N。如果两个都为0，则表示输入结束，这个样例不需要处理。 输出 每行输出一个样例的结果，因为数值会比较大，所有结果需要对109+7取模。 样例输入 1 1  2 1  3 2  1000 1000  0 0 样例输出 0  0  3  37043040  解释：  组合数和错排这里的解法都是用到的递推**组合数：**递推公式是C(n,m)=C(n-1,m-1)+C(n,m-1);因为对于m个物体中的一个物体，要么被选，要么不被选，两种可能，所以如果被选了，就只需要从剩下的m-1个物体中选n-1个即C(n-1,m-1),  如果没有被选，就需要从剩下的m-1个物体中选n个即C(n,m-1),然后相加，即C(n,m)=C(n-1,m-1)+C(n,m-1)。**错排公式：**递推公式为f(n)=(n-1)\*(f(n-1)+f(n-2));因为对于n个物体错排，如果第一个物体错排的是第k个物体(n-1)种选择，那么分析第k个物体，如果第k个物体的位置错排的是第1个物体那么剩下的n-2个物体再进行错排就可以了即f(n-2)  如果第k个物体的位置错排的不是第1个物体而是w那么就相当于把第k个物体扔了直接再第一个物体下面排w即f(n-1)然后相加。即f(n)=(n-1)\*(f(n-1)+f(n-2))。  代码如下 |

#include<iostream>

#include<cstdio>

#include<cstring>

using namespace std;

const int mod =1000000007;

long long c[1005][1005],f[1005];

void init1()

{

for(int i=0;i<1005;i++){

c[i][i]=c[0][i]=1;

}

for(int i=1;i<1005;i++){

for(int j=i+1;j<1005;j++){

c[i][j]=(c[i-1][j-1]+c[i][j-1])%mod;

}

}

}

void init2(){

f[0]=1;

f[1]=0;

f[2]=1;

for(int i=3;i<1005;i++){

f[i]=((i-1)\*(f[i-2]+f[i-1]))%mod;

}

}

int main(){

int n,k;

init1();

init2();

while(~scanf("%d%d",&n,&k),n+k){;

printf("%I64d\n",(c[n-k][n]\*f[k])%mod);

}

return 0;

}

**组合数：**

**1.卡特兰数**

令h(0)=1,h(1)=1，catalan数满足递推式[1]  ：

**h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0) (n>=2)**

例如：h(2)=h(0)\*h(1)+h(1)\*h(0)=1\*1+1\*1=2

h(3)=h(0)\*h(2)+h(1)\*h(1)+h(2)\*h(0)=1\*2+1\*1+2\*1=5

另类递推式[2]  ：

**h(n)=h(n-1)\*(4\*n-2)/(n+1);**

递推关系的解为：

**h(n)=C(2n,n)/(n+1) (n=0,1,2,...)**

递推关系的另类解为：

**h(n)=c(2n,n)-c(2n,n-1)(n=0,1,2,...)**

**卡特兰数程序：**

#include<stdio.h>

#define n 20

main()

{

long long a[20]={0,0,0,0,0,0,0,0,0,0};

int i,j;

a[0]=1;

a[1]=1;

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

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

if(i>j)

a[i]+=a[j]\*a[i-j-1];

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

printf("%lld\n",a[i]);

}

**序列：1 1 2 5 14 42 132 429**

**1430 4862 16796**

**Problem 2020 组合**

### Time Limit: 1000 mSec    Memory Limit : 32768 KB

## IMG_257Problem Description

给出组合数C(n,m), 表示从n个元素中选出m个元素的方案数。例如C(5,2) = 10, C(4,2) = 6.可是当n,m比较大的时候，C(n,m)很大！于是xiaobo希望你输出 C(n,m) mod p的值！

## IMG_258Input

输入数据第一行是一个正整数T,表示数据组数 (T <= 100) 接下来是T组数据，每组数据有3个正整数 n, m, p (1 <= m <= n <= 10^9, m <= 10^4, m < p < 10^9, p是素数)

## IMG_259Output

对于每组数据，输出一个正整数，表示C(n,m) mod p的结果。

## IMG_260Sample Input

2

5 2 3

5 2 61

## IMG_261Sample Output

1

10

## IMG_262Source

FOJ有奖月赛-2011年04月（校赛热身赛）

#include <iostream>

#include <string.h>

#include <stdio.h>

using namespace std;

typedef long long LL;

LL n,m,p;

LL quick\_mod(LL a, LL b)

{

LL ans = 1;

a %= p;

while(b)

{

if(b & 1)

{

ans = ans \* a % p;

b--;

}

b >>= 1;

a = a \* a % p;

}

return ans;

}

LL C(LL n, LL m)

{

if(m > n) return 0;

LL ans = 1;

for(int i=1; i<=m; i++)

{

LL a = (n + i - m) % p;

LL b = i % p;

ans = ans \* (a \* quick\_mod(b, p-2) % p) % p;

}

return ans;

}

LL Lucas(LL n, LL m)

{

if(m == 0) return 1;

return C(n % p, m % p) \* Lucas(n / p, m / p) % p;

}

int main()

{

int T;

scanf("%d", &T);

while(T--)

{

scanf("%I64d%I64d%I64d", &n, &m, &p);

printf("%I64d\n", Lucas(n,m));

}

return 0;

}

----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

**最短路径及最小生成树题目：**

**类型一：**

/\*

\* 1077.cpp

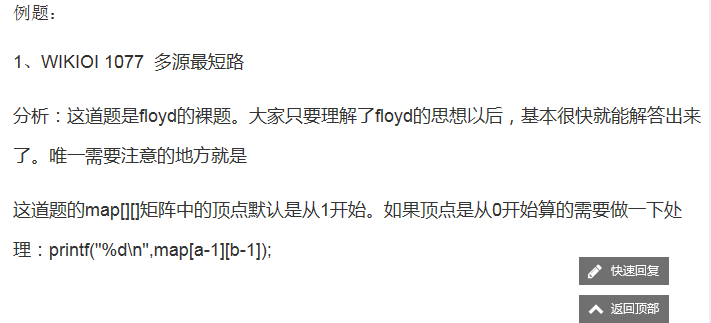
\*

\* Created on: 2014年5月23日

\* Author: pc

\*/





# 题目描述：【codevs1077】多源最短路

**2013年12月30日6170**

## 题目描述 Description

**已知n个点(n<=100)，给你n\*n的方阵，a[i,j]表示从第i个点到第j个点的直接距离。**

**现在有Q个询问，每个询问两个正整数，a和b，让你求a到b之间的最短路程。**

**满足a[i,j]=a[j,i];**

## 输入描述 Input Description

**第一行一个正整数n，接下来n行每行n个正整数，满足a[i,i]=0,再一行一个Q，接下来Q行，每行两个正整数a和b。**

## 输出描述 Output Description

**一共Q行，每行一个整数。**

## 样例输入 Sample Input

**3**

**0 1 1**

**1 0 3**

**1 3 0**

**1**

**2 3**

## 样例输出 Sample Output

**2**

## 数据范围及提示 Data Size & Hint

**n<=100，Q可能非常大。g[i][j]均>=0**

**请使用flyod算法**

**使用C/C++的同学请注意：由于输入数据较大，使用cin和cout会导致程序超时。请使用scanf与printf进行输入和输出。**

## 代码

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

int e[maxn][maxn];

int n;

const int inf = 99999999;

void initial() {

int i, j;

for (i = 1; i <= n; ++i) {

for (j = 1; j <= n; ++j) {

if (i == j) {

e[i][j] = 0;

} else {

e[i][j] = inf;

}

}

}

}

/\*\*

\*floyd算法

\*/

void floyd() {

int i, j, k;

for (k = 1; k <= n; ++k) {//遍历所有的中间点

for (i = 1; i <= n; ++i) {//遍历所有的起点

for (j = 1; j <= n; ++j) {//遍历所有的终点

if (e[i][j] > e[i][k] + e[k][j]) {//如果当前i-->j的距离大于i-->k--->j的距离之和

e[i][j] = e[i][k] + e[k][j];//更新从i--->j的最短路径

}

}

}

}

}

int main() {

while (scanf("%d", &n) != EOF) {

initial();

int i, j;

for (i = 1; i <= n; ++i) {

for (j = 1; j <= n; ++j) {

scanf("%d", &e[i][j]);

}

}

floyd();

int q;

scanf("%d", &q);

while (q--) {

int a, b;

scanf("%d %d", &a, &b);

printf("%d\n", e[a][b]);

}

}

return 0;

}

**类型二**

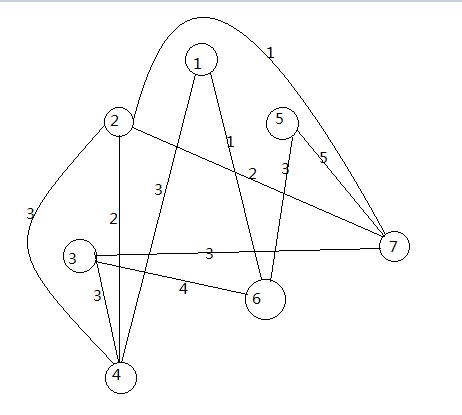
**1：求可变指定起始点到终点的最短路径（起点和终点是变化的）**

**1728: 运送牛奶**

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 31  Solved: 9  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1728)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1728)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1728)]

**Description**

德克萨斯纯朴的民眾们这个夏天正在遭受巨大的热浪！！！他们的德克萨斯长角牛吃起来不错，可是他们并不是很擅长生產富含奶油的乳製品。Farmer  John此时以先天下之忧而忧，后天下之乐而乐的精神，身先士卒地承担起向德克萨斯运送大量的营养冰凉的牛奶的重任，以减轻德克萨斯人忍受酷暑的痛苦。 FJ已经研究过可以把牛奶从威斯康星运送到德克萨斯州的路线。这些路线包括起始点和终点先一共经过T  (1  < =  T  < =  2,500)个城镇，方便地标号為1到T。除了起点和终点外地每个城镇由两条双向道路连向至少两个其它地城镇。每条道路有一个通过费用（包括油费，过路费 等等）。考虑这个有7个城镇的地图。城镇5是奶源，城镇4是终点（括号内的数字是道路的通过费用）。



经过路线5-6-3-4总共需要花费3  (5-> 6)  +  4  (6-> 3)  +  3  (3-> 4)  =  10的费用。 给定一个地图，包含C  (1  < =  C  < =  6,200)条直接连接2个城镇的道路。每条道路由道路的起点Rs，终点Re  (1  < =  Rs  < =  T;  1  < =  Re  < =  T)，和花费(1  < =  Ci  < =  1,000)组成。求从起始的城镇Ts  (1  < =  Ts  < =  T)到终点的城镇Te(1  < =  Te  < =  T)最小的总费用。

**Input**

第一行:  4个由空格隔开的整数:  T,  C,  Ts,  Te

第2到第C+1行:

第i+1行描述第i条道路。有3个由空格隔开的整数:  Rs,  Re和Ci

**Output**

一个单独的整数表示Ts到Te的最短路的长度。（不是费用麼？怎麼突然变直白了 ——译者注）数据保证至少存在一条道路。

**Sample Input**

7 11 5 4

2 4 2

1 4 3

7 2 2

3 4 3

5 7 5

7 3 3

6 1 1

6 3 4

2 4 3

5 6 3

7 2 1

**Sample Output**

7

**HINT**

5-> 6-> 1-> 4  (3  +  1  +  3)

**1．耗时：**

#include <stdio.h>

#include <string.h>

#define MIN(a,b) a<b?a:b

#define M 9999

#define N 2503

int d[N];//起点到该点的权值

int w[N][N];//a到b的权值

int vis[N];

void Dijkstra(int n,int s)

{

    int i,j,k,v,t;

    memset(vis,0,sizeof(vis));

    for(i=1; i<=n; i++)

        d[i]=w[s][i];

    d[s]=0;

    vis[s]=1;

    for(i=1; i<=n; i++)

    {

        t=M;

        for(j=1; j<=n; j++)

        {

            if(t>d[j]&&!vis[j])

            {

                t=d[j];

                v=j;

            }

        }

        vis[v]=1;

        for(k=1; k<=n; k++)

        {

            if(!vis[k])d[k]=MIN(d[k],d[v]+w[v][k]);

        }

    }

}

int main()

{

    int n,m,i,j,a,b,c,T,S,C,E;

    scanf("%d%d%d%d",&T,&C,&S,&E);

        for(i=1;i<=T;i++)

            for(j=1;j<=T;j++)

            w[i][j]=M;

       for(i=0;i<C;i++)

       {

            scanf("%d%d%d",&a,&b,&c);

            w[a][b]=MIN(w[a][b],c);

            w[b][a]=w[a][b];

        }

        Dijkstra(T,S);

        printf("%d\n",d[E]);

    return 0;

}

**2.用队列做的：**

#include<stdio.h>

#include<vector>

#include<queue>

using namespace std;

const int imax = 2505;

#define INF 1e9

vector<pair<int,int> >vec[imax];

int dis[imax];

void Dijkstra(int s)

{

priority\_queue<pair<int,int> >que;

dis[s]=0;

que.push(make\_pair(-dis[s],s));

while(!que.empty())

{

int now = que.top().second;

que.pop();

for(int i = 0; i<vec[now].size(); i++)

{

int v = vec[now][i].first;

if(dis[v]>dis[now]+vec[now][i].second)

{

dis[v]=dis[now]+vec[now][i].second;

que.push(make\_pair(-dis[v],v));

}

}

}

}

void init()

{

for(int i = 0; i<imax; i++)

vec[i].clear(),dis[i]=INF;

}

int main()

{

int n,m,s,t;

int Rs,Re,Ci;

while(scanf("%d %d %d %d",&n,&m,&s,&t)!=EOF)

{

init();

for(int i = 1; i<=m; i++)

{

scanf("%d%d%d",&Rs,&Re,&Ci);

vec[Rs].push\_back(make\_pair(Re,Ci));

vec[Re].push\_back(make\_pair(Rs,Ci));

}

Dijkstra(s);

printf("%d\n",dis[t]);

}

}

**类型三**

**2.求唯一起点到终点的最短路径（起点和终点固定不变）**

**1950: 中午饿了要定餐（该题测试数据中包含重复路径，是万能模版）**

Time Limit: 1 Sec  Memory Limit: 33 MB  
Submit: 52  Solved: 8  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1950)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1950)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1950)]

**Description**

ACM比赛正在5教进行中，转眼到了中午，这时候出去吃饭怎么可能，只有叫外卖了。外卖要从小门出发，骑自行车来到5教，因为大一学生军训，很多路线有变化，送餐员不知道怎么走了。现在请你来帮帮他，根据你知道的校园各条路的距离，规划出一条从小门到5教的最短路线。

**Input**

输入包括多组数据。每组数据第一行是两个整数N、M（N<=100，M<=10000），N表示校园内有几个路口，标号为1的路口是小门所在 地，标号为N的路口是5教所在地，M则表示在校园有几条路。N=M=0表示输入结束。接下来M行个整数A，B，C（1<=A,B<=N,1& lt;=C<=1000）,表示在路口A与路口B之间有一条路，C表示A到B的距离。  
输入保证至少存在1条小门到5教的路线。

**Output**

对于每组输入，输出一行，表示送餐员从小门到5教的最短路线

**Sample Input**

2 1

1 2 3

3 3

1 2 5

2 3 5

3 1 2

0 0

**Sample Output**

3

2

**HINT**

#include <stdio.h>

#include <string.h>

#define MIN(a,b) a<b?a:b

#define M 100000

#define N 101

int d[N];//起点到该点的权值

int w[N][N];//a到b的权值

int vis[N];

void Dijkstra(int n)

{

    int i,j,k,v,t;

    memset(vis,0,sizeof(vis));

    for(i=1; i<=n; i++)

        d[i]=w[1][i];

    d[1]=0;

    vis[1]=1;

    for(i=1; i<=n; i++)

    {

        t=M;

        for(j=1; j<=n; j++)

        {

            if(t>d[j]&&!vis[j])

            {

                t=d[j];

                v=j;

            }

        }

        vis[v]=1;

        for(k=1; k<=n; k++)

        {

            if(!vis[k])d[k]=MIN(d[k],d[v]+w[v][k]);

        }

    }

}

int main()

{

    int n,m,i,j,a,b,c;

    while(scanf("%d%d",&n,&m)&&(n!=0||m!=0))

    {

        for(i=1;i<=n;i++)

            for(j=1;j<=n;j++)

            w[i][j]=M;

        while(m--)

        {

            scanf("%d%d%d",&a,&b,&c);

            w[a][b]=MIN(w[a][b],c);

            w[b][a]=w[a][b];

        }

        Dijkstra(n);

        printf("%d\n",d[n]);

    }

    return 0;

}

**2．求已知起点到终点的最短路径：(该题测试数据不包含重复路径，老师的模板是万能的模板)**

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 9999999;

int s[maxn];//用来记录某一点是否被访问过

int map[maxn][maxn];//地图

int dis[maxn];//从原点到某一个点的最短距离(一开始是估算距离)

int n;

int target;

/\*\*

\* 返回从v---->到target的最短路径

\*/

int dijkstra(int v){

int i;

for(i = 1 ; i <= n ; ++i){//初始化

s[i] = 0;//一开始，所有的点均为被访问过

dis[i] = map[v][i];

}

for(i = 1 ; i < n ; ++i){

int min = inf;

int pos;

int j;

for(j = 1 ; j <= n ; ++j){//寻找目前的最短路径的最小点

if(!s[j] && dis[j] < min){

min = dis[j];

pos = j;

}

}

s[pos] = 1;

for(j = 1 ; j <= n ; j++){//遍历u的所有的邻接的边

if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

dis[j] = dis[pos] + map[pos][j];//对边进行松弛

}

}

}

return dis[target];

}

int main(){

int m;

while(scanf("%d%d",&n,&m)!=EOF){

int i;

int j;

for(i = 1 ; i <= n ; ++i){

for(j = 1 ; j <= n ; ++j){

if(i == j){

map[i][j] = 0;

}else{

map[i][j] = inf;

}

}

}

for(i = 1 ; i <= m ; ++i){

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

map[a][b] = map[b][a] = c;//这里默认是无向图。。所以要两个方向都做处理,只做一个方向上的处理会WA

}

target = n;

int result = dijkstra(1);

printf("%d\n",result);

}

return 0;

}

*---------------------------------------------------------------------------------------------------------------------------------*-/\*

\* NEFU\_207.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 99999999;

int e[maxn][maxn];

int n,m;

void initial(){

int i;

int j;

for(i = 1 ; i <= n ; ++i){

for(j = 1 ; j <= n ; ++j){

if(i == j){

e[i][j] = 0;

}else{

e[i][j] = inf;

}

}

}

}

void floyd(){

int i;

int j;

int k;

for(k = 1 ; k <= n ; ++k){

for(i = 1 ; i <= n ; ++i){

for(j = 1 ; j <= n ; ++j){

if(e[i][j] > e[i][k] + e[k][j]){

e[i][j] = e[i][k] + e[k][j];

}

}

}

}

}

int main(){

while(scanf("%d%d",&n,&m)!=EOF){

initial();

int i;

for(i = 1 ; i <= m ; ++i){

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

e[a][b] = e[b][a] = c;

}

floyd();

printf("%d\n",e[1][n]);

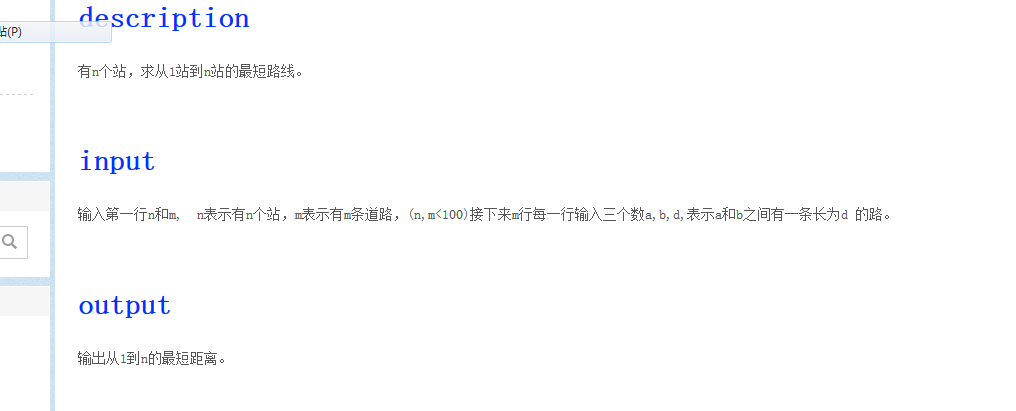
}

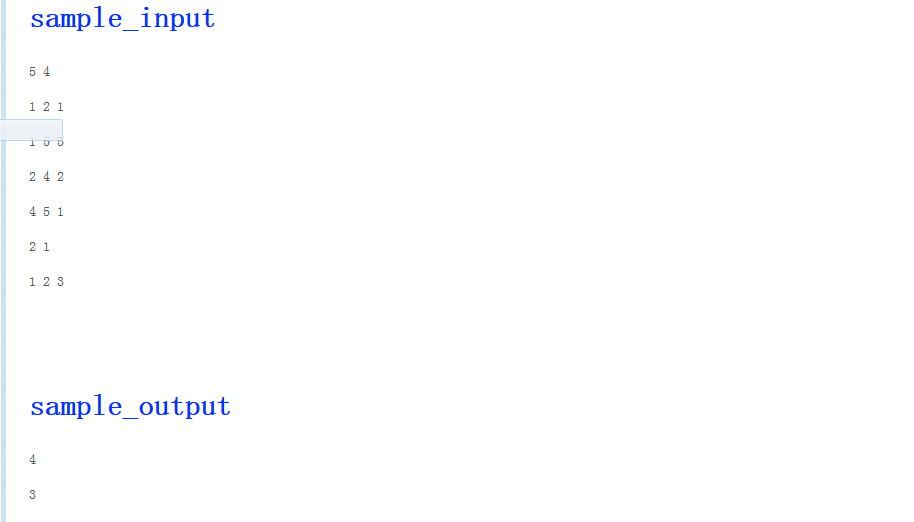
return 0;

}

**floyd：解决多源最短路径问题。求任意两个点之间的最短路径。这当然也就包含了“从a到b的这种情况”。所以这道题也可以使用floyd来解决**

**dijkstra：解决单源最短路径问题 。最典型的就是解决“从a到b的最短路径的距离”的这种问题了。**





#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 9999999;

int s[maxn];//用来记录某一点是否被访问过

int map[maxn][maxn];//地图

int dis[maxn];//从原点到某一个点的最短距离(一开始是估算距离)

int n;

int target;

/\*\*

 \* 返回从v---->到target的最短路径

 \*/

int dijkstra(int v){

    int i;

    for(i = 1 ; i <= n ; ++i){//初始化

        s[i] = 0;//一开始，所有的点均为被访问过

        dis[i] = map[v][i];

    }

    for(i = 1 ; i < n ; ++i){

        int min = inf;

        int pos;

        int j;

        for(j = 1 ; j <= n ; ++j){//寻找目前的最短路径的最小点

            if(!s[j] && dis[j] < min){

                min = dis[j];

                pos = j;

            }

        }

        s[pos] = 1;

        for(j = 1 ; j <= n ; j++){//遍历u的所有的邻接的边

            if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

                dis[j] = dis[pos] + map[pos][j];//对边进行松弛

            }

        }

    }

    return dis[target];

}

int main(){

    int m;

    while(scanf("%d%d",&n,&m)!=EOF){

      if(n==0)

      break;

        int i;

        int j;

        for(i = 1 ; i <= n ; ++i){

            for(j = 1 ; j <= n ; ++j){

                if(i == j){

                    map[i][j] = 0;

                }else{

                    map[i][j] = inf;

                }

            }

        }

        for(i = 1 ; i <= m ; ++i){

            int a,b,c;

            scanf("%d%d%d",&a,&b,&c);

            map[a][b] = map[b][a] = c;//这里默认是无向图。。所以要两个方向都做处理,只做一个方向上的处理会WA

        }

        target = n;

        int result = dijkstra(1);

        printf("%d\n",result);

    }

    return 0;

}

### 4.任意两点最短路径问题（给的是坐标距离需要自己求）

#### Problem:313

#### Time Limit:1000ms

#### Memory Limit:65536K

### Description

平面上有n个点(n<=100)，每个点的坐标均在-10000~10000之间。其中的一些点之间有连线，若有连线，则表示可从一个点到达另一个点，即两点间有通路，通路的距离为两点间的直线距离。现在的任务是找出从一点到另一点之间的最短路径。

### Input

多组数据输入.

每组输入共n+m+3行，其中：

第一行为整数n。

第2行到第n+1行（共n行），每行两个整数x和y，描述了一个点的坐标（以一个空格分隔）。

第n+2行为一个整数m，表示图中连线的个数。

此后的m行，每行描述一条连线，由两个整数I和j组成，表示第I个点和第j个点之间有连线。

最后一行：两个整数s和t，分别表示源点和目标点。

### Output

每组输出一个实数（保留两位小数），表示从s到t 的最短路径长度。

### Sample Input

5

0 0

2 0

2 2

0 2

3 1

5

1 2

1 3

1 4

2 5

3 5

1 5

### Sample Output

3.41

### Hint

### Source

OI

/\*

\* NEFU\_313.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

const int maxn = 105;

const int inf = INT\_MAX;

int s[maxn];

double dis[maxn];

double map[maxn][maxn];

int n;

int target;

struct Pointt{

double x;

double y;

};

double distance1(Pointt p1, Pointt p2){

return sqrt((p1.x - p2.x)\*(p1.x - p2.x) + (p1.y - p2.y)\*(p1.y - p2.y));

}

double dijkstra(int v){

int i;

for(i =1 ; i <= n ; ++i){

s[i] = 0;

dis[i] = map[v][i];

}

for(i = 1 ; i < n; ++i){

double min = inf;

int pos;

int j;

for(j = 1 ; j <= n ; ++j){

if(!s[j] && dis[j] < min){

min = dis[j];

pos = j;

}

}

s[pos] = 1;

for(j = 1 ; j <= n ; ++j){

if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

dis[j] = dis[pos] + map[pos][j];

}

}

}

return dis[target];

}

void printfMap(){

int i;

int j;

for(i = 1 ; i <= n ; ++i){

for(j = 1 ; j <= n ; ++j){

printf("%lf " ,map[i][j]);

}

printf("\n");

}

}

int main(){

while(scanf("%d",&n)!=EOF){

Pointt p[n+1];

int i;

for(i = 1 ; i <= n ; ++i){

scanf("%lf%lf",&p[i].x,&p[i].y);

}

int j;

for(i = 1 ; i <= n ; ++i){

for(j = 1 ; j <= n ; ++j){

if(i == j){

map[i][j] = 0;

}else{

map[i][j] = inf;

}

}

}

int m;

scanf("%d",&m);

for(i = 1 ; i <= m ; ++i){

int a,b;

scanf("%d%d",&a,&b);

map[a][b] = map[b][a] = distance1(p[a],p[b]);

}

int start;

scanf("%d%d",&start,&target);

double result = dijkstra(start);

printf("%.2lf\n",result);

}

return 0;

}

/\*

\* NEFU\_313.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

Floyd:

#include <iostream>

#include <cstdio>

#include <cmath>

using namespace std;

const int maxn = 105;

double map[maxn][maxn];

int n;

const int inf = INT\_MAX;

struct Pointt {

double x;

double y;

};

double distance1(Pointt p1, Pointt p2) {

return sqrt((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y));

}

void initial() {

int i;

int j;

for (i = 1; i <= n; ++i) {

for (j = 1; j <= n; ++j) {

if (i == j) {

map[i][j] = 0;

} else {

map[i][j] = inf;

}

}

}

}

void floyd() {

int i;

int j;

int k;

for (k = 1; k <= n; ++k) {

for (i = 1; i <= n; ++i) {

for (j = 1; j <= n; ++j) {

if (map[i][j] > map[i][k] + map[k][j]) {

map[i][j] = map[i][k] + map[k][j];

}

}

}

}

}

int main() {

while (scanf("%d", &n) != EOF) {

int i;

Pointt p[n + 1];

for (i = 1; i <= n; ++i) {

scanf("%lf%lf", &p[i].x, &p[i].y);

}

int m;

scanf("%d", &m);

initial();

for (i = 1; i <= m; ++i) {

int a, b;

scanf("%d%d", &a, &b);

map[a][b] = map[b][a] = distance1(p[a], p[b]);

}

floyd();

int start, end;

scanf("%d%d", &start, &end);

printf("%.2lf\n", map[start][end]);

}

return 0;

}

## 最小生成树问题：

## 5. 1396: Dick's back

时间限制: 1 Sec  内存限制: 128 MB  
提交: 16  解决: 6  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1396)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1396)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1396)]

**题目描述**

In an episode of the Dick Van Dyke show, little Richie connects the freckles on his Dad's back to form a picture of the Liberty Bell. Alas, one of the freckles turns out to be a scar, so his Ripley's engagement falls through.

Consider Dick's back to be a plane with freckles at various (x,y) locations. Your job is to tell Richie how to connect the dots so as to minimize the amount of ink used. Richie connects the dots by drawing straight lines between pairs, possibly lifting the pen between lines. When Richie is done there must be a sequence of connected lines from any freckle to any other freckle.

**输入**

The first line contains 0 < *n* <= 100, the number of freckles on Dick's back. For each freckle, a line follows; each following line contains two real numbers indicating the (x,y) coordinates of the freckle.

**输出**

Your program prints a single real number to two decimal places: the minimum total length of ink lines that can connect all the freckles.

**样例输入**

3

1.0 1.0

2.0 2.0

2.0 4.0

**样例输出**

3.41

**提示**

**典型的模板题（prim算法）**

 #include<stdio.h>

    #include<string.h>

    #include<math.h>

    double d[102][2], w[102][102], minCost[102];

    int n, pre[102], hash[102];

     double getDist(double x1,double y1,double x2,double y2){

        return sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));

    }

    double Prim(){

        memset(hash, 0, sizeof(hash));

        hash[1] = 1;

        for(int i=1; i<=n; ++i){

            minCost[i] = w[1][i];

            pre[i] = 1;

        }

        double sum=0;

        for(int i=1; i<n; ++i){

            int u=-1;

            for(int j=1; j<=n; ++j)if(!hash[j]){

                if(u==-1||minCost[j]<minCost[u])

                    u=j;

            }

            sum += w[pre[u]][u];

            hash[u] = 1;

            for(int j=1; j<=n; ++j)if(!hash[j]){

                if(minCost[j]>w[u][j]){

                    minCost[j] = w[u][j];

                    pre[j] = u;

                }

            }

        }

        return sum;

    }//prim

    int main(){

            scanf("%d",&n);

            for(int i=1; i<=n; ++i)

                scanf("%lf%lf",&d[i][0],&d[i][1]);

            memset(w, 0, sizeof(w));

            for(int i=1; i<=n; ++i)

                for(int j=1; j<=n; ++j)if(i!=j)

                    w[i][j] = getDist(d[i][0],d[i][1],

                            d[j][0],d[j][1]);

            printf("%.2f\n", Prim());

        return 0;

}

**6.求任意点最短路径及最少花费（即路径长度相同时优先选择花费少的）**

**1970: Shortest Path Problem**

时间限制: 1 Sec  内存限制: 128 MB  
提交: 20  解决: 5  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1970)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1970)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1970)]

**题目描述**

 Given N points, M undirected edges, every edge has length D and cost P. Now give you starting point S and ending point T, output the shortest distance from starting point to ending point and its cost. If the shortest path has several routes, print less cost one.

**输入**

     Input N,M, point is marked as 1 to N, and M lines followed, each line has 4 numbers A,B,D,P, indicating there is an edge between A and B, length is D, cost is P. The last line has two numbers S, T. If  the  N,M is 0 0,End the file.

  (1<N<=1000, 0<M<100000, S !=T)

**输出**

Two numbers, shortest distance and cost.

**样例输入**

3 2

1 2 5 6

2 3 4 5

1 3

0 0

**样例输出**

9 11

**提示**

#include <stdio.h>

#include <string.h>

#define MIN(a,b) a<b?a:b

#define M 100000

#define N 101

int d[N];//起点到该点的权值

int cos[N];

int w[N][N];//a到b的权值

int ww[N][N];

int vis[N];

int viss[N];

void Dijkstra(int n,int s)

{

    int i,j,k,v,t;

    memset(vis,0,sizeof(vis));

    for(i=1; i<=n; i++)

      {

        d[i]=w[s][i];

        cos[i]=ww[s][i];

      }

    d[s]=0;

    vis[s]=1;

    cos[s]=0;

   // viss[s]=1;

    for(i=1; i<=n; i++)

    {

        t=M;

        for(j=1; j<=n; j++)

        {

            if(t>d[j]&&!vis[j])

            {

                t=d[j];

                v=j;

            }

        }

        vis[v]=1;

        for(k=1; k<=n; k++)

        {

            if(!vis[k])

            {

                if(d[k]>d[v]+w[v][k])

                {

                    d[k]=d[v]+w[v][k];

                    cos[k]=cos[v]+ww[v][k];

                }

                if(d[k]==d[v]+w[v][k])

                {

                    if(cos[k]>cos[v]+ww[v][k])

                      cos[k]=cos[v]+ww[v][k];

                }

            }

        }

    }

}

int main()

{

    int n,m,i,j,a,b,c,cost,C,E,T,S;

    while(scanf("%d%d",&n,&m)&&(n!=0||m!=0))

    {

        for(i=1;i<=n;i++)

            for(j=1;j<=n;j++)

            {

              w[i][j]=ww[i][j]=M;

            }

        for(i=0;i<m;i++)

        {

            scanf("%d%d%d%d",&a,&b,&c,&cost);

           // w[a][b]=MIN(w[a][b],c);

          //  ww[a][b]=cost;

            if(w[a][b]>c)

            {

                w[a][b]=w[b][a]=c;

                ww[a][b]=ww[b][a]=cost;

            }

            if(w[a][b]==c)

            {

                if(ww[a][b]>cost)

                ww[a][b]=ww[b][a]=cost;

            }

        }

        scanf("%d%d",&S,&E);

        Dijkstra(n,S);

        printf("%d %d\n",d[E],cos[E]);

    }

    return 0;

}

8

### 宫锁珠帘

#### Problem:208

#### Time Limit:1000ms

#### Memory Limit:65536K

### Description

新年期间湖南卫视又上映了不少宫闱大戏，晴川穿走了，又来了个宫锁珠帘的袁珊珊，袁珊珊不够美，没锁住观众的心，特别是咱们整ACM的，对这些宫闱大戏当然是不屑一顾了，可这新亮剑的李云龙咱们还是有得一看的，最近听说A地又驻扎了一个骑兵营，李云龙心里就乐了，心想老子吃了骑兵营这么多亏，这下终于有机会自个儿也能整个啥骑兵营威风威风了，于是拿出地图决定抄最短的路立马拿下这个骑兵营。

### Input

每组数据第一行包含两个正整数N和M(0 < N < 100,0 < M < 100)，分别代表李云龙现有地图上显示的村子或者驻扎地的总数目，分别以0～N-1编号。

接下来是M行道路信息。每一行有三个整数A,B,X(0 <= A,B < N,A!=B,0< X < 10000),表示A和B之间有一条长度为X的双向道路。

再接下一行有两个整数S,T(0 <= S,T < N)，分别代表起点和终点。

### Output

输出从S到T的最短路的长度。若不存在这样的一条路，则输出-1.

### Sample Input

5 4

0 1 5

0 3 4

3 2 7

3 2 6

0 2

4 1

2 3 1

1 2

### Sample Output

10

-1

### Hint

题目与分析：

这道题抽象一下，还是“求从a到b的最短距离”。。同样可以使用floyd和dijkstra来做。。

这道题与前面的不同的地方在于：两个点之间可能有多条路(我们保存那条最短的即可)。

另外，还要理解dijkstra和floyd算法中使用到的map[][]矩阵的含义。

map[i][i] = 0.自己到自己的距离为0

map[i][j] = inf .表示两点之间无法连通

以下是分别用dijkstra、floyd、spfa这三种算法来做的代码，需要注意的是这道题顶点序号的范围是0~n-1，而之前做的题目的定点序号范围都是1~n。

/\*

\* NEFU\_208.cpp

\*

\* Created on: 2014年5月27日

\* Author: pc

\*/

#include <iostream>

#include <cstdio>

using namespace std;

const int maxn = 105;

const int inf = 10005;

int n;

int s[maxn];

int dis[maxn];

int map[maxn][maxn];

int target;

int dijkstra(int v){

int i;

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

s[i] = 0;

dis[i] = map[v][i];

}

for(i = 0 ; i < n-1 ; ++i){//这里的意思实际上是将剩下的n-1个点全部放到S集合中

int min = inf;

int pos;

int j;

for(j = 0 ; j < n ; ++j){//寻找最短路径点

if(!s[j] && dis[j] < min){

min = dis[j];

pos = j;

}

}

s[pos] = 1;

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

if(!s[j] && dis[j] > dis[pos] + map[pos][j]){

dis[j] = dis[pos] + map[pos][j];

}

}

}

return dis[target];

}

int main(){

int m;

while(scanf("%d%d",&n,&m)!=EOF){

int i;

int j;

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

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

if(i == j){

map[i][j] = 0;

}else{

map[i][j] = inf;

}

}

}

for(i = 1 ; i <= m ; ++i){

int a,b,c;

scanf("%d%d%d",&a,&b,&c);

if(map[a][b] > c){

map[a][b] = map[b][a] = c;

}

}

int start,end;

scanf("%d%d",&start,&end);

target = end;

int result = dijkstra(start);

if(result == inf){

printf("-1\n");

}else{

printf("%d\n",result);

}

}

return 0;

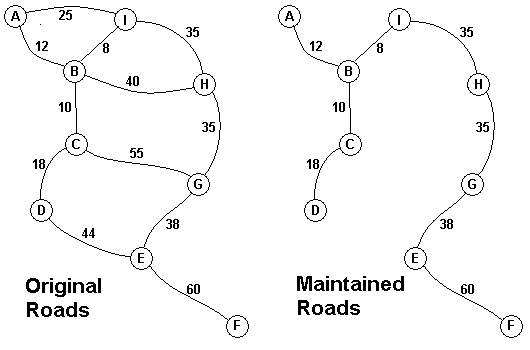
}

**并查集求最小生成树**

## **1824: J. Jungle**

Time Limit: 1 Sec  Memory Limit: 33 MB  
Submit: 8  Solved: 4  
[[Submit](http://218.27.6.151/oj/submitpage.php?id=1824)][[Status](http://218.27.6.151/oj/problemstatus.php?id=1824)][[Web Board](http://218.27.6.151/oj/bbs.php?pid=1824)]

## **Description**



The Head Elder of the tropical island of Lagrishan has a problem. A burst of foreign aid money was spent on extra roads between villages some years ago. But the jungle overtakes roads relentlessly, so the large road network is too expensive to maintain. The Council of Elders must choose to stop maintaining some roads. The map above on the left shows all the roads in use now and the cost in aacms per month to maintain them. Of course there needs to be some way to get between all the villages on maintained roads, even if the route is not as short as before. The Chief Elder would like to tell the Council of Elders what would be the smallest amount they could spend in aacms per month to maintain roads that would connect all the villages. The villages are labeled A through I in the maps above. The map on the right shows the roads that could be maintained most cheaply, for 216 aacms per month. Your task is to write a program that will solve such problems.   
  
The input consists of one to 100 data sets, followed by a final line containing only 0. Each data set starts with a line containing only a number n, which is the number of villages, 1 < n < 27, and the villages are labeled with the first n letters of the alphabet, capitalized. Each data set is completed with n-1 lines that start with village labels in alphabetical order. There is no line for the last village. Each line for a village starts with the village label followed by a number, k, of roads from this village to villages with labels later in the alphabet. If k is greater than 0, the line continues with data for each of the k roads. The data for each road is the village label for the other end of the road followed by the monthly maintenance cost in aacms for the road. Maintenance costs will be positive integers less than 100. All data fields in the row are separated by single blanks. The road network will always allow travel between all the villages. The network will never have more than 75 roads. No village will have more than 15 roads going to other villages (before or after in the alphabet). In the sample input below, the first data set goes with the map above.   
  
The output is one integer per line for each data set: the minimum cost in aacms per month to maintain a road system that connect all the villages. Caution: A brute force solution that examines every possible set of roads will not finish within the one minute time limit.

## **Input**

9

A 2 B 12 I 25

B 3 C 10 H 40 I 8

C 2 D 18 G 55

D 1 E 44

E 2 F 60 G 38

F 0

G 1 H 35

H 1 I 35

3

A 2 B 10 C 40

B 1 C 20

0

## **Output**

216

30

## **Sample Input**

9

A 2 B 12 I 25

B 3 C 10 H 40 I 8

C 2 D 18 G 55

D 1 E 44

E 2 F 60 G 38

F 0

G 1 H 35

H 1 I 35

3

A 2 B 10 C 40

B 1 C 20

0

## **Sample Output**

216

30

## **HINT**

## **Source**

#include<stdio.h>

int set[27];

void merge(int x ,int y)

{

set[x]=y;

}

int find(int x)

{

int tt;

tt=x;

while(set[tt]!=tt)

tt=set[tt];

return tt;

}

int main()

{

int t,i,j,k,sj,sk,p,q,n,r,a[27][27];

char c,d;

while(scanf("%d",&t)&&t)

{

for(i=0;i<=26;i++)

for(j=0;j<=26;j++)

{

a[i][j]=101;

set[j]=j;

}

q=t;

//printf("---%d\n",q);

while(--q)

{

//printf("---%d\n",q);

scanf("%\*c%c %d",&c,&n);

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

{

scanf("%\*c%c %d",&d,&k);

a[c-'A'][d-'A']=k;

}

}

r=sj=sk=0;

q=t;

while(--q){

p=101;

for(j=0;j<t;j++)

{

for(k=j+1;k<t;k++)

{

if(p>a[j][k])

{

if(find(j)!=find(k))

{

p=a[j][k];

sj=j;

sk=k;

}

}

}

}

r+=p;

merge(find(sj),find(sk));

// printf("-%d\n",q);

}

printf("%d\n",r);

}

return 0;

}

# 畅通工程

**Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 26026    Accepted Submission(s): 11350**

**Problem Description**

省政府“畅通工程”的目标是使全省任何两个村庄间都可以实现公路交通（但不一定有直接的公路相连，只要能间接通过公路可达即可）。经过调查评估，得到的统计表中列出了有可能建设公路的若干条道路的成本。现请你编写程序，计算出全省畅通需要的最低成本。

**Input**

测试输入包含若干测试用例。每个测试用例的第1行给出评估的道路条数 N、村庄数目M ( < 100 )；随后的 N   
行对应村庄间道路的成本，每行给出一对正整数，分别是两个村庄的编号，以及此两村庄间道路的成本（也是正整数）。为简单起见，村庄从1到M编号。当N为0时，全部输入结束，相应的结果不要输出。

**Output**

对每个测试用例，在1行里输出全省畅通需要的最低成本。若统计数据不足以保证畅通，则输出“?”。

**Sample Input**

3 3

1 2 1

1 3 2

2 3 4

1 3

2 3 2

0 100

**Sample Output**

3

?

**Source**

[浙大计算机研究生复试上机考试-2007年](http://acm.hdu.edu.cn/search.php?field=problem&key=%D5%E3%B4%F3%BC%C6%CB%E3%BB%FA%D1%D0%BE%BF%C9%FA%B8%B4%CA%D4%C9%CF%BB%FA%BF%BC%CA%D4-2007%C4%EA&source=1&searchmode=source)

**Recommend**

lcy

#include <stdio.h>

#include <stdlib.h>

#include <algorithm>

using namespace std;

const int N = 150;

int m,n,u[N],v[N],w[N],p[N],r[N];

int cmp(const int i,const int j)

{

return w[i] > w[j];

}

int find(int x)

{

return p[x]==x?x:p[x]=find(p[x]);

}

int kruskal()

{

int cnt=0,x,y,i,ans=0;

//n是点数，m是边数，汝佳那本书上是如此

//并查集初始化

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

p[i]=i;

//边编号

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

r[i]=i;

sort(r,r+m,cmp);

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

{

//取出未加入的边权最小的边的编号

int e=r[i];

x=find(u[e]);

y=find(v[e]);

if(x!=y)

{

ans += w[e];

p[x]=y;

cnt++;

}

}

//找不到最小生成树

if(cnt<n-1)

ans=0;

return ans;

}

int main()

{

int i,ans;

while(scanf("%d%d",&m,&n)!=EOF&&m)

{

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

{

scanf("%d%d%d",&u[i],&v[i],&w[i]);

}

ans=kruskal();

if(ans)

printf("%d\n",ans);

else //说明不存在最小生成树

puts("存在最小生成树!");

}

return 0;

}

**高效素数**

#include <stdio.h>

#define N 10000000

int a[N+5];

int main()

{

int sum=0;

int i,j;

a[0]=a[1]=0;

for (i=2;i<=N;i++)

if (i%2==0&&i!=2)

a[i]=0;

else

a[i]=1;

for (i=2;i<=N;i++)

if (a[i])

{

for (j=i\*2;j<=N;j+=i)

a[j]=0;

}

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

if (a[i])

sum++;

printf ("%d\n",sum);

return 0;

}

//寻找10000000以内素数的个数，codeblocks运行时间0.676s

**高效开方**

float InvSqrt(float x)

{

float xhalf = 0.5f\*x;

int i = \*(int\*)&x; // get bits for floating VALUE

i = 0x5f375a86- (i>>1); // gives initial guess y0

x = \*(float\*)&i; // convert bits BACK to float

x = x\*(1.5f-xhalf\*x\*x); // Newton step, repeating increases accuracy

x = x\*(1.5f-xhalf\*x\*x); // Newton step, repeating increases accuracy

x = x\*(1.5f-xhalf\*x\*x); // Newton step, repeating increases accuracy

return 1/x;

}

long fib(int n)

{

double z = sqrt(5.0);

double x = (1 + z)/2;

double y = (1 - z)/2;

return (pow(x, n) - pow(y, n))/z + 0.5;

}

**快速斐波那契数列求值算法**

long fib(int n)

{

double z = sqrt(5.0);

double x = (1 + z)/2;

double y = (1 - z)/2;

return (pow(x, n) - pow(y, n))/z + 0.5;

}

int div\_cou(long n)

{

int count=0;

int i;

for (i=1;i<=sqrt(n);i++)

{

if (i\*i==n)

count++;

else if (n%i==0)

count+=2;

}

return count;

}

**// 计算十亿的约数个数可以打表0.192s**

int div\_cou(long n)

{

int count=0;

int i;

for (i=1;i<=sqrt(n);i++)

{

if (i\*i==n)

count++;

else if (n%i==0)

count+=2;

}

return count;

}

// 计算十亿的约数个数可以打表0.192s

**Oj做题总结**

（一）排序方法

**1.选择排序**

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

scanf("%d",&a[i]);

for(i=0;i<n-1;i++)

{

k=i;

for(j=i+1;j<n;j++)

if(a[k]<a[j])

k=j;

if(j!=i)

{

t=a[i];

a[i]=a[k];

a[k]=t;

}

}

**2.冒泡排序**：

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

scanf("%d",&m[i]);

for(i=0;i<n-1;i++)

for(j=0;j<n-1-i;j++)

if(a[j]>a[j+1])

{

t=a[j];

a[j]=a[j+1];

a[j+1]=t;

}

**3如何在数列中插入数据：**

有n(n<=100)个整数，已经按照从小到大顺序排列好，现在另外给一个整数x，请将该数插入到序列中，并使新的序列仍然有序。

#include<stdio.h>

int main()

{

int m,n,i,j,a[100];

while(scanf("%d%d",&n,&m)&&n!=0&&m!=0)

{

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

scanf("%d",&a[i]);

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

{

if(m<a[i])

{

for(j=n-1;j>=i;j--)

a[j+1]=a[j];

a[i]=m;

break;

}

}

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

printf("%d ",a[i]);

printf("\n");

}

return 0;

}

**（二）大数运算**

**1. 求17（n）的倍数**

#include<stdio.h>

#include<math.h>

#include<string.h>

int main()

{

char a[1000]={'0'};

int i,j,l,temp,c=0;

while(gets(a))

{

temp=0;

l=strlen(a);

c++;

if(c>10||a[0]=='0') break;

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

**{**

**temp=temp\*10+(a[i]-'0');**

**temp%=17;**

**}**

if(temp==0)

printf("1\n");

else

printf("0\n");

}

return 0;

}

**2.自守数问题 //有规律**

#include <stdio.h>

int main()

{

int m=0,n=200000;

int i;

int t;

while(m<=n)

{

t=m\*(m-1);

i=1;

while(i<=m)

i\*=10;

if(t%i==0)

printf("%d ",m);

m++;

}

return 0;

}

3**二进制数转换**

#include<stdio.h>

#include<string.h>

int main()

{

char a[2000];

int x[2000]={0};

while(scanf("%s",a)!=EOF)

{

int len=strlen(a);

int m,n,k,j=0;

**for(;len!=0;len=k)**

**{**

**m=0,n,k=0;**

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

**{**

**n=m\*10+a[i]-'0';**

**m=n%2;**

**if(k==0&&n/2!=0)**

**a[k++]=n/2+'0';**

**else if(k!=0)**

**a[k++]=n/2+'0';**

**if(i==len-1)**

**x[j++]=m;**

**}**

}

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

printf("%d",x[i]);

printf("\n");

}

return 0;

}

**4.将十进制数转换为八进制数**

\*/

//我刚写了一个十进制转化为八进制的，比你要求的多一步，希望你自己好好看看啊

//你看程序时,自己手动的算一下十进制转化为二进制,以及二进制转化为八进制,这样就可以理解里面的意思了

/\*#include<stdio.h>

int a[100];//十进制数转化为八进制数 要先把十进制数转化为二进制 在把二进制转化为八进制

void fun2(int k)//二进制转化为八进制数

{

int i, sum, b[100], top = 0;

for(i = 0; i <= k - 3; i += 3)//每三位二进制转化为一位八进制数

{

b[top++] = a[i]\*1 + a[i + 1]\*2 + a[i + 2]\*4;//每三位二进制转化为一位八进制数 分别乘以 2^0 2^1 2^3

}

if(i == k - 2)//最后二进制还有三位没有算

{

b[top] = a[i]\*1 + a[i + 1]\*2 + a[i + 2]\*4;//每三位二进制转化为一位八进制数 分别乘以 2^0 2^1 2^3

}

else if(i == k - 1)//最后还有两位没算

{

b[top] = a[i]\*1 + a[i + 1]\*2;//每三位二进制转化为一位八进制数 分别乘以 2^0 2^1 2^3

}

else if(i == k)//只有一位没算了

{

b[top] = a[i]\*1;

}

printf("八进制数为: ");

for(top; top >= 0; top--)//栈, 先进入数组

{

printf("%d", b[top]);

}

printf("\n");

}

void fun(int k)//转化为二进制数

{

int top;

for(top = 0; k != 0; top++)

{

a[top] = k%2;

k /= 2;

}

fun2(top - 1);

}

int main()

{

int i, k;

printf("输入十进制数: ");

scanf("%d", &k);

fun(k);

return 0;

}

**（三）字符串处理**

**1.统计字母个数**

#include<stdio.h>

#include<string.h>

int main()

{

int i,t[27]={0};

char c;

while ((c=getchar()) && c!='#')

{ if(c>='a' && c<='z')

t[c-'a']++;

}

for(i=0;i<26;i++)

printf("%c %d\n",i+'a',t[i]);

return 0;

}

## 1189: 字符排列问题

## 题目描述

有n个字母，列出由该字母组成的字符串的全排列（相同的排列只计一次）。

## 输入

第一行输入是字母个数n，1<=n<=20。接下来一行输入的是待排列的n个字母。

## 输出

计算出的n 个字母的所有不同排列总数

## 样例输入

4

aacc

## 样例输出

6

#include<stdio.h>

#include<string.h>

int main()

{

int i,t[27]={0},n;

long long s=1,x=1,j,z=1;

char c;

scanf("%d",&n);

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

{

c=getchar();

if(c>='a' && c<='z')

t[c-'a']++;

}

getchar();

for(i=0;i<26;i++)

{ s=1;

if(t[i]==0)

t[i]=1;

for(j=1;j<=t[i];j++)

s=s\*j;

x=x\*s;

}

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

z=z\*i;

printf("%lld\n",z/x);

return 0;

}

2.十进制转换为二进制

#include<stdio.h>

#include<math.h>

int main()

{

int n,t=0,i,j,p;

int m[100]={0};

while( scanf("%d",&n) !=EOF)

{ i=0;

p=n;

if(n<0)

n=abs(n);

if(n==0)

printf("%11d-->0",n);

else

{

while(n>0)

{

m[i]=n%2;

n=(n-m[i])/2;

i++;

}

if(p<0)

printf("%11d-->-",p);

else

printf("%11d-->",p);

}

for(j=i-1;j>=0;j--)

printf("%d",m[j]);

printf("\n");

}

return 0;

}

**排序并去重**

#include <stdio.h>

#include <stdlib.h>

int cmp (const void \*a,const void \*b){

return \*((int \*)a)-\*((int \*)b); //一个函数

}

int main()

{

int arr[100];

int res[100];

int n;

scanf("%d",&n);

int i,tmp,count = 0;

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

scanf("%d",&arr[i]);

}

qsort(arr,n,sizeof(int),cmp);//排序

tmp=arr[0];

res[count]=tmp;

count++;

for(i=1;i<n;i++){//去重

if(arr[i]>tmp){

tmp = arr[i];

res[count++]=tmp;

}//if

}//for

printf("%d\n",count);

printf("%d",res[0]);

for(i=1;i<count;i++)

{

printf(" %d",res[i]);

}

printf("\n");

return 0;

}

## 求众数问题

**如何将出现次数最多的数字统计出来以及记录下它的下标**

#include <stdio.h>

int main()

{

int a[50];

int i,j,maxCount=0,index=0,nCount=0;

int n;

scanf("%d",&n);

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

{

scanf("%d",&a[i]);

}

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

{

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

{

if(a[j]==a[i])

nCount++;

}

if(nCount>maxCount)

{

maxCount=nCount;

index=i;

}

nCount=0;

}

printf("%d\n%d\n",a[index],maxCount);

}

# **c语言问题:n人围成一圈，顺序报数.从1到M，凡报到5的退出圈子，问最后剩下是原来的第几号那位？**

#include<stdio.h>

void main()

{

int \*p,m,n,h,num[10000];int i=0,k=0;

while(scanf("%d%d",&n,&h)!=EOF)

{

p=num;

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

\*(p+i)=i+1;

i=0;

k=0;

m=0;

while(m<n-1)

{

if(\*(p+i)!=0)

k++;

if(k==h)

{

\*(p+i)=0;

k=0;

m++;

}

i++;

if(i==n)

i=0;

}

while(\*p==0)

p++;

printf("%d\n",\*p);}

}

**如何输入多行后，计算每一行的最大值（不用数组）**

#include<stdio.h>

int main()

{

 int i=0,a,max;

 while(scanf("%d",&max)!=EOF)

 {

     i=0;

     while(scanf("%d",&a) && a!=0)

    {

         if(max<a) max=a;

    }

     printf("%d\n",max);

 }

 return 0;

}

WHILE（1）

{

---------------------------；

----------------------------；

}

**该种编程方法会导致程序无法结束，输出超限。**

**判断是否为回文数**

**#include <stdio.h>**

**int Palindrome(int num){//求一个数的回文数。。**

**int ret = 0, m;**

**while(num){**

**m = num % 10;**

**ret = 10 \* ret + m;**

**num /= 10;**

**}**

**return ret;**

**}**

**int main(){**

**int cas, i, num, pnum;**

**scanf("%d", &cas);**

**while(cas--){**

**scanf("%d", &num);**

**for(i = 0; i <= 8; ++i){**

**pnum = Palindrome(num);**

**if(pnum == num) break;**

**num += pnum;**

**}**

**if(i <= 8) printf("%d\n", i);**

**else printf("0\n");**

**}**

**return 0;**

**该种格式是这样编的，吸收空格，**

2

a100 zhblue 70 80 90

b200 newsclan 90 85 75

## Sample Output

a100,zhblue,70,80,90

b200,newsclan,90,85,75

#include<stdio.h>

#include<string.h>

int main()

{

int i=0,n,k,j=0;

char a[100][100];

scanf("%d",&n);

getchar();

while(i<n)

{

gets(a[i]);

int m;

m=strlen(a[i]);

for(j=0;j<m;j++)

{

if(a[i][j]==' ')

a[i][j]=',';

}

printf("%s",a[i]);

printf("\n");

i++;

}

return 0;

}

**求最大公约数和最小公倍数**

#include<stdio.h>   
int main()   
{   
int a,b,imax,i[min](https://www.baidu.com/s?wd=min&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvuWPWryNWuH63rH63uAwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1TLnHnsP1b3" \t "_blank),tend;   
int max(int,int);   
printf("please insert two [num](https://www.baidu.com/s?wd=num&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvuWPWryNWuH63rH63uAwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1TLnHnsP1b3" \t "_blank)bers:");   
scanf("%d %d",&a,&b);   
if(a<b)   
{   
tend=a;   
a=b;   
b=tend;   
}   
imax=max(a,b);   
i[min](https://www.baidu.com/s?wd=min&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvuWPWryNWuH63rH63uAwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1TLnHnsP1b3" \t "_blank)=a\*b/imax;   
printf("最大公约数是%d\n",imax);   
printf("最小公倍数是%d\n",i[min](https://www.baidu.com/s?wd=min&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvuWPWryNWuH63rH63uAwB0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6K1TL0qnfK1TL0z5HD0IgF_5y9YIZ0lQzqlpA-bmyt8mh7GuZR8mvqVQL7dugPYpyq8Q1TLnHnsP1b3" \t "_blank));   
}   
int max(int x,int y)   
{   
int z;   
while(x%y!=0)   
{   
z=x%y;   
x=y;   
y=z;   
}   
return(y);   
}

**C语言中将字符串转换为数字：**

C语言有atoi、atol、atof等库函数，可分别把ASCII编码的字符串转化为int、long、float类型的数字。  
头文件：stdlib.h  
函数原型：int atoi(const char\* nptr);  
（另外两个类似）  
  
举个例子：  
char \*str="123";  
int num=atoi(str);  
执行后，num的值被初始化为123

举例：

将字符串12转换为数字12

Int a；

Char b[10];

Scanf(“%s”,b);

A =atoi(b);

可以参看例子1081,这道题就用了这个方法、

**最长不降子序列（动态规划）**

***类型一：***

**1875: MaxSubSequenceSum**

给定K个整数的序列{ N1, N2, ..., NK }，其任意连续子序列可表示为{ Ni, Ni+1, ..., Nj }，其中 1 <= i <= j <= K。最大连续子序列是所有连续子序列中元素和最大的一个,编写程序输出最大连续子序列和。

Input

测试输入包含若干测试用例，每个测试用例占2行，第1行给出正整数K (K < 100 )，第2行给出K个整数，中间用空格分隔。当K为0时，输入结束，该用例不被处理。

Output

对于每个测试实例，输出最大连续子序列和。

Sample Input

6

-2 11 -4 13 -5 -2

Sample Output

20

代码：

#include<stdio.h>

void main()

{

int j,i,t,k,maxs,n;

while(scanf("%d",&n)!=EOF)

{

int a[700];

maxs=-1000000;

int p=0;

int s=0;

if(n==0)

break;

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

{

scanf("%d",&a[i]);

if(a[i]<0)

p++;

}

if(p==n)

{

for(i=0;i<n-1;i++)

{

k=i;

for(j=i+1;j<n;j++)

if(a[k]<a[j])

k=j;

if(j!=i)

{

t=a[i];

a[i]=a[k];

a[k]=t;

}

}

printf("%d\n",a[0]);

}

else

{

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

{

s=s+a[i];

if(s<0)

{

s=0;

}

if(s>maxs)

{

maxs=s;

}

}

printf("%d\n",maxs);

}

}

}

***类型二：***

**1855: maximal sub-rectangle**

Description

Given a two-dimensional array of positive and negative integers, a sub-rectangle is any contiguous sub-array of size 1\*1 or greater located within the whole array. The sum of a rectangle is the sum of all the elements in that rectangle. In this problem the sub-rectangle with the largest sum is referred to as the maximal sub-rectangle.

As an example, the maximal sub-rectangle of the array:

0 -2 -7 0

9 2 -6 2

-4 1 -4 1

-1 8 0 -2

is in the lower left corner:

9 2

-4 1

-1 8

and has a sum of 15.

Input

The input consists of an N \* N array of integers. The input begins with a single positive integer N on a line by itself, indicating the size of the square two-dimensional array. This is followed by N^2 integers separated by whitespace (spaces and newlines). These are the N^2 integers of the array, presented in row-major order. That is, all numbers in the first row, left to right, then all numbers in the second row, left to right, etc. N may be as large as 100. The numbers in the array will be in the range [-127,127].

Output

Output the sum of the maximal sub-rectangle.

Sample Input

4

0 -2 -7 0 9 2 -6 2

-4 1 -4 1 -1

8 0 -2

Sample Output

15

HINT

Source

动态规划

代码：

#include<stdio.h>

#include<string.h>

#define MAXN 105

int main()

{

int i,j,k,n,t,sum,max;

int a[MAXN][MAXN];

while (scanf("%d",&n)!=EOF)

{

memset(a,0,sizeof(a));

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

{

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

{

scanf("%d",&t);

a[i][j]=a[i-1][j]+t;

}

}

max=0;

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

{

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

{

sum=0;

for (k=1;k<=n;++k)

{

t=a[j][k]-a[i-1][k];

sum+=t;

if (sum<0) sum=0;

if (sum>max) max=sum;

}

}

}

printf("%d\n",max);

}

return 0;

}

***类型三：***

**1746: 最长不下降子序列**

Time Limit: 0 Sec Memory Limit: 128 MB

Submit: 59 Solved: 23

[Submit][Status][Web Board]

Description

求最长不下降子序列的长度

Input

第一行为n，表示n个数 第二行n个数

Output

最长不下降子序列的长度

Sample Input

3

1 2 3

Sample Output

3

HINT

N小于5000

for each num < =maxint

Source

动态规划

代码：

#include<stdio.h>

int main()

{

    int n,i,j,t,k,max;

    while(scanf("%d",&n)!=EOF)

    {   max=0;

       int b[5005],a[5005],s[5005];

    for(i=0;i<n;i++)

        scanf("%d",&s[i]);

        a[0]=1;

    for(i=1;i<n;i++)

    {

        max=0;

        int b[5005]={0};

      for(j=0;j<i;j++)

      {

          if(s[i]>=s[j])

              b[j]=a[j];

      }

           max=b[0];

        for(k=0;k<i;k++)

        {

            if(b[k]>max)

              max=b[k];

            a[i]=max+1;

    }   }

          max=a[0];

      for(i=0;i<n;i++)

        if(a[i]>max)

          max=a[i];

        printf("%d\n",max);}

    return 0;

}

**求质方数：**

小明天生对数字比较敏感，3岁的时候就能背诵圆周率一百位。 现在，小明慢慢长大了，但依然很喜欢数字，最近，他迷上了质数和平方数，并且自己把质数的平方命名为“质方数”。 现在，他在研究这样一个问题：距离一个正整数N最接近的质方数是多少？

#include<stdio.h>

#include<math.h>

#define Max 1000000

int a[Max+1]={1,1,0};

void main()

{

int i,j,temp;

for( i=2; i<=500000; i++ )

for( j=i<<1; j<Max+1; j+=i )

{

a[j]=1;

}//将非质数挑出来（一定要记住）

int T;

int N;

scanf("%d",&T);

while(T--)

{

scanf("%d",&N);

temp=sqrt(N);

if( temp\*temp == N && a[temp]==0 )

printf("%d\n",N);

else if( temp==1 )

printf("4\n");

else{

int i, j;

for( i=temp; i>=2; i-- ){

if( a[i]==0 )

break;

}

for( j=temp+1; j<=Max; j++ ){

if( a[j]==0 )

break;

}

if( N-i\*i <= j\*j-N )

printf("%d\n",i\*i);

else

printf("%d\n",j\*j);

}

}

}

求连续自然数和的简便算法：

Oj :1243

#include<stdio.h>

int main()

{

int a,i,n=0,m=0,s,he;

while(scanf("%d",&a)!=EOF)

{

for(n=1;n<a/2+1;n++)

{

s=0;

i=0;

for(m=n;s<a;m++)

{

s=s+m;

i++;

}

if(s==a)

{

printf("%d %d\n",m-i,m-1);

}

}

printf("\n");

}

return 0;

}

## 1926: 热烈鼓掌！！！

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 22  Solved: 11  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1926)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1926)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1926)]

## Description

玛 法里奥喜欢的妹子将参加歌唱比赛，可惜他去不了。但他希望妹子唱完歌后所有观众都能为她鼓掌。已知每个观众都有一个害羞值S,表示当有S个观众鼓掌时，他 也会跟着鼓掌。S=0表示无论如何都会鼓掌。玛法里奥想知道他最少需要安插多少自己人进去带节奏才能使所有观众都为妹子鼓掌。

## Input

多组样例，每组一行，显示一个整数Smax(0 <= Smax <= 1000)，表示观众最大的害羞值，紧接着是一个长度为Smax+1的字符串，字符串的第i位表示害羞值为i的观众人数(i从0开始计数，相同害羞值的观众数不超过9个)

## Output

玛法里奥需要安插的最少人数。

## Sample Input

4 11111

1 09

5 110011

0 1

## Sample Output

0

1

2

0

## HINT

注意：当连续输入一个数一个字符串时，这么写：

#include<stdio.h>

#include<string.h>

int main()

{

int Smax,i,m,n,len;

char s[1001];

while(scanf("%d",&Smax)!=EOF)

{

m=0;

n=0;

scanf("%s",s);

len=strlen(s);

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

{

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

if(s[i+1]!='0'&&m<(i+1))

{

n=i+1-m;

m+=n;

}

}

printf("%d\n",n);

}

return 0;

}

**分糖果**

## 1927: 分糖果

时间限制: 1 Sec  内存限制: 128 MB  
提交: 24  解决: 6  
[[提交](http://10.5.54.252/acm/submitpage.php?id=1927)][[状态](http://10.5.54.252/acm/problemstatus.php?id=1927)][[讨论版](http://10.5.54.252/acm/bbs.php?pid=1927)]

## 题目描述

Wiyr有n颗糖果，每颗糖果有一个美味值。Wiyr想要把这些糖果分成两部分，使得两部分糖果的美味值总和相等。注意，两部分糖果的数量可以不相等。

## 输入

多组数据读入。

每组数据第一行是一个正整数n(1<=n<=15),表示Wiyr有n颗糖果。

接下来一行有n个正整数ai(1<=ai<=10^8)，表示每颗糖果的美味值。

## 输出

如果Wiyr能够将糖果分成如题目描述的两部分，输出”Yes”， 否则输出”No”

## 样例输入

3

1 2 3

2

2 4

## 样例输出

Yes

No

#include<stdio.h>

#include<math.h>

#include<string.h>

int main()

{

int n,i,j,k,t,a[100];

while(scanf("%d",&n)!=EOF)

{

int sum=0;

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

{

scanf("%d",&a[i]);

sum+=a[i];

}

if(n==1||sum%2!=0)

{

printf("No\n");

continue;

}

else

{

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

{

k=i;

for(j=k;j<n;j++)

{

if(a[k]<a[j])

k=j;

}

if(i!=k)

{

t=a[k];

a[k]=a[i];

a[i]=t;

}

}

int s=0;

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

{

s+=a[i];

if(s==sum/2)

{

printf("Yes\n");

break;

}

else if(s>sum/2)

s-=a[i];

}

if(i==n)

printf("No\n");

}

}

return 0;

}

**关于C语言中字符串结束符的’\0’的用法**

## 1177: 去掉双斜杠注释

时间限制: 1 Sec  内存限制: 128 MB  
提交: 43  解决: 10  
[[提交](http://10.5.54.252/acm/submitpage.php?id=1177)][[状态](http://10.5.54.252/acm/problemstatus.php?id=1177)][[讨论版](http://10.5.54.252/acm/bbs.php?pid=1177)]

## 题目描述

|  |
| --- |
| 将C程序代码中的双斜杠注释去掉。 |

## 输入

输入数据中含有一些符合C++语法的代码行。需要说明的是，为了方便编程，规定双斜杠注释内容不含有双引号,源程序中没空行。

## 输出

输出不含有双斜杠注释的C++代码，除了注释代码之外，原语句行格式不变。

## 样例输入

//======================

// simplest program

//======================

#include

using namespace std;

//----------------------

int main(){

cout<<”hello world!\n”;

}//---------------------

## 样例输出

#include

using namespace std;

int main(){

cout<<”hello world!\n”;

}

#include <stdio.h>

#include<string.h>

int main()

{

char a[100000];

int i,j,n;

while(gets(a)!=NULL)

{

n=strlen(a);

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

if(a[i]=='/'&&a[i+1]=='/')

a[i]='\0';//截断字符串

if(a[0]!='\0') // ‘\0’为结束符标志

printf("%s\n",a);

}

return 0;

}

**1415（java运算大数）**

## 1415: Ones

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 56  Solved: 35  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1415)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1415)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1415)]

## Description

Given any integer 0 <= n <= 10000 not divisible by 2 or 5, some multiple of n is a number which in decimal notation is a sequence of 1's. How many digits are in the smallest such a multiple of n?

## Input

## Output

## Sample Input

3

7

9901

## Sample Output

3

6

12

## HINT

import java.math.\*;

import java.util.Scanner;

    public class Main {

        public static void main(String[] args)

        {

            Scanner ss=new Scanner(System.in);

            BigInteger a;

            BigInteger y,x,s,n,m;

            y=BigInteger.valueOf(10);

            x=BigInteger.valueOf(0);

            while(ss.hasNext())

            {

               int k=0,i=0;

               s=BigInteger.valueOf(0);

               n=ss.nextBigInteger();

              do

              {

                    m=y.pow(i);

                    s=s.add(m);

                    i=i+1;

                    k=k+1;

                //  System.out.println(m);

              }

              while(!(s.mod(n)).equals(x));

              System.out.println(k);

            }

        }

}

**Java大数总结**

## Java 大数运算

(2011-04-16 18:16:54)

*[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)](javascript:;)*[转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签： [大整数](http://search.sina.com.cn/?c=blog&q=%B4%F3%D5%FB%CA%FD&by=tag" \t "_blank)[加](http://search.sina.com.cn/?c=blog&q=%BC%D3&by=tag" \t "_blank)[减](http://search.sina.com.cn/?c=blog&q=%BC%F5&by=tag" \t "_blank)[乘](http://search.sina.com.cn/?c=blog&q=%B3%CB&by=tag" \t "_blank)[除](http://search.sina.com.cn/?c=blog&q=%B3%FD&by=tag" \t "_blank) |  |

**import** java.util.\*;

**import** java.math.\*;

**public** **class** Main{

**public** **static** **void** main(String args[]){

Scanner cin = **new** Scanner(System.*in*);

BigInteger a, b;

//以文件EOF结束

**while** (cin.hasNext()){

a = cin.nextBigInteger();

b = cin.nextBigInteger();

System.*out*.println(a.add(b)); //大整数加法

System.*out*.println(a.subtract(b)); //大整数减法

System.*out*.println(a.multiply(b)); //大整数乘法

System.*out*.println(a.divide(b)); //大整数除法(取整)

System.*out*.println(a.remainder(b)); //大整数取模

//大整数的比较

**if**( a.compareTo(b) == 0 ) System.*out*.println("a == b"); //大整数a==b

**else** **if**( a.compareTo(b) > 0 ) System.*out*.println("a > b"); //大整数a>b

**else** **if**( a.compareTo(b) < 0 ) System.*out*.println("a < b"); //大整数a<b

//大整数绝对值

System.*out*.println(a.abs()); //大整数a的绝对值

//大整数的幂

**int** exponent=10;

System.*out*.println(a.pow(exponent)); //大整数a的exponent次幂

//返回大整数十进制的字符串表示

System.*out*.println(a.toString());

//返回大整数p进制的字符串表示

**int** p=8;

System.*out*.println(a.toString(p));

}

}

}

## 1390: Hanoi双塔问题

## Description

给定A,B,C三根足够长的细柱，在A柱上放有2n个中间有空的圆盘，共有n个不同的尺寸，每个尺寸都有两个相同的圆盘，注意这两个圆盘是不加区分的(下图为n=3的情形）。现要将 这些国盘移到C柱上，在移动过程中可放在B柱上暂存。要求:



(1)每次只能移动一个圆盘；

(2) A、B、C三根细柱上的圆盘都要保持上小下大的顺序；

任务:设An为2n个圆盘完成上述任务所需的最少移动次数，对于输入的n，输出An。

## Input

输入文件hanoi.in为一个正整数n，表示在A柱上放有2n个圆盘。

## Output

输出文件hanoi.out仅一行，包含一个正整数，为完成上述任务所需的最少移动次数An。

## Sample Input

1

## Sample Output

2

## HINT

对于50%的数据， 1<=n<=25

对于100% 数据， 1<=n<=200

**设法建立An与An-1的递推关系式**。

import java.math.\*;

    import java.util.Scanner;

    public class Main {

        public static void main(String[] args)

        {

            Scanner ss=new Scanner(System.in);

            BigInteger a[]=new BigInteger[500];

            BigInteger y,x;

            y=BigInteger.valueOf(2);

            x=BigInteger.valueOf(1);

            //a[1]=BigInteger.valueOf(1);

            a[1]=BigInteger.valueOf(2);

              for(int i=2;i<=400;i++)

                  a[i]=y.pow(i+1).subtract(y);

            while(ss.hasNext())

            {

              int n=ss.nextInt();

              System.out.println(a[n]);

            }

        }

}

## 1817: Game of Connections

Time Limit: 1 Sec  Memory Limit: 33 MB  
Submit: 14  Solved: 9  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1817)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1817)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1817)]

## Description

This is a small but ancient game. You are supposed to write down the numbers 1, 2, 3, ... , 2n - 1, 2n consecutively in clockwise order on the ground to form a circle, and then, to draw some straight line segments to connect them into number pairs. Every number must be connected to exactly one another. And, no two segments are allowed to intersect.  
  
It's still a simple game, isn't it? But after you've written down the 2n numbers, can you tell me in how many different ways can you connect the numbers into pairs? Life is harder, right?

## Input

Each line of the input file will be a single positive number n, except the last line, which is a number -1. You may assume that 1 <= n <= 100.

## Output

For each n, print in a single line the number of ways to connect the 2n numbers into pairs.

## Sample Input

2

3

-1

## Sample Output

2

5

import java.math.BigInteger;

import java.util.\*;

public class Main {

 public static void main(String[] args) {

  Scanner ss=new Scanner(System.in);

  BigInteger[] a=new BigInteger[101];

  int i;

  BigInteger j;

  a[1]=BigInteger.ONE;

  for(i=2;i<101;i++)

  {

   j=BigInteger.valueOf(4\*i-2);

   a[i]=a[i-1].multiply(j);

   j=BigInteger.valueOf(i+1);

   a[i]=a[i].divide(j);

  }

  while(ss.hasNextInt())

  {

   i=ss.nextInt();

   if(i==-1) break;

   else{

    System.out.println(a[i]);

   }

  }

 }

}

## 1241: 诡秘的余数

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 40  Solved: 23  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1241)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1241)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1241)]

## Description

不 可否认，小强是一名数学天才，大家都这么说。天才小强的两大最新发现如下：（1） 正整数n除3的余数，等价于，正整数n的各位数字之和除3的余数；（2） 正整数n除9的余数可以通过这样的方法来计算：计算n 的各位数之和，设为m，如果m已经是一位数，那么余数就是m；否则设n=m，重新进行计算n的各位数之和m，直到m是一个一位数。但是，正整数除 1，2，4，5，6，7，8，也存在类似的性质吗？这真是一个难题啊！小强想睡觉了，不去管了。现在请你计算一下正整数n除一位数m的余数。文件中有一些 数对，一为大整数（可能大到100位）n，另一为一位数m（>0）。求其n除以m的余数。

## Input

## Output

## Sample Input

23 7

123 9

## Sample Output

2

6

import java.math.\*;

import java.util.\*;

public class Main {

    public static void main(String[] arge)

    {

        BigInteger X,Y;

        String s;

        Scanner scanner =new Scanner(System.in);

        while(scanner.hasNext())

        {

        X=scanner.nextBigInteger();

        Y=scanner.nextBigInteger();

        X=X.remainder(Y);

        System.out.println(X);

        }

    }

}

## 1219: A + B Problem II

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 168  Solved: 26  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1219)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1219)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1219)]

## Description

I have a very simple problem for you. Given two integers A and B, your job is to calculate the Sum of A + B.

## Input

The first line of the input contains an integer T(1<=T<=20) which means the number of test cases. Then T lines follow, each line consists of two positive integers, A and B. Notice that the integers are very large, that means you should not process them by using 32-bit integer. You may assume the length of each integer will not exceed 1000.

## Output

For each test case, you should output two lines. The first line is "Case #:", # means the number of the test case. The second line is the an equation "A + B = Sum", Sum means the result of A + B. Note there are some spaces int the equation. Output a blank line between two test cases.

## Sample Input

2

1 2

112233445566778899 998877665544332211

## Sample Output

Case 1:

1 + 2 = 3

Case 2:

112233445566778899 + 998877665544332211 = 1111111111111111110

import java.math.\*;

import java.util.\*;

public class Main {

    public static void main(String[] arge)

    {

        BigInteger X,Y,Z;

        int A;

        Scanner scanner =new Scanner(System.in);

        A=scanner.nextInt();

        for(int i=1;i<=A;i++)

        {

          X=scanner.nextBigInteger();

          Y=scanner.nextBigInteger();

          Z=X.add(Y);

          System.out.println("Case "+i+":");

          System.out.println(X+" + "+Y+" = "+Z+"\n");

        }

        }

}

**B:brother**

## Description

Adam gets to know a new friend, Bob, recently. One day, they realize that they are family 500 years ago. Now, Adam wants to know whether Bob is his elder, younger or brother.

## Input

There are multiple test cases.

For each test case, the first line has a single integer, n (n<=1000). The next n lines have two integers a and b (1<=a,b<=2000) each, indicating b is the father of a. One person has exactly one father, of course. Adam is numbered 1 and Bob is numbered 2.

Proceed to the end of file.

## Output

For each test case, if  Bob is Adam’s younger, print “You are my younger”. Otherwise, if Bob is Adam’s elder, print “You are my elder”. Otherwise, print “You are my brother”. The output for each test case occupied exactly one line.

## Sample Input

5

1 3

2 4

3 5

4 6

5 6

6

1 3

2 4

3 5

4 6

5 7

6 7

## Sample Output

You are my elder

You are my brother

1—3—5—6

2—4—6

1是2

#include<stdio.h>

int s[2001],y;

**int find(int x)**

**{**

**y=0;**

**int i=x;**

**while(s[i]!=0)**

**{**

**i=s[i];**

**y++;**

**}**

**return y;**

**} //该段写得好 （记录下来了）**

int main()

{

    int N;

    while(scanf("%d",&N)!=EOF)

    {

        int i,j;

        for(i=0;i<2001;i++)

            s[i]=0;

        for(i=0;i<N;i++)

        {

            scanf("%d",&j);

            scanf("%d",&s[j]);

        }

        if(find(1)>find(2))

            printf("You are my elder\n");

        else

            if(find(1)==find(2))

                printf("You are my brother\n");

            else

                printf("You are my younger\n");

    }

    return 0;

}

## 1650: 区间交交（明显的结构体存储）

刘备今年终于进入浙江传媒学院了，想当年在高中的那场数学之战真是惊心动魄。为了能够顺利借到东风，诸葛亮将两组区间放在一起。将它们共同的部分提取出来，然后向上苍祷告。于是玉皇大帝就命令风神助卧龙一臂之力了。

如今，刘备问卧龙，曰：“军师，How long 你提取出的区间？”卧龙笑而不语，羽扇一扇便说，他的学生可以告诉刘备！你可以将这两组区间相交的部分在数轴上线段的长度告诉刘备么？

## Input

数据只有两行代表当年诸葛亮用来提取的两组区间。每行的第一个自然数N (0<=N<100)表示该组区间的数量，接着N对小数si,ei表示区间的开始和结束。

## Output

输出区间交集的长度输出来，结果保留三位小数。

## Sample Input

2 1 10 14 17

3 3 7 9 15 16 18

## Sample Output

7.000

#include<stdio.h>

#include<string.h>

struct t

{

double b;

double e;

}t[101];

int main()

{

int n,m,i,j;

double x,y,s=0;

scanf("%d",&n);

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

{

scanf("%lf%lf",&t[i].b,&t[i].e);

}

scanf("%d",&m);

while(m--)

{

scanf("%lf%lf",&x,&y);

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

{

if(x>t[j].b&&y<t[j].e)

{

s+=y-x;

}

else if(x<t[j].b&&y>t[j].e)

{

s+=t[j].e-t[j].b;

}

else if(x>t[j].b&&x<t[j].e&&y>t[j].e)

{

s+=t[j].e-x;

}

else if(y>t[j].b&&y<t[j].e&&x<t[j].b)

s+=y-t[j].b;

}

}

printf("%.3lf\n",s);

return 0;

}

搜索算法

**1323 选数**

## 1323: 选数

时间限制: 1 Sec  内存限制: 128 MB  
提交: 23  解决: 15  
[[提交](http://10.5.54.252/acm/submitpage.php?id=1323)][[状态](http://10.5.54.252/acm/problemstatus.php?id=1323)][[讨论版](http://10.5.54.252/acm/bbs.php?pid=1323)]

## 题目描述

　　已知 n 个整数 x1,x2,…,xn，以及一个整数 k（k＜n）。从 n 个整数中任选 k 个整数相加，可分别得到一系列的和。例如当 n=4，k＝3，4 个整数分别为 3，7，12，19 时，可得全部的组合与它们的和为：  
3＋7＋12=22　　3＋7＋19＝29　　7＋12＋19＝38　　3＋12＋19＝34。  
现在，要求你计算出和为素数共有多少种。  
例如上例，只有一种的和为素数：3＋7＋19＝29）。

## 输入

键盘输入，格式为：  
n , k （1<=n<=20，k＜n）  
x1,x2，…,xn （1<=xi<=5000000）

## 输出

　屏幕输出，格式为：  
一个整数（满足条件的种数）。

## 样例输入

4 3

3 7 12 19

## 样例输出

1

## 提示

#include <stdio.h>

int n,k,x[21],y;

void dfs(int i,int m,int s)

{

int j;

if(m==k&&isPrime(s))

{

y++;

return;

}

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

**dfs(j+1,m+1,s+x[j]);//递归**

}

int main()

{

int i;

y=0;

scanf("%d%d",&n,&k);

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

scanf("%d",&x[i]);

dfs(0,0,0);

printf("%d\n",y);

return 0;

}

## 1505: How many 0

Time Limit: 10 Sec  Memory Limit: 128 MB  
Submit: 24  Solved: 10  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1505)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1505)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1505)]

## Description

A Benedict monk No. 16 writes down the decimal representations of all natural numbers between and including

**m and** **n , m≤ n**

. How many 0's will he write down?

Input consists of a sequence of lines. Each line contains two unsigned 32-bit integers **m** and **n**, **m** ≤ **n**. The last line of input has the value of **m** negative and this line should not be processed.

For each line of input print one line of output with one integer number giving the number of 0's written down by the monk.

## Input

## Output

## Sample Input

10 11

100 200

0 500

1234567890 2345678901

0 4294967295

-1 -1

## Sample Output

1

22

92

987654304

3825876150

## HINT

相关：[1206](http://cst2.jlnu.edu.cn/acm/problem.php?id=1206)

#include <stdio.h>

#include <math.h>

long long f(long long x)

{

    long long i,y=1;

    for(i=0; i<x; i++)

    y\*=10;

    return y;

}

int main()

{

    long long i,k,m,k1,m1,s=0,h=0;

    long long a[20]={0},n1,n2,b,p,p1,a1[20]={0},b1,p2;

    while(scanf("%lld%lld",&n1,&n2)!=EOF)

    {

        if(n1==-1&&n2==-1)

        break;

        if(n1>=2)

        {

        h=n1;

        n1=n1-1;

**for(i=0;i<11;i++)**

**a1[i]=0;**

**m1=(long long)log10(n1);**

**k1=m1;**

**while(k1>=0)**

**{**

**p1=f(k1);**

**b1=n1/p1;**

**n1%=p1;**

**for(i=(k1==m1?1:0);i<b1;i++)**

**a1[i]+=p1;**

**a1[b1]+=(n1+1);**

**for(i=0;i<10;i++)**

**a1[i]+=p1\*k1\*b1/10;**

**k1--;**

**}**

**while(m1--)**

**a1[0]-=f(m1); //一个结构集合，关键处，同下题**

       for(i=0; i<11; i++)

       a[i]=0;

        m=(long long)log10(n2);

        k=m;

        while(k>=0)

        {

            p=f(k);

            b=n2/p;

            n2%=p;

            for(i=(k==m?1:0);i<b;i++)

              a[i]+=p;

              a[b]+=(n2+1);

            for(i=0;i<10;i++)

              a[i]+=p\*k\*b/10;

              k--;

      }

     while(m--)

     a[0]-=f(m);

     printf("%lld\n",a[0]-a1[0]);

    }

     else

     {

        for(i=0; i<11; i++)

        a[i]=0;

        m=(long long)log10(n2);

        k=m;

        while(k>=0)

        {

            p=f(k);

            b=n2/p;

            n2%=p;

            for(i=(k==m?1:0);i<b;i++)

              a[i]+=p;

              a[b]+=(n2+1);

            for(i=0;i<10;i++)

              a[i]+=p\*k\*b/10;

              k--;

       }

      while(m--)

      a[0]-=f(m);

      if(n1==1)

      printf("%lld\n",a[0]);

      else

      printf("%lld\n",a[0]+1);

     }

  }

return 0;

}

## 1206: 数字统计（与上一题有密切关系）

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 61  Solved: 31  
[[Submit](http://cst2.jlnu.edu.cn/acm/submitpage.php?id=1206)][[Status](http://cst2.jlnu.edu.cn/acm/problemstatus.php?id=1206)][[Web Board](http://cst2.jlnu.edu.cn/acm/bbs.php?pid=1206)]

## Description

一本书的页码从自然数1 开始顺序编码直到自然数n。书的页码按照通常的习惯编排，   
每个页码都不含多余的前导数字0。例如，第6 页用数字6 表示，而不是06 或006 等。数   
字计数问题要求对给定书的总页码n，计算出书的全部页码中分别用到多少次数字0，1，   
2，…，9。

## Input

给出表示书的总页码的整数n（1≤n≤2^31-1）

## Output

输出10行，在第k行输出页码中用到数字k-1 的次数，k=1，2，…，10。

## Sample Input

11

## Sample Output

1

4

1

1

1

1

1

1

1

1

## HINT

#include <stdio.h>

#include <math.h>

int f(int x)

{

int i,y=1;

for(i=0; i<x; i++)y\*=10;

return y;

}

int main()

{

int i,k,m;

long a[10],n,b,p;

while(scanf("%lld",&n)!=EOF)

{

for(i=0; i<10; i++)a[i]=0;

m=(int)log10(n);

k=m;

while(k>=0)

{

p=f(k);

b=n/p;

n%=p;

for(i=(k==m?1:0); i<b; i++)a[i]+=p;

a[b]+=(n+1);

for(i=0; i<10; i++)a[i]+=p\*k\*b/10;

k--;

}

while(m--)a[0]-=f(m);

for(i=0;i<10;i++)

printf("%lld\n",a[i]);

}

return 0;

}

## 1180: 取石子游戏

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 13  Solved: 2  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1180)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1180)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1180)]

## Description

有 两堆石子，数量任意，可以不同。游戏开始由两个人轮流取石子。游戏规定，每次有两种不同的取法，一是可以在任意的一堆中取走任意多的石子；二是可以在两堆 中同时取走相同数量的石子。最后把石子全部取完者为胜者。现在给出初始的两堆石子的数目，如果轮到你先取，假设双方都采取最好的策略，问最后你是胜者还是 败者。

## Input

输入包含若干行，表示若干种石子的初始情况，其中每一行包含两个非负整数a和b，表示两堆石子的数目，a和b都不大于1,000,000,000。

## Output

输出对应也有若干行，每行包含一个数字1或0，如果最后你是胜者，则为1，反之，则为0。

## Sample Input

2 1

## Sample Output

0

## HINT

#include<math.h>

#include<stdio.h>

int main()

{

    int m,n,t,k,d;

    while(scanf("%d%d",&m,&n)!=EOF)

    {

        if (m > n)

        {

            t = m;

            m = n;

            n =t;

        }

        k=n-m;

        d=floor(k\*(1.0+sqrt(5.0))/2.0);

        if (d==m)

            printf("0\n");

        else

            printf("1\n");

        }

}//威佐夫博奕

**一定存在规则允许的某种操作可将必胜点移动到必败点；**

**证明：以某个必胜点(i,j)为例。因为所有自然数都会出现在某个必败点中，故要么i等于m(k)，要么j等于 n(k)。若i=m(k)，j>n(k)，可从j中取走j-n(k)个石子到达必败点；若i=m(k)，j<n(k)，可从两堆同时拿走 m(k)-m(j-m(k))，从而到达必败点(m(j-m(k))，m(j-m(k))+j-m(k))；若i>m(k)，j=n(k)，可从i 中取走i-m(k)个石子到达必败点；若i<m(k)，j=n(k)，需要再分两种情况，因为i一定也出现在某个必败点中，若i=m(l)，则从j 中拿走j-n(l)，若i=n(l)，则从j中拿走j-m(l)，从而到达必败点(m(l),n(l))。性质3证毕。**

**判断一个点是不是必败点的公式与黄金分割有关，为：**

**m(k) = k \* (1 + sqrt(5))/2**

**n(k) = m(k) + k**

## 1924: Powers of two

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 12  Solved: 5  
[[Submit](http://10.5.54.252/acm/submitpage.php?id=1924)][[Status](http://10.5.54.252/acm/problemstatus.php?id=1924)][[Web Board](http://10.5.54.252/acm/bbs.php?pid=1924)]

## Description

Hadi loves binary numbers. So he likes to partition everything into powers of two. He wonders in how many ways he can partition a given number n into powers of two.

For example, there are 2 ways to partition 3: 1+1+1 and 1+2.

Help Him.

## Input

The first line of input consists of a single integer T, the number of test cases. Each test case consists of a line containing n. (0 <= n <= 2,000,000)

## Output

For each test case, output a line containing number of ways in which n can be partitioned into binary numbers modulo 1,000,000.

## Sample Input

4

1

2

3

280

## Sample Output

1

2

2

93298

## HINT

## Source

#include <stdio.h>

long long n,i,a,j;

long long s[2000001]={1};

int main()

{

for(i=0;(1<<i)<2000002;i++)

for(j=(1<<i);j<2000001;j++)

{

(s[j]+=s[j-(1<<i)])%=1000000;

//printf("%d ",s[i]);

}

while(scanf("%lld",&n)!=EOF)

{

while(n--)

{

scanf("%lld",&a);

printf("%lld\n",s[a]);

}

}

return 0;

}

**搜索算法:**

## 1816: Tempter of the Bone

时间限制: 1 Sec  内存限制: 33 MB  
提交: 37  解决: 9  
[[提交](http://10.5.54.252/acm/submitpage.php?id=1816)][[状态](http://10.5.54.252/acm/problemstatus.php?id=1816)][[讨论版](http://10.5.54.252/acm/bbs.php?pid=1816)]

## 题目描述

The doggie found a bone in an ancient maze, which fascinated him a lot. However, when he picked it up, the maze began to shake, and the doggie could feel the ground sinking. He realized that the bone was a trap, and he tried desperately to get out of this maze.  
  
The maze was a rectangle with sizes N by M. There was a door in the maze. At the beginning, the door was closed and it would open at the T-th second for a short period of time (less than 1 second). Therefore the doggie had to arrive at the door on exactly the T-th second. In every second, he could move one block to one of the upper, lower, left and right neighboring blocks. Once he entered a block, the ground of this block would start to sink and disappear in the next second. He could not stay at one block for more than one second, nor could he move into a visited block. Can the poor doggie survive? Please help him.

## 输入

The input consists of multiple test cases. The first line of each test case contains three integers N, M, and T (1 < N, M < 7; 0 < T < 50), which denote the sizes of the maze and the time at which the door will open, respectively. The next N lines give the maze layout, with each line containing M characters. A character is one of the following:  
  
'X': a block of wall, which the doggie cannot enter;   
'S': the start point of the doggie;   
'D': the Door; or  
'.': an empty block.  
  
The input is terminated with three 0's. This test case is not to be processed.

## 输出

For each test case, print in one line "YES" if the doggie can survive, or "NO" otherwise.

## 样例输入

4 4 5

S.X.

..X.

..XD

....

3 4 5

S.X.

..X.

...D

0 0 0

## 样例输出

NO

YES

#include <stdio.h>

#include <stdlib.h>

int sx,sy,dx,dy,n,m,t,flag;

char a[9][9];

void dfs(int tt,int x,int y)

{

if(a[x][y]=='X')return;

if(flag==1)return;

if(tt==t)

{

if(a[x][y]=='D')

{

flag=1;

}

}

else

{

if(a[x][y]=='D')return;

a[x][y]='X';

dfs(tt+1,x-1,y);

dfs(tt+1,x+1,y);

dfs(tt+1,x,y-1);

dfs(tt+1,x,y+1);

a[x][y]='.';

}

return;

}

int main()

{

int i,j,num;

char b[9];

for(i=0;i<9;i++)

{

a[0][i]='X';

a[i][0]='X';

}

while(scanf("%d%d%d",&n,&m,&t),(n||m||t))

{

flag=0;

num=0;

//将迷宫读入二维数组，边界全部设为‘X’

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

{

scanf("%s",b);

for(j=0;j<m;j++)

{

a[i][j+1]=b[j];

if(b[j]=='S')

{

sx=i;

sy=j+1;

}

else if(b[j]=='D')

{

dx=i;

dy=j+1;

}

else if(b[j]=='X')

num++;

}

}

for(i=0;i<9;i++)

{

a[n+1][i]='X';

a[i][m+1]='X';

}

//可走步数少于时间==剪枝

if(m\*n-num-1<t)

{

printf("NO\n");

continue;

}

//步数奇偶==剪枝

if(abs((sx+sy)%2-(dx+dy)%2)!=t%2)

{

printf("NO\n");

continue;

}

//Depth First Search

dfs(0,sx,sy);

if(flag)

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

快速模取幂算法

|  |
| --- |
| **快速模取幂**      数论计算中经常出现的一种运算就是求一个数的幂ab对另外一个数n个模的运算，即计算:  ab mod n (a,b,n是正整数)      由于计算机只能表示有限位的整数，所以编程时模取幂的运算要注意值的大小范围，当ab的值超过整数范围时,mod运算便无法进行。      如何解决这个问题，我们引出一个能计算ab mod n的值的有用算法——反复平方法，首先我们必须明确：  d=ab mod n=(…((((a mod n)\*a)mod n)\*a)mod n…\*a)mod n    {共b个a}      由此可以引出一个迭代式           d:=a;           for i:=2 to b do              d:=d mod n\*a;           d:=d mod n;      时间复杂度为O（b），当b很大时，效率很低。我们可以将b转换为二进制数<bk,bk-1,...,b1,b0>,然后从最低位b0开始，由右至左逐位扫描，每次迭代时，用到下面两个恒等式：  a2c mod n =(ac)2 mod n      bi=0  a2c+1 mod n =a\*(ac)2 mod n   bi=1 (0<=c<=b)      其中c为b的二进制数的后缀（bi-1...b0）对应的十进制数，当c成倍增加时，算法保持d=ac mod n不变，直至c=b。       程序实现可如下：  long long result(long long a,long long b,long long m) {     long long d,t；     d=1;     t=a;     while (b>0)     {         if (b%2==1)         d=(d\*t)%m;         b/=2;         t=(t\*t)%m;     }     return d; }  **找出相邻的不大于1的正整数矩阵最大数目：（下面两道类似）** 1977: matrix Time Limit: 1 Sec  Memory Limit: 128 MB Submit: 37  Solved: 5 [[Submit](http://10.5.54.252/oj/submitpage.php?id=1977)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1977)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1977)] Description      There is a 1\*n matrix, the first number is fixed to be 1. The other numbers are positive integers and the difference between adjacent numbers is not more than 1. The number of possible forms of matrix is m. Input  A number n representing 1\*n matrix (n<=1000) Output  A number representing of the results of m%( 1e9 + 7 ). Sample Input 3 Sample Output 5 HINTSource #include<stdio.h>  int main()  {  long long k,n,i,j;  while(scanf("%lld",&k)!=EOF)  {  long long a[1001]={0},b[1001]={0};  **for(i=1;i<=k+1;i++)**  **{**  **a[i]=1;**  **b[i]=1;**  **}**  **for(j=1;j<k;j++)**  **{**  **for(i=1;i<=k+1;i++)**  **a[i]=(b[i-1]+b[i]+b[i+1])%1000000007;**  **for(i=1;i<=k+1;i++)**  **b[i]=a[i];**  **}**  printf("%lld\n",b[1]);  }  return 0;  } |

## 1407: Tight words

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 56  Solved: 19  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1407)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1407)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1407)]

## Description

Given is an alphabet {0, 1, ... , k}, 0 <= k <= 9 . We say that a word of length n over this alphabet is tight if any two neighbour digits in the word do not differ by more than 1.

Input is a sequence of lines, each line contains two integer numbers k and n, 1 <= n <= 100. For each line of input, output the percentage of tight words of length n over the alphabet {0, 1, ... , k} with 5 fractional digits.

## Input

## Output

## Sample Input

4 1

2 5

3 5

8 7

## Sample Output

100.00000

40.74074

17.38281

0.10130

#include<stdio.h>

int main()

{

    int k,n,i,j;

    while(scanf("%d %d",&k,&n)!=EOF)

    {

        double sum1,sum2,a[11]={0},b[11]={0};

        sum2=1;

        for(i=0;i<n;i++)

            sum2\*=(k+1);

        sum1=0;

        for(i=1;i<=k+1;i++)

        {

            a[i]=1;

            b[i]=1;

        }

        for(j=1;j<n;j++)

        {

            for(i=1;i<=k+1;i++)

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

            for(i=1;i<=k+1;i++)

                b[i]=a[i];

        }

        for(i=1;i<=k+1;i++)

            sum1+=a[i];

        printf("%0.5lf\n",sum1/sum2\*100);

    }

    return 0;

}

## C++数列全排列函数：

## 1964: 排列

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 67  Solved: 22  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1964)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1964)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1964)]

## Description

大家知道，给出正整数n，则1 到n 这n 个数可以构成n！种排列，把这些排列按照从小到大的顺序（字典顺序）列出，如n=3 时，列出1 2 3，1 3 2，2 1 3，2 3 1，3 1 2，3 2 1六个排列。  
给出某个排列，求出这个排列的下k 个排列，如果遇到最后一个排列，则下1 排列为第1 个排列，即排列1 2 3…n。  
比如：n = 3，k=2 给出排列2 3 1，则它的下1 个排列为3 1 2，下2 个排列为3 2 1，因此答案为3 2 1。

## Input

第一行是一个正整数 m，表示测试数据的个数，下面是m 组测试数据，每组测试数据第一行是2 个正整数n( 1 <= n < 1024 )和k(1<=k<=64)，第二行有n 个正整数，是1，2 … n的一个排列。

## Output

对于每组输入数据，输出一行，n 个数，中间用空格隔开，表示输入排列的下k 个排列。

## Sample Input

3

3 1

2 3 1

3 1

3 2 1

10 2

1 2 3 4 5 6 7 8 9 10

## Sample Output

3 1 2

1 2 3

1 2 3 4 5 6 7 9 8 10

## HINT

## Source

#include <stdio.h>

#include <algorithm>

using namespace std;

int k,t,i,j,p,n,s[1100];

int main(){

scanf("%d",&t);

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

{

scanf("%d %d",&n,&k);

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

scanf("%d",&s[j]);

for(j=0;j<k;j++)**//进行K次全排列**

**next\_permutation(s,s+n);//进行一次S之后的全排列**

for(j=0;j<n-1;j++)

printf("%d ",s[j]);

printf("%d\n",s[j]);

}

return 0;

}

**next\_permutation 使用方法：**

1. **int 类型的next\_permutation   
   int main()  
   {  
    int a[3];  
   a[0]=1;a[1]=2;a[2]=3;  
    do  
   {  
   cout<<a[0]<<" "<<a[1]<<" "<<a[2]<<endl;  
   } while (next\_permutation(a,a+3)); //参数3指的是要进行排列的长度  
   //如果存在a之后的排列，就返回true。如果a是最后一个排列没有后继，返回false，每执行一次，a就变成它的后继  
   }   
   输出：  
    1 2 3  
    1 3 2  
    2 1 3  
    2 3 1  
    3 1 2  
    3 2 1  
   如果改成 while(next\_permutation(a,a+2));  
   则输出：  
    1 2 3  
    2 1 3  
   只对前两个元素进行字典排序  
   显然，如果改成 while(next\_permutation(a,a+1)); 则只输出：1 2 3  
   若排列本来就是最大的了没有后继，则next\_permutation执行后，会对排列进行字典升序排序,相当于循环  
    int list[3]={3,2,1};  
   next\_permutation(list,list+3);  
   cout<<list[0]<<" "<<list[1]<<" "<<list[2]<<endl;  
   //输出: 1 2 3   
   (2) char 类型的next\_permutation  
   int main()  
   {  
    char ch[205];  
   cin >> ch;  
   sort(ch, ch + strlen(ch) );  
   //该语句对输入的数组进行字典升序排序。如输入9874563102 cout<<ch; 将输出0123456789,这样就能输出全排列了  
    char \*first = ch;  
    char \*last = ch + strlen(ch);  
    do {  
   cout<< ch << endl;  
   }while(next\_permutation(first, last));  
    return 0;  
   }  
   //这样就不必事先知道ch的大小了，是把整个ch字符串全都进行排序  
   //若采用 while(next\_permutation(ch,ch+5)); 如果只输入1562，就会产生错误，因为ch中第五个元素指向未知  
   //若要整个字符串进行排序，参数5指的是数组的长度，不含结束符  
   (3) string 类型的next\_permutation  
   int main()  
   {  
    string line;  
    while(cin>>line&&line!="#")  
   {  
    if(next\_permutation(line.begin(),line.end())) //从当前输入位置开始  
   cout<<line<<endl;  
    else cout<<"Nosuccesor\n";  
   }  
   }  
     
   int main()  
   {  
    string line;  
    while(cin>>line&&line!="#")  
   {  
   sort(line.begin(),line.end());//全排列  
   cout<<line<<endl;  
    while(next\_permutation(line.begin(),line.end()))  
   cout<<line<<endl;  
   }  
   }**
2. **next\_permutation 自定义比较函数  
   #include<iostream> //poj 1256 Anagram  
   #include<string>  
   #include<algorithm>  
   using namespace std;  
   int cmp(char a,char b) //'A'<'a'<'B'<'b'<...<'Z'<'z'.  
   {  
    if(tolower(a)!=tolower(b))  
    return tolower(a)<tolower(b);  
    else  
    return a<b;  
   }  
   int main()  
   {  
    char ch[20];  
    int n;  
   cin>>n;  
    while(n--)  
   {  
   scanf("%s",ch);  
   sort(ch,ch+strlen(ch),cmp);  
    do  
   {  
   printf("%s\n",ch);  
   }while(next\_permutation(ch,ch+strlen(ch),cmp));  
   }  
    return 0;  
   }**

**最短路径：**

#include <iostream>

using namespace std;

const int maxnum = 100;

const int maxint = 999;

const int maxnum2 = 100;

const int maxint2 = 999;

void Dijkstra(int n, int v, int \*dist,int \*dist2, int \*prev, int c[maxnum][maxnum],int d[maxnum2][maxnum2])

{

    bool s[maxnum];    // 判断是否已存入该点到S集合中

    for(int i=1; i<=n; ++i)

    {

        dist[i] = c[v][i];

        dist2[i] = d[v][i];

        s[i] = 0;     // 初始都未用过该点

        if(dist[i] == maxint)

            prev[i] = 0;

        else

           {

            prev[i] = v;

          // prev[i] = v;

        }

    }

    dist[v] = 0;

    s[v] = 1;

    // 依次将未放入S集合的结点中，取dist[]最小值的结点，放入结合S中

    // 一旦S包含了所有V中顶点，dist就记录了从源点到所有其他顶点之间的最短路径长度

    for(int i=2; i<=n; ++i)

    {

        int tmp = maxint;

        int u = v;

        // 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]<tmp)

            {

                u = j;              // u保存当前邻接点中距离最小的点的号码

                tmp = dist[j];

            }

        s[u] = 1;    // 表示u点已存入S集合中

        // 更新dist

        for(int j=1; j<=n; ++j)

            if((!s[j]) && c[u][j]<maxint)

            {

                int newdist = dist[u] + c[u][j];

                int newdist2 = dist2[u] + d[u][j];

                if(newdist < dist[j])

                {

                    dist[j] = newdist;

                    dist2[j]=newdist2 ;

                    prev[j] = u;

                }

            }

    }

}

int main()

{

    int dist[maxnum];     // 表示当前点到源点的最短路径长度

    int prev[maxnum];     // 记录当前点的前一个结点

    int c[maxnum][maxnum];   // 记录图的两点间路径长度

    int dist2[maxnum2];     // 表示当前点到源点的最短路径长度

    int prev2[maxnum2];     // 记录当前点的前一个结点

    int d[maxnum2][maxnum2];

    int n, line;         // 图的结点数和路径数

    int s,t,k;

   while(scanf("%d%d",&n,&line)!=EOF)

   {

    if(n==0||line==0)

      break;

    int p, q, len,cost;          // 输入p, q两点及其路径长度

    // 初始化c[][]为maxint

    for(int i=1; i<=n; ++i)

        for(int j=1; j<=n; ++j)

            c[i][j] = maxint;

    for(int i=1; i<=line; ++i)

    {

        cin >> p >> q >> len >> cost;

        if(len < c[p][q])       // 有重边

        {

            c[p][q] = len;      // p指向q

            c[q][p] = len;      // q指向p，这样表示无向图

            d[p][q] = cost;      // p指向q

            d[q][p] = cost;

        }

    }

   scanf("%d%d",&s,&t);

    for(int i=1; i<=n; ++i)

        {

        dist[i] = maxint;

        dist2[i] = maxint2;

    }

    k=t-s;

    Dijkstra(n, 1, dist,dist2, prev, c,d);

    printf("%d %d\n",dist[n],dist2[n]);

}

}

**动态规划问题：**

**最长不降子序列**

#include<iostream>

using namespace std;

#define N 500

int main()

{

    int a[N];

    int n,i,max,max\_tmp;

    cin>>n;

    for(i=0;i<n;++i) {

        cin>>a[i];

    }

    max = 1;

    max\_tmp = 1;

    for(i=1;i<n; ++i) {

        if(a[i] >= a[i-1]) {

            ++max\_tmp;

            if(max\_tmp > max) max = max\_tmp;

        }

        else {

            max\_tmp = 1;

        }

    }

    cout<<max<<endl;

    return 0;

}

# [编辑距离：动态规划【用最少的字符操作将字符串A 转换为字符串B】](http://blog.csdn.net/fall221/article/details/8242706)

## 1187: 字符串的修改

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 14  Solved: 13  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1187)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1187)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1187)]

## Description

设A和B是两个字符串。我们要用最少的字符操作次数，将字符串A转换为字符串B。这里所说的字符操作共有三种：   
  
1. 删除一个字符；   
2. 插入一个字符；   
3. 将一个字符改为另一个字符。   
对任给的两个字符串A和B，计算出将字符串A变换为字符串B所用的最少字符操作次数。

## Input

第一行为字符串A；第二行为字符串B；字符串A和B的长度均小于200。

## Output

只有一个正整数，为最少字符操作次数。

## Sample Input

sfdxbqw

gfdgw

## Sample Output

4

## HINT

见1793，求出a和b的最长公共子序列，接下来就好办了

#include <stdio.h>

#include <string.h>

int dyna(char \*str1, int m, char \*str2, int n)

{

int \*\*p=new int\*[m];

int x;

for (x=0; x<m; x++)

{

p[x]=new int[n];

}

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

{

p[0][x]=x;

}

for (x=0; x<m; x++)

{

p[x][0]=x;

}

int i,j;

for (i=1; i<m; i++)

{

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

{

int cost = (str1[i-1] == str2[j-1]) ? 0 : 1;

cost += p[i-1][j-1];

p[i][j] = ( cost < (p[i-1][j]+1) )? cost:(p[i-1][j]+1);

p[i][j] = (p[i][j] < (p[i][j-1] +1)) ? p[i][j] :(p[i][j-1]);

}

}

int mm=p[m-1][n-1];

for (x=0; x<m; x++)

{

delete [](p[x]);

}

return mm;

}

int main()

{

char str1[200];

char str2[200];

while(scanf("%s%s",str1,str2)!=NULL){

int n = dyna(str1,strlen(str1)+1, str2, strlen(str2)+1);

printf("%d\n", n);

}

return 0;

}

**求最长公共子序列：**

## 1793: Common Subsequence

时间限制: 1 Sec  内存限制: 128 MB  
提交: 34  解决: 22  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1793)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1793)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1793)]

## 题目描述

A subsequence of a given sequence is the given sequence with some elements (possible none) left out. Given a sequence X = <x1, x2, ..., xm> another sequence Z = <z1, z2, ..., zk> is a subsequence of X if there exists a strictly increasing sequence <i1, i2, ..., ik> of indices of X such that for all j = 1,2,...,k, xij = zj. For example, Z = <a, b, f, c> is a subsequence of X = <a, b, c, f, b, c> with index sequence <1, 2, 4, 6>. Given two sequences X and Y the problem is to find the length of the maximum-length common subsequence of X and Y.   
The program input is from a text file. Each data set in the file contains two strings representing the given sequences. The sequences are separated by any number of white spaces. The input data are correct. For each set of data the program prints on the standard output the length of the maximum-length common subsequence from the beginning of a separate line.

## 输入

## 输出

## 样例输入

abcfbc abfcab

programming contest

abcd mnp

## 样例输出

4

2

0

## 提示

## 来源

[动态规划](http://10.5.54.252/oj/problemset.php?search=%E5%8A%A8%E6%80%81%E8%A7%84%E5%88%92)

**#include<stdio.h>**

**#include<string.h>**

**char a[300],b[300];**

**int dp[300][300];**

**int max1(int a,int b)**

**{**

**if(a>b) return a;**

**else  return b;**

**}**

**int main()**

**{**

**while(scanf("%s%s",&a,&b)!=EOF)**

**{**

**int n1=strlen(a);**

**int n2=strlen(b);**

**int i,j;**

**for(i=0;i<=n1;i++)**

**dp[i][0]=0;**

**for(i=0;i<=n2;i++)**

**dp[0][i]=0;**

**for(i=0;i<n1;i++)**

**for(j=0;j<n2;j++)**

**{**

**if(a[i]==b[j])**

**dp[i+1][j+1]=dp[i][j]+1;**

**else**

**{**

**dp[i+1][j+1]=max1(dp[i][j+1],dp[i+1][j]);**

**}**

**}**

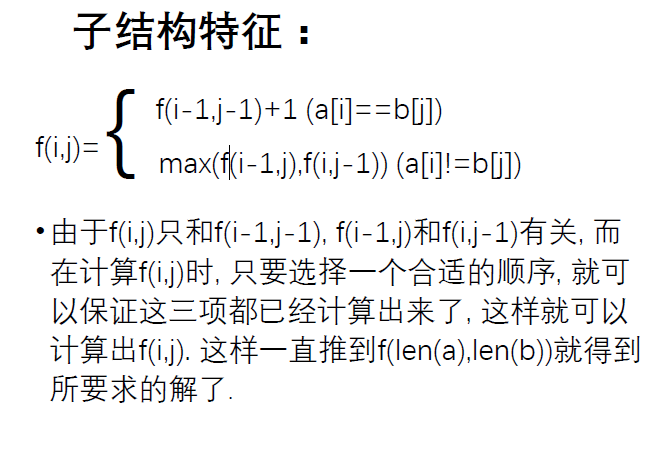
**printf("%d\n",dp[n1][n2]);**

**}**

**return 0;**

**}**

**动态转移方程：**



## 1791: 打地鼠

时间限制: 1 Sec  内存限制: 128 MB  
提交: 12  解决: 9  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1791)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1791)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1791)]

## 题目描述

威威猫最近不务正业，每天沉迷于游戏“打地鼠”。 　　每当朋友们劝他别太着迷游戏，应该好好工作的时候，他总是说，我是威威猫，猫打老鼠就是我的工作！ 　　无话可说... 　　我们知道，打地鼠是一款经典小游戏，规则很简单：每隔一个时间段就会从地下冒出一只或多只地鼠，玩游戏的人要做的就是打地鼠。 　　假设： 　　1、每一个时刻我们只能打一只地鼠，并且打完以后该时刻出现的所有地鼠都会立刻消失； 　　2、老鼠出现的位置在一条直线上，如果上一个时刻我们在x1位置打地鼠，下一个时刻我们在x2位置打地鼠，那么，此时我们消耗的能量为abs( x1 - x2 )； 　　3、打第一只地鼠无能量消耗。 　　现在，我们知道每个时刻所有冒出地面的地鼠位置，若在每个时刻都要打到一只地鼠，请计算最小需要消耗多少能量。

## 输入

输入数据包含多组测试用例；  
每组数据的第一行是2个正整数N和K（1 <= N <= 20, 1 <= K <= 10 )，表示有N个时刻，每个时刻有K只地鼠冒出地面；  
接下来的N行，每行表示一个时刻K只地鼠出现的坐标（坐标均为正整数，且<=500）。

## 输出

请计算并输出最小需要消耗的能量，每组数据输出一行。

## 样例输入

2 2

1 10

4 9

3 5

1 2 3 4 5

2 4 6 8 10

3 6 9 12 15

## 样例输出

1

1

## 提示

#include<stdio.h>

#include<math.h>

int s=10000000;

int main()

{

int n,i,k,j,m,a[100][100]={0},min[100][100];

while(scanf("%d%d",&n,&m)!=EOF)

{

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

for(j=0;j<m;j++)

{

scanf("%d",&a[i][j]);

min[i][j]=1000000;

}

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

min[0][i]=0;

for(k=1;k<n;k++)

{

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

{

for(j=0;j<m;j++)

{

**if(min[k][i]>min[k-1][j]+abs(a[k][i]-a[k-1][j]))**

**min[k][i]=min[k-1][j]+abs(a[k][i]-a[k-1][j]);**

**}//动态转移方程**

}

}

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

if(s>min[n-1][i])

s=min[n-1][i];

printf("%d\n",s);

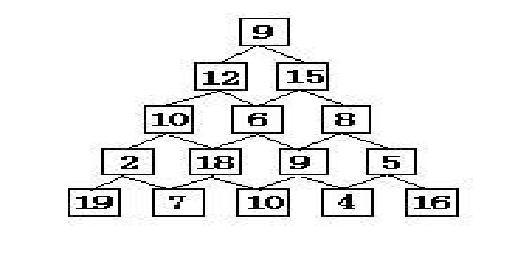
}

}

## 1788: 数塔

时间限制: 1 Sec  内存限制: 128 MB  
提交: 59  解决: 47  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1788)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1788)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1788)]

## 题目描述

在讲述DP算法的时候，一个经典的例子就是数塔问题，它是这样描述的： 有如下所示的数塔，要求从顶层走到底层，若每一步只能走到相邻的结点，则经过的结点的数字之和最大是多少？   
  
已经告诉你了，这是个DP的题目，你能AC吗?

## 输入

输入数据首先包括一个整数C,表示测试实例的个数，每个测试实例的第一行是一个整数N(1 <= N <= 100)，表示数塔的高度，接下来用N行数字表示数塔，其中第i行有个i个整数，且所有的整数均在区间[0,99]内。

## 输出

对于每个测试实例，输出可能得到的最大和，每个实例的输出占一行。

## 样例输入

1

5

7

3 8

8 1 0

2 7 4 4

4 5 2 6 5

## 样例输出

30

## 提示

## 来源

#include<stdio.h>

max(int a,int b)

{

  if(a>b) return a;

  else  return b;

}

int main()

{

 int n,i,j,a[100][100]={0},k,m;

    scanf("%d",&m);

    for(k=1;k<=m;k++)

{

    scanf("%d",&n);

 for(i=0;i<n;i++)

    for(j=0;j<i+1;j++)

    {

     scanf("%d",&a[i][j]);

    }

  for(i=n-2;i>=0;i--)

      {

        for(j=0;j<=i;j++)

**a[i][j]=a[i][j]+max(a[i+1][j],a[i+1][j+1]);**

      }**//状态转移方程**

  printf("%d\n",a[0][0]); }

 return 0;

}

## 1794: 搬寝室

时间限制: 1 Sec  内存限制: 128 MB  
提交: 44  解决: 10  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1794)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1794)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1794)]

## 题目描述

搬 寝室是很累的,xhd深有体会.时间追述2006年7月9号,那天xhd迫于无奈要从27号楼搬到3号楼,因为10号要封楼了.看着寝室里的n件物 品,xhd开始发呆,因为n是一个小于2000的整数,实在是太多了,于是xhd决定随便搬2\*k件过去就行了.但还是会很累,因为2\*k也不小是一个不 大于n的整数.幸运的是xhd根据多年的搬东西的经验发现每搬一次的疲劳度是和左右手的物品的重量差的平方成正比(这里补充一句,xhd每次搬两件东西, 左手一件右手一件).例如xhd左手拿重量为3的物品,右手拿重量为6的物品,则他搬完这次的疲劳度为(6-3)^2 = 9.现在可怜的xhd希望知道搬完这2\*k件物品后的最佳状态是怎样的(也就是最低的疲劳度),请告诉他吧.

## 输入

每组输入数据有两行,第一行有两个数n,k(2<=2\*k<=n<2000).第二行有n个整数分别表示n件物品的重量(重量是一个小于2^15的正整数).

## 输出

对应每组输入数据,输出数据只有一个表示他的最少的疲劳度,每个一行.

## 样例输入

2 1

1 3

## 样例输出

4

#include<stdio.h>

#include<stdlib.h>

int p[2002][2002]={0};

int mmin(int a,int b)

{

     if(a<b) return a;

        else  return b;

}

int cmp ( const void \*a , const void \*b )

{

return \*(int \*)a - \*(int \*)b;

}

int main()

{

    int i,j,n,t=0,x,k,a[2002]={0};

    while(scanf("%d%d",&n,&k)!=EOF)

    {

        t=0;

        for(i=1;i<=n;i++)

         scanf("%d",&a[i]);

         qsort(a,n+1,sizeof(a[1]),cmp);

            for(i=2;i<=n;i++)

            for(j=1;j<=k;j++)

            {

                if(i==2\*j)

                p[i][j]=p[i-2][j-1]+(a[i]-a[i-1])\*(a[i]-a[i-1]);

                if(i>2\*j)

                p[i][j]=mmin(p[i-1][j] ,p[i-2][j-1]+(a[i]-a[i-1])\*(a[i]-a[i-1]));

            }

        printf("%d\n",p[n][k]);

    }

    return 0;

}

## 1789: 免费馅饼

时间限制: 1 Sec  内存限制: 128 MB  
提交: 51  解决: 19  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1789)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1789)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1789)]

## 题目描述

都 说天上不会掉馅饼，但有一天gameboy正走在回家的小径上，忽然天上掉下大把大把的馅饼。说来gameboy的人品实在是太好了，这馅饼别处都不掉， 就掉落在他身旁的10米范围内。馅饼如果掉在了地上当然就不能吃了，所以gameboy马上卸下身上的背包去接。但由于小径两侧都不能站人，所以他只能在 小径上接。由于gameboy平时老呆在房间里玩游戏，虽然在游戏中是个身手敏捷的高手，但在现实中运动神经特别迟钝，每秒种只有在移动不超过一米的范围 内接住坠落的馅饼。现在给这条小径如图标上坐标：

http://acm.hdu.edu.cn/data/images/1176_1.jpg

为了使问题简化，假设在接下来的一段时间里，馅饼都掉落在0-10这11个位置。开始时gameboy站在5这个位置，因此在第一秒，他只能接到4,5,6这三个位置中其中一个位置上的馅饼。问gameboy最多可能接到多少个馅饼？（假设他的背包可以容纳无穷多个馅饼）

## 输入

输入数据有多组。每组数据的第一行为以正整数 n(0<n<100000)，表示有n个馅饼掉在这条小径上。在结下来的n行中，每行有两个整数 x,T(0<T<100000),表示在第T秒有一个馅饼掉在x点上。同一秒钟在同一点上可能掉下多个馅饼。n=0时输入结束。

## 输出

每一组输入数据对应一行输出。输出一个整数m，表示gameboy最多可能接到m个馅饼。

## 样例输入

6

5 1

4 1

6 1

7 2

7 2

8 3

0

## 样例输出

4

## 提示

本题的输入数据量比较大，建议用scanf读入，用cin可能会超时。

#include<stdio.h>

#include<string.h>

int a[100002][12];

int max(int a,int b)

{

    if(a>b)

    return a;

    else

    return b;

}

int max2(int a,int b,int c)

{

    int max1;

    if(a>b)

    max1=a;

    else

    max1=b;

    if(max1<c)

    max1=c;

    return max1;

}

int main()

{

    int i,j,n,t,x,s;

    while(scanf("%d",&n)!=EOF)

    {

        if(n==0)

        break;

        s=0;

        for(i=0;i<n;i++)

        {

             scanf("%d%d",&x,&t);

             a[t][x]++;

             if(s<t)

             s=t;

        }

        for(i=s-1;i>=0;i--)

        {

            for(j=1;j<=9;j++)

**a[i][j]+=max2(a[i+1][j-1],a[i+1][j],a[i+1][j+1]);**

**a[i][0]+=max(a[i+1][0],a[i+1][1]);**

**a[i][10]+=max(a[i+1][10],a[i+1][9]);**

        }//动态规划状态转移方程

        printf("%d\n",a[0][5]);

    }return 0;

}

**逆元：**

**搜索算法：**

## 1816: Tempter of the Bone

Time Limit: 1 Sec  Memory Limit: 33 MB  
Submit: 55  Solved: 15  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1816)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1816)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1816)]

## Description

The doggie found a bone in an ancient maze, which fascinated him a lot. However, when he picked it up, the maze began to shake, and the doggie could feel the ground sinking. He realized that the bone was a trap, and he tried desperately to get out of this maze.  
  
The maze was a rectangle with sizes N by M. There was a door in the maze. At the beginning, the door was closed and it would open at the T-th second for a short period of time (less than 1 second). Therefore the doggie had to arrive at the door on exactly the T-th second. In every second, he could move one block to one of the upper, lower, left and right neighboring blocks. Once he entered a block, the ground of this block would start to sink and disappear in the next second. He could not stay at one block for more than one second, nor could he move into a visited block. Can the poor doggie survive? Please help him.

## Input

The input consists of multiple test cases. The first line of each test case contains three integers N, M, and T (1 < N, M < 7; 0 < T < 50), which denote the sizes of the maze and the time at which the door will open, respectively. The next N lines give the maze layout, with each line containing M characters. A character is one of the following:  
  
'X': a block of wall, which the doggie cannot enter;   
'S': the start point of the doggie;   
'D': the Door; or  
'.': an empty block.  
  
The input is terminated with three 0's. This test case is not to be processed.

## Output

For each test case, print in one line "YES" if the doggie can survive, or "NO" otherwise.

## Sample Input

4 4 5

S.X.

..X.

..XD

....

3 4 5

S.X.

..X.

...D

0 0 0

## Sample Output

NO

YES

搜索算法：

#include<stdio.h>

#include<math.h>

char a[9][9];//上课老师讲解做法

int g[4][2]={{1,0},{0,1},{-1,0},{0,-1}};//四个方向

int n,m,t,dx,dy,f;

void dfs(int x,int y,int t)

{

int i,sx,sy,s;

s=t-abs(x-dx)-abs(y-dy);//

if(x>n||y>m||x<=0||y<=0)//判断边界 ----- 剪枝 3

return;

if(t==0&&s==0)

f=1;

if(f) return;

if(s<0||s&1) return;//判断奇偶 ---- 剪枝 2

for(i=0;i<4;i++)//上下左右4个方向

{

sx=x+g[i][0];

sy=y+g[i][1];

if(a[sx][sy]!='X')

{

a[sx][sy]='X';

dfs(sx,sy,t-1);

a[sx][sy]='.';//回朔恢复到之前状态

}

}

return;

}

int main()

{

while(scanf("%d%d%d",&n,&m,&t)&&(n!=0||m!=0||t!=0))

{

int i,j,x,y,b=0;

getchar();//接收回车

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

{

for(j=1;j<=m;j++)

{

scanf("%c",&a[i][j]);

if(a[i][j]=='S')

{

x=i;

y=j;

}//起始点

else if(a[i][j]=='D')

{

dx=i;

dy=j;

}//终点

else if(a[i][j]=='X')//计算有多少墙

b++;

}

getchar();//接收回车

}

if(m\*n-b<=t)//-------剪枝 1

{

printf("NO\n");

continue;

}

f=0;

//a[x][y]='X';

dfs(x,y,t);//搜索

if(f)

printf("YES\n");

else

printf("NO\n");

}

return 0;

}

速算24点



## 1246: 速算24点

时间限制: 1 Sec  内存限制: 128 MB  
提交: 25  解决: 8  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1246)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1246)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1246)]

## 题目描述

速算24点相信绝大多数人都玩过。就是随机给你四张牌，包括 A(1),2,3,4,5,6,7,8,9,10,J(11),Q(12),K(13)。要求只用'+','-','\*','/'运算符以及括号改变运算 顺序，使得最终运算结果为24(每个数必须且仅能用一次)。游戏很简单，但遇到无解的情况往往让人很郁闷。你的任务就是针对每一组随机产生的四张牌，判断 是否有解。我们另外规定，整个计算过程中都不能出现小数。

## 输入

输入数据占一行，给定四张牌。

## 输出

如果有解则输出"Y"，无解则输出"N"。

## 样例输入

A 2 3 6

## 样例输出

Y

#include<stdio.h>

int s[5];

int guess(int n)

{

if(n==1)

{

if(s[0]==24)

return 1;

else

return 0;

}

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

{

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

{

int a=s[i],b=s[j];

s[j]=s[n-1];

s[i]=a+b;

if(guess(n-1))

return 1;

s[i]=a-b;

if(guess(n-1))

return 1;

s[i]=b-a;

if(guess(n-1))

return 1;

s[i]=b\*a;

if(guess(n-1))

return 1;

if(b!=0&&(a%b)==0)

{

s[i]=a/b;

if(guess(n-1))

return 1;

}

if(a!=0&&(b%a)==0)

{

s[i]=b/a;

if(guess(n-1))

return 1;

}

s[i]=a;

s[j]=b;

}

}

return 0;

}

int change(char a[100])

{

if(a=="A")

return 1;

if(a=="J")

return 2;

if(a=="Q")

return 3;

if(a=="K")

return 4;

if(a=="10")

return 10;

return a[0]-'0';

}

int main()

{

int i;

char a[4][100];

scanf("%s%s%s%s",a[0],a[1],a[2],a[3]);

for(i=0;i<4;i++)

s[i]=change(a[i]);

if(guess(4))

printf("Y\n");

else

printf("N\n");

return 0;

}

## 1971: Recurrence sequence

时间限制: 1 Sec  内存限制: 128 MB  
提交: 85  解决: 14  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1971)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1971)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1971)]

## 题目描述

There is a sequence that is defined in this way：

f(1) = 1,

f(2) = 1,

f(n) = (A \* f(n - 1) + B \* f(n - 2)) mod 7.

Given A, B and N, calculate the value of f(n).

## 输入

Input three numbers: A, B, N, separated by spaces.

(-10000 <= A, B <= 10000, 1 <= N <= 10^9)

## 输出

Output the value of f(n).

## 样例输入

3 -1 5

## 样例输出

6

## 提示

## 来源

#include<stdio.h>

#include<math.h>

int h[100],s;

int f(int a,int b)

{

int s,i;

h[1]=h[2]=1;

for(i=3;;i++)

{

s=fabs((a\*h[i-1]+b\*h[i-2])%7);

if(((a\*h[i-1]+b\*h[i-2])%7)>=0)

h[i]=s;

else

h[i]=7-s;

if(h[i]==1&&h[i-1]==1)

break;

}

h[0]=h[i-2];

return i-2;

}

int main()

{

int n,i,t,k,a,b;

while(scanf("%d%d%d",&a,&b,&n)!=EOF)

{

t=f(a,b);

printf("%d\n",h[n%t]);

}

return 0;

}

**并查集：**int father[MAX];   /\* father[x]表示x的父节点\*/  
int rank[MAX];     /\* rank[x]表示x的秩\*/  
/\* 初始化集合\*/  
void Make\_Set(int x)  
{  
    father[x] = x; //根据实际情况指定的父节点可变化  
    rank[x] = 0;   //根据实际情况初始化秩也有所变化  
}  
/\* 查找x元素所在的集合,回溯时压缩路径\*/  
int Find\_Set(int x)  
  
{  
    if (x != father[x])  
    {  
        father[x] = Find\_Set(father[x]); //这个回溯时的压缩路径是精华  
    }  
    return father[x];  
}  
/\*  
   按秩合并x,y所在的集合  
   下面的那个if else结构不是绝对的，具体根据情况变化  
   但是，宗旨是不变的即，按秩合并，实时更新秩。  
\*/  
void Union(int x, int y)  
  
{  
    x = Find\_Set(x);  
    y = Find\_Set(y);  
    if (x == y) return;  
    if (rank[x] > rank[y])  
    {  
        father[y] = x;  
    }  
    else  
    {  
        if (rank[x] == rank[y])  
        {  
            rank[y]++;  
        }  
        father[x] = y;  
    }  
}

**求三维空间中点的最小生成树（并查集）**

## 1664: A MST Problem

Time Limit: 1 Sec  Memory Limit: 32 MB  
Submit: 20  Solved: 7  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1664)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1664)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1664)]

## Description

It is just a mining spanning tree ( 最小生成树 ) problem, what makes you a little difficult is that you are in a 3D space.

## Input

The first line of the input contains the number of test cases in the file. And t he first line of each case

contains one integer numbers n(0<n<30) specifying the number of the point . The n next n line s, each line

contain s Three Integer Numbers xi,yi and zi, indicating the position of point i.

## Output

For each test case, output a line with the answer, which should accurately rounded to two decimals .

## Sample Input

2

2

1 1 0

2 2 0

3

1 2 3

0 0 0

1 1 1

## Sample Output

1.41

3.97

## HINT

## Source

#include<stdio.h>

#include<string.h>

#include<math.h>

double d[102][3], w[102][102], minCost[102];

int n, pre[102], hash[102];

double getDist(double x1,double y1,double z1,double x2,double y2,double z2){

return sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2)+(z1-z2)\*(z1-z2));

}

double Prim(){

memset(hash, 0, sizeof(hash));

hash[1] = 1;

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

minCost[i] = w[1][i];

pre[i] = 1;

}

double sum=0;

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

int u=-1;

for(int j=1; j<=n; ++j)if(!hash[j]){

if(u==-1||minCost[j]<minCost[u])

u=j;

}

sum += w[pre[u]][u];

hash[u] = 1;

for(int j=1; j<=n; ++j)if(!hash[j]){

if(minCost[j]>w[u][j]){

minCost[j] = w[u][j];

pre[j] = u;

}

}

}

return sum;

}//prim

int main(){

int t;

while(scanf("%d",&t)!=EOF)

{

while(t--)

{

scanf("%d",&n);

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

scanf("%lf%lf%lf",&d[i][0],&d[i][1],&d[i][2]);

memset(w, 0, sizeof(w));

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

for(int j=1; j<=n; ++j)if(i!=j)

w[i][j] = getDist(d[i][0],d[i][1],d[i][2],

d[j][0],d[j][1],d[j][2]);

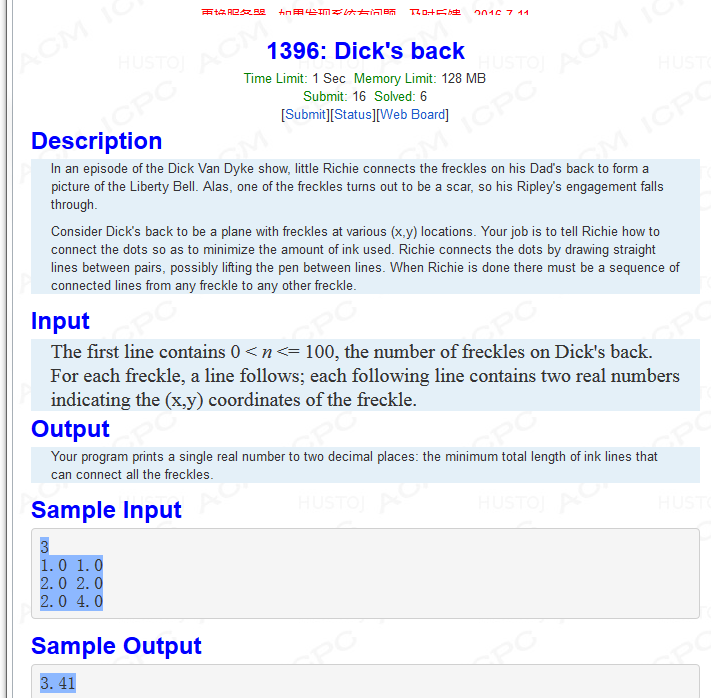
printf("%.2f\n", Prim());

}

}

return 0;

}



   #include<stdio.h>

    #include<string.h>

    #include<math.h>

    double d[102][2], w[102][102], minCost[102];

    int n, pre[102], hash[102];

     double getDist(double x1,double y1,double x2,double y2){

        return sqrt((x1-x2)\*(x1-x2)+(y1-y2)\*(y1-y2));

    }

    double Prim(){

        memset(hash, 0, sizeof(hash));

        hash[1] = 1;

        for(int i=1; i<=n; ++i){

            minCost[i] = w[1][i];

            pre[i] = 1;

        }

        double sum=0;

        for(int i=1; i<n; ++i){

            int u=-1;

            for(int j=1; j<=n; ++j)if(!hash[j]){

                if(u==-1||minCost[j]<minCost[u])

                    u=j;

            }

            sum += w[pre[u]][u];

            hash[u] = 1;

            for(int j=1; j<=n; ++j)if(!hash[j]){

                if(minCost[j]>w[u][j]){

                    minCost[j] = w[u][j];

                    pre[j] = u;

                }

            }

        }

        return sum;

    }//prim

    int main(){

            scanf("%d",&n);

            for(int i=1; i<=n; ++i)

                scanf("%lf%lf",&d[i][0],&d[i][1]);

            memset(w, 0, sizeof(w));

            for(int i=1; i<=n; ++i)

                for(int j=1; j<=n; ++j)if(i!=j)

                    w[i][j] = getDist(d[i][0],d[i][1],

                            d[j][0],d[j][1]);

            printf("%.2f\n", Prim());

        return 0;

    }

并查集

## 1796: 畅通工程

Time Limit: 1 Sec  Memory Limit: 128 MB  
Submit: 34  Solved: 28  
[[Submit](http://10.5.54.252/oj/submitpage.php?id=1796)][[Status](http://10.5.54.252/oj/problemstatus.php?id=1796)][[Web Board](http://10.5.54.252/oj/bbs.php?pid=1796)]

## Description

某省调查城镇交通状况，得到现有城镇道路统计表，表中列出了每条道路直接连通的城镇。省政府“畅通工程”的目标是使全省任何两个城镇间都可以实现交通（但不一定有直接的道路相连，只要互相间接通过道路可达即可）。问最少还需要建设多少条道路？

## Input

测试输入包含若干测试用例。每个测试用例的第1行给出两个正整数，分别是城镇数目N ( < 1000 )和道路数目M；随后的M行对应M条道路，每行给出一对正整数，分别是该条道路直接连通的两个城镇的编号。为简单起见，城镇从1到N编号。   
注意:两个城市之间可以有多条道路相通,也就是说  
3 3  
1 2  
1 2  
2 1  
这种输入也是合法的  
当N为0时，输入结束，该用例不被处理。

## Output

对每个测试用例，在1行里输出最少还需要建设的道路数目。

## Sample Input

4 2

1 3

4 3

3 3

1 2

1 3

2 3

5 2

1 2

3 5

999 0

0

## Sample Output

1

0

2

998

## HINT

Huge input, scanf is recommended.

## Source

[并查集](http://10.5.54.252/oj/problemset.php?search=%E5%B9%B6%E6%9F%A5%E9%9B%86)

**#include<stdio.h>**

**int set[1004];**

**void merge(int x ,int y)**

**{**

**set[x]=y;**

**}**

**int find(int x)**

**{**

**int r;**

**r=x;**

**while(set[r]!=r)**

**r=set[r];**

**return r;**

**}**

**int main()**

**{**

**int a,b,m,n,i,x,y;**

**int t;**

**while(scanf("%d%d",&m,&n)&&m!=0)**

**{**

**if(m==0)**

**break;**

**for(i=1;i<=m;++i)**

**{**

**set[i]=i;**

**}**

**while(n--)**

**{**

**scanf("%d%d",&a,&b);**

**x=find(a);**

**y=find(b);**

**if(x!=y)**

**merge(x,y);**

**}**

**int s=0;**

**for(i=1;i<=m;++i)**

**{**

**if(set[i]==i)**

**s++;**

**}**

**printf("%d\n",s-1);**

**}**

**return 0;**

**}**

**动态规划**

**最小代价子母树**

#include "iostream"

#include "cstdio"

#include "queue"

#define qwq 21000000

int n,m,q;

using namespace std;

int f[1000][1000];

int g[1000][1000];

int gra[1000],minn=-2100000;

int main()

{

int n;

cin>>n;

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

cin>>gra[i];

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

g[i][i]=gra[i];

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

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

g[i][j]=g[i][j-1]+gra[j];

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

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

f[i][j]=(i!=j)?qwq:0; //i -> j(i)的代价为 +∞(0)

for (int p=2;p<=n;p++)//穷举堆数//因为是递推(Recursion)所以从小堆数向大堆数

for (int i=1;i<=(n-p+1);i++)//一段的确定性(Certainty) 列举起点

{

int j=i+p-1; //终点(Destination)

for (int k=i;k<=j-1;k++)//穷举隔开位置 //注意超界所致的溢出(Overflow)

f[i][j]=(f[i][j]>f[i][k]+f[k+1][j]+g[i][j])?f[i][k]+f[k+1][j]+g[i][j]:f[i][j];//三目运算符不清楚的可以百度，下面也有简介

} //正确写出状态转移方程//无后效性 (Unfollow-up Effect)

cout<<f[1][n]<<endl;

}

**1686: 取数**

时间限制: 0 Sec  内存限制: 128 MB  
提交: 35  解决: 7  
[[提交](http://10.5.54.252/oj/submitpage.php?id=1686)][[状态](http://10.5.54.252/oj/problemstatus.php?id=1686)][[讨论版](http://10.5.54.252/oj/bbs.php?pid=1686)]

**题目描述**

给出长度为N的数列{A\_i}，每次可以从最左边或者最右边取走一个数，第i次取数得到的价值是i  \*  A\_j。求价值之和最大的取数方案。 N  < =  2000  ,  A\_i  < =  1000

**输入**

        第一行，一个整数，表示数列长度N。         接下来N行，每行一个整数，表示数列A\_i。

**输出**

一个整数，表示最大的价值之和。

**样例输入**

5

1

3

1

5

2

**样例输出**

43

**提示**

考虑第i次取数时最右边（或者最左边）取了j个数的最大值

**来源**

#include<stdio.h>

#include<string.h>

int a[2002];

int max(int a,int b)

{

    if(a>b)

    return a;

    else

    return b;

}

int main()

{

    int n,i,m=-100000000,j;

    while(scanf("%d",&n)!=EOF)

    {

        for(i=1;i<=n;i++)

        scanf("%d",&a[i]);

        int b[2002]={0};

        b[0]=a[1];b[1]=a[n];

        for(i=2;i<=n;i++)

        {

            for(j=i;j>=0;j--)

            {

                if(j>0)

                b[j]=max(b[j-1]+i\*a[n-j+1],b[j]+i\*a[i-j]);

                else

                b[j]+=i\*a[i];

              //  printf("%d ",b[j]);

            }

        }

        for(i=0;i<=n;i++)

        if(b[i]>m)

        m=b[i];

        printf("%d\n",m);

    }

}

**能量项链：**

#include <iostream>

using namespace std;

int n, e=0;

int a[111\*2];

int dp[111\*2][111\*2];

int main()

{

cin >> n;

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

{

cin >> a[i];

a[i+n] = a[i];//用数组模拟环，减少一层循环

}

for(int j=2; j<=n+n; j++)

{

for(int i=j-1; i>=1&&j-i<n; i--)

{

int maxV = 0;

for(int k=i; k<j; k++)

{

if(maxV < dp[i][k]+dp[k+1][j]+a[i]\*a[k+1]\*a[j+1])

maxV = dp[i][k]+dp[k+1][j]+a[i]\*a[k+1]\*a[j+1];

}

dp[i][j] = maxV;

if(e < dp[i][j]) e = dp[i][j];

}

}

cout << e;

return 0;

}

**石子归并：**

#include <iostream>

using namespace std;

int sum[111] = {0};

int n,minV;

int w[111];

int dp[111][111] = {0};

int main()

{

cin >> n;

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

{

cin >> w[i];

sum[i] = sum[i-1] + w[i];

}

for(int len=2; len<=n; len++)

{

for(int i=1; i<=n-len+1; i++)

{

int j=i+len-1;

minV = 0x7fffffff;

for(int k=i; k<j; k++)

{

if(minV > dp[i][k]+dp[k+1][j]+sum[j]-sum[i-1])

minV = dp[i][k]+dp[k+1][j]+sum[j]-sum[i-1];

}

dp[i][j] = minV;

}

}

cout << dp[1][n];

return 0;

}