2016级软件学院数据结构习题整理

第1次上机

A Mdd的swap函数

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

在C++中，往一个函数传递参数可以有3种方法，分别是值传递，引用传递，指针传递，而mdd想知道这3种方法的区别，你能展示给他看吗。

请不要水过，务必实现3个函数。

输入

输入多组数据。

对于每组数据，输入两个数a，b（均在int范围内）。

输出

输出三行，分别为按值传递，引用传递，指针传递来实现，a，b交换后的值。

输入样例

```
1 2
```

输出样例

```
1 2
2 1
2 1
```

Warning

直接交换位置输出，使用std::swap等方法水过的，不给分。

知识点

指针 引用

AC Code

```
#include<cstdio>
void swap_by_value(int a, int b) {
  int c = a;
  a = b;
  b = c;
}
void swap_by_quote(int& a, int& b) {
  int& c = a;
  int d = a;
  a = b;
  b = d;
}
void swap_by_pointer(int* a, int* b) {
  int c;
  c = *a;
  *a = *b;
  *b = c;
}
int m, n;
int tmpm, tmpn;
int main() {
  while (scanf("%d%d", &m, &n) != EOF) {
    tmpm = m; tmpn = n;
    swap_by_value(tmpm, tmpn);
    printf("%d %d\n", tmpm, tmpn);
    tmpm = m; tmpn = n;
    swap_by_quote(tmpm, tmpn);
    printf("%d %d\n", tmpm, tmpn);
    tmpm = m; tmpn = n;
    swap_by_pointer(&tmpm, &tmpn);
    printf("%d %d\n", tmpm, tmpn);
  }
}
```


B 巧克力游戏

时间限制：2000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh和Syw喜欢玩游戏，现在他们准备玩一个新游戏。他们把nn块巧克力排成一排。Gzh从左往右吃，Syw从右往左吃。他们吃巧克力的速度一样并且吃每块巧克力的时间是已知的。当一个人吃完一块巧克力时，他将立刻开始吃下一块。游戏规定不允许一个人同时吃两块，不允许吃一块巧克力不吃完而剩下，也不允许吃的过程中有停顿。如果Gzh和Syw同时开始吃一块巧克力，那么Syw会把这块巧克力让给Gzh。

按照这个规则，最后Gzh和Syw会各吃多少块巧克力呢?

输入

多组输入数据。

每组输入数据为2行。第一行一个整数n(1≤n≤105)n(1≤n≤105)，代表巧克力数量。第二行nn个整数t1,t2,...,tn(1≤ti≤1000)t1,t2,...,tn(1≤ti≤1000)，其中titi表示吃从左到右算第ii块巧克力所需要的时间。

输出

每组数据输出一行，包含两个整数，分别表示Gzh吃的巧克力数量和Syw吃的巧克力数量，用空格隔开。

输入样例

```
5
2 9 8 2 7
```

输出样例

```
2 3
```

知识点

数组 贪心

AC Code

```
#include<cstdio>
#define maxn 100010
int sweet[maxn];
int acnt, bcnt;
int amark, bmark;
int tot;
int n;
int main() {
  while (scanf("%d", &n) != EOF) {
    amark = 1; bmark = n;
    for (int i = 1; i <= n; ++i)scanf("%d", sweet + i);
```

```
    if (n == 1) { puts("1 0"); continue; }
    while (amark <= bmark && amark + n - bmark + 1 < n) {
      if (sweet[amark] > sweet[bmark]) {
        sweet[amark] -= sweet[bmark];
        --bmark;
        ++tot;
      }
      else if (sweet[bmark] > sweet[amark]) {
        sweet[bmark] -= sweet[amark];
        ++amark;
        ++tot;
      }
      else if (amark == bmark)amark++;
      else {
        sweet[bmark] = sweet[amark] = 0;
        ++amark, --bmark;
      }
    }
    printf("%d %d\n", amark, n - amark);
  }
}
```

c Mdd来排队

时间限制：500ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

在舞蹈课上，老师一般都是将队伍排为两列，一列男生，一列女生。一开始每个人都是和自己的舞伴站在同一排的，然而老师却要求按照身高从矮到高顺序排队，mdd很好奇，再一次排完队后，每个人的舞伴都在哪个位置呢？

输入

多组输入数据。

对于每组数据，第一行为人数n（1<=n<=10000），接下来n行，每行共2个数，分别为男生女生各自的身高，保证身高各不相同。

输出

对于每组数据输出2行，分别为2列队伍中，排完队之后各自舞伴的位置，以空格隔开，从1开始计数。

输入样例

```
3
1.82 1.67
1.81 1.69
1.84 1.68
```

输出样例

```
3 1 2
2 3 1
```

Hint

对于样例，一开始给他们标号，分别为1-3号男生，1-3号女生，排完队后男生队列是2 1 3，女生队列是1 3 2，所以先输出2号男生对应舞伴即2号女生的位置：3。

知识点

结构体排序

AC Code

```cpp
#include<cstdio>
#include<algorithm>
using namespace std;
#define maxn 10010
int n;
int pos[maxn];
struct student {
  double height;
  int number;
  bool operator <(const student& b) const {
    return height < b.height;
  }
}male[maxn], female[maxn];
int main() {
  while (scanf("%d", &n) != EOF) {
    for (int i = 1; i <= n; ++i) {
      scanf("%lf%lf", &female[i].height, &male[i].height);
      female[i].number = i, male[i].number = i;
    }
    sort(female + 1, female + n + 1);
    sort(male + 1, male + n + 1);
    for (int i = 1; i <= n; ++i)pos[male[i].number] = i;
    for (int i = 1; i <= n; ++i)printf("%d ", pos[female[i].number]);
    puts("");
    for (int i = 1; i <= n; ++i)pos[female[i].number] = i;
    for (int i = 1; i <= n; ++i)printf("%d ", pos[male[i].number]);
```

```
    puts("");
  }
}
```

D Mdd的成绩单

时间限制：100ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

公公是个懒惰的人，虽然他是一名离散猪脚，但成绩统计这么麻烦的事情他是不会做的，于是他把这件事托付(py)给了 mdd。虽然mdd接受了他的任务，但他看到这么多人的成绩还是很头疼，你能够帮帮他吗?

输入

多组输入数据。

对于每组数据，第一个数为n（1<=n<=10000），表示共有n名同学，接下来分别为各位同学的姓名（不超过12个字符，没有空格），学号（int范围内），成绩（0<=grade<=100）。

接下来一个数m，表示共有m组查询，每行只有一个学号，请输出该同学对应的信息。

输出

每组查询输出一行，分别为该同学的姓名，学号，成绩（保留两位小数），以及该同学的排名。

排名按照降序排列，即成绩高的在前，若成绩相同，则学号小的在前。

如果查询的学号没有对应的同学，输出”Only god knows where he is."。

输入样例

```
2
mike 16211111 90.5
marry 16211102 100
2
16211111
16211101
```

输出样例

```
mike 16211111 90.50 2
Only god knows where he is.
```

Hint

请使用结构体
建议使用scanf printf

知识点

结构体排序 二分查找

AC Code

```
#pragma G++ optimize(2)
#include<cstdio>
#include<algorithm>
using namespace std;
#define maxn 10010
int n, m;
struct Student {
    char name[100];
    int num;
    double grade;
    int position;
}stu[maxn];
int num[maxn];
bool cmp(Student a, Student b)
{
    if (a.grade == b.grade)
    {
        return (a.num < b.num);
    }
    return (a.grade > b.grade);
}
bool cmp2(Student a, Student b) {
    return a.num < b.num;
}
int main()
{
    while (~scanf("%d", &n))
    {
        for (int i = 0; i < n; i++)
        {
            scanf("%s %d %lf", &stu[i].name, &stu[i].num, &stu[i].grade);
        }
        sort(stu, stu + n, cmp);
        for(int i=0;i<n;i++)
        {
            stu[i].position = i + 1;
```

```
        }
        sort(stu, stu + n, cmp2);
        for (int i = 0; i < n; i++)num[i] = stu[i].num;
        scanf("%d", &m);
        int search1;
        for (int i = 0; i < m; i++)
        {
            scanf("%d", &search1);
            int* pos = lower_bound(num, num + n, search1);
            if (num[pos - num] == search1) {
                printf("%s %d %.2lf %d\n", stu[pos - num].name, stu[pos -
num].num, stu[pos - num].grade, stu[pos - num].position);
            }
            else {
                printf("Only god knows where he is.\n");
            }
        }
    }
}
```

E Gzh又去上自习

时间限制：5ms 内存限制：1800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

众所周知，Gzh是一个十分热爱学习的人，所以这一次，他不仅自己去上自习，还带领了他的小伙伴们一起去上自习。假设他有mm个小伙伴，有nn个自习室，每个小伙伴都可能去任意一个自习室，请给出他们上自习所有可能情况的数量。

输入

第一个数为数据组数t(1≤t≤100)t(1≤t≤100)

接下来tt行，每行2个整数m,n(1≤m,n≤100)m,n(1≤m,n≤100)

输出

对于每组数据，输出一行，为可能的情况的数量

输入样例

```
1
7 3
```

输出样例

```
8
```

Hint

每个自习室都是相同的，只考虑自习室中的人数不考虑是谁，即假设有4个小伙伴，5个自习室，那么每个自习时分别有1 1 1 1 0和0 1 1 1 1个人算是一种情况

知识点

动态规划

顺带一提，n个球进m个箱子这样的模型，随着球和箱子是否重复是否可空，情况十分复杂，在组合数学中也是重点内容，读者可以就此好好总结

AC Code

```
#pragma GCC optimize(2)
#include<stdio.h>
#define maxn 105
typedef long long ll;
int dp[maxn][maxn];
inline void buildDP() {
  dp[1][1] = 1;
  for (int i = 2; i < maxn; ++i) dp[1][i] = dp[i][1] = 1;
  for (int i = 2; i < maxn; ++i) {
    for (int j = 2; j < i; ++j) {
      dp[i][j] = ((i - j >= j) ? dp[i - j][j] : dp[i - j][i - j]) + dp[i][j -
1];
    }
    dp[i][i] = 1 + dp[i][i - 1];
  }
}
int t, m, n;
int main() {
  buildDP();
  scanf("%d", &t);
  while (t--) {
    scanf("%d%d", &m, &n);
    if (m < n)printf("%d\n", dp[m][m]);
    else printf("%d\n", dp[m][n]);
  }
}
```

F ModricWang的下午茶

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

ModricWang是个吃货，在一个晴朗的下午，他把竖式里的一些数字吃掉了，但是他不记得自己到底吃了哪些数字，需要你们根据剩下的数字来判定被吃掉的数字。来看一个简单的例子：

```
   43#9865#045
+    8468#6633
   44445509678
```

其中#号代表被ModricWang吃掉的数字。根据算式，我们很容易判断：第一行的两个数字分别是5和3，第二行的数字是5。

现在，我们对问题做两个限制：

首先，我们只考虑加法。这里的加法是N进制加法，并且算式中三个数都有N位，允许有前导的0。

其次，ModricWang把所有的数都吃光了，我们只知道哪些数字是相同的，我们将相同的数字用相同的字母表示，不同的数字用不同的字母表示。如果这个算式是N进制的，我们就取英文字母表午的前N个大写字母来表示这个算式中的0到N-1这N个不同的数字(但是这N个字母并不一定顺序地代表0到N-1)。输入数据保证N个字母分别至少出现一次。

```
      BADC
+     CBDA
      DCCC
```

上面的算式是一个4进制的算式。很显然，我们只要让ABCD分别代表0123，便可以让这个式子成立了。你的任务是，对于给定的N进制加法算式，求出N个不同的字母分别代表的数字，使得该加法算式成立。

输入数据保证有且仅有一组解。

输入

输入包含4行。

第一行有一个正整数N(N<=26)

后面的3行每行有一个由大写字母组成的字符串，分别代表两个加数以及和。

这3个字符串左右两端都没有空格，从高位到低位，并且恰好有N位。

对于30%的数据，保证有N<＝10；

对于50%的数据，保证有N<＝15；

对于全部的数据，保证有N<＝26。

输出

输出包含一行。

在这一行中，应当包含唯一的那组解。

解是这样表示的：输出N个数字，分别表示A，B，C……所代表的数字，相邻的两个数字用一个空格隔开，不能有多余的空格。

输入样例

```
5
ABCED
BDACE
EBBAA
```

输出样例

```
1 0 3 4 2
```

知识点

搜索+多重剪枝

AC Code

```
#include<iostream>
#include<string>
using namespace std;
/*
这道题肯定是要搜索的这个没有办法...
但是通过一定的约束条件，可以剪枝，将问题的规模大大的降低
这种题必须要多见多想，很重要!

(1)1+1=3这种明显不合法的问题
每次搜索的时候，从后向前判断是否有不合法的式子
(2)配合(1)接着做，按照字母的出现顺序搜索，提前减掉一些废枝
实际上还有第三个要点(以下*为已知，?为未知)
    A * * * ? * * *
+   B * ? * * ? * *
----------------------
    C * * * ? ? ? *
(不一定只是这一种特定的情况,只是想举一个特定位的例子而已)
只是说，这种情况并不能直接确定A+B=C是不是合法
```

```
但是比如说(A+B)%N与(A+B+1)%N都不等于C,那就可以直接剪了
类似的A+?=C的时候，看(C-A+N)%N与(C-A-1+N)%N这两个数是否用过
如果都用过，就直接剪了
*/
//基本的流程：初始都是ascii码的字母，都比具体数字要大，当知道一个值之后，就把具体字母换成数字
string a, b, c;
string words;
bool hashlist[256], used[27];//指这个有没有用上
bool finish;//一旦找出正确结果，不再接着找，直接断掉
int n;
int stack[27];
//利用类似栈结构来记录，该栈与最终的结果words一一对应
int ans[27];

inline void addChar(char ch) {
  if (!hashlist[ch]) { hashlist[ch] = true; words += ch; }
  //不重复添加字符
}
inline string change(string str, char x, char y){
  for (int i = 0; i < n; ++i) {
    if (str[i] == x)str[i] = y;
  }
  return str;
}
//第一种剪枝方式，出现非法返回true
inline bool bad() {
  //检查某一结果的值是否合法
  int p, g = 0;//g表示进位 p表示a+b当前位再加进位之后的位置
  for (int i = n - 1; i >= 0; --i) {
    if (a[i] >= n || b[i] >= n || c[i] >= n)return false;//目前该组合是未知的
    p = a[i] + b[i] + g;
    if (p % n != c[i])return true;//明显的不合法结果
    g = p / n;
  }
  return false;
}
//第二种剪枝方式，出现非法返回true
inline bool checkMod() {
  int p, p1, p2, g = 0;
  //A+B=C
  for (int i = n - 1; i >= 0; --i) {
    if (a[i] >= n || b[i] >= n || c[i] >= n)continue;//目前是未知数，还无法判断是否
合法
    p = (a[i] + b[i]) % n;
    if (!(p % n == c[i] || (p + 1) % n == c[i]))return true;//上面已经分析过了
  }
  //A+? = C
  for (int i = n - 1; i >= 0; --i) {
    if (!(a[i] < n && b[i] >= n && c[i] < n))continue;
```

```
    p1 = (c[i] - a[i] + n) % n;
    p2 = (c[i] - a[i] - 1 + n) % n;
    if (used[p1] && used[p2])return true;
  }
  //?+B = C
  for (int i = n - 1; i >= 0; --i) {
    if (!(a[i] >= n && b[i] < n && c[i] < n))continue;
    p1 = (c[i] - b[i] + n) % n;
    p2 = (c[i] - b[i] - 1 + n) % n;
    if (used[p1] && used[p2])return true;
  }
  //A+B = ?
  for (int i = n - 1; i >= 0; --i) {
    if (!(a[i] < n && b[i] < n && c[i] >= n))continue;
    p1 = (a[i] + b[i]) % n;
    p2 = (a[i] + b[i] + 1) % n;
    if (used[p1] && used[p2])return true;
  }
  return false;
}
inline void outputResult() {
  for (int i = 0; i < n; ++i) {
    ans[words[i] - 65] = stack[i];
  }
  for (int i = 0; i < n; i++) {
    cout << ans[i] << " ";
  }
  finish = true;
}
inline void dfs(int level) {
  int i;
  string tmpA, tmpB, tmpC;
  if (finish)return;
  if (bad())return;
  if (checkMod())return;
  if (level == n) {
    outputResult();
    return;
  }

  for (i = n - 1; i >= 0; --i) {
    if (!used[i]) {
      used[i] = true;//往下走一层，假设某个数就是i了
      tmpA = a; a = change(tmpA, words[level], i);
      tmpB = b; b = change(tmpB, words[level], i);
      tmpC = c; c = change(tmpC, words[level], i);
      stack[level] = i;
      dfs(level + 1);
      used[i] = false; a = tmpA, b = tmpB, c = tmpC;
```

```
    }
  }
}
int main() {
  ios::sync_with_stdio(false);
  cin >> n >> a >> b >> c;
  words = "";//记录出现了哪些值
  for (int i = n - 1; i >= 0; --i) {
    addChar(a[i]); addChar(b[i]); addChar(c[i]);
  }
  dfs(0);
  return 0;

}
```

第2次上机

A Mdd的线性表

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd想实现一个线性表，不过代码那部分还是得请你们帮他实现。这个线性表是固定大小的，支持3种操作：

ins x pos，将x插入到线性表的某个位置上。

del x，将第一个值为x的数删掉。

sum，求所有元素之和并输出。

输入

多组输入数据

第一行一个数n，代表共有n个操作，可能为ins,del,sum中的一种，表的大小为500。

输出

对于每组操作，如果操作不合法则输出“invalid operation!"。

输入样例

```
4
sum
ins 1 0
sum
del 0
```

输出样例

```
invalid operation!
1
invalid operation!
```

Hint

操作不合法包括插入位置不合法，要删除的元素不存在，对空表进行求和，对一个满的表进行插入等等。
对于一个长度为2的表，它的可插入位置为0，1，2，分别对应表头，原有的两个元素中间，表尾。

知识点

顺序表的基本操作

AC Code

```
#include<iostream>
using namespace std;
#define maxn 500
typedef struct list
{
    long long  data[maxn+1];
    int number;
    int length;
};
list a;
int n;
void initlist(list &a)
{
    a.length=maxn;
    a.number=0;
}
void ins(list &a,long long  x,int y)
{
    if(a.number==a.length)
        cout<<"invalid operation!"<<endl;
    else if(y<0||y>a.number)
        cout<<"invalid operation!"<<endl;
    else
```

```
    {
        //cout<<"OK"<<endl;
        a.number++;
        for(int i=a.number;i>y;i--)
        {
            a.data[i]=a.data[i-1];
        }
        a.data[y]=x;
    }
}
void del(list &a,long long  x)
{
    if(a.number==0)
        cout<<"invalid operation!"<<endl;
    else
    {
        int flag=0;
        for(int i=0;i<a.number;i++)
        {
            if(a.data[i]==x)
            {
                for(int j=i;j<a.number-1;j++)
                {
                    a.data[j]=a.data[j+1];
                }
                a.number--;
                flag=1;
                break;
            }
        }
        if(!flag)
            cout<<"invalid operation!"<<endl;
    }
}
void summ(list a)
{
    if(a.number==0)
        cout<<"invalid operation!"<<endl;
    else
    {
        long long  sum1=0;
        for(int i=0;i<a.number;i++)
        {
            sum1+=a.data[i];
        }
        cout<<sum1<<endl;
    }
}
int main(){
```

```
    ios::sync_with_stdio(false);
    int n;
    char operat[5];
    long long  x,y;
    while(cin>>n)
    {
        initlist(a);
        while(n--)
        {
            cin>>operat;
            switch(operat[0])
            {
                case 's':
                    summ(a);
                    break;
                case 'i':
                    cin>>x>>y;
                    ins(a,x,y);
                    break;
                case 'd':
                    cin>>x;
                    del (a,x);
                    break;
            }
        }
    }
  return 0;
}
```

B Mdd的链表

时间限制：1000ms 内存限制：10000kb

通过率：/ (%) 正确率：/ (%)

题目描述

题目要求很简单，给你一组排好序的数，把重复的去掉。

输入

多组输入数据

第一个数为n，代表共有n个非负数。

接下来一行，共有n个已排好序的数。

输出

对于每组数据，输出2行，第一行为刚建立完的链表，第二行为去重之后的链表。

输入样例

```
4
1 2 2 3
```

输出样例

```
1 2 2 3
1 2 3
```

Hint

链表的声明大致如下：

```
struct SqList {
    int val;
    SqList *next;
};
```

请务必先建表再输出。

知识点

有序数组/链表的去重

AC Code

```
#include<cstdio>
#define maxn 100010
int n;
int a[maxn];
void uniqfy(int* array, int* size) {
  int i = 0, j = 0;
  while (++j < *size) {
    if (array[i] != array[j])//跳过雷同者
      array[++i] = array[j];
  }
  *size = ++i;
  return;
}
int main() {
  while (scanf("%d", &n) != EOF) {
    for (int i = 0; i < n; ++i)scanf("%d", a + i);
```

```
    for (int i = 0; i < n; ++i)printf("%d ", a[i]);
    uniqfy(a, &n); puts("");
    for (int i = 0; i < n; ++i)printf("%d ", a[i]);
    puts("");
  }
}
```

c ModricWang的文本排版

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang穿越到了1985年的一个编辑部， 变成了一个编辑。这天ModricWang接到一段英文，要对其进行排版，人工排版非常麻烦，他想借助于计算机进行自动排版。现要求每行宽度为n个字符。如果一行的最后一个单词超出了本行的n个字符的范围，则应把它移到下一行去，并在每个单词前增加一些空格，以便每行的末尾准确地处于第n个字符处。（首尾共n个字符，且单词与单词间空格较均匀)。如果一行只有一个单词，则直接输出。

如果空格不能均匀分布，那么使前面尽量多的空格数相等，最后一个空位允许多空一些，但最后一个空位的空格数不能比前面少。如果是8个空格，3个空，应该排列成2-2-4。

最后一行不需要右端对齐，单词间空1个空格即可。

输入

输入有2行，第1行为1个正整数，表示每行宽度N(N<100)个字符，第2行有若干个单词组织成的句子(长度不超过250)，单词与单词之间有空格隔开(约定每个单词字符的个数小于n)。

输出

输出时每行宽度为n个字符。如果一行的最后一个单词超出了本行的n个字符的范围，则应把它移到下一行去，并在每个单词前增加一些空格，以便每行的末尾准确地处于第n个字符处。（首尾共n个字符，且单词与单词间空格较均匀)。如果一行只有一个单词，则直接输出。

输入样例

```
20
Angela    dreamed    many times     about    going    fishing
```

输出样例

```
Angela dreamed  many
times  about   going
fishing
```

HINT

链表

知识点

字符串处理

AC Code

```cpp
#include<cstdio>
#include<cstring>
#include<string>
#include<vector>
#define maxn 256
using namespace std;
int n;
int cur;
vector<string> line;
string real_input;
char input[maxn];
inline void handleSpace() {
  int need = line.size() - 1;
  int require = n - cur;
  if (need) {
    int cnt = require / need;
    int last = require - (need - 1) * cnt;
    string out = "";
    for (int i = 0; i < line.size() - 1; ++i) {
      out += line[i];
      out.insert(out.end(), cnt, ' ');
    }
    if(last > cnt)out.insert(out.end(), last - cnt, ' ');
    out += line[line.size() - 1];
    printf("%s\n", out.c_str());
  }
  else {
    string out = "";
    //out.insert(out.end(), require, ' ');
    out += line[0];
    printf("%s\n", out.c_str());
  }
  line.clear();
  cur = 0;
```

```
}
int main() {
  scanf("%d", &n);
  cur = 0;
  while (scanf("%s", input) != EOF) {
    real_input = input;
    if (real_input.size() + line.size() + cur > n) {
      if (cur) { handleSpace(); line.push_back(input); cur +=
real_input.size(); }
      else printf("%s\n", input);
    }
    else {
      line.push_back(input); cur += real_input.size();
    }
  }
  if (!line.empty()) {
    for (string a : line)printf("%s ", a.c_str());
  }
}
```

D DH的新手机

时间限制：1000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH最近买了个新手机，这个新手机的系统是一个新系统叫DS。这部手机安装了nn个应用程序，每个程序都有它的图标。这些图标在不同的屏幕上，一个屏幕上有kk个图标。第11个到第kk个图标在第一个屏幕上，第k+1k+1个到第2k2k个图标在第二个屏幕上，以此类推（最后一个屏幕可能会不足kk个图标）。

开始时手机菜单显示的是第一个屏幕，想要启动第tt个屏幕上的图标，DH需要先滑动屏幕t−1t−1次，再点击该图标。

这个程序启动以后，将回到第一个屏幕，也就是说想要启动下一个应用程序又要从第一个屏幕开始先滑动再点击了。

应用程序的编号是从11到nn的，我们现在知道刚开始每个应用程序所在的位置，不过因为DS系统的一个特殊功能，应用程序的位置是会随着DH的使用改变的。DS系统会把更常用的应用程序放在前边，更准确地说，当DH启动了一个应用程序后，这个应用程序会和它前边一位的应用程序交换位置，当然这两个相邻的程序可能在不同的屏幕上，这不影响，该交换还是会交换的，不过如果这个应用程序在最前边，就不会有交换顺序的事情发生了。

DH已经计划好了启动应用程序的顺序，他想知道这么做的话他需要做多少次操作（滑动屏幕和点击都算操作）。

需要注意的是一个应用程序可能会被启动多次。

输入

多组输入数据

对于每组数据，第一行有三个整数n,m,k(1≤n,m,k≤105)n,m,k(1≤n,m,k≤105)，分别表示DH新手机安装的应用程序数量，DH想要启动多少次应用程序，每个屏幕上的图标数量。

第二行有nn个整数，a1,a2,...,ana1,a2,...,an，表示刚开始时图标的顺序，aiai代表编号为aiai的程序在第ii位。保证a1,a2,...,ana1,a2,...,an为11到nn的一个排列。

第三行有mm个整数,b1,b2,...,bnb1,b2,...,bn，表示DH计划启动应用程序的顺序，bibi代表编号DH计划启动的第ii个程序是bibi。一个程序可能被启动多次。

输出

对于每组数据，输出一行，为一个整数，表示DH按他计划的启动应用程序的顺序需要进行的操作次数。

输入样例1

```
8 3 3
1 2 3 4 5 6 7 8
7 8 1
```

输出样例1

```
7
```

输入样例2

```
5 4 2
3 1 5 2 4
4 4 4 4
```

输出样例2

```
8
```


知识点

优化建模

AC Code

```
#include<cstdio>
#include<algorithm>
#define maxn 100010
```

```cpp
using namespace std;
typedef long long ll;

ll pos[maxn], num[maxn];
ll n, m, k;
ll x;
ll res, Search;
ll pos1, num1, pos2, num2;
int main() {
  while (scanf("%lld%lld%lld", &n, &m, &k) != EOF) {
    res = 0;
    for (ll i = 1; i <= n; ++i) {
      scanf("%lld", &x);
      pos[x] = i, num[i] = x;
    }
    for (ll i = 1; i <= m; ++i) {
      scanf("%lld", &Search);
      res += (pos[Search] - 1) / k + 1;
      if (pos[Search] != 1) {
        pos1 = pos[Search], num1 = Search;
        num2 = num[pos1 - 1], pos2 = pos[num2];
        swap(num[pos1], num[pos2]);
        swap(pos[num1], pos[num2]);
      }
    }
    printf("%lld\n", res);
  }
}
```

E Gzh最后一次上自习

时间限制：35ms 内存限制：3000kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

Gzh觉得天天都上自习的日子实在是太无聊了，所以他决定他这次上自习就会是他最后一次上自习了。

众所周知，Gzh上自习的方式和他去的自习室总是与众不同的，这次也不例外。

Gzh这次要去的自习室，每个自习室有n个小房间(3<=n<=100000)，每个小房间可以坐1-6个人，相邻的两个房间的人数之差不能大于4且至少有三个容量不同的小房间，那么，Gzh想知道自习室的小房间数目为n时，自习室构造的所有情况。结果对7777777取模。

输入

第一个数为数据组数t 接下来n行，每行1个整数n

输出

对于每组数据，输出一行，为自习室小房间数目为n时，自习室所有可能的构造的数量。

输入样例

```
1
3
```

输出样例

```
104
```

知识点

组合数学/排列组合 动态规划

这题的时间空间限制就注定了这题就是一个纯数学推导的问题，码量很少，推不出来就是不会做

具体的推导过程较复杂，不多说，仅给几个提示

这题很显然是1和6不能紧邻，且必须出现1-6的3种及以上。所以可以分为仅需要满足1-6不紧邻的排列an和1-6不紧邻但是只出现1种和2种的情况bn 所求的cn=an-bn

同样的，也可以写一个纯搜索的程序来验证前面10的正确性。

然后打表的时候，还要注意到mod与减法的问题。~~所以场上A掉这题基本上不可能的，当时场上也确实没人A掉~~

AC Code

```cpp
#pragma G++ optimize(2)
#include<cstdio>
#define MOD 7777777
#define maxn 100010
typedef long long ll;
ll dp[maxn];
inline void buildDP() {
  dp[1] = 0;
  dp[2] = 0;
  dp[3] = 104;
  dp[4] = 904;
  for (int i = 5; i < maxn; ++i) {
    dp[i] = 7 * dp[i - 1] - 6 * dp[i - 2] - 8 * dp[i - 3] + 176;
```

```
    while (dp[i] < 0)dp[i] += MOD;
    dp[i] %= MOD;
  }
}
int n,t;
int main() {
  buildDP();
  scanf("%d", &t);
  while (t--) {
  scanf("%d", &n);
    printf("%d\n", dp[n]);
  }
}
```

F DH的矩阵游戏

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH是个菜鸡，不过他热爱矩阵。现在他得到了一个nn行mm列的矩阵，他想进行qq次操作，每一次操作是将其中的两个子矩阵交换。

对于每一次操作，DH会想出6个整数ai,bi,ci,di,hi,wiai,bi,ci,di,hi,wi，aiai代表第一个矩形左上角所在的行，bibi代表其所在的列，cici代表第二个矩形左上角所在的行，didi代表其所在的列，hihi表示矩形所占行数，wiwi表示矩形所占列数。

保证一次操作的两个矩形不会有重叠部分，也不会有边与边的接触，即不会有一个元素同时在这两个矩形中，存在于不同矩形的元素不会有重合的边。不过两个矩形是允许有一个相接触的角的。

现在他想知道当所有的操作都进行完以后矩阵的样子。

输入

多组输入数据 对于每一组输入数据，第一行为3个整数 n,m,q(2≤n,m≤1000,1≤q≤10000)n,m,q(2≤n,m≤1000,1≤q≤10000)，表示DH开始时得到的矩阵的行数、列数，DH想进行的操作次数。

接下来nn行，每行mm个整数vi,j(1≤vi,j≤109)vi,j(1≤vi,j≤109)，表示矩阵元素初始值。

接下来qq行，每行6个整数 ai,bi,ci,di,hi,wi(1≤ai,ci,hi≤n,1≤bi,di,wi≤m)ai,bi,ci,di,hi,wi(1≤ai,ci,hi≤n,1≤bi,di,wi≤m)。

输出

每组数据输出nn行，每行mm个整数，表示最终的矩阵。

输入样例1

```
4 4 2
1 1 2 2
1 1 2 2
3 3 4 4
3 3 4 4
1 1 3 3 2 2
3 1 1 3 2 2
```

输出样例1

```
4 4 3 3
4 4 3 3
2 2 1 1
2 2 1 1
```

输入样例2

```
4 2 1
1 1
1 1
2 2
2 2
1 1 4 1 1 2
```

输出样例2

```
2 2
1 1
2 2
1 1
```

知识点

十字链表

提示：链表的表示不止有指针，利用下标数字会使操作更方便

AC Code

```
#pragma G++ optimize(2)
#include<cstdio>
```

```cpp
#include<algorithm>
#define maxn 1010
using namespace std;
struct node {
  int right;
  int down;
  int value;
}Node[maxn * maxn];
int a[maxn][maxn], b[maxn][maxn];
int idx;
inline void write(int x) {
  if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
inline int move(int p, int x, int y) {
  while (x--) p = Node[p].down;
  while (y--)p = Node[p].right;
  return p;
}
inline void print(int n, int m) {
  int p = b[0][0];
  for (int i = 1; i <= n; ++i) {
    p = move(p, 1, 0);
    int ip = p;
    for (int j = 0; j < m; ++j) {
      ip = Node[ip].right;
      write(Node[ip].value);
      if (j == m - 1)putchar('\n');
      else putchar(' ');
    }
  }
}
int n, m, q;
int ai, bi, ci, di, hi, wi;
int main() {
```

```
  while (scanf("%d%d%d", &n, &m, &q) != EOF) {
    idx = 0;
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= m; ++j) {
        a[i][j] = read();
      }
    }
    //初始化十字链表
    for (int i = 0; i <= n; ++i) {
      for (int j = 0; j <= m; ++j) {
        b[i][j] = idx++;
        Node[b[i][j]].value = a[i][j];
        if (j)Node[b[i][j - 1]].right = b[i][j];
        if (i)Node[b[i - 1][j]].down = b[i][j];
      }
    }
    //读入操作
    while (q--) {
      ai = read(), bi = read(), ci = read(), di = read(), hi = read(), wi =
read();
      int p1 = move(b[0][0], ai - 1, bi - 1);
      int p2 = move(b[0][0], ci - 1, di - 1);

      int dp1 = move(p1, 1, 0), dp2 = move(p2, 1, 0);

      for (int i = 0; i < hi; ++i) {
        swap(Node[dp1].right, Node[dp2].right);
        dp1 = move(dp1, 1, 0), dp2 = move(dp2, 1, 0);
      }

      int rp1 = move(p1, 0, 1), rp2 = move(p2, 0, 1);
      for (int i = 0; i < wi; ++i) {
        swap(Node[rp1].down, Node[rp2].down);
        rp1 = move(rp1, 0, 1), rp2 = move(rp2, 0, 1);
      }

      dp1 = move(p1, 1, wi), dp2 = move(p2, 1, wi);

      for (int i = 0; i < hi; ++i) {
        swap(Node[dp1].right, Node[dp2].right);
        dp1 = move(dp1, 1, 0), dp2 = move(dp2, 1, 0);
      }

      rp1 = move(p1, hi, 1), rp2 = move(p2, hi, 1);
      for (int i = 0; i < wi; ++i) {
        swap(Node[rp1].down, Node[rp2].down);
        rp1 = move(rp1, 0, 1), rp2 = move(rp2, 0, 1);
      }
```

```
    }
    print(n, m);
  }
}
```

第3次上机

A Mdd的链表(II)

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

题目很简单，给你一个链表，要求把**倒数第n个**数去掉。（签到题）

输入

共一组数据。

第一行一个数n(保证n合法)，第二行为一个链表

输出

输出两行，第一行为去掉的数，第二行为操作之后的链表。

输入样例

```
1
1 2 3 4
```

输出样例

```
4
1->2->3
```

Note

```
链表的声明大致如下：
struct SqList {
    int val;
    SqList *next;
};
```

知识点

水

AC Code

```
#include<cstdio>
#include<string>
#define maxn 100010
FILE* in;
int n;
int size;
int a[maxn];
int main() {
  //in = freopen("708.in", "r", stdin);
  scanf("%d", &n);
  while (scanf("%d", &a[size]) != EOF)size++;
  printf("%d\n", a[size - n]);
  int cnt = 0;
  for (int i = 0; i < size; ++i) {
    if (i != size - n) {
      printf("%d", a[i]);
      if (++cnt != size - 1)printf("->");
    }
  }
}
```


B Mdd玩炉石

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

玩过炉石传说的人都知道，每个玩家手上的牌次序是不可以变的，早抓的牌就在手牌的左边，所以有经验的玩家有时候可以根据卡牌的位置猜到这张牌。假如有一张牌一直都在卡在手上，那很大几率是一张AOE。所以为了防止对手猜到手牌，mdd决定每次都交换卡牌的位置。虽然在现实中是不可能的，因为设计师觉得这种设计很cooooool.

输入

只有一组数据

接下来两行，第一行一个正整数k，第二行一个链表。

输出

输出一行，链表中每有k个数便将这k个数反转 (请不要采取仅仅交换值的方式)，输出整个链表。

输入样例

```
3
1 2 3 4 5 6 7 8
```

输出样例

```
3 2 1 6 5 4 7 8
```

Note

```
链表的声明大致如下：
struct SqList {
    int val;
    SqList *next;
};
```

知识点

链表

AC Code

```
#include<iostream>
using namespace std;
typedef struct node
{
    int data;
    struct node *next;
};
node *head;
int length=0;
int flag=0;
node *create()
{
    node *p,*q,*r,*head;
```

```
    head=new node;
    p=new node;
    int x;
    cin>>x;
    p->data=x;
    head->next=p;
    q=p;
    length++;
    while(cin>>x)
    {
        p=new node;
        p->data=x;
        p->next=NULL;
        q->next=p;
        q=p;
        length++;
    }
    q->next=NULL;
    return head;
}
node *reverse(node *head,int n)
{
    node *p,*q,*r,*s,*c;
    p=head;
    q=p->next;
    r=q->next;
    s=q;
    c=head;
    int counter=1;
    if(n>length)
       return head;
    int cnt=length/n;
    int a=0;
    for(int i=0;i<cnt;i++)
    {
        while(counter<n)
        {
            int x=r->data;
            q->next=r->next;
            delete r;
            r=new node;
            r->data=x;
            p->next=r;
            r->next=s;
            s=r;
            //a++;
            counter++;
            r=q->next;
        }
```

```
        //cout<<p->data<<" "<<q->data<<" "<<r->data<<endl;
        p=q;
        if(q==NULL||q->next==NULL||q->next==NULL)
            break;
        q=q->next;
        r=q->next;
        s=q;
        counter=1;
        //cout<<p->data<<" "<<q->data<<" "<<r->data<<endl;
    }
    return head;
}
void print(node *head)
{
    node *p=head->next;
    while(p!=NULL)
    {
        cout<<p->data<<" ";
        p=p->next;
    }
    cout<<endl;
}
int main()
{
    int n;
    struct node *head;
    cin>>n;
    head=create();
    reverse(head,n);
    print(head);
}
```


C Gzh之女篮风云

时间限制：500ms 内存限制：5000kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh混入了软院女篮的qq群，这是众所周知的。这一天，他突然偷听到女篮队员们要玩一个游戏：

所有女篮队员们会站成一个圈，并以一个人为起点，依次标号1，2.....

教练首先会给1，2号球员每人一个球，然后给4号一个球，然后给7号一个球，然后给11号一个球，以此类推（每次跳过的人数+1）。

那么，Gzh想知道，当女篮人数为n时，最后会不会每个人手里都有球呢。

输入

多组输入数据

每组一个数n，代表女篮的人数 （n在long long之内）

输出

对于每组数据，输出一行，如果可以，输出GzhIsSoHandsome，否则输出GzhIsHandsome

输入样例

```
2
```

输出样例

```
GzhIsSoHandsome
```

Hint

此题严查代码，交流的注意了

知识点

找 规 律 （说实话我到现在都不会证）

AC Code

```
#include<cstdio>
typedef unsigned long long ull;
ull n;
int main(){
  while(scanf("%llu",&n)!=EOF){
    int res = 0;
    while(n)n=n&(n-1),res++;
    if(res==1)puts("GzhIsSoHandsome");
    else puts("GzhIsHandsome");
  }
}
```


D ModricWang's JOSEPHUS Problem

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

约瑟夫问题是个有名的问题：N个人围成一圈，从第一个开始报数，第M个将被杀掉，最后剩下一个，其余人都将被杀掉。例如N=6，M=5，被杀掉的顺序是：5，4，6，2，3。1存活了下来。

可怕的事情发生了，ModricWang的妹妹也在这个圈里，幸好ModricWang能暗中改变所有人的相对位置，那么，ModricWang需要将他的妹妹安排在哪个位置，才能让妹妹存活?

输入

两个数字，N和M

N≤1e7，M≤1e7N≤1e7，M≤1e7

输出

对于每组数据，输出一行，可以存活的位置

输入样例

```
6 5
```

输出样例

```
1
```

知识点

找规律（但是这个规律是公认的，而且具体数学书上专门介绍过）

AC Code

```
#include<cstdio>
int n, m;
inline int cir(int n, int m)
{
  int p = 0;
  for (int i = 2; i <= n; i++)
  {
    p = (p + m) % i;
  }
  return p + 1;
}
int main() {
  while (scanf("%d%d%d", &n, &m) != EOF) {
    printf("%d\n",cir(n, m));
  }
```

```
}
```

E DH的杀人游戏

时间限制：1000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH是一个变态杀人狂（看你们的上机就知道了），这一天他抓来了nn个人，强迫他们进行了一场杀人游戏（真•杀人游戏）。游戏开始时，他分给这nn个人每人一个编号，为了方便，这个编号就是11至nn中的某一个数，并且他们的编号互不相同。然后DH把这nn个人排成一排，告诉了他们游戏规则。活着的所有人同时向右举枪，如果一个人紧右边的人编号比自己小，他就开枪杀了他紧右边的人，最右边的人因为右边没人所以不会杀人，这称为一轮游戏。注意一轮中每个人是同时开枪的，所以有可能出现一个人同时杀人和被杀的可能。每一轮结束后，DH会让人把尸体抬出去，所有活人向左看齐补位。游戏终有结束的一天，那么什么时候算做结束呢，就是不会再有人死了。DH想要知道这个游戏最多可以进行几轮?

输入

多组输入数据

对于每一组数据，第一行为一个整数n(1≤n≤105)n(1≤n≤105)，表示参加杀人游戏的人数。

第二行nn个整数，第ii个整数代表刚开始的一排中从左往右数第ii个人的编号。

输出

对于每组数据，输出一行，包含一个整数，为游戏可以进行的轮数

输入样例1

```
10
10 9 7 8 6 5 3 4 2 1
```

输出样例1

```
2
```

输入样例2

```
6
1 2 3 4 5 6
```

输出样例2

```
0
```

知识点

线段树/单调栈（推荐）

维护一个严格递减的单调栈，如果进来了一个破坏单调性的元素，则意味着前面所有比他小的元素，都将在同一轮之内被击杀。以此特点，建立时间轴进行求解即可。

AC Code

```cpp
#include<cstdio>
#include<stack>
#include<algorithm>
#include<cstring>
#define maxn 100010
#define mp make_pair
using namespace std;
typedef pair<int, int> PII;
stack<PII>s;
int a[maxn];
int l[maxn];
int n;
int ans;
int main() {
  while (scanf("%d", &n) != EOF) {
    ans = 0;
    memset(a, 0, sizeof(a));
    memset(l, 0, sizeof(l));
    for (int i = 1; i <= n; ++i)scanf("%d", a + i);
    while (!s.empty())s.pop();
    int Max = 0, ans = 0;
    for (int i = 1; i <= n; ++i) {
      Max = 0;
      while (!s.empty() && a[s.top().first] < a[i]) {
        Max = max(Max, s.top().second);
        s.pop();
      }
      if (s.empty())ans = max(ans, Max);
      else Max = max(Max, s.top().second), s.top().second = Max + 1;
      s.push(mp(i, 0));
    }
```

```
    while (!s.empty())
      ans = max(ans, s.top().second), s.pop();
    printf("%d\n", ans);

  }
}
```

F ModricWang's JOSEPHUS Problem II

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

原始的Josephus问题的描述见前一题。

ModricWang的新问题是：假设有k个好人和k个坏人。好人的编号是1到k，坏人的编号是k+1到2k。我们希望求出m的最小值，使得最先出列的k个人都是坏人。

输入

输入文件仅有一行包含一个整数k (0<k<14)。

输出

输出文件仅有一行包含一个整数，表示使得最先出列的k个人都是坏人的m的最小值。

输入样例

```
4
```

输出样例

```
30
```

知识点

模拟

AC Code

```
#include<cstdio>
```

```
int n;
int main() {
  while (scanf("%d", &n) != EOF) {
    bool flag = true;
    int m = n;
    while (flag) {
      int ptr = 0;
      m++;
      for (int i = 0; i < m; ++i) {
        ptr = (ptr + m - 1) % (2 * n - i);
        if (ptr < n)break;
        if (i == n - 1)flag = 0;
      }
    }
    printf("%d\n", m);
  }
}
```

第4次上机

A Mdd来排队(II)

时间限制：1000ms 内存限制：4096kb

通过率：/ (%) 正确率：/ (%)

题目描述

每天中午食堂都会有很多人，为了维持秩序，食堂的负责人决定钦点mdd来负责排队。因为食堂空间有限，**每条队伍只能容纳20个人**，人满的话是进不了队伍的。

输入

一组输入数据

对于每组数据，一开始队伍为空

每组数据都有多行命令，分别是offer x，poll，peek，分别表示让编号为x的同学进入队伍，让队首的同学离开，查询队首的编号。

输出

如果命令是peek，输出队首同学的编号。在操作过程中，如果操作不合法，则输出“fool!”

输入样例

```
offer 1
peek
poll
peek
```

输出样例

```
1
fool!
```

知识点

队列

AC Code

```cpp
#include<cstdio>
#include<queue>
#include<cstring>
#define MAXN 20
using namespace std;
queue<int>q;
char input[10];
char offer[7] = "offer";
int op;
char poll[7] = "poll";
char peek[7] = "peek";
int main() {
  while (scanf("%s", input) != EOF) {
    if (!strcmp(input, offer)) {
      scanf("%d", &op);
      if (q.size() >= MAXN)puts("fool!");
      else q.push(op);
    }
    else if (!strcmp(input, poll)) {
      if (q.empty())puts("fool!");
      else q.pop();
    }
    else {
      if (q.empty())puts("fool!");
      else printf("%d\n", q.front());
    }
  }
}
```

B ModricWang合并果子

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

在一个果园里，ModricWang已经将所有的果子打了下来，而且按果子的不同种类分成了不同的堆。ModricWang决定把所有的果子合成一堆。

每一次合并，ModricWang可以把两堆果子合并到一起，消耗的体力等于两堆果子的重量之和。可以看出，所有的果子经过n-1次合并之后，就只剩下一堆了。ModricWang在合并果子时总共消耗的体力等于每次合并所耗体力之和。

因为还要花大力气把这些果子搬回家，所以ModricWang在合并果子时要尽可能地节省体力。假定每个果子重量都为1，并且已知果子的种类数和每种果子的数目，你的任务是设计出合并的次序方案，使ModricWang耗费的体力最少，并输出这个最小的体力耗费值。

例如有3种果子，数目依次为1，2，9。可以先将1、2堆合并，新堆数目为3，耗费体力为3。接着，将新堆与原先的第三堆合并，又得到新的堆，数目为12，耗费体力为12。所以ModricWang总共耗费体力=3+12=15。可以证明15为最小的体力耗费值。

输入

两行

第一行是一个整数n(1≤n≤10000)(1≤n≤10000)，表示果子的种类数。

第二行包含n个整数，用空格分隔，第i个整数ai(1≤ai≤20000)(1≤ai≤20000)是第i种果子的数目。

输出

最小的体力耗费值，保证在int范围内

输入样例

```
3
1 2 9
```

输出样例

```
15
```

知识点

优先队列应用

这其实算的就是哈夫曼树的带权路径长度

顺带一提，如果n的数量再变大的话，则需要使用桶排序/计数排序等O(n)的方法，建议读者尝试

AC Code（O(nlogn) Ver）

```
#include<cstdio>
#include<queue>
using namespace std;
unsigned long long result;
long long counter;
int n;
int tmpint;

int main() {

  while (scanf("%d", &n) != EOF) {
    result = 0;
    priority_queue<long long, vector<long long>, greater<long long> > lalala;
    for (int i = 0; i < n; ++i) {
      scanf("%d", &tmpint);
      lalala.push((long long)tmpint);
    }
    while (!lalala.empty()) {
      counter = 0;
      counter += lalala.top();
      lalala.pop();
      if (lalala.empty())break;
      counter += lalala.top();
      lalala.pop();
      result += counter;
      lalala.push(counter);
    }
    printf("%lld\n", result);
  }
}
```

AC Code（计数排序 O(n) Ver）

```
#include<cstdio>
#include<queue>
#define maxn 100010
using namespace std;
typedef unsigned long long ull;
queue<ull>q1, q2;
int bucket[maxn];
inline void write(ull st) {
  //if (x < 0)putchar('-'), x = -x;
  if (st > 9)write(st / 10);
```

```
  putchar(st % 10 + 48);
}
inline ull read() {
  ull k = 0;// f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;// *f;
}
int main() {
  ull n = read();
  for (int i = 1; i <= n; ++i) {
    ull a = read();
    bucket[a]++;
  }
  for (int i = 1; i < maxn; ++i) {
    while (bucket[i]) {
      bucket[i]--, q1.push(i);
    }
  }
  ull ans = 0;
  for (int i = 1; i < n; ++i) {
    ull x, y;
    if (q2.empty() || (q1.front() < q2.front() && !q1.empty()))
      x = q1.front(), q1.pop();
    else
      x = q2.front(), q2.pop();
    if (q2.empty() || (q1.front() < q2.front() && !q1.empty()))
      y = q1.front(), q1.pop();
    else
      y = q2.front(), q2.pop();
    ans += (x + y);
    q2.push(x + y);
  }
  write(ans), putchar('\n');
}
```


c Gzh之返老还童

时间限制：1000ms 内存限制：3000kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh一觉醒来，发现自己穿越到了15年之前，摆在他面前的有一堆积木，于是他就兴致勃勃的开始玩起了积木。

积木只有两种样式，分别是长6厘米宽3厘米的和长6厘米宽6厘米的。

无聊的Gzh想把积木一直拼下去，假设最后的长度为n，那么Gzh会把积木拼成一个形状为6*n的长条。

那么请问，当积木的长度为n时，共有多少种拼法呢?

输入

多组输入数据

每组输入一个数n（0<n<751），保证n为3的倍数。

输出

对于每组数据，输出一行，拼法的数量

输入样例

```
6
```

输出样例

```
3
```

知识点

高精 动态规划

AC Code

```
#include<cstdio>
#include<cstring>
#define radix 1000000000
typedef unsigned long long ull;
struct BigInteger {
  ull num[110];
  int size;
  BigInteger() :size(0) {
    memset(num, 0, sizeof(num));
  }
  BigInteger(const BigInteger& b) {
```

```
    for (int i = 1; i <= b.size; ++i) {
      num[i] = b.num[i];
    }
    size = b.size;
  }
  void print() {
    printf("%llu", num[size]);
    for (int i = size - 1; i > 0; --i)printf("%09llu", num[i]);
    //0处不放东西,就算答案是0也没问题
    putchar('\n');
  }
  void mul(const BigInteger& b) {
    BigInteger c;
    for (int i = 1; i <= size; ++i) {
      for (int j = 1; j <= b.size; ++j) {
        c.num[i + j - 1] += (num[i] * b.num[j]);
      }
    }
    c.size = size + b.size + 2;
    for (int i = 1; i <= c.size; ++i)
      c.num[i + 1] += c.num[i] / radix, c.num[i] %= radix;
    while (!c.num[c.size])--c.size;
    *this = c;
  }
  BigInteger operator +(const BigInteger& b) const {
    BigInteger c;
    c.size = size;
    if (c.size < b.size)c.size = b.size;
    for (int i = 1; i <= c.size; ++i) {
      c.num[i] += num[i] + b.num[i];
      c.num[i + 1] = c.num[i] / radix;
      c.num[i] %= radix;
    }
    if (c.num[c.size + 1])++c.size;
    return c;
  }
  bool operator <=(const BigInteger& b) const {
    if (size != b.size)return size < b.size;
    for (int i = size; i; --i) {
      if (num[i] != b.num[i])return num[i] < b.num[i];
    }
    return true;
  }
};
BigInteger dp[255];
BigInteger two;
inline void buildDP() {
  dp[0].num[1] = 1, dp[0].size = 1;
  dp[1].num[1] = 1, dp[1].size = 1;
```

```
  two.num[1] = 2, two.size = 1;
  for (int i = 2; i < 255; ++i) {
    dp[i] = dp[i - 2];
    dp[i].mul(two);
    dp[i] = dp[i] + dp[i - 1];
  }
}
int n;
int main() {
  buildDP();
  while (scanf("%d", &n) != EOF)dp[n / 3].print();
}
```

D Mdd的异世界

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd现在在一个异世界，这个异世界仅仅只有两个维度，可以把这个世界看成一个处在一条直线上的世界。这个世界上有很多长方形的巨石，一天，这个世界下雨了，mdd想知道这些巨石之间能存多少水呢。

输入

多组输入数据

第一行一个数n，0 <= n <= 1000。

接下来n个数，代表这些巨石的高度h，h >= 0，巨石的宽度都为1。

输出

对于每组数据，输出一行，蓄水的最大值。

输入样例

```
3
1 0 1
4
3 1 2 3
```

输出样例

```
1
3
```

Hint

第一组样例代表的是一组成凹字形的巨石阵|_|，中间凹下去的既是可以蓄水的部分。

知识点

数组 贪心

AC Code

```
#include<cstdio>
#define maxn 1010
typedef long long ll;
/*****************************************
注意后面是否有比他高的，然后贪心就可以了
*****************************************/
int a[maxn];
int n;
ll res;
bool flag;
int main() {
  while (scanf("%d", &n) != EOF) {
    for (int i = 0; i < n; ++i)scanf("%d", a + i);
    flag = false;
    res = 0;
    for (int i = 0; i < n - 2; ++i) {
      if (a[i] > a[i + 1]) {
        flag = false;
        for (int j = i + 1; j < n; ++j) {
          if (a[j] >= a[i]) {
            flag = true;
            for (int k = i + 1; k < j; ++k)res += a[i] - a[k];
            i = j - 1;
            break;
          }
        }
        if (!flag) {
          int maxi = 0, maxIndex = 0;
          for (int j = i + 1; j < n; ++j) {
            if (a[j] > maxi)maxi = a[j], maxIndex = j;
          }
          for (int j = i + 1; j < maxIndex; ++j)res += maxi - a[j];
          if (maxIndex == n - 1)break;
          else i = maxIndex - 1;
```

```
        }
      }
    }
    printf("%lld\n", res);
  }
}
```

E ModricWang's Hanoi Tower（栈）

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

汉诺塔来源于印度传说的一个故事，上帝创造世界时作了三根金刚石柱子，在一根柱子上从下往上按大小顺序摞着64片黄金圆盘。上帝命令婆罗门把圆盘从下面开始按大小顺序重新摆放在另一根柱子上。并且规定，在小圆盘上不能放大圆盘，在三根柱子之间一回只能移动一个圆盘。有预言说，这件事完成时宇宙会在一瞬间闪电式毁灭。也有人相信婆罗门至今仍在一刻不停地搬动着圆盘。恩，当然这个传说并不可信。

ModricWang想让这个游戏更有趣一些，规定所有的圆盘只能在相邻的柱子间移动。规定三根柱子为A、B、C，请输出所有的n个盘子从A移动到C的步骤。

输入

一个数字，n (0≤n≤12)(0≤n≤12)

输出

对于每组数据，输出若干行，表示移动步骤，格式见样例

输入样例

```
2
```

输出样例

```
A --> B
B --> C
A --> B
C --> B
B --> A
B --> C
A --> B
B --> C
```

Hint

务必用栈模拟，否则不给分，严查代码，禁止抄袭。

知识点

栈

AC Code

```
#pragma G++ optimize(2)
#include<iostream>
#include<stack>
#include<cstdlib>
#include<cstring>
using namespace std;
struct Hanoi {
  stack<int> tower;
  char name;
} a,b,c;
inline void move(Hanoi& a, Hanoi& b) {
  if (!a.tower.empty()) {
    int k = a.tower.top();
    a.tower.pop();
    b.tower.push(k);
    printf("%c --> %c\n", a.name, b.name);
  }
}
inline void simuHanoi(int n, Hanoi& a, Hanoi& b, Hanoi& c) {
  if(n){
    simuHanoi(n - 1, a, b, c);
    move(a, b);
    simuHanoi(n - 1, c, b, a);
                move(b, c);
    simuHanoi(n - 1, a, b, c);
         }
}
int main() {
  a.name = 'A', b.name = 'B', c.name = 'C';
```

```
  int n;
  while (~scanf("%d", &n)) {
    while (!c.tower.empty())c.tower.pop();
    for (int i = n; i ; --i) {
      a.tower.push(i);
    }
    simuHanoi(n, a, b, c);
    //printf("Finished!\n");
  }
}
```

F DH的数字序列

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

某一天，DH得到了一个数字序列，他想要找到这个数字序列的一个最长连续子序列，满足其中最大数和最小数差不超过1。他想要知道这个最长连续子序列的长度，请你帮帮他。

输入

多组输入数据。

对于每组数据，第一行为一个整数n(2≤n≤100000)n(2≤n≤100000)。

第二行为nn个整数a1,a2,...,an(1≤ai≤100000)a1,a2,...,an(1≤ai≤100000)。

输出

对于每组数据，输出一行，为一个整数，表示满足要求的最长连续子序列的长度。

输入样例1

```
5
1 2 3 3 2
```

输出样例1

```
4
```

输入样例2

```
11
5 4 5 5 6 7 8 8 8 7 6
```

输出样例2

```
5
```

知识点

数组

AC Code

```
#include<cstdio>
#define maxn 100010
#define Max(a,b) (((a)>(b))?(a):(b))
#define Min(a,b) (((a)<(b))?(a):(b))
int a[maxn];
int n;
int tmpLen, len;
int tmpMax, tmpMin;
int main() {
  while (scanf("%d", &n) != EOF) {
    int j1 = 0, j2 = 1;
    tmpLen = 1, len = 0;
    for (int i = 0; i < n; ++i)scanf("%d", a + i);
    for (j1 = 0; j2 < n; ++j1) {
      j2 = j1 + 1;
      tmpLen = 1;
      while (a[j1] == a[j2] && j2 < n) {
        j2++; tmpLen++;
      }
      j1 = j2 - 1;
      if (a[j1] - 1 == a[j2] || a[j1] + 1 == a[j2]) {
        tmpMax = Max(a[j1], a[j2]);
        tmpMin = Min(a[j1], a[j2]);
        while ((a[j2] == tmpMax || a[j2] == tmpMin) && j2 < n) {
          j2++, tmpLen++;
        }
      }
      if (tmpLen > len) len = tmpLen;
    }
    printf("%d\n", len);
  }
}
```

G Gzh之表达式求值

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

如题，给你个数学表达式，求值。

输入

多组输入数据，每组数据为一行字符串（长度小于100）。输入只包含整数（32位int范围内），'+'，'-'，'*'，'/'，'('，')'，'%'，'**'。（两个乘号为乘方，具有最高优先级）

输入数据保证运算合法，保证不会出现除乘方外两个运算符连续出现的情况（比如"++1"），不会出现小数。

输出

对于每组数据，输出一行，表达式的值。

输入样例

```
1+1
```

输出样例

```
2
```

知识点

栈 运算符的优先级

AC Code

```
#include<iostream>
#include<algorithm>
#include<cctype>
#include<cstdlib>
#include<stack>
#include<string>
#include<vector>
#include<cmath>
#define N_OPTR 10
```

```
#define RADIX 10
#define debug 0
using namespace std;
typedef long long ll;
//逆波兰表达式还是留着洛谷那个题练吧
string expr;
stack<double>opnd;//运算数
stack<char>optr;//运算符
enum Operator { ADD, SUB, MUL, DIV, MOD, POW, FAC, L_P, R_P, EOE };
//                +    -    *    /    %    ^    !    (    )   nul
//本题分为以下几个:左 加减 乘除模 次方 阶乘 右括号 空
/*************************************************************
核心的主算法:
先把结尾的标识符填进去做铺垫，然后正式读入表达式
判断读入数字或者运算符
数字直接入栈
运算符则需要根据与栈顶的优先级进行分别处理

先说优先级处理:
如果栈顶的更低的话，先把当前运算符进栈，延迟计算
如果栈顶的更高的话，则运算符出栈
根据运算符是一元还是二元决定弹出几个，计算结果再入栈
相等只有栈顶左括号，当前右括号的时候，或两个都空
这意味着这个括号内的全部计算全部完成，直接出栈

再说优先级表
普通的这几级倒是好理解，只要记住以下几点:
如果同级或者相同的话，栈顶的比当前的大，否则就是小

栈顶的左括号比别的都要小（右括号则是等于）
栈顶和当前都是左括号则默认栈顶的小（这也没啥好说的）
但是阶乘符号后面不会直接接左括号，所以这个时候判无效

栈顶除左括号之外都比当前的右括号大
但是不会有存在栈顶的右括号，所以这里全都是无效的

空运算符永远是最小的，无论是在栈顶还是当前 但是两个空运算符相等
但是栈顶左括号不可能迎来空，所以无效
*************************************************************/
const char pri[N_OPTR][N_OPTR] = {//运算符优先等级[栈顶][当前]
  /*       +    -    *    /    %    ^    !    (    )   nul   */
  /* + */ '>', '>', '<', '<', '<', '<', '<', '<', '>', '>',
  /* - */ '>', '>', '<', '<', '<', '<', '<', '<', '>', '>',
  /* * */ '>', '>', '>', '>', '>', '<', '<', '<', '>', '>',
  /* / */ '>', '>', '>', '>', '>', '<', '<', '<', '>', '>',
  /* % */ '>', '>', '>', '>', '>', '<', '<', '<', '>', '>',
  /* ^ */ '>', '>', '>', '>', '>', '>', '<', '<', '>', '>',
  /* ! */ '>', '>', '>', '>', '>', '>', '>', ' ', '>', '>',
  /* ( */ '<', '<', '<', '<', '<', '<', '<', '<', '=', ' ',
```

```
  /* ) */ ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ',
  /*nul*/ '<', '<', '<', '<', '<', '<', '<', '<', ' ', '='
};
inline Operator optr2rank(char op) {
  switch (op) {
    case '+' : return ADD;
    case '-' : return SUB;
    case '*' : return MUL;
    case '/' : return DIV;
    case '%' : return MOD;
    case '^' : return POW;
    case '!' : return FAC;
    case '(' : return L_P;
    case ')' : return R_P;
    default  : return EOE;
  }
}
inline char orderBetween(char op1, char op2) {
  return pri[optr2rank(op1)][optr2rank(op2)];
}
inline ll fac(int n) {
  ll ret = 1;
  while (n > 1)ret *= n--;
  return ret;
}
inline double cal(double a, char op, double b) {
  switch (op) {
  case '+': return a + b;
  case '-': return a - b;
  case '*': return a * b;
  case '/': return a / b;
  case '%': return (double)((ll)a % (ll)b);
  case '^': return pow(a, b);
  default : return 0;
  }
}
inline double cal(double a, char op) {
  switch (op) {
    case '!': return (double)fac((int)a);
    default : return 0;
  }
}
inline void init(string& expr) {
  while (!opnd.empty())opnd.pop();
  while (!optr.empty())optr.pop();
  int findstr;
  while ((findstr = expr.find("**")) != expr.npos) {
    expr.replace(findstr, 2, "^");
  }
```

```
}
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> expr) {
    init(expr);
    optr.push('#');//开始标志
    opnd.push(0);//防止初始负数的情况
    int len = expr.length(), i = 0;
    expr += "#";//加入结束标志
    while (!opnd.empty() && !optr.empty() && i <= len) {
      if (isdigit(expr[i])) {
        double tmp = 0;
        while (isdigit(expr[i])) {
          tmp = tmp * RADIX + expr[i] - '0';
          ++i;
        }
        if (expr[i] == '.') {
          double fraction = 1;
          ++i;
          while (isdigit(expr[i])) {
            fraction /= 10;
            tmp += (expr[i] - '0') * fraction;
            ++i;
          }
        }
        opnd.push(tmp);
      }
      else {
        if (i >= 1 && expr[i] == '-' && expr[i - 1] == '(')opnd.push(0);
        char op;
        switch (orderBetween(optr.top(), expr[i])) {
        case '<':
          optr.push(expr[i]);
          ++i;
          break;
        case '=':
          optr.pop();
          ++i;
          break;
        case '>':
          op = optr.top(); optr.pop();
          if ('!' == op) {
            double a = opnd.top(); opnd.pop();
            opnd.push(cal(a, op));
          }
          else {
            double b = opnd.top(); opnd.pop();
            double a = opnd.top(); opnd.pop();
```

```
            opnd.push(cal(a, op, b));
          }
          break;
        default: exit(-1);
        }
      }
    }
    printf("%lld\n", (ll)opnd.top());
  }
}
```

第5次上机

A ModricWang的瑞士轮

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

前排提示

这题虽然题面很长，但是没有什么坑点，总体难度比较简单

题目描述

在双人对决的竞技性比赛，如乒乓球、羽毛球、国际象棋中，最常见的赛制是淘汰赛和 循环赛。前者的特点是比赛场数少，每场都紧张刺激，但偶然性较高。后者的特点是较为公 平，偶然性较低，但比赛过程往往十分冗长。

本题中介绍的瑞士轮赛制，因最早使用于 1895 年在瑞士举办的国际象棋比赛而得名。 它可以看作是淘汰赛与循环赛的折衷，既保证了比赛的稳定性，又能使赛程不至于过长。

2*N 名编号为1~2N 的选手共进行R 轮比赛。每轮比赛开始前，以及所有比赛结束后， 都会按照总分从高到低对选手进行一次排名。选手的总分为第一轮开始前的初始分数加上已 参加过的所有比赛的得分和。总分相同的，约定编号较小的选手排名靠前。

每轮比赛的对阵安排与该轮比赛开始前的排名有关：第 1 名和第2 名、第3 名和第4 名、……、第2K – 1 名和第2K 名、…… 、第 2N – 1 名和第2N 名，各进行一场比赛。每 场比赛胜者得1 分，负者得0 分。也就是说除了首轮以外，其它轮比赛的安排均不能事先确 定，而是要取决于选手在之前比赛中的表现。

现给定每个选手的初始分数及其实力值，试计算在 R 轮比赛过后，排名第Q 的选手编号是多少。我们假设选手的实力值两两不同，且每场比赛中实力值较高的总能获胜。

输入

输入的第一行是三个正整数 N、R、Q，每两个数之间用一个空格隔开，表示有2*N 名 选手、R 轮比赛，以及我们关心的名次Q。

第二行是 2*N 个非负整数s1, s2, ..., s2N，每两个数之间用一个空格隔开，其中si 表示编 号为i 的选手的初始分数。

第三行是 2*N 个正整数w1, w2, ..., w2N，每两个数之间用一个空格隔开，其中wi 表示编 号为i 的选手的实力值。

1 ≤ N ≤ 10000

1 ≤ R ≤ 50

输出

输出只有一行，包含一个整数，即 R 轮比赛结束后，排名第Q 的选手的编号。

输入样例

```
2 4 2
7 6 6 7
10 5 20 15
```

输出样例

```
1
```

HINT

~~这里应该有个图的，可是他又裂了~~

知识点

结构体排序

AC Code

```
#include<cstdio>
#include<algorithm>
using namespace std;
#define maxn 25000
struct stu
{
    int data;
    int num;
    int shili;
};
stu a[maxn];
```

```
bool cmp(stu x,stu y)
{
    if(x.data==y.data)
        return x.num<y.num;
    return x.data>y.data;
}
int n,r,q;
int main(){
    scanf("%d%d%d",&n,&r,&q);
    for(int i=0;i<2*n;i++)scanf("%d",&a[i].data),a[i].num=i;
    for(int i=0;i<2*n;i++) scanf("%d",&a[i].shili);
    for(int i=0;i<r;i++)
    {
        sort(a,a+2*n,cmp);
        for(int i=0;i<2*n;i+=2)
        {
            if(a[i].shili<a[i+1].shili)
            {
                a[i+1].data++;
            }
            else if(a[i].shili>a[i+1].shili)
            {
                a[i].data++;
            }
        }
    }
    sort(a,a+2*n,cmp);
    printf("%d\n",a[q-1].num+1);
  return 0;
}
```

B ModricWang's Marshaling

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

前排提示

ModricWang的题一般没有单文件内多组数据，如果有多组数据一定会说明。

此题不限制做法。

题目描述

货运火车要在编组站根据挂靠车厢到达目的地重新分组，如果一列火车有4节车厢，经过编组后，车厢的编组顺序为3，2，4，1，你知道编组站是怎么编组的吗？ModricWang到编组站参观后发现编组站的铁路有很多岔道，火车在岔道上来来回回地开动，最后列车编组就完成了。ModricWang想到学习过的栈操作，发现火车编组的过程就是若干个进栈出栈操作构成的，于是ModricWang编了一个程序，只要知道最后的编组要求，就能将编组方案输出。

输入

两行

第一行一个正整数n（n<=100）

第二行n个小于等于n的正整数。表示有n节车厢，编号为1，2，3.....n，编组时按照编号进栈，第二行数据表示列车经过编组后的车厢编号顺序。

输出

一个由大写字母A和B构成的字符串，A表示进栈，B表示出栈。表示编组时进栈出栈的操作序列。

输入样例

```
4
3 2 4 1
```

输出样例

```
AAABBABB
```

样例说明

AAABBABB，先入栈3个元素，此时栈中元素为 1 2 3，然后出栈两个元素，出栈顺序为3 2，再入栈一个元素并出栈两个元素，栈空，出栈顺序为4 1，因此总的出栈顺序为3 2 4 1。你要做的就是通过3 2 4 1求出原有的操作。

知识点

栈混洗

AC Code

```
#include<cstdio>
#include<stack>
using namespace std;
bool flag;
stack<int>s;
int n;
int input;
```

```
int curMax;
int main() {
  scanf("%d", &n);
  flag = true;
  for (int i = 1; i <= n; ++i) {
    scanf("%d", &input);
    if (!flag)continue;
    if (input > curMax) {
      for (int j = curMax + 1; j <= input; ++j) {
        s.push(j); putchar('A');
      }
      curMax = input;
      s.pop(), putchar('B');
    }
    else {
      if (input == s.top()) s.pop(), putchar('B');
      else flag = false;
    }
  }
}
```

c Mdd的回文串

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

在字符串匹配中，可以用通配符代表一些不确定的字符，比如字符 * 可以匹配零到多个字符，字符？可以匹配任意一个字符。现在给你一串包含通配符的字符串，请你来判断是否为回文串。

输入

多组输入数据

每组数据共一行字符串，长度不超过100，**只包括[a-z A-Z ?]中的字符**。

输出

对于每组数据，输出一行，如果是回文串，输出“Yes”，否则输出“No"。

输入样例

```
ab?
ab
abc?b
```

输出样例

```
Yes
No
No
```

知识点

水

AC Code

```
#include<iostream>
#include<cstring>
using namespace std;
#define maxn 110
char a[maxn];
char b[maxn];
int main(){
  ios::sync_with_stdio(false), cin.tie(0),cout.tie(0);
    while(cin>>a)
    {
        int flag=0;
        int length=strlen(a);
        int i=0,j=0;
        for(i=0;i<length;i++)
        {
            b[length-i-1]=a[i];
        }
        for(i=0;i<length;i++)
        {
            if(a[i]==b[i])
            {
                flag=1;
            }
            else
            {
                if(a[i]=='?')
                {
                    flag=1;
                }
                else if(b[i]=='?')
                {
```

```
                    flag=1;
                }
                else
                {
                    flag=0;
                    break;
                }
            }
        }
        if(flag==0)
            cout<<"No"<<endl;
        else
            cout<<"Yes"<<endl;
    }
  return 0;
}
```

D 括号匹配

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

给你一些括号，问它们是否可以完全匹配。

输入

多组输入数据

每组共一行字符串，长度小于100，只包括"()[]{}<>"中的字符。

输出

对于每组数据，输出一行，如果合法，输出”Yes"，不合法则输出“No"。

输入样例

```
()
[]]
<{}>
```

输出样例

```
Yes
No
Yes
```

知识点

栈

AC Code

```cpp
#include<iostream>
#include<stack>
#include<string>
using namespace std;

stack<char> m;
int t;
string a;
bool flag;
inline void check(char x, char y) {
  while (!m.empty())m.pop();
  for (int i = 0; i < a.size(); ++i) {
    if (a[i] == y) {
      if (m.empty() || m.top() != x) { flag = false; break; }
      else {
        m.pop();
      }
    }
    else if (a[i] == x)m.push(x);
  }
  if (!m.empty())flag = false;
}
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> a) {
    flag = true;
    check('(', ')');
    check('[', ']');
    check('{', '}');check('<', '>');
    cout << (flag ? "Yes" : "No") << endl;
  }
}
```


E Mdd学数学

时间限制：1000ms 内存限制：65536kb

通过率：/ `(%)` 正确率：/ `(%)`

题目描述

小学老师曾经教过mdd，写在表达式里常用的括号共有3种：大括号"{ }"，中括号"[ ]"，小括号"( )"，而且小括号里面不能够有其他括号，中括号里面只能有小括号，只有大括号可以包含大括号。不过mdd总是忘记这些规则，你能够帮助他判断这些表达式是否正确吗?

输入

多组输入数据

每组输入共一行，一串只含有括号的字符串，长度不超过100。

输出

对于每组数据，输出一行，如果表达式正确，输出"Yes"，否则输出"No"。

输入样例

```
[]
{{}}
{[]}
{()}
(())
```

输出样例

```
Yes
Yes
Yes
No
No
```

知识点

栈

AC Code

```
#include<iostream>
#include<stack>
#include<string>
using namespace std;
stack<char> m;
```

```cpp
int t;
string a;
bool flag;
inline void check(char x1, char y1, char x2, char y2, char x3, char y3) {
  while (!m.empty())m.pop();
  for (int i = 0; i < a.size(); ++i) {
    if (a[i] == x1) {
      if (!m.empty() && m.top() != x1) { flag = false; break; }
      else m.push(a[i]);
    }
    else if (a[i] == y1) {
      if (m.empty() || m.top() != x1) { flag = false; break; }
      else m.pop();
    }
    if (a[i] == x2) {
      if (!m.empty() && m.top() != x1) { flag = false; break; }
      else m.push(a[i]);
    }
    else if (a[i] == y2) {
      if (m.empty() || m.top() != x2) { flag = false; break; }
      else m.pop();
    }
    if (a[i] == x3) {
      if (!m.empty() && m.top() != x2) { flag = false; break; }
      else m.push(a[i]);
    }
    else if (a[i] == y3) {
      if (m.empty() || m.top() != x3) { flag = false; break; }
      else m.pop();
    }
  }
  if (!m.empty())flag = false;
}
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> a) {
    flag = true;
    check('{', '}', '[', ']', '(', ')');
    cout << (flag ? "Yes" : "No") << endl;
  }
}
```


F DH的新手机II

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH觉得之前那个DS手机不好用，又换了一部新手机，这个新手机什么都好，就是没办法知道有多少条未读信息，现在他请你帮忙写个程序告诉他手机上有多少条未读信息。

这部手机上安装了nn个应用程序，刚开始没有信息，下面会发生qq个事件，事件属于以下三种类型之一：

1、 xx号应用程序出现了一条新信息（未读的）。

2、 DH阅读了xx号应用程序至今的所有信息（他可能重复读了某些信息）。

3、 DH阅读了最开始的tt个信息（由第1种事件产生的最开始的tt个信息），保证在此之前由第1种事件产生的信息至少有tt个。需要注意的是他不是阅读前tt个未读信息，他读的是最开始的tt个信息，所以他可能会重复阅读某些信息。

请你告诉DH每一个事件发生后手机上未读信息的数量。

输入

多组输入数据。

对于每组数据，第一行有两个整数n,q(1≤n,q≤300000)n,q(1≤n,q≤300000)，表示应用程序数量和事件数量，接下来qq行，每一行两个整数，第一个整数是typeitypei，如果typei=1typei=1或typei=2typei=2，第二个数为xixi，否则第二个数为titi(1≤typei≤3,1≤xi≤n,1≤ti≤q)(1≤typei≤3,1≤xi≤n,1≤ti≤q)。

输出

对于每组数据，输出qq行，每一行一个整数，第ii行表示第ii个事件发生完了以后未读信息的数量。

输入样例1

```
3 4
1 3
1 1
1 2
2 3
```

输出样例1

```
1
2
3
2
```

输入样例2

```
4 6
1 2
1 4
1 2
3 3
1 3
1 3
```

输出样例2

```
1
2
3
0
1
2
```

知识点

建模优化

AC Code

```cpp
#include<cstdio>
#include<cstring>
#define maxn 300010
enum Status {noNotice, unRead, Read};
struct App {
  Status status;
  int app_id;
}app[maxn];
inline void write(int x) {
  if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
```

```
  return k * f;
}
int n, q;
int res, time;
int readTime;
int type, op;
int main() {
  while (scanf("%d%d", &n, &q) != EOF) {
    res = 0, time = 0;
    memset(app, 0, sizeof(app));
    readTime = 0;
    for (int i = 0; i < q; ++i) {
      type = read(), op = read();
      if (type == 1) {
        app[time].app_id = op;
        app[time++].status = unRead;
        res++;
      }
      else if (type == 2) {
        for (int j = readTime; j < time; ++j) {
          if (app[j].app_id == op) {
            if (app[j].status == unRead)app[j].status = Read, res--;
          }
        }
      }
      else {
        for (int j = readTime; j < op; ++j) {
          if (app[j].status == unRead)app[j].status = Read, res--;
        }
        if (op > readTime)readTime = op;
      }
      write(res), putchar('\n');
    }
  }
}
```


G ModricWang的星灵棋

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

ModricWang上次出了两个题，可是被校园网坑了，所以这一次准备出个一题更比四题强的题来维系宇宙平衡。

星灵棋是一种流行于Aiur的棋类游戏。ModricWang经常和他的表哥Artanis下星灵棋。

星灵棋规则如下：在一个4*4的棋盘上摆放了14颗棋子，其中有7颗白色棋子，7颗黑色棋子，有两个空白地带，任何一颗黑白棋子都可以向上下左右四个方向移动到相邻的空格，这叫行棋一步，黑白双方交替走棋，任意一方可以先走，如果某个时刻使得任意一种颜色的棋子形成四个一线（包括斜线），这样的状态为目标棋局，先达到目标棋局者赢。

由于ModricWang还有很多份解题报告没有批，所以他根本不关心输赢，只想赶紧把手里这一盘下完走人。给定当前的棋局状态，请你帮ModricWang算出最少还要多少步才能下完这盘棋。

输入

只有一组评测数据。

一个4*4的初始棋局，黑棋子用B表示，白棋子用W表示，空格地带用O表示。

保证输入格式严格符合规范，总共只有4个连续的可见行，每行只有4个连续的可见字符。不存在前导空行或前导空格。

保证输入数据完全符合规则，总共有7个B，7个W和2个O。

输出

用最少的步数移动到目标棋局的步数。

输入样例

```
BWBO
WBWB
BWBW
WBWO
```

输出样例

```
5
```

HINT

可能ModricWang已经赢了

知识点

八数码难题变种 状态压缩+bfs

AC Code

```cpp
#include<cstdio>
#include<cstring>
#include<queue>
#include<unordered_map>
#include<algorithm>
#define x first
#define y second
#define mp make_pair
#define debug 0
using namespace std;
typedef unsigned long long ull;
typedef pair<int, int> PII;
typedef pair<PII, PII> OPlace;
/*****************************
类似八数码难题，直接bfs就完事了
*****************************/
char map[4][4];
unordered_map<ull, int> a;
queue<ull>q_black_first;
queue<ull>q_white_first;
int ans;
inline int transNum(char a) {
  return (a == 'O') ? 0 : (a == 'B') ? 1 : 2;
}
inline char transChar(int a) {
  return (a == 0) ? 'O' : (a == 1) ? 'B' : 'W';
}
inline char next_black_first(int a) {
  return (a & 1) ? 'B' : 'W';
}
inline char next_white_first(int a) {
  return (a & 1) ? 'W' : 'B';
}
inline bool illegal(int _x, int _y) {
  return _x < 0 || _x > 3 || _y < 0 || _y > 3;
}
inline bool check() {
  bool a1 = map[0][0] == map[0][1] && map[0][0] == map[0][2] && map[0][0] ==
map[0][3];
  bool a2 = map[1][0] == map[1][1] && map[1][0] == map[1][2] && map[1][0] ==
map[1][3];
  bool a3 = map[2][0] == map[2][1] && map[2][0] == map[2][2] && map[2][0] ==
map[2][3];
  bool a4 = map[3][0] == map[3][1] && map[3][0] == map[3][2] && map[3][0] ==
map[3][3];
  bool b1 = map[0][0] == map[1][0] && map[0][0] == map[2][0] && map[0][0] ==
map[3][0];
```

```
  bool b2 = map[0][1] == map[1][1] && map[0][1] == map[2][1] && map[0][1] ==
map[3][1];
  bool b3 = map[0][2] == map[1][2] && map[0][2] == map[2][2] && map[0][2] ==
map[3][2];
  bool b4 = map[0][3] == map[1][3] && map[0][3] == map[2][3] && map[0][3] ==
map[3][3];
  bool c1 = map[0][0] == map[1][1] && map[0][0] == map[2][2] && map[0][0] ==
map[3][3];
  bool c2 = map[3][0] == map[2][1] && map[3][0] == map[1][2] && map[3][0] ==
map[0][3];
  return a1 || a2 || a3 || a4 || b1 || b2 || b3 || b4 || c1 || c2;
}
int nx[4] = { -1,0,0,1 };
int ny[4] = { 0,-1,1,0 };
inline ull pack_map() {
  ull ret = 0;
  for (int i = 0; i < 4; ++i) {
    for (int j = 0; j < 4; ++j) {
      ret = (ret << 1) + (ret << 3) + transNum(map[i][j]);
    }
  }
  return ret;
}
inline void print_debug(int i) {
  printf("Now is round %d\nmap is \n\n", i);
  for (int i = 0; i < 4; ++i) {
    for (int j = 0; j < 4; ++j)
      putchar(map[i][j]);
    putchar('\n');
  }
  putchar('\n');
}
inline OPlace unpack_map(ull pack) {
  OPlace ret;
  bool flag = false;
  for (int i = 3; i >= 0; --i) {
    for (int j = 3; j >= 0; --j) {
      map[i][j] = transChar(pack % 10);
      pack /= 10;
      if (map[i][j] == 'O') {
        if (!flag)ret.x.x = i, ret.x.y = j, flag = true;
        else ret.y.x = i, ret.y.y = j;
      }
    }
  }
  return ret;
}
int main() {
  //freopen("output BUAAOJ516.txt", "w", stdout);
```

```
for (int i = 0; i < 4; ++i)scanf("%[^\n]", map[i]), getchar();
ull first = pack_map();
q_black_first.push(first);
q_white_first.push(first);
a[first] = 0;
while (!q_black_first.empty()) {
  ull u = q_black_first.front();
  q_black_first.pop();
  memset(map, 0, sizeof(map));
  OPlace place = unpack_map(u);
  PII O1 = place.x, O2 = place.y;
  int round = a[u];
  if (debug)print_debug(round);
  if (check()) { ans = round; break; }
  char side = next_black_first(round);
  int x_2 = O1.x, y_2 = O1.y;
  for (int i = 0; i < 4; ++i) {
    int x3 = x_2 + nx[i], y3 = y_2 + ny[i];
    if (illegal(x3, y3))continue;
    if (map[x3][y3] != side)continue;
    swap(map[x_2][y_2], map[x3][y3]);
    ull ns = pack_map();
    if (!a.count(ns)) {
      a[ns] = round + 1;
      q_black_first.push(ns);
    }
    swap(map[x_2][y_2], map[x3][y3]);
  }
  x_2 = O2.x, y_2 = O2.y;
  for (int i = 0; i < 4; ++i) {
    int x3 = x_2 + nx[i], y3 = y_2 + ny[i];
    if (illegal(x3, y3))continue;
    if (map[x3][y3] != side)continue;
    swap(map[x_2][y_2], map[x3][y3]);
    ull ns = pack_map();
    if (!a.count(ns)) {
      a[ns] = round + 1;
      q_black_first.push(ns);
    }
    swap(map[x_2][y_2], map[x3][y3]);
  }
}
a.clear();
a[first] = 0;
while (!q_white_first.empty()) {
  ull u = q_white_first.front();
  q_white_first.pop();
  memset(map, 0, sizeof(map));
  OPlace place = unpack_map(u);
```

```
    PII O1 = place.x, O2 = place.y;
    int round = a[u];
    if (debug)print_debug(round);
    if (check()) { ans = (ans < round) ? ans : round; break; }
    char side = next_white_first(round);
    int x_2 = O1.x, y_2 = O1.y;
    for (int i = 0; i < 4; ++i) {
      int x3 = x_2 + nx[i], y3 = y_2 + ny[i];
      if (illegal(x3, y3))continue;
      if (map[x3][y3] != side)continue;
      swap(map[x_2][y_2], map[x3][y3]);
      ull ns = pack_map();
      if (!a.count(ns)) {
        a[ns] = round + 1;
        q_white_first.push(ns);
      }
      swap(map[x_2][y_2], map[x3][y3]);
    }
    x_2 = O2.x, y_2 = O2.y;
    for (int i = 0; i < 4; ++i) {
      int x3 = x_2 + nx[i], y3 = y_2 + ny[i];
      if (illegal(x3, y3))continue;
      if (map[x3][y3] != side)continue;
      swap(map[x_2][y_2], map[x3][y3]);
      ull ns = pack_map();
      if (!a.count(ns)) {
        a[ns] = round + 1;
        q_white_first.push(ns);
      }
      swap(map[x_2][y_2], map[x3][y3]);
    }
  }
  printf("%d\n", ans);
}
```

第6次上机

A Gzh渣基三

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh最近沉迷基三（剑侠情缘三），在游戏里他是一个手法犀利意识风骚英俊潇洒风流倜傥的男子（和现实中一样）

~~图怎么又没了~~

Gzh也很苦恼，他现在做一个阵营日常都有无数奶妈小姐姐想要组队焦点奶他，然而一个团队只能有25个人，如果满了就不能再进入团队了，所以奶妈小姐姐们自发排队入队。（Gzh不占队伍名额）

队伍有三种操作：

1.add x 让ID为x的奶妈入队，其中x为一个字符串，长度不超过20

2.delete q 让队首的前q个奶妈离开队伍

3.get m 得到位置为m的奶妈小姐姐的ID并输出

输入

多组输入数据，第一个数为操作个数个数n（0<n<101），保证q和m在int范围内。

接下来n行，每行一个操作。

输出

按操作要求输出，如果操作不合法，输出GzhIsSoHandsome并不执行此次操作。

输入样例

```
4
add baiexian
get 1
delete 2
get 1
```

输出样例

```
baiexian
GzhIsSoHandsome
baiexian
```

Hint

请使用队列完成，另外数据要多加考虑

知识点

队列

AC Code

```cpp
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<iostream>
using namespace std;
#define maxn 110
typedef struct node
{
    string data;
    struct node* next;
}queuenode, * queueptr;
typedef struct
{
    queueptr front;
    queueptr rear;
}LinkQueue;

void initQueue(LinkQueue& q)
{
    q.front = q.rear = new node;
    q.front->next = NULL;
}
bool isEmpty(LinkQueue& q)
{
    if (q.front == q.rear)
        return 1;
    else
        return 0;
}
int queueSize(LinkQueue& q)
{
    int counter = 0;
    queueptr p = q.front->next;
    while (p != NULL)
    {
        p = p->next;
        counter++;
    }
    return counter;
}
int EnQueue(LinkQueue& q, string e)
{
    queueptr p = new queuenode;
    p->data = e;
    if (isEmpty(q))
    {
        q.front->next = p;
        q.rear = p;
        p->next = NULL;
```

```
    }
    else
    {
        q.rear->next = p;
        q.rear = p;
        p->next = NULL;
    }
    return 1;
}
int DeQueue(LinkQueue& q)
{
    if (isEmpty(q))
        return 0;
    else
    {
        queueptr p;
        p = q.front->next;
        q.front->next = p->next;
        if (q.rear == p)
        {
            q.rear = q.front;
        }
        delete p;
    }
    return 1;
}
int getId(LinkQueue& q, int m)
{
    int i = 1;
    queueptr p = q.front->next;
    while (i < m && p != NULL)
    {
        p = p->next;
        i++;
    }
    printf("%s\n", p->data.c_str());
    return 1;
}
void destroyQueue(LinkQueue& q)
{
    while (q.front)
    {
        q.rear = q.front->next;
        delete q.front;
        q.front = q.rear;
    }
}
int main()
{
```

```
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    int n;
    while (cin >> n)
    {
        LinkQueue q;
        initQueue(q);
        string opera;
        for (int i = 0; i < n; i++)
        {
            cin >> opera;
            if (opera == "add")
            {
                string str;
                cin >> str;
                if (queueSize(q) >= 25)
                    puts("GzhIsSoHandsome");
                else
                {
                    EnQueue(q, str);
                }
            }
            else if (opera == "get")
            {
                int m;
                cin >> m;
                int size1 = queueSize(q);
                if (m <= 0 || m > size1)
                {
                    puts("GzhIsSoHandsome");
                }
                else
                {
                    getId(q, m);
                }
            }
            else if (opera == "delete")
            {
                int m;
                cin >> m;
                if (m<0 || m>queueSize(q))
                {
                    puts("GzhIsSoHandsome");
                }
                else
                {
                    for (int i = 0; i < m; i++)
                    {
                        DeQueue(q);
                    }
```

```
                }
            }
        }
        destroyQueue(q);
    }
}
```

B ModricWang的序列修改

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

前排提示

此题没有多组数据。

此题推荐使用自己实现的数据结构，也允许使用STL，但是不能直接输入输出水过（例如输入到数组里，输出的时候跳过那个数）。

题目描述

ModricWang是一个仁慈的人，他希望大家数据结构都能拿高分。

ModricWang创造了一个序列，经过推算，发现其中第K个位置上的数破坏了整体的和谐，于是他想把这个数删掉。

输入

第一行，两个数，序列长度n，需要被删掉的位置K，保证输入合法

接下来一行，n个数，表示ModricWang创造的序列，都在int范围内

输出

输出一行，修改后的序列

输入样例

```
4 2
255 666 511 1023
```

输出样例

```
255 511 1023
```

知识点

水

AC Code

```
#include<cstdio>
int n,k;
int input;
int main(){scanf("%d%d",&n,&k);for(int i = 1; i <= n; ++i)
{scanf("%d",&input);if(i!=k)printf("%d ",input);}}
```

C Mdd的数字

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

在信息论里面，有个概念叫做汉明距离，它表示两个相同长度字对应位不同的数量。现在给你两个数，请求出它们的汉明距离。

输入

多组输入数据

接下来多行，每行两个数a，b，均在int范围内，**可能为负数**。

输出

对于每组数据，输出一行，a，b的汉明距离

输入样例

```
2 4
3 4
2 3
```

输出样例

```
2
3
1
```

Hint

```
2可以表示为(0, 1, 0)，3可以表示为(0, 1, 1)，4可以表示为(1, 0, 0)
```

知识点

异或和

AC Code

```
#include<cstdio>
int a, b;
int main() {
  while (scanf("%d%d", &a, &b) != EOF) {
    int res = a ^ b;
    int ans = 0;
    while (res)res &= (res - 1), ans++;
    printf("%d\n", ans);
  }
}
```

D DH的回文数

时间限制：1000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH喜欢回文数，更喜欢长度为偶数的回文数。现在他想知道从小到大第nn个偶长度回文数是多少（即从1到无穷大这一组数中第n个偶长度回文数），你能帮帮他吗?

输入

多组输入数据，每组数据一行，包含一个整数n(1≤n≤10100000)n(1≤n≤10100000)

输出

每组数据输出一行，包含一个整数，表示从小到大第nn个偶长度回文数。

输入样例

```
1
```

输出样例

```
11
```

知识点

水

AC Code

```
#include<iostream>
#include<string>
using namespace std;
string a;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(nullptr);
  while (cin >> a) {
    cout << a;
    for (int i = a.length()-1; i >= 0; --i)cout << a[i];
    cout << endl;
  }
}
```


E Gzh之返老还童II

时间限制：200ms 内存限制：2500kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh现在觉得回到童年的他什么都想玩，而不仅仅局限于积木。这不，他又拿来了一堆英文字母（a-i）和字符*。Gzh把英文字母摆成了一个形如

aababcabcdabcdeabcdefabcdefgabcdefghabcdefghi

abcdefghia*

abcdefghia*aa

abcdefghia*aaab

abcdefghia*aaabac

.....

abcdefghia*aaabacadaeafagahaib*

.....

的串，现在Gzh想知道从左往右数的第n个字符是什么，可是他太笨了，一个一个数不知道要数到什么时候。所以他能来求助机智的你们了。

输入

第一个数为数据组数n

接下来n行，每行1个整数m(保证m<2147483648)

输出

对于每组数据，输出一行，为第m个位置的字符。

输入样例

```
1
1
```

输出样例

```
a
```

知识点

c风格字符串处理 （如果没有空间限制，用string会舒服很多）

AC Code

```
#include <stdio.h>
char rowNow[2000000];
int main() {
     int n,m,numPerRow;
     scanf("%d",&n);
    for(int i=0;i<n;i++) {
         scanf("%d",&m);
        int digitNow=1;
        rowNow[0]='a';
        numPerRow=1;
        while(m>numPerRow) {
           m-=numPerRow;
            int k;
          for(k=0;k<digitNow;k++) {
                if(rowNow[numPerRow-k-1]!='i') break;
           }
            int flag;
```

```
            if(k==digitNow) {
            digitNow++;
               flag=1;
         }
             numPerRow+=digitNow;
            int c=0;
            if(rowNow[numPerRow-digitNow-1]=='*')  rowNow[numPerRow-1]='a';
          else rowNow[numPerRow-1]=rowNow[numPerRow-digitNow-1]+1;
            int j;
         for(j=0;j<digitNow;j++) {
            if(rowNow[numPerRow-1-j]=='j') {
                  rowNow[numPerRow-1-j]='*';
                c=1;
              }
                else {
                   if(j==0) continue;
                    rowNow[numPerRow-1-j]=c+rowNow[numPerRow-digitNow-1-j];
                    if(rowNow[numPerRow-1-j]=='*'+1) {
                        rowNow[numPerRow-1-j]='a';
                       c=0;
                    }
                 else if(rowNow[numPerRow-1-j]=='j') {
                        rowNow[numPerRow-1-j]='*';
                      c=1;
                    }
                    else c=0;
              }
           }
            if(flag==1) {
                rowNow[numPerRow-digitNow]='a';
          }
            flag=0;
       }
       putchar(rowNow[m-1]),putchar('\n');
       //printf("%c\n",rowNow[m-1]);
    }
}
```


F Gzh之软院联谊

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

软院男女比8:1，耿导因为担心大家的感情问题，甚是苦恼。于是，耿导找来了Gzh，让他和中华女子学院（耿导钦点的，谈起他们和女院联谊的时候还一脸怀念）联谊。

假设软院和中华女子学院各有n个班级，他们之间各有一个满意优先度，如果1和2联谊了，3和4联谊了（1,3是软院班级，2,4是女院班级），但1比起2更满意4,同时4比起3更满意1，这时联谊就会破裂，现在给出每个班级的满意优先度，请你求出一份不会破裂的联谊配对（以更令软院班级满意的结果优先考虑）。

输入

第一个数为数据组数m接下来m组，每组第一行为班级数n。接下来一行为2*n个字符，分别表示软院班级和女院班级，软院班级为一个小写字母，女院班级为一个大写字母。接下来为*2n行，每行表示一个班级的满意优先度（先给出软院，后给出女院）。

输出

对于每组数据，输出n行，为最终的联谊配对，以软院班级名字升序排列，每组数据之间有一个空行。

输入样例

```
1
2
a b A B
a:AB
b:AB
A:ab
B:ab
```

输出样例

```
a A
b B
```

知识点

二分图稳定婚姻匹配问题 Gale-Shapley算法

AC Code

```
#include<cstdio>
#include<cstring>
#include<queue>
#include<algorithm>
#define maxn 40
using namespace std;
```

```
/*********************************************************
原题:poj3487 Gale-Shapley算法
问题模型:二分图稳定婚姻匹配
给定男士女士的婚姻匹配优先级
求出一份不会破裂的联谊配对
(即不会出现其中两对,有一对男女更加中意彼此但是被分离了)
题目优先保证男方的满意度,这样的话求解就是唯一的了
说白了,算法需要采取男生主动追求女生的形式

具体算法流程如下:
第一轮的时候,每个男生都选择自己名单上排在首位的女生,并表白
如果这个女生没被追求过,就接受男生的追求
如果女生已经接受其他男生的追求,则和现任男友比较
如果现男友好就拒绝,否则就抛弃现男友(白 色 相 簿)
第一轮结束之后会出现有的男生有伴侣,有的女生没有的情况
第二轮则是所有单身男从没有拒绝过他的女孩中选择最高的那个
不论她是否单身,直接追求
以此类推,直到所有人都不单身为止
*********************************************************/
char str[maxn], name[maxn << 1];
int t, n, x;
int MaleRank[maxn][maxn];//i女士心中,j男士的排名
int FemaleRank[maxn][maxn];//i男士心中,j女士的排名
int Next[maxn];//i男士下一个求婚对象的在男士心中的排名
int ans[2][maxn];//0表示女士的未婚夫,1表示男士的未婚妻
queue<int>single;//依旧单身的男士
inline void init() {
  memset(str, 0, sizeof(str));
  memset(name, 0, sizeof(name));
  memset(MaleRank, 0, sizeof(MaleRank));
}
inline void Paired(int x, int y) {//x男士y女士
  //激动人心的白色相簿环节(x
  //将原有配对的男士设置单身,重新配对
  int m = ans[0][y];
  if (m)ans[1][m] = 0, single.push(m);
  ans[1][x] = y, ans[0][y] = x;
}
int main() {
  scanf("%d", &t);
  while (t--) {
    //init();
    scanf("%d", &n);
    for (int i = 1; i <= (n << 1); ++i)scanf("%s", str), name[i] = str[0];
    sort(name + 1, name + n + 1);//姓名排序
    for (int i = 1; i <= n; ++i) {
      scanf("%s", str);
      for (int j = 2; j < n + 2; ++j)FemaleRank[str[0] - 'a' + 1][j - 1] =
str[j] - 'A' + 1;
```

```
      Next[str[0] - 'a' + 1] = 1, ans[1][str[0] - 'a' + 1] = 0;
      single.push(str[0] - 'a' + 1);
    }
    for (int i = 1; i <= n; ++i) {
      scanf("%s", str);
      for (int j = 2; j < n + 2; ++j)MaleRank[str[0] - 'A' + 1][str[j] - 'a' +
1] = j - 1;
      ans[0][str[0] - 'A' + 1] = 0;
    }
    while (!single.empty()) {
      int u = single.front();
      single.pop();
      int v = FemaleRank[u][Next[u]++];
      if (!ans[0][v])Paired(u, v);
      else if (MaleRank[v][u] < MaleRank[v][ans[0][v]])Paired(u, v);
      else single.push(u);
    }
    for (int i = 1; i <= n; ++i)
      printf("%c %c\n", name[i], ans[1][name[i] - 'a' + 1] + 'A' - 1);
    putchar('\n');
  }
}
```

G DH的01串

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

某一天DH得到了一个01串（一个字符串只包含'0''1'两种字符），他很好奇其中最长的交替子串（0和1交替出现（10,01均可），不要求一定是原串的连续子串）有多长，不过这似乎太简单了，于是他决定将原始01串的一个非空连续子串进行取反操作（1变为0,0变为1），现在他想知道经过这样的操作后最长的交替子串的最大长度是多少。

输入

多组输入数据。

对于每组输入数据，包含两行。

第一行为一个整数n(1≤n≤100000)n(1≤n≤100000)，表示01串的长度

第二行为长度为nn的一个01串

输出

每组数据输出一行，包含一个整数，表示题目所求结果

输入样例1

```
8
10000011
```

输出样例1

```
5
```

输入样例2

```
2
01
```

输出样例2

```
2
```

知识点

找规律

AC Code

```
#include<cstdio>
#include<cstring>
int n;
char s[100010];
int main() {
  while (scanf("%d", &n) != EOF) {
    scanf("%s", s);
    int flag;
    int cnt;
    flag = s[0];
    cnt = 1;
    for (int i = 1; i < n; ++i) {
      if (s[i] != flag)cnt++, flag = s[i];
    }
    if (cnt + 2 > n)printf("%d\n", n);
    else printf("%d\n", cnt + 2);
  }
}
```

期中上机

A Gzh的a+b

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

签到题，如题，计算a+b。

输入

一组输入数据，2个整数a,b(保证a,b在int范围内)

输出

输出一行，a+b的值

输入样例

```
1 2
```

输出样例

```
3
```

知识点

还用我说?

AC Code

还用我写?

B ModricWang的链表

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang要求你们实现一个带头结点单链表的结构用于存储32位有符号整型数，并实现包括初始化、用尾插法建表、查找、插入、删除、定位、判表空、求表长、释放整个表等操作的实现。

ModricWang会给你一个操作队列，操作指令都在以下范围：

new 建立新的链表，丢弃原来的链表，要求不造成内存泄漏

insert x 向链表尾部插入元素x

find x 找到元素x第一次出现的位置

erase x 找到元素x第一次出现的位置，并将其删除

size 求出链表内的元素个数

list_empty 判断表是否为空

delete 清空这个链表，要求不造成内存泄漏

可能会有连续的delete，正常输出即可，但是要注意细节，很多同学OE/RE就是因为这个

你需要对操作给出一些回应：

new 需要输出一行，内容为"A new list is created"

insert x 需要输出一行，内容为"A new element is inserted into the list"

find x 如果找到了元素，需要输出一行，内容为"Element %d is found at %d"，第一个参数为元素值，第二个参数为元素位置；如果没有找到元素，需要输出一行，内容为"404 Not Found"

erase x 如果元素存在，需要输出一行，"Element %d is erased at %d"，第一个参数为元素值，第二个参数为元素位置；否则输出出一行，内容为"404 Not Found"

size 输出一行，"Size : %d"，参数为链表长度

list_empty 输出一行，如果链表为空，输出"Empty"，否则输出"Not Empty"

delete 输出一行，"It vanishes"

输入

第一行，一个整数N，表示操作数

接下来N行，每行一个指令

保证第一条指令为new

输出

N行，对每个指令，给出回应

输入样例

```
9
new
erase 2
insert 2
find 2
size
new
list_empty
find 4
delete
```

输出样例

```
A new list is created
404 Not Found
A new element is inserted into the list
Element 2 is found at 0
Size : 1
A new list is created
Empty
404 Not Found
It vanishes
```

知识点

链表

AC Code

```
#include<cstdio>
#include<cstring>
#include<malloc.h>
typedef struct node
{
  int data;
  struct node *link;
}LNode, *LinkList;
int allpos;
LinkList CREAT(int n)
{
  LinkList p, r, list = NULL;
  int a = 0;
  int i;
  for(i = 0; i < n; i++)
  {
    p = (LinkList)malloc(sizeof(LNode));
    p->data = a;
```

```
    p->link = NULL;
    if(list == NULL)
      list = p;
    else
      r->link = p;
    r = p;
  }
  return list;
}
int LENGTH(LinkList list)
{
  LinkList p = list;
  int n = 0;
  while(p != NULL)
  {
    n++;
    p = p->link;
  }
  return n;
}
int ISEMPTY(LinkList list)
{
  return list == NULL;
}
int FIND(LinkList list, int a)
{
  LinkList p = list;
  int fpos = 0;
  while(p != NULL && p->data != a)
  {
    p = p->link;
    fpos++;
  }
  return p == NULL ? -1 : fpos;
}
void DELETELIST(LinkList list)
{
  LinkList p = list;
  while(p != NULL)
  {
    list = p->link;
    free(p);
    p = list;
  }
}
LinkList ERASE(LinkList list, int a)
{
  LinkList p = list->link, q = list;
  if(q->data == a)
```

```
  {
    list = list->link;
    free(q);
    return list;
  }
  allpos++;
  int found = 0;
  while(p != NULL)
  {
    if(p->data == a)
    {
      q->link = p->link;
      free(p);
      p = q->link;
      found = 1;
      break;
    }
    else
    {
      q = p;
      p = p->link;
      allpos++;
    }
  }
  if(found == 0)
    allpos = -1;
  return list;
}
void INSERT(LinkList list, int a)
{
  LinkList p = list, r;
  while(p->link != NULL)
  {
    p = p->link;
  }
  r = (LinkList)malloc(sizeof(LNode));
  r->data = a;
  r->link = NULL;
  p->link = r;
}

int main()
{
  int N;
  scanf("%d", &N);
  LinkList head = NULL;
  while(N--)
  {
    char str[20];
```

```
    scanf("%s", str);
    if(strcmp(str, "new") == 0)
    {
      if(head != NULL)
      {
        DELETELIST(head);
        head = NULL;
      }
      printf("A new list is created\n");
    }
    else if(strcmp(str, "erase") == 0)
    {
      int a;
      scanf("%d", &a);
      if(head != NULL)
      {
        head = ERASE(head, a);
        if(allpos >= 0)
          printf("Element %d is erased at %d\n", a, allpos);
        else
          printf("404 Not Found\n");
      }
      else
        printf("404 Not Found\n");
      allpos = 0;
    }
    else if(strcmp(str, "insert") == 0)
    {
      int a;
      scanf("%d", &a);
      if(head == NULL)
      {
        head = CREAT(1);
        head->data = a;
      }
      else
        INSERT(head, a);
      printf("A new element is inserted into the list\n");
    }
    else if(strcmp(str, "find") == 0)
    {
      int a, pos = 0;
      scanf("%d", &a);
      pos = FIND(head, a);
      if(pos != -1)
        printf("Element %d is found at %d\n", a, pos);
      else
        printf("404 Not Found\n");
    }
```

```
    else if(strcmp(str, "size") == 0)
    {
      printf("Size : %d\n", LENGTH(head));
    }
    else if(strcmp(str, "delete") == 0)
    {
      if(head)
        DELETELIST(head);
      head = NULL;
      printf("It vanishes\n");
    }
    else if(strcmp(str, "list_empty") == 0)
    {
      if(ISEMPTY(head))
        printf("Empty\n");
      else
        printf("Not Empty\n");
    }
  }
}
```

C 进制转换

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

题目要求很简单，给你一个非负十进制数，把它转换为为二进制数。

请用栈实现

输入

多组输入数据

每组共一个数n，0 <= n <= INT_MAX。

输出

对于每组数据，输出n的二进制表示。

输入样例

```
5
7
18
```

输出样例

```
101
111
10010
```

知识点

水

AC Code

```
#include<cstdio>
#define RADIX 2
void write(int x) {
  //if(x < 0)putchar('-'),x=-x;
  if (x > RADIX-1)write(x / RADIX);
  putchar(x % RADIX + 48);
}
int n;
int main(){while(scanf("%d",&n)!=EOF){write(n);putchar('\n');}}
```

D Gzh之简单的约瑟夫环

时间限制：1000ms 内存限制：3000kb

通过率：/ (%) 正确率：/ (%)

题目描述

如题所示，简单的约瑟夫环，真的没有任何坑点。约瑟夫环的描述如下：

已知n个人(不妨分别以编号1，2，3，...，n 代表)围坐在一张圆桌周围，从编号为1的开始数，数到k的人从1开始报数，数到m的那个人出列，他的下一个人又从1开始继续报数，数到m的那个人出列...,依此重复下去，直到圆桌周围只剩一个人。

输入

多组输入数据，第一个数为人数n，第二个数为k，第三个数为m（保证n，m，k不超过1000）。

输出

输出最后的那个人的编号。

输入样例

```
2
1 1
```

输出样例

```
2
```

Hint

务必使用循环链表实现，否则不给分！！

知识点

链表 但是现在用链表很容易TLE，所以就只贴数学规律的代码了

AC Code

```
#include<cstdio>
#define maxn 1010
bool alive[maxn];
int n, k, m;
int realm;
int cur;
int l;
inline int cir(int n, int m)
{
  int p = 0;
  for (int i = 2; i <= n; i++)
  {
    p = (p + m) % i;
  }
  return (p + l) % n + 1;
}
int main() {
  while (scanf("%d%d%d", &n, &k, &m) != EOF) {
    k--;
    k %= n;
    l = k;
    printf("%d\n",cir(n, m));

  }
}
```

E 杨辉三角

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

杨辉三角，是二项式系数在三角形中的一种几何排列。在欧洲，这个表叫做帕斯卡三角形。帕斯卡（1623----1662）是在1654年发现这一规律的，比杨辉要迟393年，比贾宪迟600年。早在我国南宋数学家杨辉1261年所著的《详解九章算法》一书里就出现了。这又是我国数学史上的一个伟大成就（摘自百度百科）。请你用**循环队列**实现打印出杨辉三角。

输入

多组输入数据

每组一个数n，0 < n <= 20。

输出

对于每组数据，输出杨辉三角的前n行，同一行的数用空格隔开。

输入样例

```
1
2
3
```

输出样例

```
1
1
1 1
1
1 1
1 2 1
```

知识点

组合数学 动态规划 循环队列就不写了

AC Code

```
#include<cstdio>
```

```
#define maxn 101
typedef unsigned long long ull;
ull dp[maxn][maxn];
int n;
/*************************************
杨辉三角状态转移方程：
dp[i][j] = dp[i-1][j]+dp[i-1][j-1]
*************************************/
inline void buildDP() {
  dp[1][1] = 1;
  for (int i = 2; i < maxn; ++i) {
    for (int j = 1; j <= i; ++j) {
      dp[i][j] = dp[i - 1][j] + dp[i - 1][j - 1];
    }
  }
}
inline void print(int n) {
  for (int i = 1; i <= n; ++i) {
    //for (int j = 0; j < ((n - i) << 2); ++j) putchar(' ');
    for (int j = 1; j <= i; ++j) {
      //printf("%4llu    ", dp[i][j]);
      printf("%llu ", dp[i][j]);
    }
    putchar('\n');
  }
  //putchar('\n');
}
int main() {
  buildDP();
  while (scanf("%d", &n) != EOF)print(n);
}
```

F Mdd的小游戏

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd有一个十分幼稚的数数游戏，给你一个串数字，要你把它每个数字都数出来。

比如，“1”是“一个1”，可以表示成“11”；“11”是“两个1”，可以表示成“21”。"21"是“一个2一个1”，可以表示成“1211”，“1211”是“一个1一个2两个1”，可以表示成“111221”。

起始只有一个数“1”，如果这个游戏一直进行下去，就会形成一个这样的序列：

1, 11, 21, 1211, 111221, 312211...

输入

多组输入数据

每组一个数n，n不超过50。

输出

对于每组数据，输出一行，这个序列的第n串数字。

输入样例

```
3
5
```

输出样例

```
21
111221
```

知识点

趣题 Say What You See

AC Code

```
#include<cstdio>
#include<cstring>
#define maxn 51000
int s1[maxn], s2[maxn];
void solve(int x) {
  int cnt = 0;
  s1[cnt++] = 1;
  s1[cnt] = -1;
  while (x--) {
    int cnt2 = 0, pre;
    for (int i = 0, pre = 0; i < cnt; ++i) {
      if (s1[i] != s1[i + 1]) {
        s2[cnt2++] = i - pre + 1;
        s2[cnt2++] = s1[i];
        pre = i + 1;
      }
    }
    s2[cnt2] = -1;
    for (int i = 0; i <= cnt2; ++i)s1[i] = s2[i];
    cnt = cnt2;
  }
```

```
  for (int i = 0; i < cnt; ++i)printf("%d", s1[i]);
  putchar('\n');
}
int t, n;
int main() {
  while (scanf("%d", &n)!=EOF) {
    solve(n-1);
  }
}
```

G DH的字符串游戏

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH最近好闲啊，于是乎开始玩儿起了字符串。现在他有两个等长的字符串a和b，他想知道这两个字符串是否等价。

字符串aa和bb等价的定义：

1、 若单个字符串长度为奇数，则两字符串只有完全相等才可称为等价

2、 若单个字符串长度为偶数，则将aa从中间分为长度相等的两半，记为a1a1和a2a2，将bb从中间分为长度相等的两半，记为b1b1和b2b2，满足a1a1与b1b1等价且a2a2与b2b2等价，或者a1a1与b2b2等价且a2a2与b1b1等价，称aa与bb等价

输入

多组输入数据。 每组输入数据两行，每行一个只由小写字母组成的字符串且字符串长度在[1,200000]范围内，保证两字符串等长。

输出

对于每组输入数据，输出一行，如两字符串等价输出"YES"，否则输出"NO"（均不含双引号）。

输入样例1

```
aaba
abaa
```

输出样例1

```
YES
```

输入样例2

```
aabb
abab
```

输出样例2

```
NO
```

知识点

递归分析 类似难题可参照17级算法“W型串”和18级算法“pair”

AC Code

```
#include<iostream>
#include<string>
using namespace std;
inline bool cmp(string a, string b) {
  //printf("cmp %s %s\n", a.c_str(), b.c_str());
  string a1 = a.substr(0, a.size() / 2);
  string a2 = a.substr(a.size() / 2, a.size() - a.size() / 2);
  string b1 = b.substr(0, b.size() / 2);
  string b2 = b.substr(b.size() / 2, b.size() - b.size() / 2);

  if (a1.size() & 1) {
    if ((a1 == b1 && a2 == b2) || (a1 == b2 && a2 == b1))return true;
    else return false;
  }
  else {
    if ((cmp(a1, b1) && cmp(a2, b2)) || (cmp(a1, b2) && cmp(a2, b1)))return
true;
    else return false;
  }
}
string a, b;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> a >> b) {
    if (a.size() & 1) {
      if (a == b)puts("YES");
      else puts("NO");
    }
    else {
      if (cmp(a, b))puts("YES");
```

```
        else puts("NO");
      }
    }
}
```

H Gzh之返老还童III

时间限制：3000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

年幼的Gzh这次又想要玩字符串了。这次，摆在他面前的有三个字符串a，b，c。

Gzh想要找到一个最长的串，但是这个串必须有一定要求，这样才不会显得太过于无趣。

于是他决定他要找的这个串必须是a和b的子串，而且c必须不是它的子串，下面请你们来帮帮他吧。

输入

多组输入数据，每组为三个串a，b，c。其中a，b的长度不超过50000，c的长度不超过10000。

输出

每组输入数据输出一行，为Gzh想找到的串的长度。

输入样例

```
abc abcd a
```

输出样例

```
2
```

知识点

KMP（替换掉a和b中的所有c）

二分答案+字符串哈希（较慢 TLE） 后缀数组（中等 TLE） 后缀自动机（最快 AC）（求a和b最长公共子串）

AC Code

```cpp
#include<cstdio>
#include<cstring>
#include<string>
#define Max(a,b) (((a)>(b))?(a):(b))
using namespace std;
typedef unsigned long long ull;
const int CHAR = 130;
const int N = 250010;
struct SAM_Node {
    SAM_Node* fa, * next[CHAR];
    int len;
    int id, pos;
    SAM_Node() {}
    SAM_Node(int _len) {
        fa = 0;
        len = _len;
        memset(next, 0, sizeof(next));
    }
};
SAM_Node pool[N << 1], * root, * last;
int Size;
SAM_Node* nnode(int len) {
    pool[Size] = SAM_Node(len);
    pool[Size].id = Size;
    return &pool[Size++];
}
SAM_Node* nnode(SAM_Node* p) {
    pool[Size] = *p;
    pool[Size].id = Size;
    return &pool[Size++];
}
void SAM_init() {
    Size = 0;
    root = last = nnode(0);
    pool[0].pos = 0;
}
void add(int x, int len) {
    SAM_Node* p = last, * np = nnode(p->len + 1);
    np->pos = len;
    last = np;
    for (; p && !p->next[x]; p = p->fa)
        p->next[x] = np;
    if (!p) {
        np->fa = root;
        return;
    }
    SAM_Node* q = p->next[x];
    if (q->len == p->len + 1) {
        np->fa = q;
```

```
        return;
    }
    SAM_Node* nq = nnode(q);
    nq->len = p->len + 1;
    q->fa = nq;
    np->fa = nq;
    for (; p && p->next[x] == q; p = p->fa)
        p->next[x] = nq;
}
void SAM_build(char* s) {
    SAM_init();
    int len = strlen(s);
    for (int i = 0; i < len; i++)
        add(s[i], i + 1);
}
char str1[N], str2[N], str3[N];
int main() {
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    int t = 100;
    while (t--) {
        scanf("%s%s%s", str1, str2, str3);
        string a = str1, b = str2, c = str3;
        int pos;
        while ((pos = a.find(c)) != a.npos)a.replace(pos, c.size(), "");
        while ((pos = b.find(c)) != b.npos)b.replace(pos, c.size(), "");
        strcpy(str1, a.c_str()); strcpy(str2, b.c_str());
        SAM_build(str1);//以字符串1建立后缀自动机
        int len = strlen(str2);
        SAM_Node* tmp = root;
        int ans = 0;
        int t = 0;//匹配长度
        for (int i = 0; i < len; i++) { //对于字符串2的每一个前缀做后缀匹配
            if (tmp->next[str2[i]]) { //如果可以从上一步往下走（是相同后缀）
                tmp = tmp->next[str2[i]];
                t++;//往下走，长度+1
            }
            else {
                while (tmp && !tmp->next[str2[i]])//tmp不为空且能走第i位
                    tmp = tmp->fa; //沿着失配指针向上，寻找后缀（去掉一位前缀）
                if (tmp == NULL) { //失配到底了。。
                    tmp = root;
                    t = 0;
                }
                else {
                    t = tmp->len + 1;//tmp节点代表的最小长度
                    tmp = tmp->next[str2[i]];
                }
            }
```

```
            ans = Max(ans, t);
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

第7次上机

A ModricWang的意大利炮

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang有一门意大利炮，还有N枚炮弹。每一枚炮弹只能打到固定的距离(如800米)，并对目标造成一定的伤害。假设敌人的生命值为H，距离为D，炮弹只能按照给你的顺序发射，如果距离合适则必然命中。如果因为距离不符合而没有命中敌人，则称这一枚炮弹被浪费了。请问至少要浪费多少枚炮弹才能消灭敌人?

输入

第一行三个整数，N，D，H

接下来N行，每行两个整数，表示每个炮弹的距离和伤害值

输出

输出一行，浪费的炮弹数。

输入样例

```
4 400 20
1000 99
300 4
400 24
500 9
```

输出样例

```
2
```

知识点

水

AC Code

```
#include<iostream>
using namespace std;

int main(){
    int n,d,h;
    cin>>n>>d>>h;
    int counter=0;
    int a[100000][2];
    for(int i=0;i<n;i++)
    {
        cin>>a[i][0]>>a[i][1];
    }
    for(int i=0;i<n;i++)
    {
        if(a[i][0]!=d)
            counter++;
        else
        {
            if(a[i][1]>=h)
                break;
            else
                h-=a[i][1];
        }
    }
    cout<<counter<<endl;
  return 0;
}
```


B 摄影师ModricWang

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

Rocky山脉有n个山峰，一字排开，从西向东依次编号为1, 2, 3, ..., n。每个山峰的高度都是不一样的。编号为i的山峰高度为hihi。

ZKx从西往东登山。每到一座山峰，ModricWang就帮她拍一张照片，设照片中能看到的山峰数为sisi。

何谓“能看到”？如果在第i座山峰，存在 j < k < i，hj<hk，那么第j座山峰就是不可见的。除了不可见的山峰，其余的山峰都是可见的。

回家之后，ZKx把所有的si加起来得到S作为她此次旅行快乐值。现在n座山峰的高度都提供给你了，你能计算出ZKx的快乐值吗?

输入

第一行一个数，为山峰数n，n<=10000

第二行n个数，为山峰高度

输出

输出一行，题中所述的快乐值

输入样例

```
5
2 1 3 5 9
```

输出样例

```
5
```

知识点

单调栈

AC Code

```
#include<cstdio>
#include<stack>
using namespace std;
int n;
int tmp;
int ans;
stack<int> s;
int main() {
  scanf("%d", &n);
  s.push(2147483647);
  for (int i = 1; i <= n; ++i) {
    ans += s.size() - 1;
    scanf("%d", &tmp);
    while (s.top() < tmp)s.pop();
    s.push(tmp);
  }
```

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

c Mdd的完全二叉树

时间限制：1000ms 内存限制：4000kb

通过率：/ (%) 正确率：/ (%)

题目描述

Mdd有一棵完全二叉树，也就是只有最后一层可能不是满的，他想知道中序遍历这棵树是什么样子的，你能帮帮他吗。

请用**非递归**方式实现中序遍历。

输入

一组输入数据

接下来一行，接下来n个数，为这棵树的层次遍历的节点值。

输出

对于每组数据，输出一行，这棵树的中序遍历，节点之间用空格隔开。

输入样例

```
1 2 3 4 5 6
```

输出样例

```
4 2 5 1 6 3
```

输入样例

```
1 2 3 4 5 6 7 8 9 10
```

输出样例

```
8 4 9 2 10 5 1 6 3 7
```

知识点

完全二叉树构造 非递归中序遍历（在邓俊辉数据结构中，对非递归的前中后遍历有详细介绍）

AC Code

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#include<queue>
#include<stack>
#include<algorithm>
using namespace std;
struct Tree {
  int value;
  int lchild, rchild, parent;
} tree[100010];
int Size;
inline void inorder_iteration() {
  stack<int> s;
  int cur = 1;
  while (cur || !s.empty()) {
    if (cur) {
      s.push(cur);
      cur = tree[cur].lchild;
    }
    else {
      cur = s.top();
      printf("%d ", tree[cur].value);
      s.pop();
      cur = tree[cur].rchild;
    }
  }
}
int main() {
  int n;
  while (scanf("%d", &n) != EOF) {
    tree[++Size].value = n;
    tree[Size].lchild = tree[Size].rchild = 0;
    tree[Size].parent = Size >> 1;
  }
  for (int i = 1; i <= Size; ++i) {
    if ((i << 1) <= Size)tree[i].lchild = i << 1;
    if (((i << 1) + 1) <= Size)tree[i].rchild = (i << 1) + 1;
  }
  inorder_iteration();
}
```

D Mdd的二叉树

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

既然二叉树是一棵树，那么它就会有枝干和叶子，给你一颗二叉树的前序遍历序列，请你数一数它的叶子节点的数目。

输入

多组输入数据

每组一行字符串，长度小于100，每个字符代表一个节点，'#'代表空节点。

输出

对于每组数据，输出一行，叶子节点的数目

输入样例

```
a##
ab##c##
```

输出样例

```
1
2
```

知识点

叶子节点 本题不一定需要建树

AC Code

```
#include<iostream>
#include<string>
#define maxn 110
using namespace std;
string a;
int res;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> a) {
```

```
    res = 0;
    int len = a.length() - 2;
    for (int i = 0; i < len; ++i) {
      if (a[i] != '#' && a[i + 1] == '#' && a[i + 2] == '#')++res;
    }
    cout << res << endl;
  }
}
```

E Gzh的二叉搜索树遍历

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

输入一系列整数，并建立二叉搜索树，并进行前序，中序，后序和层次遍历。

输入

多组输入数据

每组数据为二叉树的节点值

输出

可能有多组测试数据，对于每组数据，将题目所给数据建立一个二叉搜索树（第一个点为根节点，按输入顺序建树），并对二叉排序树进行前序、中序、后序和层次遍历。 每种遍历结果输出一行。

输入样例

```
1 6 5 9 8
```

输出样例

```
1 6 5 9 8
1 5 6 8 9
5 8 9 6 1
1 6 5 9 8
```

知识点

一行的不定量数据处理 二叉搜索树基础

AC Code（递归遍历 Ver）

```
#include<cstdio>
#include<queue>
#include<iostream>
#include<string>
#include<sstream>
using namespace std;
int n;
struct BST {
  int value;
  BST* lchild, * rchild;
};
void InsertBST(BST*& t, int value) {
  if (t == NULL) {
    t = new BST;
    t->lchild = t->rchild = NULL;
    t->value = value;
  }
  else {
    if (value < t->value)InsertBST(t->lchild, value);
    else InsertBST(t->rchild, value);
  }
}
void printPre(BST* t) {
  if (t) {
    printf("%d ", t->value);
    printPre(t->lchild);
    printPre(t->rchild);
  }
}
void printMi(BST* t) {
  if (t) {
    printMi(t->lchild);
    printf("%d ", t->value);
    printMi(t->rchild);
  }
}
void printBack(BST* t) {
  if (t) {
    printBack(t->lchild);
    printBack(t->rchild);
    printf("%d ", t->value);
  }
}
void printBFS(BST* t) {
  queue<BST*> q;
  q.push(t);
```

```
  while (!q.empty()) {
    BST* x = q.front();
    q.pop();
    printf("%d ", x->value);
    if (x->lchild)q.push(x->lchild);
    if (x->rchild)q.push(x->rchild);
  }
}
string input;
int main() {
  ios::sync_with_stdio(false);
  while (getline(cin, input)) {
    BST* root = NULL;
    stringstream s1(input);
    int n;
    while(s1 >> n)InsertBST(root, n);
    printPre(root); putchar('\n');
    printMi(root); putchar('\n');
    printBack(root); putchar('\n');
    printBFS(root); putchar('\n');
  }
}
```

AC Code（迭代遍历 Ver）

```
#include<cstdio>
#include<queue>
#include<stack>
#include<iostream>
#include<string>
#include<sstream>
using namespace std;
int n;
struct BST {
  int value;
  BST* lchild, * rchild, *parent;
};
void InsertBST(BST*& t, int value) {
  if (t == NULL) {
    t = new BST;
    t->lchild = t->rchild = t->parent = NULL;
    t->value = value;
  }
  else {
    if (value < t->value)InsertBST(t->lchild, value);
    else InsertBST(t->rchild, value);
  }
}
```

```
void updateParent(BST*& t) {
  if (t) {
    if (t->lchild) { t->lchild->parent = t; updateParent(t->lchild); }
    if (t->rchild) { t->rchild->parent = t; updateParent(t->rchild); }
  }
}
void printPre_iteration(BST* t) {
  stack<BST*>s;
  BST* x = t;
  while (true) {
    while (x) {
      printf("%d ", x->value);
      s.push(x->rchild);
      x = x->lchild;
    }
    if (s.empty())break;
    x = s.top(), s.pop();
  }
}
void printMi_iteration(BST* t) {
  stack<BST*>s;
  BST* x = t;
  while (true) {
    while (x) {
      s.push(x);
      x = x->lchild;
    }
    if (s.empty())break;
    x = s.top(), s.pop();
    printf("%d ", x->value);
    x = x->rchild;
  }
}
void printBack_iteration(BST* t) {
  stack<BST*>s;
  BST* x = t;
  if (x)s.push(x);
  while (!s.empty()) {
    if (s.top() != x->parent) {
      while (BST* tmp = s.top()) {
        if (tmp->lchild) {
          if (tmp->rchild)s.push(tmp->rchild);
          s.push(tmp->lchild);
        }
        else {
          s.push(tmp->rchild);
        }
      }
      s.pop();
```

```
    }
    x = s.top(), s.pop();
    printf("%d ", x->value);
  }
}
void printBFS(BST* t) {
  queue<BST*> q;
  q.push(t);
  while (!q.empty()) {
    BST* x = q.front();
    q.pop();
    printf("%d ", x->value);
    if (x->lchild)q.push(x->lchild);
    if (x->rchild)q.push(x->rchild);
  }
}
string input;
int main() {
  ios::sync_with_stdio(false);
  while (getline(cin, input)) {
    BST* root = NULL;
    stringstream s1(input);
    int n;
    while (s1 >> n)InsertBST(root, n);
    updateParent(root);
    printPre_iteration(root); putchar('\n');
    printMi_iteration(root); putchar('\n');
    printBack_iteration(root); putchar('\n');
    printBFS(root); putchar('\n');
  }
}
```

F DH的出行规划

时间限制：3000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH生活在一个神奇的地区，这个地区有许多城市，每个城市都有自己唯一的编号，城市之间只有高速路连通，且编号为ii的城市与编号为2*i2*i和2*i+12*i+1的城市相连，所有高速路都是双向且免费的，现在政府考虑到经济危机，可能会在某些路上增加收费，而DH又由于各种原因经常要从一个地方跑到另一个地方去，现在他想在每次出发前知道这一次出行要花费多少（出行只考虑单程）？

输入

多组输入数据。

对于每组数据，每组数据第一行包含一个整数q(1≤q≤1000)q(1≤q≤1000)，表示发生的事件数（政府宣布对某些路增加收费或者DH计划一次出行均为事件）

接下来qq行，表示事件的情况，首先会有一个整数，如果这个整数为11，后边会有3个整数v,u,wv,u,w，表示政府宣布对从vv到uu的最短路径上的每一条路收费增加ww；如果这个整数为22，后边会有两个整数v,uv,u，表示DH要从vv出发去uu。

说明，最短路指经过路的数量最少的一条路径。 1≤v,u≤1018,v≠u,1≤w≤1091≤v,u≤1018,v≠u,1≤w≤109

输出

对DH的每一次出行计划进行计算，每次输出一行，为一个整数，表示DH这次出行的花费。

输入样例

```
7
1 3 4 30
1 4 1 2
1 3 6 8
2 4 3
1 6 1 40
2 3 7
2 2 4
```

输出样例

```
94
0
32
```

知识点

满二叉树 最近公共祖先 离散化处理

AC Code

```
#include<cstdio>
#include<unordered_map>
#include<algorithm>
#include<cmath>
using namespace std;
typedef unsigned long long ull;
unordered_map<ull, ull> costmap;
inline void write(ull x) {
  //if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
```

```
  putchar(x % 10 + 48);
}
inline ull read() {
  ull k = 0;// , f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;// *f;
}
ull q;
ull op, u, v, w;
ull ret;
inline void addEdge(ull u, ull v, ull w) {
  while (u != v) {
    if (log(u) > log(v)) {
      costmap[u] += w;
      u >>= 1;
    }
    else {
      costmap[v] += w;
      v >>= 1;
    }
  }
}
inline void getCost(ull u, ull v) {
  ret = 0;
  while (u != v) {
    if (log(u) > log(v)) {
      ret += costmap[u];
      u >>= 1;
    }
    else {
      ret += costmap[v];
      v >>= 1;
    }
  }
  write(ret), putchar('\n');
}
int main() {
  while (scanf("%llu", &q) != EOF) {
    costmap.clear();
    while (q--) {
      op = read();
```

```
      if (op == 1) {
        u = read(), v = read(), w = read();
        addEdge(u, v, w);
      }
      else {
        u = read(), v = read();
        getCost(u, v);
      }
    }
  }
}
```

G DH的二叉搜索树

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH喜欢树，但更喜欢一些特殊的树，因为它们长得好看，今天他想来跟大家聊一聊二叉搜索树，什么叫二叉搜索树，首先它是一棵二叉树，其次，这棵树的每个节点的值都不一样且对于每一个节点它的左子树的任何节点的值都比它的值小，右子树的任何节点的值都比它的值大。现在DH想知道由值为1,2,3,...,n1,2,3,...,n的nn个节点组成的二叉搜索树里，高度不小于hh的有多少?

输入

多组输入数据，每组输入数据一行，包含两个数n(1≤n≤35),h(1≤h≤n)n(1≤n≤35),h(1≤h≤n)

输出

每组数据输出一行，为该组数据对应的答案

输入样例

```
1 1
```

输出样例

```
1
```

知识点

计数原理 动态规划

AC Code

```
#include<cstdio>
typedef unsigned long long ull;
ull dp[40][40];
/*****************************************
首先凡是这种和二叉搜索树数个数相关的
跟搜索树的性质一点关系都没有
权当做普通的二叉树来做就可以了

那剩下的就是乘法计数原理dp的事情了
设dp[n][h]为小于等于h的二叉树形态数
然后选择其中一个点做根节点
剩下的分成左右两个子树
*****************************************/
inline void buildDP() {
  for (int i = 0; i <= 35; ++i) {
    dp[0][i] = 1;
  }
  for (int i = 1; i <= 35; ++i) {
    for (int j = 1; j <= 35; ++j) {
      for (int k = 1; k <= i; ++k) {
        dp[i][j] += dp[k - 1][j - 1] * dp[i - k][j - 1];
      }
    }
  }
}
int n, h;
int main() {
  buildDP();
  while (scanf("%d%d", &n, &h) != EOF) {
    printf("%llu\n", dp[n][n] - dp[n][h - 1]);
  }
}
```


第8次上机

A Gzh的二叉树

时间限制：15ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh得到了一颗二叉树，他想知道这颗二叉树满不满足以下两特性。

1.从根节点对折后左右子树能重合。(对应节点内容也要相同，且不需要把节点内容也反转)

2.左右子树平移后可以重合。(结构相同即可，不考虑节点内容，即节点内容可以不同)

输入

第一个数为数据组数n(n<=20)

每组输入一行，为二叉树的广义表表示形式。

输出

对于每组数据，输出一行。

若满足1，贼输出GzhIsHandsome

若满足2，贼输出GzhIsSoHandsome

若都满足，贼输出GzhIsSoSoHandsome

若都不满足，贼输出GzhIsSoSoSoHandsome

输入样例

```
3
AAA(BBB,CCC)
AAA
A(B,)
```

输出样例

```
GzhIsSoHandsome
GzhIsSoSoHandsome
GzhIsSoSoSoHandsome
```

知识点

广义表转二叉树 对称二叉树 同构二叉树

AC Code

```
#include<iostream>
#include<cstring>
using namespace std;
#define maxn 1000
```

```
string list;
int n;
typedef struct tree
{
    string data;
    struct tree *lch,*rch;
};
void createtree(tree *&b,string list)
{
   tree *sta[maxn],*p=NULL;
   int top=-1,k=0,j=0;
   char ch;
   b=NULL;
   ch=list[j];
   for(j=0;list[j]!='\0';j++)
   {
       ch=list[j];
       switch(ch)
       {
           case '(':
               top++;
               sta[top]=p;
               k=1;
               break;
           case ',':
               k=2;
               break;
           case ')':
               top--;
               break;
           default:
               p=new tree;

while(list[j+1]!='('&&list[j+1]!=','&&list[j+1]!=')'&&list[j+1]!='\0')
               {
                   p->data+=list[j];
                   j++;
               }
               p->data+=list[j];
               p->lch=NULL;
               p->rch=NULL;
               if(b==NULL)
                   b=p;
               else
               {
                   if(k==1)
                       sta[top]->lch=p;
                   else
                       sta[top]->rch=p;
```

```
                }
                break;
        }
    }
}
bool judge(tree *lch,tree *rch)
{
    if(lch!=NULL&&rch!=NULL)
    {
        if(lch->data!=rch->data)
            return false;
        if(judge(lch->lch,rch->rch)&&judge(lch->rch,rch->lch))
            return true;
        else
            return false;
    }
    else if(lch==NULL&&rch==NULL)
        return true;
    else
        return false;
}
bool duizhe(tree *&head)
{
    if(head==NULL)
        return 1;
    bool re=judge(head->lch,head->rch);
    return re;
}
bool judge2(tree *lch,tree *rch)
{
    if(lch!=NULL&&rch!=NULL)
    {
        if(judge2(lch->lch,rch->lch)&&judge2(lch->rch,rch->rch))
            return true;
        else
            return false;
    }
    else if(lch==NULL&&rch==NULL)
        return true;
    else
        return false;
}
bool pingyi(tree *&head)
{
    if(head==NULL)
        return 1;
    return judge2(head->lch,head->rch);
}
void destroy(tree *&head)
```

```
{
    if(head==NULL)
        return ;
    else
    {
        destroy(head->lch);
        destroy(head->rch);
        delete head;
    }
}
int main(){
    int n;
    cin>>n;
    while(n--)
    {
        tree *head;
        cin>>list;
        bool flag1=0;
        bool flag2=0;
        createtree(head,list);
        if(duizhe(head))
            flag1=1;
        if(pingyi(head))
            flag2=1;
        if(flag1==1&&flag2==1)
            cout<<"GzhIsSoSoHandsome"<<endl;
        else if(flag1==1&&flag2==0)
            cout<<"GzhIsHandsome"<<endl;
        else if(flag1==0&&flag2==1)
            cout<<"GzhIsSoHandsome"<<endl;
        else
            cout<<"GzhIsSoSoSoHandsome"<<endl;
        destroy(head);
    }
  return 0;
}
```


B Mdd的信

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd有一封信，不过mdd很无聊，他想把这封信的**所有字符(空格，换行符也算)**用哈夫曼编码后再寄出去，他想知道编码之后他的信有多长呢。

输入

只有一组数据。

每组数据包括多行。

输出

对于每组数据，输出一行，编码后的长度。

输入样例

```
Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
```

输出样例

```
797
```

Hint

```
可以用while ((c = getchar()) != EOF) { // your code }, 读入一个字符c。
```

知识点

哈夫曼树的带权路径长度

AC Code

```
#include<cstdio>
#include<cstring>
#include<cstdlib>
#define print 0
#define sheet 150
#define maxn 300
#define len 10010
#define INF 0x3f3f3f3f
struct Text {
  char input[len];
  bool occur[sheet];
  int weight[sheet];
  int size;
```

```
  inline int read() {
    return scanf("%s", input);
  }
  inline int read2() {
    int i = 0;
    char c;
    while ((c = getchar()) != EOF) {
      input[i++] = c;
    }
  }
  inline void init() {
    memset(weight, 0, sizeof(weight));
    memset(occur, false, sizeof(occur));
    size = 0;
  }
  inline void buildWeight() {
    int l = strlen(input);
    for (int i = 0; i < l; ++i) {
      weight[input[i]]++;
      occur[input[i]] = true;
    }
    for (int i = 1; i < sheet; ++i) {
      if (occur[i]) {
        if (print) {
          printf("%c weight is %d\n", i, weight[i]);
        }
        ++size;
      }
    }
    if (print)printf("size is %d\n", size);
  }
};
struct HuffmanTree {
  struct Node {
    int data;
    int weight;
    int lchild, rchild, parent;
  }nodes[maxn];
  int size;
  inline void init() {
    memset(nodes, 0, sizeof(nodes));
    size = 0;
  }
  inline void create(int weight[], int n) {
    int p1 = 0, p2 = 0, min1, min2;
    int cnt = 1;
    for (int i = 1; i <= n && cnt < sheet; cnt++) {
      if (weight[cnt]) {
        nodes[i].weight = weight[cnt];
```

```
        if (print)printf("weight[%d] is %d\n", i, nodes[i].weight);
        i++;
      }
    }
    for (int i = n + 1; i < (n << 1); ++i) {
      min1 = min2 = INF;
      for (int j = 1; j < i; ++j) {
        if (!nodes[j].parent) {
          if (nodes[j].weight < min1) {
            p2 = p1;
            min2 = min1;
            p1 = j;
            min1 = nodes[j].weight;
          }
          else if (nodes[j].weight < min2) {
            p2 = j;
            min2 = nodes[j].weight;
          }
        }
      }
      nodes[i].lchild = p1, nodes[i].rchild = p2;
      nodes[i].weight = nodes[p1].weight + nodes[p2].weight;
      nodes[p1].parent = nodes[p2].parent = i;
    }
    size = n;
  }
  inline void printHuffmanLength() {
    //所求长度即为非叶子节点的权值和
    int ans = 0;
    for (int i = 1; i <= size; ++i) {
      int dep = 0, pos = i;
      while (nodes[pos].parent) {
        dep++;
        pos = nodes[pos].parent;
      }
      ans += nodes[i].weight * dep;
    }
    printf("%d\n", ans);
  }
};
Text text;
HuffmanTree a;
int n;
int main() {

    text.init(), a.init();
    text.read2();
    text.buildWeight();
    a.create(text.weight, text.size);
```

```
    a.printHuffmanLength();

}
```

C DH的公司聚会

时间限制：3000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH的公司有nn名员工，每名员工有至多一名直属上司，现在他为了酬谢员工，准备举办一次公司聚会，不过他准备将员工分成若干组，每一组中任意两人之间不存在上司与下属的关系，因为如果存在上司下属关系可能会使组内气氛略显尴尬，问DH最少需要将员工分为多少组？

输入

多组输入数据。

对于每一组数据第一行为一个整数n(1≤n≤2000)n(1≤n≤2000)，接下来nn行，每行一个整数pi(1≤pi≤norpi=−1)pi(1≤pi≤norpi=−1)，表示第ii名员工的直属上司是pipi，如果pi=−1pi=−1，代表该员工无直属上司。数据保证pi≠ipi≠i，且不会出现环。

输出

每组数据输出一行，为该组数据答案。

输入样例

```
5
-1
1
2
1
-1
```

输出样例

```
3
```

知识点

图的层数

AC Code

```
#include<cstdio>
#include<queue>
#include<vector>
#include<cstring>
#define maxn 2010
using namespace std;
int ans;
vector<int> g[maxn];
//优先遍历最小的元素
bool occur[maxn];
//是否已经遍历过了
int n, op;
inline void eraseMap() {
  for (int i = 0; i < maxn; ++i)
    g[i].clear();
  memset(occur, 0, sizeof(occur));
}
void dfs(int x, int dis) {
  //printf("%d %d\n", x, dis);
  if (dis > ans)ans = dis;
  occur[x] = true;
  for (int a : g[x]) {
    if (!occur[a])dfs(a, dis + 1);
  }

}
int main() {
  while (scanf("%d", &n) != EOF) {
    ans = 0;
    eraseMap();
    for (int i = 1; i <= n; ++i) {
      scanf("%d", &op);
      if (op == -1)g[0].push_back(i);
      else g[op].push_back(i);
    }
    dfs(0, 0);
    printf("%d\n", ans);
  }
}
```


D Mdd的二叉搜索树

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

二叉搜索树是一棵这样的树，它的左子树所有的节点都比根节点小，它的右子树的所有节点都比根节点要大。现在给你n个不同节点，请问这n个节点可以组成多少种不同结构的二叉搜索树，要求把这n个节点都用上。

输入

多组输入数据

每组共一行，一个数n，0 < n <= 15，代表节点数目。

输出

对于每组数据，输出一行，这n个节点可以组成的二叉搜索树的数目。

输入样例

```
1
2
3
```

输出样例

```
1
2
5
```

知识点

组合数学 卡特兰数

AC Code

```cpp
#include<cstdio>
typedef long long ll;
int n;
int main() {
  while (scanf("%d", &n) != EOF) {
    if (n == 0)puts("0");
    else {
      ll a = 1;
      for (int i = 1; i <= n; ++i) {
        a *= (n + i), a /= i;
      }
```

```
        a /= (n + 1);
        printf("%lld\n", a);
      }
    }
}
```

E Gzh的射击游戏

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

Gzh最近在玩一个射击游戏，首先他站在起点，起点和终点的距离为x(x<111111111)。终点有一个标志物。而射击距离越近，则射击准确度越高。

因为起点终点距离太远，直接从起点射击终点的标志物难度太高，所以Gzh决定在起点和终点之间加上一些落脚点。

假设Gzh一共加上了m个落脚点(m<77777)，并给出每个落脚点到起点的距离y(保证y合法)。

每个落脚点上面都有一个标志物，Gzh现在只能从起点射击到离他最近的标志物，然后到那个落脚点上去射击下一个离他最近的标志物（直到射击到终点的标志物）。

玩了一会的Gzh觉得这个游戏又变得过于简单，于是他决定去掉t个落脚点(保证t合法)。可是他不知道该去掉哪些落脚点，于是他便想出了一些限制条件。

他决定去掉了t个落脚点之后，要保证剩下的之后的射击的准确度最高的那次射击的准确度尽可能的低，此次射击记为p。(强行增加游戏难度)

那么Gzh想知道p这次射击的距离为多少。

输入

多组输入数据，每组输入第一行为三个整数x，m，t。

加下来m行，每行一个数y。

输出

对于每组数据，输出一行，为p的射击距离。

输入样例

```
17 4 2
2
7
11
14
```

输出样例

```
4
```

知识点

二分答案 前缀和优化

```cpp
#include<cstdio>
#include<algorithm>
#include<climits>
#include<cstring>
using namespace std;
#define Max(a,b) (((a)>(b))?(a):(b))
#define Min(a,b) (((a)<(b))?(a):(b))
#define maxn 100010
#define debug 0
int lo, hi, mi;
/**************************************
本题属于二分答案找最大值

每当发现有一个不够的时候
就利用前缀和+lower_bound即可

比如说发现a[i]<mi的时候
从s当中找到s[cur]+mi-a[cur]即可
**************************************/
inline void write(int x) {
  //if(x < 0)putchar('-'),x=-x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0;// int f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    //printf("get here!\n");
```

```
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;
}
int x, m, t;
int a[maxn];
int s[maxn];
int pre, now;
inline void printAS() {
  puts("a is ");
  for (int i = 1; i <= m; ++i)printf("%d ", a[i]);
  puts("");
  for (int i = 1; i <= m; ++i)printf("%d ", s[i]);
  puts("");
}

int main() {
  while (scanf("%d%d%d", &x, &m, &t) != EOF) {
    memset(a, 0, sizeof(a));
    memset(s, 0, sizeof(s));
    pre = 0;
    lo = hi = x;
    for (int i = 1; i <= m; ++i) {
      s[i] = read();
    }
    s[++m] = x;
    sort(s + 1, s + m + 1);
    for (int i = 1; i <= m; ++i) {
      a[i] = s[i] - s[i - 1];
      if (a[i] < lo)lo = a[i];
    }
    if (debug)printAS();
    if (t == m - 1) { write(x), putchar('\n'); continue; }
    else if (t == 0) { write(lo), putchar('\n'); continue; }
    while (lo <= hi) {
      if (debug)printf("lo is %d, hi is %d\n", lo, hi);
      mi = (lo + hi) >> 1;
      if (debug)printf("mi is %d\n", mi);
      int needK = 0;
      int cur = 1;
      while (cur <= m) {
        if(debug) printf("cur is %d\n", cur);
        if (cur == m) {
          if (a[cur] < mi)needK += 1;
          break;
        }
        else if (a[cur] < mi) {
          int* pos = lower_bound(s + cur + 1, s + m + 1, s[cur] + mi - a[cur]);
```

```
            needK += pos - s - cur;
            if (debug)printf("needK plus %d\n", pos - s - cur);
            cur = pos - s + 1;
          }
          else {
            cur++;
          }
        }
        if (needK > t) {
          if (debug)puts("hi is too high");
          hi = mi - 1;
        }
        else {
          if (debug)puts("lo is too low");
          lo = mi + 1;
        }
      }
      write(lo - 1); putchar('\n');
    }
}
```

F DH去旅行

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH生活在一个美丽的世界，这个世界有nn个城市，这nn个城市形成一颗树，而且是以DH的家所在的1号城市为根的，而树的叶节点的城市是有旅游景点的，DH希望能够挑一个带旅游景点的城市去玩，但他很胆小，有些城市的一些情况非常不好，他并不很喜欢通过，但是为了去旅行，他克服了一部分心中的恐惧，但是还是只能够连续通过不超过mm个情况不好的城市，现在他希望知道在这样的情况下，他还有多少个带旅游景点的城市可供选择?

输入

多组输入数据。

对于每一组数据，第一行包括两个整数n,m(2≤n≤105,1≤m≤n)n,m(2≤n≤105,1≤m≤n)，第二行包括nn个整数a1,a2,...,ana1,a2,...,an，表示这个城市的情况好不好（好为0，不好为1），接下来n−1n−1行，每行包括两个整数xi,yi(1≤xi,yi≤n,xi≠yi)xi,yi(1≤xi,yi≤n,xi≠yi)，表示xixi号城市与yiyi号城市直接相连，保证数据是一棵树

输出

对于每组数据，输出一行，为一个整数，代表该组数据的答案

输入样例1

```
4 1
1 1 0 0
1 2
1 3
1 4
```

输出样例1

```
2
```

输入样例2

```
7 1
1 0 1 1 0 0 0
1 2
1 3
2 4
2 5
3 6
3 7
```

输出样例2

```
2
```

知识点

图的bfs

AC Code

```
#include<cstdio>
#include<cstring>
#include<queue>
#include<vector>
#include<algorithm>
#define maxn 100010
#define mp make_pair
#define x first
#define y second
using namespace std;
typedef pair<int, int> PII;
```

```cpp
inline void write(int x) {
  //if(x < 0)putchar('-'),x=-x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0;// int f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;//*f
}
int bad_city[maxn];
bool occur[maxn];
vector<int> g[maxn];
queue<PII> q;
inline void eraseMap() {
  for (int i = 0; i < maxn; ++i)g[i].clear();
  memset(occur, 0, sizeof(occur));
}
int n, m;
int u, v;
int ans;
inline void bfs() {
  while (!q.empty())q.pop();
  q.push(mp(1, 0 + bad_city[1]));
  while (!q.empty()) {
    PII tmp = q.front();
    q.pop();
    int u = tmp.x;
    occur[u] = true;
    int bad = tmp.y;
    if (bad > m)continue;
    if (g[u].size() == 1 && u != 1) { ++ans; continue; }
    for (int i = 0; i < g[u].size(); ++i) {
      int v = g[u][i];
      if (occur[v])continue;
      if (bad_city[v])q.push(mp(v, bad + 1));
      else q.push(mp(v, 0));
    }
  }
}
int main() {
```

```
  while (scanf("%d%d", &n, &m) != EOF) {
    eraseMap();
    ans = 0;
    for (int i = 1; i <= n; ++i)bad_city[i] = read();
    for (int i = 1; i < n; ++i) {
      u = read(), v = read();
      g[u].push_back(v);
      g[v].push_back(u);
    }
    bfs();
    write(ans), putchar('\n');
  }
}
```

第9次上机

A ModricWang请客吃饭

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang有很多朋友，有一天ModricWang准备找个手机尾号第M小的朋友出来吃顿饭。给定ModricWang的朋友们的手机后四位，请你帮忙找到这些手机尾号中第M小的。

输入

第两个数，为ModricWang的朋友数n和序号M，1≤M≤n≤1e61≤M≤n≤1e6

接下来一行，共n个数字，范围在0~9999之间

输出

输出一行，第M小的手机尾号

输入样例

```
9 3
1234 2345 3456 0987 9876 8765 7654 4321 5432
```

输出样例

```
2345
```

知识点

stl之nth_element

AC Code

```
#include <cstdio>
#include <algorithm>
#define maxn 1000010
using namespace std;
int a[maxn];
int n, k;
int main() {
  while (scanf("%d%d",&n,&k)!=EOF) {
    for (int i = 0; i < n; ++i) scanf("%d",&a[i]);
    nth_element(a, a + k, a + n);
    printf("%d\n", a[k - 1]);
  }
}
```

B Ugly Number

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

对于一给定的素数集合 S = {p1, p2, ..., pK}, 来考虑那些质因数全部属于S 的数的集合。

这个集合包括，p1, p1p2, p1p1, 和 p1p2p3 (还有其它)。这是个对于一个输入的S的丑数集合。

注意：我们不认为1 是一个丑数。

你的工作是对于输入的集合S去寻找集合中从小到大排列的第N个丑数。

longint(signed 32-bit)对于程序是足够的。

输入

第 1 行: 二个整数:K 和 N， 1<= K<=100， 1<= N<=100,000.

第 2 行: K 个整数:集合S的元素

输出

单独的一行，写上对于输入的S的第N个丑数。

输入样例

```
4 19
2 3 5 7
```

输出样例

```
27
```

知识点

记忆化搜索

AC Code

```
#include<cstdio>
#include<algorithm>
#include<climits>
#define maxn 100010
using namespace std;
typedef unsigned long long ull;
/*********************************
这题这么考虑
寻找第i大的丑数需要
比i-1的要大，且每次循环都要
乘以从最小到某一位置，直到大于i-1
然后找到上面所述的范围中最小的那个
*********************************/
const ull INF = ULLONG_MAX;
ull a[maxn];
ull locate[maxn];
ull ans[maxn];
int k, n;
int main() {
  scanf("%d%d", &k, &n);
  for (int i = 1; i <= k; ++i) {
    scanf("%llu", a + i);
    locate[i] = 1;
  }
  sort(a + 1, a + k + 1);
  ans[1] = 1;
  for (int i = 2; i <= n + 1; ++i) {
    ans[i] = INF;
    for (int j = 1; j <= k; ++j) {
      while (a[j] * ans[locate[j]] <= ans[i - 1]) locate[j]++;
      if (a[j] * ans[locate[j]] < ans[i])ans[i] = a[j] * ans[locate[j]];
```

```
    }
  }
  printf("%llu\n", ans[n + 1]);
}
```

C ModricWang的新显卡

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang省吃俭用了整整一个月，终于买到了心爱的显卡。

ModricWang分配了N个任务给显卡，每个任务在单位时间内的计算量为Wi，运行在[Bi,Ei]时间内（对于显卡来说，任务是可以并行的，并且这张显卡的计算能力是正无穷）。

请你帮ModricWang算出，在[T1,T2]时间内，显卡总共完成了多少计算量的任务?

输入

第一行，三个整数，任务总数N，T1，T2,1≤N≤1e61≤N≤1e6

接下来N行，每行3个整数，Wi， Bi，Ei

输出

输出一个整数，在[T1,T2]时间内，显卡总共完成了多少计算量

输入样例

```
5 5 15
20 0 9
21 4 5
40 3 30
2 1 99
30 50 100
```

输出样例

```
583
```

知识点

水

AC Code

```
#include<cstdio>
typedef unsigned long long ull;
int n, t1, t2;
int w, b, e;
ull ans;
inline int realTime(int b, int e) {
  if (b >= t1 && b <= t2) {
    if (e >= t2)return t2 - b + 1;
    else return e - b + 1;
  }
  else if (b <= t1) {
    if (e >= t1 && e <= t2) return e - t1 + 1;
    else if (e > t2)return t2 - t1 + 1;
    else return 0;
  }
  else return 0;
}
int main() {
  scanf("%d%d%d", &n, &t1, &t2);
  while (n--) {
    scanf("%d%d%d", &w, &b, &e);
    ans += w * realTime(b, e);
  }
  printf("%llu\n", ans);
}
```


D Mdd的二叉树(II)

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

春天到了，路边光秃秃的树都长出了新叶。二叉树就像现实中的树一样，又有根，又有叶。那么一棵二叉树的根节点，到最近叶子节点要经过多少个节点呢?

输入

多组输入数据

每组数据共一行，为二叉树的前序遍历，长度不超过100，"#"代表空节点。

输出

对于每组数据，输出一行，根节点到最近的叶节点经过的节点数。

输入样例

```
a##
#
abc###d##
```

输出样例

```
1
0
2
```

知识点

前序遍历建树

AC Code

```
#include<cstdio>
#define maxn 1010
struct Tree {
  char value;
  Tree* lchild, * rchild;
  Tree(Tree* l = NULL, Tree* r = NULL, char a = '\0') {
    value = a;
    lchild = l;
    rchild = r;
  }
};
char input[maxn];
int index;
int n;
void createBinTree(Tree*& t) {
  ++index;
  if (input[index] == '\0')return;
  if (input[index] != '#') {
    t = new Tree;
    t->value = input[index];
    createBinTree(t->lchild);
    createBinTree(t->rchild);
  }
  else t = NULL;
```

```
}
int ans;
void dfs(Tree* t, int dis) {
  if (t->lchild == NULL && t->rchild == NULL && ans > dis) { ans = dis; return;
}
  if (t->lchild)dfs(t->lchild, dis + 1);
  if (t->rchild)dfs(t->rchild, dis + 1);
}
Tree* root;
int main() {

  while (scanf("%s", input + 1) != EOF) {
    ans = 114514;
    index = 0;
    createBinTree(root);
    if (root == NULL)puts("0");
    else {
      dfs(root, 1);
      printf("%d\n", ans);
    }
  }
}
```

E DH的满k叉树

时间限制：1000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH有一颗无穷的满kk叉树，且每一个节点与它的子节点相连的边的权值分别为1,2,...,k1,2,...,k，DH现在想知道在这棵树上有多少条从根出发不重复经过边的路径满足经过的边权值之和为nn而且至少经过一条权值不小于dd的边

输入

多组输入数据，对于每一组输入数据，为一行，包括三个整数
n,k,d(1≤n,k≤100,1≤d≤k)n,k,d(1≤n,k≤100,1≤d≤k)

输出

对于每一组数据，输出一行，为该组数据答案对1000000007(109+7)1000000007(109+7)取模的结果

输入样例

```
1 1 1
```

输出样例

```
1
```

知识点

动态规划（所以说放在算法课当中就不是难题了）

AC Code

```
#include<cstdio>
#define MOD 1000000007
#define maxn 110
typedef unsigned long long ull;
/**************************************************
这题说是满k叉树，但是总感觉跟树一点关系都没有
完全当做算法的dp问题来做也没问题
参考限高二叉树的基础思想，这题拿计数原理也就能做了
首先定义dp[i][j]为路径和为i,只拿1到j作为路径的方法数

所以答案就是dp[n][k]-dp[n][d-1]啦

边界全都是0
那首先dp[0][i]=1不多说 i从1到k

然后dp[i][j] = ∑dp[i-k][j] k≤i 且k≤j
**************************************************/
ull dp[maxn][maxn];
int n, k, d;
inline void buildDP() {
  for (int i = 1; i < maxn; ++i)dp[0][i] = 1;
  for (int i = 1; i < maxn; ++i) {
    for (int j = 1; j < maxn; ++j) {
      for (int k = 1; k <= i && k <= j; ++k) {
        dp[i][j] += dp[i - k][j];
        dp[i][j] %= MOD;
      }
    }
  }
}
int main() {
  buildDP();
```

```
  while (scanf("%d%d%d", &n, &k, &d) != EOF) {
    printf("%llu\n", (dp[n][k] - dp[n][d - 1] + MOD) % MOD);
  }
}
```

F Gzh的数学游戏

时间限制：20ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

Gzh有一个数轴，他现在就站在了原点。

他每次的移动被限制了，也就是说他每次只能从给定的一组数据中选择移动的距离。

会有两个数字n和m来限制这组数据，这组数据有n+1个数字，最大的数字为m（这组数据必须存在至少一个m），其余的数字都不超过m且允许重复，也就是说，当给出了n和m之后，会有m^n组数据。

Gzh每次的移动可以从一组数据中任意选择一个数，然后向左或者向右移动该数字距离的单位，可以走任意次，且数字可以被重复选取。

Gzh想知道，给出n和m之后，产生的m^n组数据中，有多少组可以让他移动到（1,0）处。

输入

多组输入数据

每组两个数字，为n和m。(n <= 15 , m <= 11)。

输出

对于每组数据，输出一行，为对应的组数。

输入样例

```
2 2
```

输出样例

```
3
```

Hint

(1,1,2) (1,2,2) (2,1,2) (2,2,2)中，前三组均可以实现。（向右走2，再向左走1）

知识点

裴蜀定理（数论当中的重要理论之一，高代里面也讲过） 容斥原理

顺便，这道题的原本限制是m<=100000000，可以考虑尝试做一下，合数的分解处理可能会比较麻烦，可能还需要练一下快速幂

AC Code

```
#include<cstdio>
#include<cmath>
unsigned long long ans;
/**************************************************
这题需要利用数学相关知识
裴蜀定理
设a,b均为非零整数
则存在x,y使得ax+by=gcd(a,b)
a,b互质的话那答案就是1了

所以说这题反过来说
只要有ab两数是互质的话，在这里就成立了

那剩下的基本上就好办了
对于m是质数的话
那唯独只有全都是自己的排列无法满足
如果是合数的话
就在全都是自己的基础上再除去他质因数的全部情况
这就是所谓容斥原理的问题了
**************************************************/
int n, m;
int main() {
  while (scanf("%d%d", &n, &m) != EOF) {
    if (m == 1)ans = 1;
    else if (m == 2 || m == 3 || m == 5 || m == 7 || m == 11) {
      ans = pow(m, n) - 1;
    }
    else if (m == 4 || m == 9) {
      ans = pow(m, n) - pow((int)sqrt(m), n);
    }
    else if (m == 6 || m == 10) {
      ans = pow(m, n) - pow(2, n) - pow(m >> 1, n) + 1;
    }
    else if (m == 8) {
      ans = pow(m, n) - pow(4, n);
    }
    printf("%llu\n", ans);
  }
```

```
}
```

G DH的邮件系统

时间限制：1000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH无聊的时候写了一个邮件系统，现在有一封很重要的邮件正在转发中，一共会有nn次转发，每次转发都是一个有这封邮件的人转发给一个没有这封邮件的人，刚开始这封邮件只有一个人有，转发的格式为name1 reposted name2name1 reposted name2，表示name2name2这个人转发给name1name1这个人，邮件系统对名字的大小写不敏感，现在DH想知道经过了这nn次转发后，最长的转发链有多长?

输入

多组输入数据，对于每一组数据，第一行为一个整数n(1<=n<=200)n(1<=n<=200)，接下来nn行，每行都是name1 reposted name2name1 reposted name2格式的一行字符串，保证name1,name2name1,name2长度在[2,24][2,24]区间内且只包括大小写字母和数字。

输出

对于每组数据，输出一行，为该组数据的答案。

输入样例1

```
5
tourist reposted Polycarp
Petr reposted Tourist
WJMZBMR reposted Petr
sdya reposted wjmzbmr
vepifanov reposted sdya
```

输出样例1

```
6
```

输入样例2

```
6
Mike reposted Polycarp
Max reposted Polycarp
EveryOne reposted Polycarp
111 reposted Polycarp
VkCup reposted Polycarp
Codeforces reposted Polycarp
```

输出样例2

```
2
```

输入样例3

```
1
SoMeStRaNgEgUe reposted PoLyCaRp
```

输出样例3

```
2
```

知识点

图的层数

AC Code

```cpp
#include<unordered_map>
#include<string>
#include<algorithm>
#include<cstring>
#include<cctype>
#include<vector>
#define maxn 1010
using namespace std;
unordered_map<string, int>namelist;
int n, Size, ans;
vector<int>g[maxn];
bool occur[maxn];
inline void eraseMap() {
  for (int i = 0; i < maxn; ++i)g[i].clear();
  namelist.clear();
  Size = ans = 0;
  memset(occur, 0, sizeof(occur));
}
```

```
inline void toLower(string& a) {
  for (int i = 0; i < a.size(); ++i) {
    a[i] = tolower(a[i]);
  }
}
void dfs(int x, int dis) {
  if (dis > ans)ans = dis;
  occur[x] = true;
  for (int a : g[x])if (!occur[a])dfs(a, dis + 1);
}
char a1[25], b1[25], c1[25];
string a, b, c;
int main() {
  while (scanf("%d", &n) != EOF) {
    eraseMap();
    while (n--) {
      scanf("%s%s%s", a1, b1, c1);
      a = a1, c = c1;
      toLower(a), toLower(c);
      if (!namelist.count(c)) namelist[c] = ++Size;
      if (!namelist.count(a)) namelist[a] = ++Size;
      g[namelist[c]].push_back(namelist[a]);
    }
    dfs(1, 1);
    printf("%d\n", ans);
  }
}
```

第10次上机

A ModricWang的树

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

给出一棵二叉树的前序遍历序列，用"#"表示空结点，建立二叉树并求该树的深度。

输入

输入第一行包括一个整数n(1<=n<=100)，表示数据组数。
接下来的n行每行给出一棵二叉树的前序遍历序列，其长度小于1000。

输出

输出树的深度

输入样例

```
2
AB###
ABC##D###
```

输出样例

```
2
3
```

知识点

树的深度

AC Code

```
#include<cstdio>
#define maxn 1010
struct Tree {
  char value;
  Tree* lchild,* rchild;
  Tree(Tree* l = NULL, Tree* r = NULL, char a = '\0') {
    value = a;
    lchild = l;
    rchild = r;
  }
};
char input[maxn];
int index;
int n;
void createBinTree(Tree*& t) {
  ++index;
  if (input[index] == '\0')return;
  if (input[index] != '#') {
    t = new Tree;
    t->value = input[index];
    createBinTree(t->lchild);
    createBinTree(t->rchild);
  }
  else t = NULL;
}
```

```
int height(Tree* t) {
  if (t == NULL)return 0;
  else {
    int l = height(t->lchild);
    int r = height(t->rchild);
    return (l < r) ? r + 1 : l + 1;
  }
}
Tree* root;
int main() {
  scanf("%d", &n);
  while (n--) {
    scanf("%s", input + 1);
    index = 0;
    createBinTree(root);
    printf("%d\n", height(root));
  }
}
```

B DH的传送带

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH在一条传送带上，传送带有nn个点，第ii个点有一个值aiai，表示接下来他要被传送到第i+aii+ai个点处，最后一个点没有值，认为不会再传送，现在DH在第1个点，他想知道他是否能到达第tt个点

输入

多组输入数据，对于每一组数据，第一行两个整数n(3≤n≤3×104),t(2≤t≤n)n(3≤n≤3×104),t(2≤t≤n)

第二行n−1n−1个整数a1,a2,...,an−1(1≤ai≤n−i)a1,a2,...,an−1(1≤ai≤n−i)

输出

对于每一组数据，输出一行，如果DH可以到达目的地，输出"YES"，否则输出"NO"（均不带引号）

输入样例1

```
8 4
1 2 1 2 1 2 1
```

输出样例1

```
YES
```

输入样例2

```
8 5
1 2 1 2 1 1 1
```

输出样例2

```
NO
```

知识点

水

AC Code

```
#include<cstdio>
#include<cstring>
#define maxn 50000
inline void write(int x) {
  if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
int a[maxn];
int n, t;
int main() {
  while (scanf("%d%d", &n, &t) != EOF) {
    memset(a, 0, sizeof(a));
    for (int i = 1; i < n; ++i)a[i] = read();
    int t1 = a[1];
    int j = 1;
```

```
    int ans = 0;
    bool flag = false;
    while (ans < n) {
      ans = t1 + j;
      if (ans > n)break;
      j = ans;
      t1 = a[ans];
      if (ans == t) { flag = true; break; }
    }
    if (flag)puts("YES");else puts("NO");
  }
}
```

c Gzh之图的深度优先遍历

时间限制：2000ms 内存限制：102400kb

通过率：/ (%) 正确率：/ (%)

题目描述

如题所示，给定一个有向图，给出它的深度优先遍历结果，遍历起始点的值按从小到大依次输出，每次遍历也按从小到大的顺序进行dfs。

输入

多组输入数据。

对于每组数据，第一行是两个整数k,m（0< k <1001，0<m<k*k-k），表示有k个顶点和m条边，**顶点的值小于5000**，保证每组数据的边中出现所有顶点。

下面的m行，每行是空格隔开的两个整数u，v，表示一条连接节点值为u，v顶点的有向边。

输出

对于每组数据，输出k行，每行为以改行起点dfs遍历的结果。

输入样例

```
5 4
0 1
0 2
3 0
4 0
```

输出样例

```
0 1 2
1
2
3 0 1 2
4 0 1 2
```

Hint

仔细读题。数据量较大，oj比较不稳定，如果你觉得你的做法没错但是tle了可以重新提交试试。

知识点

图的dfs

但是这道题其实学的应该是不同数据量的情况下，读写函数不同对于运行时间与占用内存的影响，以及不同的优化指令。建议读者好好总结

AC Code

```
#include<cstdio>
#include<queue>
#include<algorithm>
#include<vector>
#include<cstring>
#include<iomanip>
#include<cctype>
#include<set>
#pragma G++ optimize(2)
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-funroll-loops")
```

```cpp
#pragma GCC optimize("-fwhole-program")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-fstrict-overflow")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-skip-blocks")
#pragma GCC optimize("-fcse-follow-jumps")
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("-funsafe-loop-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#define maxn 5010
using namespace std;
typedef set<int>::iterator It;
FILE* in;
bool isEnd;
void write(int x) {
  //if(x < 0)putchar('-'),x=-x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
int read() {
  int k = 0;// int f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;//*f
}
```

```cpp
vector<int> g[maxn];
//优先遍历最小的元素
bool occur[maxn];
//是否已经遍历过了
int t;
int tmpA, tmpB;
int k,m;
set<int> s;//此处做优化，只遍历存在的点
void dfs(int x) {
  occur[x] = true;
  write(x);
  putchar(' ');
  for (int u : g[x]) {
    if(!occur[u])dfs(u);
  }
}
int main() {

  while (scanf("%d%d",&k,&m)!=EOF) {
    while (m--) {
      tmpA = read(); tmpB = read();
      g[tmpA].push_back(tmpB);
      s.insert(tmpA), s.insert(tmpB);
    }
    It it;
    for (it = s.begin(); it != s.end(); ++it) {
      sort(g[(*it)].begin(), g[(*it)].end());
    }
    for (it = s.begin(); it != s.end(); ++it) {
      memset(occur, 0, sizeof(occur));
      dfs((*it));
      putchar('\n');
    }
    for (it = s.begin(); it != s.end(); ++it)g[(*it)].clear();
    s.clear();
  }
}
```


D DH的整数魔法

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH喜欢整数，现在他有一个整数nn，他可以对其施展若干次魔法，DH会两种魔法，一种是将一个数变成它的2倍，另一种是让一个数减1，现在他想要把nn变成整数mm，问他最少需要施展多少次魔法

输入

多组输入数据，对于每一组数据，为一行，包含两个整数n,m(1≤n,m≤104)n,m(1≤n,m≤104)

输出

对于每一组数据，输出一行，为该组数据答案

输入样例

```
1 1
```

输出样例

```
0
```

知识点

脑筋急转弯?

AC Code

```
#include<cstdio>
int n, m;
long long res;
int main() {
  while (scanf("%d%d", &n, &m) != EOF) {
    res = 0;
    while (n < m) {
      ++res;
      if (!(m & 1))m >>= 1;
      else m += 1;
    }
    printf("%lld\n", res + (1ll * n - 1ll * m));
  }
}
```


E Mdd的二叉树(III)

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

春天到了，路边光秃秃的树都长出了新叶。二叉树就像现实中的树一样，又有根，又有叶。那么一棵二叉树的根节点，到一个**叶节点**所要经过节点数的期望是多少呢?

输入

多组输入数据

每组数据共一行，为二叉树的前序遍历，长度不超过100，"#"代表空节点。

输出

对于每组数据，输出一行，根节点到**叶节点**经过的节点数的期望，**保留两位小数**。

输入样例

```
a##
#
abc###d##
```

输出样例

```
1.00
0.00
2.50
```

知识点

前序遍历建树

AC Code

```
#include<cstdio>
#define maxn 1010
struct Tree {
  char value;
  Tree* lchild, * rchild;
  Tree(Tree* l = NULL, Tree* r = NULL, char a = '\0') {
    value = a;
    lchild = l;
    rchild = r;
  }
};
char input[maxn];
```

```
int index;
int n;
void createBinTree(Tree*& t) {
  ++index;
  if (input[index] == '\0')return;
  if (input[index] != '#') {
    t = new Tree;
    t->value = input[index];
    createBinTree(t->lchild);
    createBinTree(t->rchild);
  }
  else t = NULL;
}
int ans, cnt;
double res;
void dfs(Tree* t, int dis) {
  if (t->lchild == NULL && t->rchild == NULL) { ans += dis; ++cnt; return; }
  if (t->lchild)dfs(t->lchild, dis + 1);
  if (t->rchild)dfs(t->rchild, dis + 1);
}
Tree* root;
int main() {

  while (scanf("%s", input + 1) != EOF) {
    ans = cnt = 0;
    index = 0;
    createBinTree(root);
    if (root == NULL)puts("0.00");
    else {
      dfs(root, 1); res = ans * 1.0 / cnt;
      printf("%.2lf\n", res);
    }
  }
}
```


F Mdd的二叉树(IV)

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

春天到了，路边光秃秃的树都长出了新叶。二叉树就像现实中的树一样，又有根，又有叶。那么一棵二叉树的根节点，到一个**任意节点**所要经过节点数的期望是多少呢?

输入

多组输入数据

每组数据共一行，为二叉树的前序遍历，长度不超过100，"#"代表空节点。

输出

对于每组数据，输出一行，根节点到**任意节点**经过的节点数的期望，**保留两位小数**。

输入样例

```
a##
#
abc###d##
```

输出样例

```
1.00
0.00
2.00
```

知识点

前序遍历建树

AC Code

```
#include<cstdio>
#define maxn 1010
struct Tree {
  char value;
  Tree* lchild, * rchild;
  Tree(Tree* l = NULL, Tree* r = NULL, char a = '\0') {
    value = a;
    lchild = l;
    rchild = r;
  }
};
char input[maxn];
int index;
int n;
void createBinTree(Tree*& t) {
  ++index;
  if (input[index] == '\0')return;
  if (input[index] != '#') {
    t = new Tree;
```

```
    t->value = input[index];
    createBinTree(t->lchild);
    createBinTree(t->rchild);
  }
  else t = NULL;
}
int ans, cnt;
double res;
void dfs(Tree* t, int dis) {
  ans += dis; ++cnt;
  if (t->lchild)dfs(t->lchild, dis + 1);
  if (t->rchild)dfs(t->rchild, dis + 1);
}
Tree* root;
int main() {

  while (scanf("%s", input + 1) != EOF) {
    ans = cnt = 0;
    index = 0;
    createBinTree(root);
    if (root == NULL)puts("0.00");
    else {
      dfs(root, 1); res = ans * 1.0 / cnt;
      printf("%.2lf\n", res);
    }
  }
}
```

G DH滑冰

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

题目描述

DH喜欢滑冰，不过他是初学者，只能在冰面上向平行于x轴或y轴的方向沿直线滑行，遇到空地才可以停下来，现在他想要能够从任何一个空地到达另一个空地，需要至少再增加几个空地

输入

多组输入数据，对于每一组数据，第一行包含一个整数n(1≤n≤100)n(1≤n≤100)，表示现在空地的数量，接下来nn行，每行两个整数xi,yi(1≤xi,yi≤1000)xi,yi(1≤xi,yi≤1000)，表示第ii个空地的坐标，数据保证任意两个空地坐标不同

输出

对于每一组数据，输出一行，为该组数据答案

输入样例1

```
2
2 1
1 2
```

输出样例1

```
1
```

输入样例2

```
2
2 1
4 1
```

输出样例2

```
0
```

知识点

并查集

AC Code

```
#include<cstdio>
int n;
struct node {
  int x, y;
  int father;
};
node info[110];
inline int getFather(int x) {
  return info[x].father == x ? x : info[x].father = getFather(info[x].father);
}
int main() {
  while (scanf("%d", &n) != EOF) {
    for (int i = 1; i <= n; ++i) {
      scanf("%d%d", &info[i].x, &info[i].y);
      info[i].father = i;
    }
    for (int i = 1; i <= n; ++i) {
```

```
      for (int j = 1; j <= n; ++j) {
        if (i == j)continue;
        if (info[i].x == info[j].x || info[i].y == info[j].y) {
          int a = getFather(i);
          int b = getFather(j);
          if (a != b)info[a].father = b;
        }
      }
    }
    int ans = 0;
    for (int i = 1; i <= n; ++i)if (info[i].father == i)++ans;
    printf("%d\n", ans - 1);
  }
}
```

第11次上机

A Mdd去旅游

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd总是幻想着去各地旅游，他选取了很多个旅游胜地，准备都去一遍，有些景点之间可以直接坐公交到达，所以如果有公交线路的话，mdd总是会选择坐公交。如果没有线路，那么mdd只能坐飞机到另一个景点，请问mdd从学校出发，要想走遍所有的景点至少要坐几次飞机。

输入

多组输入数据

对于每组数据，第一行两个数，k(0 < k < 100)，m (0 < m < k * k)，分别代表景点总数，公交线路总数，景点编号为0到k - 1，接下来m行，每行两个景点编号u，v，代表u到v有一条公交线路(**双向**)。

输出

对于每组数据，输出一行，mdd走遍所有景点最少坐飞机次数。

输入样例

```
4 3
0 1
1 3
3 0
```

输出样例

```
2
```

知识点

并查集

AC Code

```
#include<iostream>
#include<algorithm>
#include<climits>
#include<cstring>
#define maxn 105
using namespace std;

int f[maxn];
//并查集板子
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
int n, m;
int u, v;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> n >> m) {
    for (int i = 0; i < n; ++i)f[i] = i;
    while (m--) {
      cin >> u >> v;
      int a = getFather(u);
      int b = getFather(v);
      if (a != b)
        f[a] = b;
    }
    int res = 0;
    for (int i = 0; i < n; ++i)
      if (f[i] == i)res++;
    cout << res << endl;
  }
}
```

B Gzh之颜值同化

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

很简单的问题。有n个人，分成m组，若有一个组中有人的颜值为正无穷，则这个人将会把组里所有人的人的颜值同化为正无穷。n个人的编号分别为0-n-1。若一开始只有0号成员的颜值为正无穷，那么最后颜值为正无穷的成员会有多少个。

输入

多组输入数据，每组数据第一行为两个数n，m。

接下来m行，每行第一个数为小组成员数t，接下来t个数为小组成员的编号。（水题，数据都很小，不用考虑数据大小）

输出

对于每组数据，输出一行，为最后颜值为正无穷的人数。

输入样例

```
100 3
2 0 1
2 1 2
2 2 3
```

输出样例

```
4
```

知识点

并查集

AC Code

```
#include<cstdio>
#include<algorithm>
#include<climits>
#include<cstring>
```

```cpp
#include<vector>
#define maxn 10005
using namespace std;
int f[maxn];
//并查集板子
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
int n, m;
int u, v;
int t;
vector<int>lala;
int main() {

  while (scanf("%d%d",&n,&m)!=EOF) {
    for (int i = 0; i < n; ++i)f[i] = i;
    while (m--) {
      lala.clear();
      scanf("%d", &t);
      while (t--)scanf("%d", &u), lala.push_back(u);
      sort(lala.begin(), lala.end());
      int a = getFather(lala[0]);
      //printf("a = %d\n", a);
      for (int i = 1; i < lala.size(); ++i) {
        int b = getFather(lala[i]);
        //printf("b = %d\n", b);
        if (b < a)f[a] = b;
        else if (a < b)f[b] = a;
      }
    }
    int res = 0;
    for (int i = 0; i < n; ++i)
      if (f[i] == 0)res++;
    printf("%d\n", res);
  }
}
```


c Gzh之物品分类

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

Gzh有n个物品，Gzh想知道这些物品有多少类，请你来帮帮他吧。

输入

多组输入数据，每组第一行为两个数n和m，n代表共有编号为1-n的n个物品。

接下来m行，每行2个整数a,b，为一组相同的物品，没出现的物品视为自己一组。（水题，数据都很小，不用考虑数据大小）

输出

对于每组数据，输出一行，为物品的种类数。

输入样例

```
5 2
1 2
2 3
```

输出样例

```
3
```

知识点

并查集

AC Code

```
#include<iostream>
#include<algorithm>
#include<climits>
#include<cstring>
#define maxn 1105
using namespace std;

int f[maxn];
//并查集板子
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
int n, m;
int u, v;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> n >> m) {
    for (int i = 1; i <= n; ++i)f[i] = i;
    while (m--) {
```

```
      cin >> u >> v;
      int a = getFather(u);
      int b = getFather(v);
      if (a != b)
        f[a] = b;
    }
    int res = 0;
    for (int i = 1; i <= n; ++i)
      if (f[i] == i)res++;
    cout << res << endl;
  }
}
```

D ModricWang的布线问题

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang需要将n台计算机连接起来，不同的2台计算机之间的连接费用可能是不同的。为了节省费用，我们考虑采用间接数据传输结束，就是一台计算机可以间接地通过其他计算机实现和另外一台计算机连接。

为了使得任意两台计算机之间都是连通的（不管是直接还是间接的），需要在若干台计算机之间用网线直接连接，现在想使得总的连接费用最省，让你编程计算这个最小的费用。

输入

输入第一行为两个整数n,m，表示计算机总数，和可以互相建立连接的连接个数。

接下来m行，每行三个整数a,b,c 表示在机器a和机器b之间建立连接的话费是c。

(题目保证一定存在可行的连通方案, 数据中可能存在权值不一样的重边，但是保证没有自环)

对于60分的数据，2<=n<=100，2<=m<=1000

对于100分的数据，2<=n<=10000，2<=m<=600000

输出

输出只有一行一个整数，表示最省的总连接费用。

输入样例

```
3 3
1 2 1
1 3 2
2 3 1
```

输出样例

```
2
```

知识点

最小生成树

AC Code

```
#include<cstdio>
#include<algorithm>
#include<cstdlib>
#include<cstring>
#define maxn 10100
#define maxm 600100
using namespace std;
/*****************************************
最小生成树一般还是用kruskal
因为kruskal可以判断两点之间的重边情况
kruskal与prim结合可以将复杂度优化至nlogn

一般来说点多用kruskal，点不多就prim
prim在没有重边的时候用堆优也挺快
*****************************************/
struct edge {
  int u, v;//起始点与终点
  int w;//权值
  bool operator < (const edge& a) {
    return w < a.w;
  }
} edges[maxm];
int f[maxn];//并查集
int n, m;
int p, q;
int cnt;
long long res;
inline void initFather() {
  for (int i = 1; i <= n; ++i) {
    f[i] = i;
  }
}
```

```
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
inline void write(long long x) {
  //if(x < 0)putchar('-'),x=-x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0;// int f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;//*f
}
inline void kruskal() {
  cnt = 0, res = 0;
  for (int i = 0; i < m; ++i) {
    p = getFather(edges[i].u);
    q = getFather(edges[i].v);
    if (p != q) {
      f[p] = q;//合并集合
      res += edges[i].w;
      cnt++;
    }
    if (cnt == n - 1)break;//边的数量为n-1就完成了
  }
}
int main() {
  while (scanf("%d%d", &n, &m) != EOF) {
    initFather();
    for (int i = 0; i < m; ++i) {
      edges[i].u = read();
      edges[i].v = read();
      edges[i].w = read();
    }
    sort(edges, edges + m);//递减排序
    kruskal();
    write(res);
    putchar('\n');
  }
}
```

```cpp
#include<cstdio>
#include<cstring>
#include<vector>
#include<queue>
#define maxn 10010
#define INF 0x3f3f3f3f
using namespace std;
inline void write(int x) {
  if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
struct Node {
  int v, w;
  Node(int _v = 0, int _w = 0) {
    v = _v, w = _w;
  }
  bool operator < (const Node& o) const {
    return o.w < w;
  }
};
int n, m;
int u, v, w;
int d[maxn];
bool occur[maxn];
vector<Node> g[maxn];
inline void init() {
  memset(d, 0x3f, sizeof(d));
  memset(occur, 0, sizeof(occur));
  for (int i = 0; i < maxn; ++i)g[i].clear();
}
int Prim(int s) {
  priority_queue<Node>q;
  q.push(Node(s, 0));
  int ans = 0;
  while (!q.empty()) {
```

```
    Node tmp = q.top();
    q.pop();
    int v = tmp.v;
    if (occur[v])continue;
    ans += tmp.w;
    occur[v] = true;
    for (int i = 0; i < g[v].size(); ++i) {
      int v2 = g[v][i].v;
      int w2 = g[v][i].w;
      if (!occur[v2] && d[v2] > w2) {
        d[v2] = w2;
        q.push(Node(v2, d[v2]));
      }
    }
  }
  return ans;
}
int main() {
  while (scanf("%d%d", &n, &m) != EOF) {
    init();
    while (m--) {
      u = read(), v = read(), w = read();
      g[u].push_back(Node(v, w));
      g[v].push_back(Node(u, w));
    }
    write(Prim(1)), putchar('\n');
  }
}
```

E DH的礼物

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH是一个神奇的国家的国王，这个国家有NN个城市，MM条双向路。每条路连接两个城市，两个城市间可以有多条路，一条路也有可能将一个城市与它自己连接起来形成自环。

不过这个国家频发抢劫，每条路都经常发生这样的事情。有一天，劫匪们觉得抢劫太无聊了，于是他们想要DH给他们一笔钱，他们就不再抢劫了。劫匪们想要的这笔钱包括一些黄金和一些白银，也开出了相应的条件。对于每一条路，有两个值，gigi，表示需要黄金的最少数量，sisi，表示需要白银的最少数量。意味着如果DH给劫匪的礼物里有aa块黄金，bb块白银的话，劫匪们就将停止在所有满足gi≤agi≤a且si≤bsi≤b条件的路上抢劫。

但是很不幸的是DH这个国家并没有黄金也没有白银，不过DH可以去找其他国家买，买一块黄金需要GG元，买一块白银需要SS元。DH希望把礼物给劫匪以后每两个城市间都有一条安全的路径，他现在想让你帮他算算他最少需要花多少元置办礼物

输入

多组输入数据，对于每组数据，第一行包含两个整数N,M(2≤N≤200,1≤M≤50000)N,M(2≤N≤200,1≤M≤50000)，表示这个国家的城市数和路的数量。第二行包含两个整数G,S(1≤G,S≤109)G,S(1≤G,S≤109)，表示黄金和白银的价钱。接下来MM行就是绑匪开出的条件，每一行包含四个整数xi,yi,gi,sixi,yi,gi,si，表示这条路连接城市xixi和yiyi，需要的最少黄金数量为gigi，白银数量为sisi。(1≤xi,yi≤N,1≤gi,si≤109)(1≤xi,yi≤N,1≤gi,si≤109)城市编号为11到NN，两个城市之间可能连有多条路，一条路也有可能将一个城市与它自己连接起来形成自环。

输出

对于每组数据，输出一行，为DH置办礼物的最小花销。如果怎样都满足不了条件，输出−1−1

输入样例

```
3 3
2 1
1 2 10 15
1 2 4 20
1 3 5 1
```

输出样例

```
30
```

知识点

双关键字的最小生成树

AC Code

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<climits>
#define INF 1145141919810LL//单纯记一个define long long的好方法，没有别的意思
#define Max(a,b) (((a)>(b))?(a):(b))
#define Min(a,b) (((a)<(b))?(a):(b))
#define maxn 410
#define maxm 50010
using namespace std;
typedef unsigned long long ull;
```

```
/*****************************************
说白了这题其实就是双关键字的最小生成树
本题的答案就是留下来的所有路中
G*(g的最长边权)+S*(s的最长边权)
有重边和自环的话，那就kruskal伺候

感觉这种权值算法不同的MST问题可以多练
然后问题就是要多跑几遍kruskal
*****************************************/
inline void write(ull x) {
  //if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline ull read() {
  int k = 0;// f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;// *f;
}
struct edge {
  int u, v;
  ull g, s;
  bool occur;
  bool operator < (const edge& o) {
    if (g != o.g)return g < o.g;
    else return s < o.s;
  }
};
edge edges[maxm], mst[maxm], tmp[maxm];
//原始数据，当前MST，候补MST
int f[maxn];
ull n, m, cnt, ans, S, G;
inline void init() {
  memset(edges, 0, sizeof(edges));
  memset(mst, 0, sizeof(mst));
  memset(tmp, 0, sizeof(tmp));
  ans = ULLONG_MAX;
  cnt = 0;
}
inline void initFather() {
  for (int i = 1; i <= n; ++i) {
```

```
    f[i] = i;
  }
}
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
inline void kruskal(ull maxg) {
  //在已知了最大的g值之后，就求s值，若加和更小则更新
  //复制数据
  for (int i = 1; i <= cnt; ++i) {
    tmp[i] = mst[i], tmp[i].occur = false;
  }
  ull maxs = 0, tot = 0;
  initFather();
  for (int i = 1; i <= cnt; ++i) {
    int a = getFather(mst[i].u);
    int b = getFather(mst[i].v);
    if (a != b) {
      maxs = Max(maxs, mst[i].s);
      f[a] = b;
      tot++;
      tmp[i].occur = 1;
    }
    if (tot == n - 1) {
      int num = 0;
      for (int i = 1; i <= cnt; ++i) {
        if (tmp[i].occur)mst[++num] = tmp[i];
      }
      cnt = num;
      ans = Min(ans, maxg + maxs);
      break;
    }
  }

}
inline void solve() {
  for (ull i = 1; i <= m; ++i) {
    if (edges[i].g + edges[i].s > ans)continue;
    ull pos = cnt + 1;
    for (ull j = 1; j <= cnt; ++j)
      if (mst[j].s > edges[i].s) { pos = j; break; }
    if (pos == cnt + 1) mst[++cnt] = edges[i];
    else {
      ++cnt;
      for (ull j = cnt; j >= pos + 1; --j)mst[j] = mst[j - 1];
      mst[pos] = edges[i];
    }
    kruskal(edges[i].g);
  }
```

```
}
int main() {
  while (scanf("%llu%llu", &n, &m) != EOF) {
    init();
    G = read(), S = read();
    for (ull i = 1; i <= m; ++i) {
      edges[i].u = read(), edges[i].v = read(), edges[i].g = read(), edges[i].s
= read();
      edges[i].g *= G, edges[i].s *= S;
    }
    sort(edges + 1, edges + m + 1);
    solve();
    if (ans == ULLONG_MAX)puts("-1");
    else write(ans), putchar('\n');
  }
}
```

F DH的魔法矩阵

时间限制：5000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH有一个n∗nn∗n的非负矩阵，他想知道这个矩阵具不具备魔法，一个非负矩阵具备魔法的条件是它是对称的(aij=aji),aii=0(aij=aji),aii=0，对于任意i,j,ki,j,k，满足aij≤max(aik,ajk)aij≤max(aik,ajk)，注意i,j,k不一定是不同的，现在他想让你帮忙判断这个矩阵是否具备魔法

输入

多组输入数据，对于每组数据，第一行包含一个整数n(1≤n≤2500)n(1≤n≤2500)，表示这个非负矩阵的阶数，接下来nn行每行包含nn个整数aij(0≤aij<109)aij(0≤aij<109)，表示矩阵中的元素，注意所给的矩阵不一定是对称的，可以是任意的。

输出

对于每组数据，输出一行，如果该矩阵具备魔法，输出”MAGIC””MAGIC”，否则输出”NOT MAGIC””NOT MAGIC”（均不含引号）

输入样例1

```
3
0 1 2
1 0 2
2 2 0
```

输出样例1

```
MAGIC
```

输入样例2

```
2
0 1
2 3
```

输出样例2

```
NOT MAGIC
```

输入样例3

```
4
0 1 2 3
1 0 3 4
2 3 0 5
3 4 5 0
```

输出样例3

```
NOT MAGIC
```

知识点

（首先要能成功转换题意）邻接矩阵建图 无向图判断是否成环 并查集

AC Code

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#define maxn 2505
#define maxm maxn*maxn/2
using namespace std;
```

```
/*******************************************************
这种题看着就很蹊跷，肯定不是当纯矩阵做的题
1,2是可以直接判定的，这个耗时O(n^2)没办法
最重要的是3如何看
于是乎可以直接将矩阵A当做一个图G的邻接矩阵
而且很明显可以看出是一个无自环的无向图
(因为1,2就可以直接pass非这一类的图了)
这样的话第三个条件就好说了
如果出现wij>wik 且wij > wkj就是不符合条件
所以说i到j的边与两个严格比他小的边形成的三元环
该三元环的边权可以看做是严格递增的
那可以这么想，假设形成了一个比较大的环
i->p1->p2->...->j->i
如果有i->p2>=i->j的话，那i->p1->p2->i就已经不合法了
反之则可以缩小这个环
实际上就会发现只要形成环了，就一定不合法
所以只要按照边权从小到大排序，然后加入边
只要成环了，就肯定不合法。

这里正好练一下并查集检验无向图是否成环
只要每次unite的时候两者已有共同祖先的话，就成环了
然后边权相同的话需要特殊处理一下，要先查询
全部合法的话就一并加进去
*******************************************************/
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
struct edge {
  int u, v, w;
  bool operator <(const edge& o) const {
    return w < o.w;
  }
}edges[maxm];
int f[maxn];
int n, cnt;
inline void initFather() {
  for (int i = 1; i <= n; ++i)f[i] = i;
}
inline int getFather(int x) {
```

```
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
inline bool check(int x, int y) {
  int a = getFather(x);
  int b = getFather(y);
  if (a != b) return true;
  else return false;
}
inline void merge(int x, int y) {
  int a = getFather(x);
  int b = getFather(y);
  if (a != b) f[a] = b;
}
inline void addEdge(int u, int v, int w) {
  edges[cnt].u = u, edges[cnt].v = v, edges[cnt++].w = w;
}
int a[maxn][maxn];
bool flag;
int main() {
  while (scanf("%d", &n) != EOF) {
    cnt = 0;
    memset(edges, 0, sizeof(edges));
    flag = true;
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) {
        a[i][j] = read();
      }
    }
    //1和2条件判断
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j <= n; ++j) {
        if (i == j && a[i][j]) {
          flag = false; break;
        }
        if (a[i][j] != a[j][i]) {
          flag = false; break;
        }
      }
      if (!flag)break;
    }
    if (!flag) {
      puts("NOT MAGIC");
      continue;
    }
    //3条件判断
    initFather();
    for (int i = 1; i <= n; ++i) {
      for (int j = 1; j < i; ++j) {
        addEdge(i, j, a[i][j]);
```

```
      }
    }
    sort(edges, edges + cnt);
    edges[cnt].w = -1;//防止一并加入相同边权被边界爆破
    for (int i = 0; i < cnt; ++i) {
      int j = i;
      while (edges[j + 1].w == edges[i].w)j++;
      for (int k = i; k <= j; ++k) {
        if (!(flag = check(edges[k].u, edges[k].v)))break;
      }
      if (!flag)break;
      for (int k = i; k <= j; ++k)merge(edges[k].u, edges[k].v);
      i = j;
    }
    if (flag)puts("MAGIC");
    else puts("NOT MAGIC");
  }
}
```

第12次上机

A Mdd的excel

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

用过Microsoft Excel的同学应该知道，Excel的列不是用数字表示的，而是用字母表示的，'A'=>1，'B'=>2,...,'AA'=>27,'AB'=>28。现在给你一串字符串，请问它代表的是第几列。

输入

多组输入数据

接下来多行，每组一个字符串，**长度不超过10**，只包括大写字母。

输出

对于每组数据，输出一行，该字符串代表的列数

输入样例

```
A
Z
AB
```

输出样例

```
1
26
28
```

知识点

水

AC Code

```cpp
#include<cstdio>
#include<cstring>
typedef unsigned long long ull;
char s[15];
int main() {
  ull ans;
  ull lala;
  while (scanf("%s", s) != EOF) {
    ans = 0, lala = 1;
    for (int i = strlen(s) - 1; i >= 0; --i) {
      ans += 1ll * (s[i] - 'A' + 1) * lala;
      lala *= 26;
    }
    printf("%llu\n", ans);
  }
}
```


B ModricWang的布线问题 II

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang需要将n台计算机连接起来，不同的2台计算机之间的连接费用可能是不同的。为了节省费用，我们考虑采用间接数据传输结束，就是一台计算机可以间接地通过其他计算机实现和另外一台计算机连接。

为了使得任意两台计算机之间都是连通的（不管是直接还是间接的），需要在若干台计算机之间用网线直接连接，现在想使得总的连接费用最省，让你编程计算这个最小的费用。

需要注意的是，ModricWang的计算机并不全都很强。他有一台计算机的承载能力有限，只能接入一根网线。

输入

输入第一行为两个整数n, m, k，表示计算机总数，可以互相建立连接的连接个数和只能接入一根网线的计算机的编号。

接下来m行，每行三个整数a,b,c 表示在机器a和机器b之间建立连接的话费是c。

(题目保证一定存在可行的连通方案, 数据中可能存在权值不一样的重边，但是保证没有自环)

2≤n≤10000，2≤m≤6000002≤n≤10000，2≤m≤600000

输出

输出只有一行一个整数，表示最省的总连接费用。

输入样例

```
4 4 4
1 2 1
1 3 2
2 3 1
1 4 2
```

输出样例

```
4
```

知识点

最小生成树（稍微变了一下而已）

AC Code

```
#include<cstdio>
#include<algorithm>
#include<cstdlib>
#include<cstring>
#define maxn 10100
#define maxm 600100
using namespace std;
struct edge {
  int u, v;//起始点与终点
```

```
  int w;//权值
  bool operator < (const edge& a) {
    return w < a.w;
  }
} edges[maxm];
int f[maxn];//并查集
int n, m, skip;
int u, v, w;
int mine;
bool flag;
int p, q;
int cnt;
int top;
long long res;
inline void initFather() {
  for (int i = 1; i <= n; ++i) {
    f[i] = i;
  }
}
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
inline void write(long long x) {
  //if(x < 0)putchar('-'),x=-x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0;// int f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k;//*f
}
inline void kruskal() {
  cnt = 0, res = flag ? mine : 0;
  for (int i = 0; i < top; ++i) {
    p = getFather(edges[i].u);
    q = getFather(edges[i].v);
    if (p != q) {
      f[p] = q;//合并集合
      res += edges[i].w;
      cnt++;
```

```
    }
    if (cnt == n - 2)break;//边的数量为n-1就完成了
  }
}
int main() {
  while (scanf("%d%d%d", &n, &m, &skip) != EOF) {
    mine = 0x3f3f3f3f, flag = false;
    top = 0;
    initFather();
    for (int i = 0; i < m; ++i) {
      u = read(), v = read(), w = read();
      if (u != skip && v != skip) {
        edges[top].u = u;
        edges[top].v = v;
        edges[top++].w = w;
      }
      else {
        flag = true;
        mine = min(mine, w);
      }
    }
    sort(edges, edges + top);//递减排序
    kruskal();
    write(res);
    putchar('\n');
  }
}
```

C Mdd去旅游(II)

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd总是幻想着去各地旅游，他选取了很多个旅游胜地，准备都去一遍，有些景点之间可以直接坐公交到达，所以如果有公交线路的话，mdd总是会选择坐公交。如果没有线路，那么mdd只能坐飞机到另一个景点，不过这次mdd为了省钱，决定只坐公交车，假如mdd可以从任意一个景点出发，他能否把所有景点全部浏览一遍。

输入

多组输入数据

对于每组数据，第一行两个数，k(1 < k < 100)，m (0 <= m < k * k)，分别代表景点总数，公交线路总数，景点编号为0到k - 1，接下来m行，每行两个景点编号u，v，代表u到v有一条公交线路(双向)。

输出

对于每组数据，输出一行，如果能，则输出"Yes"，否则输出“No"。

输入样例

```
4 3
0 1
1 3
3 0
```

输出样例

```
No
```

知识点

并查集

AC Code

```
#include<iostream>
#include<algorithm>
#include<climits>
#include<cstring>
#define maxn 1105
using namespace std;

int f[maxn];
//并查集板子
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
int n, m;
int u, v;
int main() {
  ios::sync_with_stdio(false);
  cin.tie(0), cout.tie(0);
  while (cin >> n >> m) {
    for (int i = 0; i < n; ++i)f[i] = i;
    while (m--) {
      cin >> u >> v;
      int a = getFather(u);
      int b = getFather(v);
      if (a != b)
        f[a] = b;
```

```
    }
    int res = 0;
    for (int i = 0; i < n; ++i)
      if (f[i] == i)res++;
    cout << (res==1?"Yes":"No") << endl;
  }
}
```

D ModricWang的局域网

时间限制：1000ms 内存限制：65536kb

通过率：/ (%) 正确率：/ (%)

题目描述

ModricWang有n台计算机和m条网线。

ModricWang已经把这些线排布好了，知道了每条网线分别连着的计算机的序号。然而，ModricWang发现，这些计算机并没有连成一个整体，有些计算机间无法直接或间接连通。

ModricWang想知道，这些计算机组成了多少个相互独立的网络。

相互独立是指：两个网络间没有直接或间接连接的计算机。

输入

第一行两个数，计算机数n和网线数m, 2≤n≤1000,1≤m≤10002≤n≤1000,1≤m≤1000

接下来m行，每行两个整数，表示网线的两个端点，保证数据合法。

输出

相互独立的网络的个数

输入样例

```
3 1
1 2
```

输出样例

```
2
```

样例说明

样例数据中有两个相互独立的网络，{1,2}和{3}

知识点

并查集

为难助教了，15年出一堆前序中序遍历建树的板子，16年出了一堆并查集的板子，真就不忍挂掉一大片呗

AC Code

和“Gzh之物品分类”一样

E Mdd去旅游(III)

时间限制：500ms 内存限制：10240kb

通过率：/ (%) 正确率：/ (%)

题目描述

mdd总是幻想着去各地旅游，他选取了很多个旅游胜地，准备都去一遍，有些景点之间可以直接坐公交到达，所以如果有公交线路的话，mdd总是会选择坐公交。如果没有线路，那么mdd只能坐飞机到另一个景点，不过这次mdd为了省钱，决定只坐公交车，假如mdd可以从任意一个景点出发，他最多可以浏览多少个景点。

输入

一组输入数据

对于每组数据，第一行两个数，n，m，，n，m，分别代表景点总数，公交线路总数，景点编号为0~n−10~n−1，接下来mm行，每行两个景点编号u，vu，v，代表uu到vv有一条公交线路(双向)。

1≤n≤1e5，0≤m≤1e6。1≤n≤1e5，0≤m≤1e6。

输出

对于每组数据，输出一行，mdd最多可以浏览多少个景点。

输入样例

```
4 3
0 1
1 3
3 0
```

输出样例

```
3
```

知识点

并查集

AC Code

```
#include<cstdio>
#include<algorithm>
#include<climits>
#include<cstring>
#include<unordered_map>
#define maxn 100005
using namespace std;
typedef pair<int, int>PII;
int ans;
int f[maxn];
unordered_map<int, int>lala;
//并查集板子
inline int getFather(int x) {
  return f[x] == x ? x : f[x] = getFather(f[x]);
}
int n, m;
int u, v;
int main() {
  while (scanf("%d%d", &n, &m) != EOF) {
    lala.clear();
    for (int i = 0; i < n; ++i)f[i] = i;
    while (m--){
      scanf("%d%d", &u, &v);
      int a = getFather(u);
      int b = getFather(v);
      if (a != b)
        f[a] = b;
    }
    ans = 0;
    for (int i = 0; i < n; ++i)
      getFather(i),lala[f[i]]++;
    for (PII n : lala)if (lala[n.first] > ans)ans = lala[n.first];
    printf("%d\n", ans);
  }
}
```

F DH去看球

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH十分喜欢足球，这一天他准备出发去看球。DH很穷，没有自己的车，就只能打车了。DH所在的城市有nn个主节点，其中的一些被一些双向路所相连，路的长度是以米为单位的正整数，路的长度不一定都相同。

刚开始每一个主节点上有且只有一辆出租车，第ii个主节点的出租车司机可以把DH送到距离不超过titi的另一个主节点处，花费为固定的cici元，与距离无关。出租车不能停在路中间，一辆出租车也不能多次使用。DH只能在出租车初始的主节点打这辆车。

刚开始DH在主节点xx，球场在主节点yy，请你帮他计算一下，他最少需要花多少钱才能够到球场

输入

多组输入数据，对于每一组数据，第一行包含两个整数n,m(1≤n≤1000,0≤m≤1000)n,m(1≤n≤1000,0≤m≤1000)，表示主节点的数量和路的数量，第二行包含两个整数x,y(1≤x,y≤n)x,y(1≤x,y≤n)，表示DH的初始位置和球场位置，接下来mm行，每行包含三个整数ui,vi,wi(1≤ui,vi≤n,1≤wi≤109)ui,vi,wi(1≤ui,vi≤n,1≤wi≤109)，表示主节点ui,viui,vi被一条长度为wiwi的路相连。接下来nn行，每行包含两个整数ti,ci(1≤ti,ci≤109)ti,ci(1≤ti,ci≤109)，表示第ii个主节点上的出租车能走的最长距离和价格。主节点自己与自己不会被路相连，但两个主节点间可以有多条路。

输出

对于每组数据，输出一行，如果DH到不了球场，输出”−1””−1”（不包含引号），否则输出该组数据答案

输入样例

```
4 4
1 3
1 2 3
1 4 1
2 4 1
2 3 5
2 7
7 2
1 2
7 7
```

输出样例

```
9
```

知识点

单源最短路 优先队列+dijkstra 重建图 重边处理

AC Code

```
#include<cstdio>
#include<algorithm>
#include<vector>
#include<queue>
#include<climits>
#include<cstring>
#include<unordered_map>
#define mp make_pair
#define x first
#define y second
#define maxn 1010
#define INF 1000000000000000000LL
using namespace std;
typedef unsigned long long ull;
typedef pair<ull, ull> PLL;//v,w
/*****************************************
说白了这题就是在一个单源最短路的基础上
加一个重建图的限制
首先跑一遍最短路算出每两个主节点的最短距离
用n次单源dijkstra,floyd在n=1000的时候承受不住
然后根据t判断是否可以从一个节点到另一个
然后再拿c作为边权重建图，反复跑最短结果即可
复杂度O(n^2logn)
*****************************************/
inline void write(ull st) {
  //if (x < 0)putchar('-'), x = -x;
  if (st > 9)write(st / 10);
  putchar(st % 10 + 48);
}
inline ull read() {
  ull k = 0;// f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    //if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
```

```
  return k;// *f;
}
ull dis[maxn][maxn];
ull cost[maxn];
struct node {
  ull v, w;
  node(ull _v = 0, ull _w = 0) { v = _v; w = _w; }
  bool operator < (const node& o)const {
    return o.w < w;
  }
};
unordered_map<ull, ull>g[maxn];
ull n, m, start, finish, u, v, w, t, c;
bool occur[maxn];
inline void init() {
  memset(dis, 0, sizeof(dis));
  memset(cost, 0, sizeof(cost));
  for (int i = 0; i < maxn; ++i)g[i].clear();
}
inline void init_dis(ull *d) {
  for (int i = 1; i <= n; ++i)d[i] = INF, occur[i] = false;
}
inline void dijstra(ull s, ull* d) {
  priority_queue<node>q;
  q.push(node(s, 0));
  init_dis(d);
  d[s] = 0;
  while (!q.empty()) {
    node tmp = q.top();
    q.pop();
    v = tmp.v;
    if (occur[v])continue;
    occur[v] = true;
    for (auto& edge : g[v]) {
      ull v2 = edge.x;
      ull w = edge.y;
      if (!occur[v2] && d[v2] > w + d[v]) {
        d[v2] = w + d[v];
        q.push(node(v2, d[v2]));
      }
    }
  }
}
int main() {
  while (scanf("%llu%llu", &n, &m) != EOF) {
    init();
    start = read(), finish = read();
    for (ull i = 0; i < m; ++i) {
      u = read(), v = read(), w = read();
```

```
      if (u != v) {
        if (!g[u].count(v) || g[u][v] > w) g[u][v] = w;
        if (!g[v].count(u) || g[v][u] > w) g[v][u] = w;
      }
    }
    for (ull i = 1; i <= n; ++i)dijstra(i, dis[i]);
    for (ull i = 1; i <= n; ++i)g[i].clear();
    for (ull i = 1; i <= n; ++i) {
      t = read(), c = read();
      for (ull j = 1; j <= n; ++j) {
        if (i != j && dis[i][j] <= t)g[i][j] = c;
      }
    }
    dijstra(start, cost);
    if (cost[finish] < INF)write(cost[finish]), putchar('\n');
    else puts("-1");
  }
}
```

G DH的魔法光束

时间限制：2000ms 内存限制：204800kb

通过率：/ (%) 正确率：/ (%)

难题慎入

题目描述

DH到达了一个神奇的地方，这个地方是一个n∗mn∗m的矩阵，里边有一些魔法镜，当DH的魔法光束通过的时候，DH可以选择使用一次魔法，使光束向上下左右四个方向发散，也可以不使用魔法，这样光束就会继续沿直线行进，现在DH在左上角的格子左边向右发射魔法光束，他想要有光束从右下角的格子向右射出，问他最少需要使用多少次魔法

输入

多组输入数据，对于每组数据，第一行为两个整数n,m(2≤n,m≤1000)n,m(2≤n,m≤1000)，接下来nn行每行包含mm个字符，表示矩阵的情况，只有两种字符，”.””.”表示空地，”#””#”表示魔法镜

输出

对于每组数据，输出一行，如果光束最后不能从右下角的格子向右射出，输出”-1””-1”（不包含引号），否则输出该组数据答案

输入样例1

```
3 3
.#.
...
.#.
```

输出样例1

```
2
```

输入样例2

```
4 3
##.
...
.#.
.#.
```

输出样例2

```
2
```

知识点

二分图 最短路

```
#include<cstdio>
#include<cstring>
#include<vector>
#include<algorithm>
#include<queue>
#include<climits>
#define maxn 2333
#define INF 0x3f3f3f3f
using namespace std;
/*************************************************
嗯这题就很神奇，竟然能和二分图联系到一块
但是仔细想想其实并不奇怪
以行列建二分图，如果s[i][j]=='#'
那么i行和j列对应的点就建双向边，边权是1
然后二分图跑最短路
目标就是能到达最后一行
*************************************************/
struct node {
  int v, w;
  node(int _v = 0, int _w = 0) { v = _v, w = _w; }
```

```
  bool operator < (const node& o) const {
    return o.w < w;
  }
};
vector<node> g[maxn];
int n, m;
char s[maxn][maxn];
bool occur[maxn];
int d[maxn];
inline void init() {
  for (int i = 0; i < maxn; ++i)g[i].clear();
  memset(s, 0, sizeof(s));
}
inline void dijkstra(int s) {
  for (int i = 0; i < maxn; ++i)d[i] = INF, occur[i] = false;
  priority_queue<node>q;
  q.push(node(s, 0));
  d[s] = 0;
  while (!q.empty()) {
    node tmp = q.top();
    q.pop();
    int v = tmp.v;
    if (occur[v])continue;
    occur[v] = true;
    for (int i = 0; i < g[v].size(); ++i) {
      int v2 = g[v][i].v;
      int w = g[v][i].w;
      if (!occur[v2] && d[v2] > w + d[v]) {
        d[v2] = w + d[v];
        q.push(node(v2, d[v2]));
      }
    }
  }
}
int main() {
  while (scanf("%d%d", &n, &m) != EOF) {
    init();
    for (int i = 0; i < n; ++i)
      scanf("%s", s[i]);
    for (int i = 0; i < n; ++i) {
      for (int j = 0; j < m; ++j) {
        if (s[i][j] == '#') {
          g[i].push_back(node(j + n, 1));
          g[j + n].push_back(node(i, 1));
        }
      }
    }
    dijkstra(0);
    if (d[n - 1] != INF)printf("%d\n", d[n - 1]);
```

```
    else puts("-1");
  }
}
```

备选题

A Mdd的栈(中)

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 3 总提交人数: 5

题目描述

Mdd有一个很普通的栈，它与其他栈没有什么不同，支持入栈和出栈，不过他想再增加一个查询栈内最大值的操作，你能够帮他实现吗?

输入

一组输入数据

每组数据第一行为n，m，1≤n≤1e5，1≤m≤3e6，n，m，1≤n≤1e5，1≤m≤3e6，分别代表栈的最大容量和指令总数, 接下来共mm行, 每行一个指令, 为下面4个指令中的一个:

push x，pop，top，max。push x，pop，top，max。

输出

对于每个指令，如果操作失败，则输出**ErrorError**，否则输出每个指令对应的输出。

push x，pop：push x，pop：不输出 ， top：top：输出栈顶的值，max：max：输出栈内的最大值。

输入样例

```
10 4
pop
push 1
max
top
```

输出样例

```
Error
1
1
```

知识点

栈

AC Code

```
#pragma G++ optimize(2)
#include<cstdio>
#include<stack>
#include<cstring>
#define Max(a,b) (((a)>(b))?(a):(b))
using namespace std;
int n, m;
char instr[5];
int opnum;
stack<int>curstack, maxstack;
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    //printf("get here!\n");
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
int main() {

  n = read(), m = read();
  maxstack.push(0);
  while (m--) {
    scanf("%s", instr);
    if (!strcmp(instr,"push")) {
      opnum = read();
      if (curstack.size() == n)puts("Error");
      else {
        curstack.push(opnum);
        maxstack.push(Max(opnum, maxstack.top()));
      }
    }
```

```
    else if (!strcmp(instr, "pop")) {
      if(curstack.empty())puts("Error");
      else curstack.pop(), maxstack.pop();
    }
    else if (!strcmp(instr, "top")) {
      if (curstack.empty())puts("Error");
      else printf("%d\n",curstack.top());
    }
    else if (!strcmp(instr, "max")) {
      if (curstack.empty())puts("Error");
      else printf("%d\n", maxstack.top());
    }
  }
}
```

B DH的注册系统

时间限制: 5000 ms 内存限制: 65536 kb

总通过人数: 1 总提交人数: 2

题目描述

DH最近写了一个注册系统，当用户输入一个要注册的用户名时，系统会先检查这个用户名是否被注册过，如果没有被注册过，系统会将这个用户名插入数据库，然后给用户返回信息显示"OK"，否则系统会生成一个新的用户名插入数据库并返回给用户，新的用户名是这么生成的：在用户输入的用户名后边增加一个数，这个数是从1开始最小的满足加上这个数后的用户名在之前没有被注册过的数。现在给你用户按顺序每次输入的用户名，问每一次系统会返回什么信息

输入

多组输入数据，对于每组数据，第一行为一个整数n(1≤n≤105)n(1≤n≤105)，表示用户将进行输入的次数，接下来nn行每行一个非空字符串，表示用户这一次输入的用户名，保证输入的字符串全部由小写字母组成且长度不超过32

输出

对于每组数据，输出nn行，为每一次系统返回给用户的信息

输入样例1

```
4
abacaba
acaba
abacaba
acab
```

输出样例1

```
OK
OK
abacaba1
OK
```

输入样例2

```
6
first
first
second
second
third
third
```

输出样例2

```
OK
first1
OK
second1
OK
third1
```

知识点

哈希表

AC Code

```
#pragma G++ optimize(2)
#include<iostream>
#include<stack>
#include<string>
#include<unordered_map>
#define mp make_pair
using namespace std;
typedef long long ll;
char s[33];
string a;
unordered_map<string, ll> reglist;
int n;
int main() {
```

```
  while (~scanf("%d",&n)) {
    reglist.clear();
    while (n--) {
      scanf("%s", s);
      a = s;
      if (reglist.count(a)) {
        printf("%s%lld\n", s, reglist[a]);
        reglist[a]++;
      }
      else {
        puts("OK");
        reglist.insert(mp(a, 1));
      }
    }
  }
}
```

C DH的括号匹配

时间限制: 2000 ms 内存限制: 204800 kb

总通过人数: 1 总提交人数: 1

难题慎入

题目描述

DH现在有一个括号序列，仅包含英文小写括号，他想知道这个序列中最长的连续完全匹配括号序列有多长，有多少个这么长的连续完全匹配括号序列

输入

多组输入数据，对于每组数据，为一行，为一个非空字符串，表示DH的括号序列，字符串长度不超过106106

输出

对于每组数据，输出一行，包含两个整数，表示序列中最长连续完全匹配括号序列的长度和长度为此长度的连续完全匹配括号序列的数量，如果DH的括号序列中没有连续完全匹配括号序列，则输出”0 1””0 1”

输入样例1

```
)(((((((()())
```

输出样例1

```
6 2
```

输入样例2

```
))(
```

输出样例2

```
0 1
```

知识点

动态规划 （大概在算法课里介于中等和困难题中间吧）

AC Code

```
#pragma G++ optimize(2)
#include<cstdio>
#include<cstring>
#define maxn 1000010
char s[maxn];
int dp[maxn];
int res, cnt;
inline void buildDP() {
  memset(dp, 0, sizeof(dp));
  res = 0, cnt = 0;
  for (int i = 2; s[i] != '\0'; ++i) {
    if (s[i] == ')') {
      if (s[i - 1] == '(') dp[i] = 2 + dp[i - 2];
      else {
        int k = i - dp[i - 1] - 1;
        if (s[k] == '(') {
          dp[i] = dp[i - 1] + 2;
          if (k > 1)dp[i] += dp[k - 1];
        }
      }
    }
    if (dp[i] > res)res = dp[i], cnt = 1;
    else if (dp[i] == res) ++cnt;
  }
}
int main() {
  while (scanf("%s", s + 1) != EOF) {
```

```
    buildDP();
    if (!res)puts("0 1");
    else printf("%d %d\n", res, cnt);
  }
}
```

D Mdd玩积木(中/难)

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 1 总提交人数: 1

题目描述

mdd有一堆编好号的积木，它们的编号各不相同。mdd很无聊，他一开始把这些积木按照从小到大排成了一列，然后从中间的某个位置起，把这个位置之前所有积木都移到了最后面。现在他想知道这些积木都有哪些编号，不过他很蠢，只会从左往右数，不过这样太慢了，你能帮帮他吗?

输入

一组输入数据

对于每组数据，第一行两个个数n, m(1≤n≤1e6, 1≤m≤1e6)n, m(1≤n≤1e6, 1≤m≤1e6)，代表共有nn个积木，mm个查询。第二行nn个数, (0≤i≤231−1)(0≤i≤231−1)。接下来mm行，每行一个数kk。

输出

对于每组数据，输出mm行，如果存在编号为kk的积木，则输出 **YesYes**，否则输出 **NoNo**

输入样例

```
7 2
3 4 5 6 7 1 2
0
3
```

输出样例

```
No
Yes
```

知识点

哈希表

由于本题多组查询的限制，只能空间复杂度O(n)，单次时间复杂度O(1)

而实际上在leetcode中的限制为空间复杂度O(1)时间复杂度O(logn)，建议读者私下尝试

AC Code

```
#include<cstdio>
#include<cstring>
#include<unordered_set>
using namespace std;
unordered_set<int> a;
int n, m;
int input;
int main() {
  scanf("%d%d", &n, &m);
  for (int i = 0; i < n; ++i) {
    scanf("%d", &input);
    a.insert(input);
  }
  for (int i = 0; i < m; ++i) {
    scanf("%d", &input);
    printf("%s\n", a.count(input) ? "Yes" : "No");
  }
}
```

AC Code(二分查找Ver)

```
#include<cstdio>
#define maxn 1000010
int a[maxn];
int n, m;
int k;
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
inline int Bsearch(int k) {
  if (!n)return -1;
  if (n == 1)return a[0] == k ? 0 : -1;
  int lo = 0, hi = n - 1, mi;
```

```
  while (lo <= hi) {
    mi = (lo + hi) >> 1;
    if (k == a[mi])return mi;
    if (a[0] <= a[mi]) {
      if (a[0] <= k && k < a[mi])
        hi = mi - 1;
      else
        lo = mi + 1;
    }
    else {
      if (a[mi] < k && k <= a[n - 1])
        lo = mi + 1;
      else
        hi = mi - 1;
    }
  }
  return -1;
}
int main() {
  n = read(), m = read();
  for (int i = 0; i < n; ++i)
    a[i] = read();
  while (m--) {
    k = read();
    puts(Bsearch(k) >= 0 ? "Yes" : "No");
  }
}
```

E DH的城市安全

时间限制: 4000 ms 内存限制: 204800 kb

总通过人数: 1 总提交人数: 1

难题慎入

题目描述

DH是一个神奇的城市的市长，这个城市被山环绕，人们都住在美丽的山上，每座山上都有一个监察者，维护人们的安全，当有危险的时候，一个监察者会立即点火作为信号，引起其他监察者注意，一个监察者能看到另一个监察者发出的信号当且仅当连接他们的圆弧中间的山没有一座山是高过这两个观察者所在的山中的任意一座的。因为对于任意两座山都有两条弧将他们相连，所以当至少有一个弧满足之前那个条件时，一个监察者就可以看到另一个监察者的信号。一个城市的安全系数就是有多少对监察者满足互相可以看到对方信号，求DH的城市的安全系数

输入

多组输入数据，对于每组数据，第一行为一个整数n(3≤n≤106)n(3≤n≤106)，表示山的数量，第二行为n个整数，表示按顺时针方向山的高度，山的高度在[1,109]内

输出

对于每组数据，输出一行，为该组数据答案

输入样例

```
5
1 2 4 5 3
```

输出样例

```
7
```

知识点

环转链（stl之rotate） 单调栈 动态规划

AC Code

```
#include<cstdio>
#include<algorithm>
#include<stack>
#include<cstring>
#define debug 0
#define maxn 1000010
using namespace std;
typedef long long ll;
/*************************************************************
这题我一开始看错了...
没看到是环，以为单调栈就能解决了

所以这题的做法为:选择环中的最大值作为起点
(因为构成的答案不存在经过这个点还满足条件的,除非是相等的)
然后链后再加一个最大值
此乃环转链

数组从左到右遍历得left[i]，从右往左遍历得到right[i]
left是左边第一个大于a[i]的坐标 right就是右边第一个大于的
(这里还是要用典型的单调栈的)
same[i]是[left[i],i)中和v[i]值相等的个数
最后(i,left[i])一对(i,right[i])一对，然后再加上same即可

求left,right上面已写，求same就是碰到坐标一个相等的j时
same[i]=same[j]+1
```

```
************************************************************/
inline void write(ll x) {
  if (x < 0)putchar('-'), x = -x;
  if (x > 9)write(x / 10);
  putchar(x % 10 + 48);
}
inline int read() {
  int k = 0, f = 1;
  char c = getchar();
  while (c < '0' || c>'9') {
    if (c == '-')f = -1;
    c = getchar();
  }
  while (c >= '0' && c <= '9') {
    k = (k << 1) + (k << 3) + c - 48;
    c = getchar();
  }
  return k * f;
}
int n;
int a[maxn],left[maxn],right[maxn],same[maxn];
ll ans;
stack<int>s;
int main() {
  while (scanf("%d", &n) != EOF) {
    memset(a, 0, sizeof(a));
    memset(left, 0, sizeof(left));
    memset(right, 0, sizeof(right));
    memset(same, 0, sizeof(same));
    for (int i = 0; i < n; ++i)a[i] = read();
    rotate(a, max_element(a, a + n), a + n);
    a[n] = a[0];
    if (debug) {
      for (int i = 0; i <= n; ++i)printf("%d ", a[i]);
      putchar('\n');
      //这里旋转没问题
    }
    while (!s.empty())s.pop();
    s.push(0);
    left[0] = -1;
    for (int i = 1; i < n; ++i) {
      while (!s.empty() && a[s.top()] <= a[i]) {
        if (a[s.top()] == a[i])same[i] = same[s.top()] + 1;
        s.pop();
      }
      if (s.empty())left[i] = -1;
      else left[i] = s.top();
      s.push(i);
    }
```

```
    while (!s.empty())s.pop();
    s.push(n); right[n] = n;
    for (int i = n - 1; i >= 1; --i) {
      while (!s.empty() && a[s.top()] <= a[i]) s.pop();
      if (s.empty())right[i] = n + 1;
      else right[i] = s.top();
      s.push(i);
    }
    ans = 0;
    for (int i = 1; i < n; ++i) {
      if (left[i] != -1)++ans;
      if (right[i] != n + 1)++ans;
      if (left[i] == 0 && right[i] == n)--ans;
      ans += same[i];
    }
    write(ans), putchar('\n');

  }
}
```

F ModricWang的星际旅行（中/难）

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 0 总提交人数: 1

难题慎入

题目描述

某星系有n个星球(2≤n≤15)(2≤n≤15)，ModricWang要到各个星球去游览。

各星球之间的路程s(1≤s≤1000)(1≤s≤1000)是已知的，且由于引力作用，A星到B星与B星到A星的时间大多不同。

为了提高效率，他从母星出发，到达每个星球一次（不允许重复），最后返回母星所在的星球，假设母星所在的星球为1，他不知道选择什么样的路线才能使所飞行的时间最短。请你帮他选择一条最省时的路线。

输入

第一行一个数，星球总数

接下来n行，每行n个数，为星球间飞行耗时的邻接矩阵

输出

一行，一个整数，最短的时间的值

输入样例

```
3
0 2 1
1 0 2
2 1 0
```

输出样例

```
3
```

知识点

经典NPC问题 最短哈密顿回路 floyd+状压dp

```
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
/************************************************
dp思路:
假设从顶点s出发
令d(i,V)表示顶点i出发经过点集V中的个顶点仅一次
最后再回到s的最短路径长度

当v为空集的时候d(i,V)表示已经回来了d(i,V)=c[i][s]
当v不为空的时候,那就是子问题最优求解，只能穷举

d(i,V)=min(cik+d(k,V-(k)))
然后这里就需要状压了...

Trick:这个图就是个邻接矩阵而已,所以还要跑个floyd
************************************************/

int dp[65540][20];
int a[20][20];
int main() {
    int n;

    while (scanf("%d", &n) != EOF) {
        n--;
        for (int i = 0; i <= n; i++)
            for (int j = 0; j <= n; j++) scanf("%d", &a[i][j]);

        for (int k = 0; k <= n; k++)
            for (int i = 0; i <= n; i++)
```

```
                for (int j = 0; j <= n; j++)
                    a[i][j] = min(a[i][j], a[i][k] + a[k][j]);

        int m = (1 << n) - 1;
        for (int S = 0; S <= m; S++)
            for (int i = 1; i <= n; i++)
                if (S & (1 << (i - 1)))
                {
                    if (S == (1 << (i - 1)))
                        dp[S][i] = a[0][i];
                    else
                    {
                        dp[S][i] = (1 << 30);
                        for (int j = 1; j <= n; j++)
                        {
                            if (S & (1 << (j - 1)) && j != i)
                                dp[S][i] = min(dp[S][i],
                                    dp[S ^ (1 << (i - 1))][j] + a[j][i]);
                        }
                    }
                }

        int ans = (1 << 31) - 1;
        for (int j = 1; j <= n; j++)
        {
            ans = min(ans, dp[(1 << n) - 1][j] + a[j][0]);
        }
        printf("%d\n", ans);
    }

    return 0;
}
```

G Mdd的二叉树(简单)

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 0 总提交人数: 1

废除题目（现已被隐藏）

题目描述

春天到了，路边光秃秃的树都长出了新叶。二叉树就像现实中的树一样，又有根，又有叶。二叉树某一层的宽度是一层所有节点数，那么一棵二叉树有多宽呢?

输入

多组输入数据

每组数据共一行，为二叉树的前序遍历，长度不超过100，"#"代表空节点。

输出

对于每组数据，输出一行，二叉树的宽度。

输入样例

```
a##
#
abc###d##
```

输出样例

```
1
0
2
```

知识点

树的宽度

AC Code

```
#include<cstdio>
#include<queue>
#include<cstring>
#define maxn 110
using namespace std;
struct Tree {
  char value;
  int depth;
  Tree* lchild, * rchild;
  Tree(Tree* l = NULL, Tree* r = NULL, char a = '\0', int d = 0) {
    value = a;
    lchild = l;
    rchild = r;
    depth = d;
  }
};
char input[maxn];
int Index;
int cnt[maxn];
```

```
queue<Tree*> q;
void createBinTree(Tree*& t, int d) {
  ++Index;
  if (input[Index] == '\0')return;
  if (input[Index] != '#') {
    t = new Tree;
    t->value = input[Index];
    t->depth = d;
    createBinTree(t->lchild, d + 1);
    createBinTree(t->rchild, d + 1);
  }
  else t = NULL;
}
int width(Tree* t) {
  int ret = 0;
  while (!q.empty())q.pop();
  memset(cnt, 0, sizeof(cnt));
  if(t)q.push(t);
  while (!q.empty()) {
    Tree* tmp = q.front();
    ++cnt[tmp->depth];
    q.pop();
    if (tmp->lchild)q.push(tmp->lchild);
    if (tmp->rchild)q.push(tmp->rchild);
  }
  for (int i = 0; i < maxn; ++i)if (cnt[i] > ret)ret = cnt[i];
  return ret;
}
Tree* root;
int main() {
  while (scanf("%s", input + 1) != EOF) {
    Index = 0;
    createBinTree(root, 1);
    printf("%d\n", width(root));
  }
}
```


H Gzh的最佳损友（中等）

时间限制: 10 ms 内存限制: 65536 kb

总通过人数: 1 总提交人数: 1

难题慎入

题目描述

众所周知，北航软件学院15级的Zdh是Gzh的好儿子，所以每当Gzh赚到一笔钱，Zdh都会要求和Gzh平分。

假设Gzh赚到了m块钱，放在一个大小为m的钱袋里。Gzh和Zdh分别有一个大小为p和q的钱袋，保证p+q=m，每次操作可以从三个钱袋互相转移钱，但是由于Gzh和Zdh都很蠢，他们认不出钱的面值，只能通过袋子的大小确定每次转移钱的数目。

那么请问，当给出了m，p，q的情况下，若可以满足两人平分金钱，则输出评分所需要的操作数，否则输出“GzhIsSoHandsome”。（详见样例和Hint）

输入

多组输入数据

每行三个数为m，p，q，保证都在int范围内。

输出

对于每组数据，输出一行，为操作数或者字符串。

输入样例

```
7 3 4
4 1 3
6 3 3
```

输出样例

```
GzhIsSoHandsome
3
1
```

Hint

第二组数据，先用m装满q，这样三个袋子分别有1 0 3，在用q装满p，这样三个袋子分别有1 1 2，在把p里面的钱都转移到m里，这三三个袋子分别有2 0 2，三次完成平分。

第三组数据，直接用m装满p或q，直接完成平分。

知识点

找规律 但是找规律的方法比较困难，不如直接写一个bfs来看

AC Code

```
#pragma GCC optimize (2)
#include<stdio.h>
int a, b, c;
```

```
int gcd(int a, int b) {
  while (b ^= a ^= b ^= a %= b);
  return a;
}
int main() {
  while (~scanf("%d%d%d", &a, &b, &c)) {
    if (a & 1)puts("GzhIsSoHandsome");
    else {
      int tmp = gcd(b, c);
      if((a>>1)%tmp)puts("GzhIsSoHandsome");
      else {
        printf("%d\n", a / tmp - 1);
      }
    }
  }
}
```

TLE Code(BFS 找规律 Ver)

```
#pragma G++ optimize(2)
#include<cstdio>
#include<cstdlib>
#include<cstring>
#include<algorithm>
#include<unordered_set>
#include<set>
#include<queue>
#define mp make_pair
#define x first
#define y second
#define maxn 100010
#define print 0
using namespace std;
typedef pair<int, int> PII;
set<PII>vis;//离散标记是否走过
struct StateNode {
    int cur[3];//当前3个钱袋
    //int v[3];//原始体积
    int step;//步数
};
queue<StateNode> que;
bool paired(StateNode a, StateNode b) {
    return (a.cur[0] == b.cur[0] && a.cur[1] == b.cur[1] && a.cur[2] ==
b.cur[2]);
}
int ans;
int v[3];//原始钱袋的体积
StateNode from, to;
```

```
int main() {
    while (scanf("%d%d%d", v, v + 1, v + 2) != EOF) {
        if (v[0] & 1) {
            //第一种情况:是奇数的话，一定不可能达到
            puts("GzhIsSoHandsome");
        }
        else if (v[1] == v[2]) {
            puts("1");
        }
        else {
            //先构造起点终点
            from.cur[0] = v[0], from.cur[1] = 0, from.cur[2] = 0;
            to.cur[0] = (v[0] >> 1);
            from.step = 0;
            if (v[1] > v[2]) to.cur[1] = (v[0] >> 1); else to.cur[2] = (v[0] >>
1);
            while (!que.empty())que.pop();
            vis.clear();
            ans = -1;
            if (print)printf("target is (%d %d %d)\n", to.cur[0], to.cur[1],
to.cur[2]);
            que.push(from);
            vis.insert(mp(from.cur[1], from.cur[2]));
            //开始bfs
            while (!que.empty()) {
                StateNode u = que.front();
                que.pop();
                if (print)printf("que pop (%d %d %d)\n", u.cur[0], u.cur[1],
u.cur[2]);
                if (paired(u, to)) {
                    ans = u.step; break;
                }
                //开始倒水bfs
                for (int i = 0; i <= 2; ++i) {
                    for (int j = 0; j <= 2; ++j) {
                        if (i != j) {
                            int require = v[j] - u.cur[j];
                            //第j个杯子还差多少倒满
                            if (u.cur[i] < require)require = u.cur[i];
                            //如果第i个杯子倒不满，那就只倒这点
                            StateNode uu = u;
                            uu.cur[i] -= require;//i倒掉这么多
                            uu.cur[j] += require;
                            uu.step++;
                            PII mark = mp(uu.cur[1], uu.cur[2]);
                            if (vis.find(mark)==vis.end()) {
                                que.push(uu);
                                vis.insert(mark);
                            }
```

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

I ModricWang的小火球术（简单）

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 1 总提交人数: 1

题目描述

ModricWang是一个初级魔法师，学习的科目是火系魔法。目前他掌握了最基本的点火技能，点一次火需要耗费15点魔法值。

大魔法师Gretchen给了ModricWang N块魔法木板，让ModricWang试着点燃。某些木板间有双向的能量连接，可以耗费X点魔法值触发这个连接，使火焰能在两块木板间传递。

Gretchen是一个严格的老师，她要求ModricWang只能点一次火，但是可以触发任意多个能量连接。请问ModricWang至少需要耗费多少魔法值，才能让这些木板都被点燃?

输入

第一行两个整数，表示木板数N和连接数M，1≤N≤1000，1≤M≤10001≤N≤1000，1≤M≤1000

接下来M行，每行3个整数，X，Y和V，X和Y表示木板的两端，V表示耗费的魔法值。

输出

输出一行，点燃所有木板需要耗费的最小的魔法值

输入样例

```
3 5
3 2 29069
3 1 21397
2 1 4557
3 1 13203
3 2 18029
```

输出样例

```
17775
```

知识点

最小生成树

AC Code

同“ModricWang的布线问题”代码，答案再加个15就行

J ModricWang的益智游戏（难）

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 0 总提交人数: 1

难题慎入

数据有误（现已被隐藏）

题目描述

在3×3的棋盘上，摆有八个棋子，每个棋子上标有1至8的某一数字。棋盘中留有一个空格，空格用0来表示。空格周围的棋子可以移到空格中。

为了方便表示，将9个格子写成一个字符串，例如，123804765表示的棋盘状态为：
123
804
765

要求解的问题是：

给出一种初始布局（初始状态）和目标布局（为了使题目简单,设目标状态为123804765），找到一种最少步骤的移动方法，实现从初始布局到目标布局的转变。

输入

输入初试状态，一行九个数字，空格用0表示

输出

只有一行，该行只有一个数字，表示从初始状态到目标状态需要的最少移动次数(保证有解)

输入样例

```
283104765
```

输出样例

```
4
```

知识点

经典问题 八数码难题 状态压缩+bfs 和“ModricWang的星灵棋”性质相同 想做的同学不放去洛谷做一下

AC Code

```
#include<cstdio>
#include<cstring>
#include<unordered_map>
#include<queue>
#include<algorithm>
using namespace std;
int nx[4] = { -1,0,0,1 };
int ny[4] = { 0,-1,1,0 };
int n;
queue<int>q;
unordered_map<int, int> a;
int c[3][3];
int main() {
  scanf("%d", &n);
  q.push(n);
  a[n] = 0;
  while (!q.empty()) {
    int u = q.front();
    memset(c, 0, sizeof(c));
    int f = 0, g = 0, cur = u;
    q.pop();
    for (int i = 2; i >= 0; --i) {
      for (int j = 2; j >= 0; --j) {
        c[i][j] = cur % 10, cur /= 10;
        if (!c[i][j])f = i, g = j;//留出空
      }
    }

    for (int i = 0; i < 4; ++i) {
      int f2 = f + nx[i], g2 = g + ny[i];
      int ns = 0;
      if (f2 < 0 || f2 > 2 || g2 < 0 || g2 > 2)continue;
      swap(c[f][g], c[f2][g2]);
      for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
          ns = (ns << 1) + (ns << 3) + c[i][j];
        }
      }
      if (!a.count(ns)) {
        a[ns] = a[u] + 1;
```

```
        q.push(ns);
      }
      swap(c[f][g], c[f2][g2]);
    }
  }
  printf("%d\n", a[123804765]);
}
```