目录

[第二章：经典入门 7](#_Toc475117105)

[一、排序 7](#_Toc475117106)

[题目1202：排序 7](#_Toc475117107)

[评价 8](#_Toc475117108)

[代码 8](#_Toc475117109)

[变题：倒序输出 8](#_Toc475117110)

[题目1061：成绩排序 9](#_Toc475117111)

[评价 10](#_Toc475117112)

[代码 10](#_Toc475117113)

[Version 1.0 10](#_Toc475117114)

[Version 2.0 11](#_Toc475117115)

[练习题1：（题目1185）特殊排序 12](#_Toc475117116)

[评价 12](#_Toc475117117)

[代码 12](#_Toc475117118)

[练习题2：（题目1023）excel排序 13](#_Toc475117119)

[评价 14](#_Toc475117120)

[代码 15](#_Toc475117121)

[练习题3：（题目1054）字符串内排序 16](#_Toc475117122)

[评价 16](#_Toc475117123)

[代码 16](#_Toc475117124)

[二、日期类问题 17](#_Toc475117125)

[题目1096：日期差值 17](#_Toc475117126)

[评价 17](#_Toc475117127)

[代码 18](#_Toc475117128)

[三、hash的应用 20](#_Toc475117129)

[练习题2：（题目1088）剩下的树 20](#_Toc475117130)

[评价 20](#_Toc475117131)

[代码 20](#_Toc475117132)

[四、排版题 21](#_Toc475117133)

[题目1432：叠筐 21](#_Toc475117134)

[评价 22](#_Toc475117135)

[代码 22](#_Toc475117136)

[练习题：（题目1161）Repeater 24](#_Toc475117137)

[评价 27](#_Toc475117138)

[代码 27](#_Toc475117139)

[五、查找 29](#_Toc475117140)

[题目1069：查找学生信息 29](#_Toc475117141)

[评价 30](#_Toc475117142)

[代码 30](#_Toc475117143)

[练习题：（题目1126）打印极值点下标 32](#_Toc475117144)

[评价 33](#_Toc475117145)

[代码 33](#_Toc475117146)

[六、贪心算法 34](#_Toc475117147)

[练习题：（题目1435）迷瘴 34](#_Toc475117148)

[评价 35](#_Toc475117149)

[代码 35](#_Toc475117150)

[练习题：（题目1436）Repair the Wall 36](#_Toc475117151)

[评价 37](#_Toc475117152)

[代码 38](#_Toc475117153)

[练习题：（题目1437）To Fill or Not to Fill 38](#_Toc475117154)

[评价 40](#_Toc475117155)

[代码 40](#_Toc475117156)

[第三章：数据结构 43](#_Toc475117157)

[一、栈的应用 43](#_Toc475117158)

[题目1153：括号匹配问题 43](#_Toc475117159)

[评价 43](#_Toc475117160)

[代码 44](#_Toc475117161)

[题目1019：简单计算器 45](#_Toc475117162)

[评价 45](#_Toc475117163)

[代码 45](#_Toc475117164)

[Version1.0 45](#_Toc475117165)

[Version 2.0 48](#_Toc475117166)

[练习题：（题目1101）计算表达式 49](#_Toc475117167)

[评价 49](#_Toc475117168)

[代码 49](#_Toc475117169)

[二、哈夫曼树 50](#_Toc475117170)

[题目1172：哈夫曼树 50](#_Toc475117171)

[评价 50](#_Toc475117172)

[代码 51](#_Toc475117173)

[题目1107：搬水果 51](#_Toc475117174)

[评价 52](#_Toc475117175)

[代码 52](#_Toc475117176)

[三、二叉树 53](#_Toc475117177)

[题目1078：二叉树遍历 53](#_Toc475117178)

[评价 54](#_Toc475117179)

[代码 54](#_Toc475117180)

[练习题：（题目1113）二叉树 56](#_Toc475117181)

[评价 57](#_Toc475117182)

[代码 57](#_Toc475117183)

[Version 1.0 57](#_Toc475117184)

[Version2.0 57](#_Toc475117185)

[练习题：（题目1176）树查找 58](#_Toc475117186)

[评价 59](#_Toc475117187)

[代码 59](#_Toc475117188)

[四、二叉排序树 60](#_Toc475117189)

[题目1201：二叉排序树 60](#_Toc475117190)

[评价 60](#_Toc475117191)

[代码 61](#_Toc475117192)

[Version 1.0 61](#_Toc475117193)

[Version 2.0 63](#_Toc475117194)

[题目1009：二叉搜索树 64](#_Toc475117195)

[评价 65](#_Toc475117196)

[代码 65](#_Toc475117197)

[第四章：数学问题 68](#_Toc475117198)

[一、%运算符 68](#_Toc475117199)

[练习题：（题目1015）还是A+B 68](#_Toc475117200)

[评价 68](#_Toc475117201)

[代码 69](#_Toc475117202)

[练习题：（题目1183）守形数 69](#_Toc475117203)

[评价 70](#_Toc475117204)

[代码 70](#_Toc475117205)

[二、数位拆解 70](#_Toc475117206)

[题目1083：特殊乘法 70](#_Toc475117207)

[评价 71](#_Toc475117208)

[代码 71](#_Toc475117209)

[七、分解素因数 72](#_Toc475117210)

[题目1104：整除问题 72](#_Toc475117211)

[评价 72](#_Toc475117212)

[代码 72](#_Toc475117213)

[Version 1.0 72](#_Toc475117214)

[Version 2.0 74](#_Toc475117215)

[练习题：（题目1087）约数的个数 74](#_Toc475117216)

[评价 75](#_Toc475117217)

[代码 75](#_Toc475117218)

[八、二分求幂 76](#_Toc475117219)

[题目1441：人见人爱 A ^ B 76](#_Toc475117220)

[评价 77](#_Toc475117221)

[代码 77](#_Toc475117222)

[Version 1.0 77](#_Toc475117223)

[Version 2.0 78](#_Toc475117224)

[练习题（题目1442）A sequence of numbers 78](#_Toc475117225)

[评价 79](#_Toc475117226)

[代码 79](#_Toc475117227)

[练习题：（题目1443）Tr A 80](#_Toc475117228)

[评价 81](#_Toc475117229)

[代码 81](#_Toc475117230)

[九、高精度整数 82](#_Toc475117231)

[题目1198：a+b 82](#_Toc475117232)

[评价 83](#_Toc475117233)

[代码 83](#_Toc475117234)

[Version 1.0 83](#_Toc475117235)

[Version 2.0 83](#_Toc475117236)

[Version 3.0 84](#_Toc475117237)

[题目1076：N的阶乘 85](#_Toc475117238)

[评价 86](#_Toc475117239)

[代码 86](#_Toc475117240)

[题目1080：进制转换 87](#_Toc475117241)

[评价 88](#_Toc475117242)

[代码 88](#_Toc475117243)

[练习题：（题目1137）浮点数加法 90](#_Toc475117244)

[评价 91](#_Toc475117245)

[代码 91](#_Toc475117246)

[练习题：（题目1190）大整数排序 95](#_Toc475117247)

[评价 95](#_Toc475117248)

[代码 95](#_Toc475117249)

[练习题：（题目1208）10进制 VS 2进制 97](#_Toc475117250)

[评价 98](#_Toc475117251)

[代码 98](#_Toc475117252)

[第五章：图论 101](#_Toc475117253)

[二、并查集 101](#_Toc475117254)

[题目1012：畅通工程 101](#_Toc475117255)

[评价 102](#_Toc475117256)

[代码 102](#_Toc475117257)

[题目1444：More is better 103](#_Toc475117258)

[评价 104](#_Toc475117259)

[代码 104](#_Toc475117260)

[练习题：（题目1109）连通图 105](#_Toc475117261)

[评价 106](#_Toc475117262)

[代码 106](#_Toc475117263)

[练习题：（题目1446）Head of a Gang 107](#_Toc475117264)

[评价 108](#_Toc475117265)

[代码 108](#_Toc475117266)

[三、最小生成树 111](#_Toc475117267)

[题目1017：还是畅通工程 111](#_Toc475117268)

[评价 112](#_Toc475117269)

[代码 112](#_Toc475117270)

[Version 1.0 112](#_Toc475117271)

[Version 2.0 113](#_Toc475117272)

[题目1144：Freckles 114](#_Toc475117273)

[评价 115](#_Toc475117274)

[代码 115](#_Toc475117275)

[四、最短路径 117](#_Toc475117276)

[题目1447：最短路 117](#_Toc475117277)

[评价 118](#_Toc475117278)

[代码 118](#_Toc475117279)

[Version 1.0 118](#_Toc475117280)

[Version 2.0 119](#_Toc475117281)

[题目1008：最短路径问题 120](#_Toc475117282)

[评价 121](#_Toc475117283)

[代码 121](#_Toc475117284)

[练习题：（题目1100）最短路径 122](#_Toc475117285)

[评价 123](#_Toc475117286)

[代码 123](#_Toc475117287)

[Version 1.0 123](#_Toc475117288)

[Version 2.0 126](#_Toc475117289)

[五、拓扑排序 128](#_Toc475117290)

[题目1448：Legal or Not 128](#_Toc475117291)

[评价 129](#_Toc475117292)

[代码 129](#_Toc475117293)

[练习题：（题目1449）确定比赛名次 130](#_Toc475117294)

[评价 131](#_Toc475117295)

[代码 131](#_Toc475117296)

[练习题：（题目1450）产生冠军 132](#_Toc475117297)

[评价 133](#_Toc475117298)

[代码 133](#_Toc475117299)

[第六章：搜索 134](#_Toc475117300)

[一、枚举 134](#_Toc475117301)

[题目1045：百鸡问题 134](#_Toc475117302)

[评价 135](#_Toc475117303)

[代码 135](#_Toc475117304)

[练习题：（题目1036）Old Bill 135](#_Toc475117305)

[评价 137](#_Toc475117306)

[代码 137](#_Toc475117307)

[二、广度优先搜索（BFS） 137](#_Toc475117308)

[题目1456：胜利大逃亡 137](#_Toc475117309)

[评价 138](#_Toc475117310)

[代码 138](#_Toc475117311)

[题目1457：非常可乐 140](#_Toc475117312)

[评价 141](#_Toc475117313)

[代码 141](#_Toc475117314)

[三、递归 144](#_Toc475117315)

[题目1458：汉诺塔III 144](#_Toc475117316)

[评价 145](#_Toc475117317)

[代码 145](#_Toc475117318)

[Version 1.0 145](#_Toc475117319)

[Version 2.0 145](#_Toc475117320)

[四、递归的应用 146](#_Toc475117321)

[题目1459：Prime ring problem 146](#_Toc475117322)

[评价 147](#_Toc475117323)

[代码 147](#_Toc475117324)

[题目1460：Oil Deposit 148](#_Toc475117325)

[评价 149](#_Toc475117326)

[代码 150](#_Toc475117327)

[Version 1.0 150](#_Toc475117328)

[Version 2.0 151](#_Toc475117329)

[练习题：（题目1120）全排列 152](#_Toc475117330)

[评价 153](#_Toc475117331)

[代码 153](#_Toc475117332)

[五、深度优先搜索（DFS） 154](#_Toc475117333)

[评价 156](#_Toc475117334)

[代码 156](#_Toc475117335)

[第七章：动态规划 157](#_Toc475117336)

[一、递推求解 157](#_Toc475117337)

[题目1205：N阶楼梯上楼问题 157](#_Toc475117338)

[评价 157](#_Toc475117339)

[代码 158](#_Toc475117340)

[题目1451：不容易系列之一 158](#_Toc475117341)

[评价 159](#_Toc475117342)

[代码 159](#_Toc475117343)

[二、最长递增子序列 160](#_Toc475117344)

[题目1112：拦截导弹 160](#_Toc475117345)

[评价 160](#_Toc475117346)

[代码 160](#_Toc475117347)

[练习题：（题目1131）合唱队形 161](#_Toc475117348)

[评价 162](#_Toc475117349)

[代码 162](#_Toc475117350)

[三、最长公共子序列 163](#_Toc475117351)

[题目1042：Coincidence 163](#_Toc475117352)

[评价 163](#_Toc475117353)

[代码 164](#_Toc475117354)

[五、动态规划问题分析举例 164](#_Toc475117355)

[题目1452：搬寝室 164](#_Toc475117356)

[评价 165](#_Toc475117357)

[代码 165](#_Toc475117358)

[题目1453：Greedy Tino 166](#_Toc475117359)

[评价 167](#_Toc475117360)

[代码 167](#_Toc475117361)

[六、背包 168](#_Toc475117362)

[题目1123：采药 168](#_Toc475117363)

[评价 169](#_Toc475117364)

[代码 169](#_Toc475117365)

[Version 1.0 169](#_Toc475117366)

[Version 2.0 170](#_Toc475117367)

[题目1454：Piggy-Bank 170](#_Toc475117368)

[评价 172](#_Toc475117369)

[代码 172](#_Toc475117370)

[题目1455：珍惜现在，感恩生活 172](#_Toc475117371)

[评价 173](#_Toc475117372)

[代码 173](#_Toc475117373)

[第八章、其他技巧 174](#_Toc475117374)

[一、标准模板库（STL） 174](#_Toc475117375)

[题目1168：字符串的查找删除 174](#_Toc475117376)

[评价 175](#_Toc475117377)

[代码 175](#_Toc475117378)

[题目1450：产生冠军 176](#_Toc475117379)

[评价 177](#_Toc475117380)

[代码 177](#_Toc475117381)

[练习题：（题目1111）单词替换 178](#_Toc475117382)

[评价 179](#_Toc475117383)

[代码 179](#_Toc475117384)

[练习题：（题目1049）字符串去特定字符 179](#_Toc475117385)

[评价 180](#_Toc475117386)

[代码 180](#_Toc475117387)

# 第二章：经典入门

## 一、排序

### 题目1202：排序

**题目描述：**

    对输入的n个数进行排序并输出。

**输入：**

    输入的第一行包括一个整数n(1<=n<=100)。  
    接下来的一行包括n个整数。

**输出：**

    可能有多组测试数据，对于每组数据，将排序后的n个整数输出，每个数后面都有一个空格。  
    每组测试数据的结果占一行。

**样例输入：**

4

1 4 3 2

**样例输出：**

1 2 3 4

#### 评价

基础中的基础

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAX 100

int main(){

int n;

int num[MAX];

int i;

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

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

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

}

sort(num,num+n);

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

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

}

printf("\n");

}

return 0;

}

#### 变题：倒序输出

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAX 100

int cmp(int x,int y){

return x>y;

}

int main(){

int n;

int num[MAX];

int i;

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

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

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

}

sort(num,num+n,cmp);

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

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

}

printf("\n");

}

return 0;

}

### 题目1061：成绩排序

有N个学生的数据，将学生数据按成绩高低排序，如果成绩相同则按姓名字符的字母序排序，如果姓名的字母序也相同则按照学生的年龄排序，并输出N个学生排序后的信息。

**输入：**

    测试数据有多组，每组输入第一行有一个整数N（N<=1000），接下来的N行包括N个学生的数据。  
    每个学生的数据包括姓名（长度不超过100的字符串）、年龄（整形数）、成绩（小于等于100的正数）。

**输出：**

    将学生信息按成绩进行排序，成绩相同的则按姓名的字母序进行排序。  
    然后输出学生信息，按照如下格式：  
    姓名 年龄 成绩

**样例输入：**

3

abc 20 99

bcd 19 97

bed 20 97

**样例输出：**

bcd 19 97

bed 20 97

abc 20 99

#### 评价

结构体的排序

注意点：字符不超过100，那就定义101

#### 代码

##### Version 1.0

#include<stdio.h>

#include<string.h>

#include<algorithm>

#define MAXNAME 101

#define MAXSTU 1000

using namespace std;

typedef struct Student{

char name[MAXNAME];

int age;

int grade;

}Student;

bool cmp(Student stu1,Student stu2){

int tmp;

if(stu1.grade!=stu2.grade){

return stu1.grade<stu2.grade;//成绩从低到高

}else if((tmp=strcmp(stu1.name,stu2.name))!=0){

return tmp<0;

}else{

return stu1.age<stu2.age;

}

}

int main(){

int i;

int n;

Student stu[MAXSTU];

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

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

scanf("%s",stu[i].name);

scanf("%d",&stu[i].age);

scanf("%d",&stu[i].grade);

}

sort(stu,stu+n,cmp);

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

printf("%s %d %d\n",stu[i].name,stu[i].age,stu[i].grade);

}

}

return 0;

}

##### Version 2.0

#include<stdio.h>

#include<string.h>

#include<algorithm>

#define MAXNAME 101

#define MAXSTU 1000

using namespace std;

typedef struct Student{

char name[MAXNAME];

int age;

int grade;

bool operator <(const Student &s)const{

int tmp;

if(grade!=s.grade){

return grade<s.grade;//成绩从低到高

}else if((tmp=strcmp(name,s.name))!=0){

return tmp<0;

}else{

return age<s.age;

}

}

}Student;

int main(){

int i;

int n;

Student stu[MAXSTU];

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

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

scanf("%s",stu[i].name);

scanf("%d",&stu[i].age);

scanf("%d",&stu[i].grade);

}

sort(stu,stu+n);

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

printf("%s %d %d\n",stu[i].name,stu[i].age,stu[i].grade);

}

}

return 0;

}

### 练习题1：（题目1185）特殊排序

**题目描述：**

输入一系列整数，将其中最大的数挑出，并将剩下的数进行排序。

**输入：**

输入第一行包括1个整数N，1<=N<=1000，代表输入数据的个数。

接下来的一行有N个整数。

**输出：**

可能有多组测试数据，对于每组数据，

第一行输出一个整数，代表N个整数中的最大值，并将此值从数组中去除，将剩下的数进行排序。

第二行将排序的结果输出。

**样例输入：**

4

1 3 4 2

**样例输出：**

4

1 2 3

**提示：**

如果数组中只有一个数，当第一行将其输出后，第二行请输出"-1"。

#### 评价

太简单了，注意一下输出格式

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAX 1000

int main(){

int n;

int num[MAX];

int i;

int max;

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

if(n==1){

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

printf("%d\n-1\n",num[0]);

continue;

}

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

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

}

sort(num,num+n);

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

for(i=0;i<n-2;i++){

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

}

printf("%d\n",num[n-2]);

}

return 0;

}

### 练习题2：（题目1023）excel排序

**题目描述：**

    Excel可以对一组纪录按任意指定列排序。现请你编写程序实现类似功能。

    对每个测试用例，首先输出1行“Case i:”，其中 i 是测试用例的编号（从1开始）。随后在 N 行中输出按要求排序后的结果，即：当 C=1 时，按学号递增排序；当 C=2时，按姓名的非递减字典序排序；当 C=3   
时，按成绩的非递减排序。当若干学生具有相同姓名或者相同成绩时，则按他们的学号递增排序。

**输入：**

    测试输入包含若干测试用例。每个测试用例的第1行包含两个整数 N (N<=100000) 和 C，其中 N 是纪录的条数，C 是指定排序的列号。以下有N行，每行包含一条学生纪录。每条学生纪录由学号（6位数字，同组测试中没有重复的学号）、姓名（不超过8位且不包含空格的字符串）、成绩（闭区间[0, 100]内的整数）组成，每个项目间用1个空格隔开。当读到 N=0 时，全部输入结束，相应的结果不要输出。

**输出：**

    对每个测试用例，首先输出1行“Case i:”，其中 i 是测试用例的编号（从1开始）。随后在 N 行中输出按要求排序后的结果，即：当 C=1 时，按学号递增排序；当 C=2时，按姓名的非递减字典序排序；当 C=3   
时，按成绩的非递减排序。当若干学生具有相同姓名或者相同成绩时，则按他们的学号递增排序。

**样例输入：**

3 1

000007 James 85

000010 Amy 90

000001 Zoe 60

4 2

000007 James 85

000010 Amy 90

000001 Zoe 60

000002 James 98

4 3

000007 James 85

000010 Amy 90

000001 Zoe 60

000002 James 90

0 0

**样例输出：**

Case 1:

000001 Zoe 60

000007 James 85

000010 Amy 90

Case 2:

000010 Amy 90

000002 James 98

000007 James 85

000001 Zoe 60

Case 3:

000001 Zoe 60

000007 James 85

000002 James 90

000010 Amy 90

#### 评价

挺简单的，sort运用的似乎很熟练。实例最后输入0 0，但我最后只输入0，题目意思应该是这样，然后我就accepted了

#### 代码

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

typedef struct Student{

int No;

char name[9];

int score;

}Stu;

#define MAX 100000

Student stu[MAX];

int C;

bool cmp(Student s1,Student s2){

int tmp;

if(C==1){

return s1.No<s2.No;

}else if(C==2){

if((tmp=strcmp(s1.name,s2.name))!=0){

return tmp<0;

}else{

return s1.No<s2.No;

}

}else if(C==3){

if(s1.score!=s2.score)

return s1.score<s2.score;

else

return s1.No<s2.No;

}

}

int main(){

int N;

int i=1;

while(scanf("%d",&N)!=EOF&&N!=0){

//input

scanf("%d",&C);

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

scanf("%d",&stu[i].No);

scanf("%s",stu[i].name);

scanf("%d",&stu[i].score);

}

//dealing

sort(stu,stu+N,cmp);

//output

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

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

printf("%06d %s %d\n",stu[i].No,stu[i].name,stu[i].score);

}

}

return 0;

}

### 练习题3：（题目1054）字符串内排序

**题目描述：**

输入一个字符串，长度小于等于200，然后将输出按字符顺序升序排序后的字符串。

**输入：**

测试数据有多组，输入字符串。

**输出：**

对于每组输入,输出处理后的结果。

**样例输入：**

bacd

**样例输出：**

abcd

#### 评价

做一做，就会发现，挺简单的。还是每道题都做过去吧

#### 代码

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

#define MAX 201

int main(){

char str[MAX];

int len;

while(gets(str)!=NULL){

len=strlen(str);

sort(str,str+len);

puts(str);

}

return 0;

}

## 二、日期类问题

### 题目1096：日期差值

**题目描述：**

有两个日期，求两个日期之间的天数，如果两个日期是连续的我们规定他们之间的天数为两天

**输入：**

有多组数据，每组数据有两行，分别表示两个日期，形式为YYYYMMDD

**输出：**

每组数据输出一行，即日期差值

**样例输入：**

20110412

20110422

**样例输出：**

11

#### 评价

书上说要空间换时间，我先用了自己方法

scanf("%4d%2d%2c",&a,&b,&c)发现这运用还是挺灵活的

#### 代码

#include<stdio.h>

typedef struct Date{

int year;

int month;

int day;

};

int buf[13][2]=//0为平年，1为闰年

{

0,0,

31,31,

28,29,

31,31,

30,30,

31,31,

30,30,

31,31,

31,31,

30,30,

31,31,

30,30,

31,31

};

int countCurrentYearDays(Date date){

int countDays=0;

if(date.month>=3){

if(date.year%4==0&&date.year%100!=0||date.year%400==0){

for(int i=1;i<date.month;i++){

countDays+=buf[i][1];

}

}else{

for(int i=1;i<date.month;i++){

countDays+=buf[i][0];

}

}

countDays+=date.day;

return countDays;

}else if(date.month==2){

return 31+date.day;

}else{

return date.day;

}

}

int count(Date date1,Date date2){

int gap;

int countYear[2]={0};//0为平年，1为闰年

int countDays[2]={0};//0为初年，1为末年

for(int i=date1.year;i<date2.year;i++){//减去初年天数，加上末年天数

if(i%4==0&&i%100!=0||i%400==0){

countYear[1]++;

}else{

countYear[0]++;

}

}

countDays[0]=countCurrentYearDays(date1);//初年

countDays[1]=countCurrentYearDays(date2);//末年

gap=countYear[0]\*365+countYear[1]\*366+countDays[1]-countDays[0]+1;

return gap;

}

int main(){

int a1,a2;

int gap;

Date date1,date2;

while(scanf("%d%d",&a1,&a2)!=EOF){

date1.year=a1/10000;

date2.year=a2/10000;

date1.month=a1/100%100;

date2.month=a2/100%100;

date1.day=a1%100;

date2.day=a2%100;

int tmp=a1-a2;

if(tmp<0){

gap=count(date1,date2);

}else{

gap=count(date2,date1);

}

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

}

return 0;

}

## 三、hash的应用

### 练习题2：（题目1088）剩下的树

**题目描述：**

    有一个长度为整数L(1<=L<=10000)的马路，可以想象成数轴上长度为L的一个线段，起点是坐标原点，在每个整数坐标点有一棵树，即在0,1,2，...，L共L+1个位置上有L+1棵树。  
    现在要移走一些树，移走的树的区间用一对数字表示，如 100 200表示移走从100到200之间（包括端点）所有的树。  
    可能有M(1<=M<=100)个区间，区间之间可能有重叠。现在要求移走所有区间的树之后剩下的树的个数。

**输入：**

    两个整数L(1<=L<=10000)和M(1<=M<=100)。  
    接下来有M组整数，每组有一对数字。

**输出：**

    可能有多组输入数据，对于每组输入数据，输出一个数，表示移走所有区间的树之后剩下的树的个数。

**样例输入：**

500 3

100 200

150 300

470 471

**样例输出：**

298

#### 评价

很简单的hash

#### 代码

#include<stdio.h>

#define MAXM 100

#define MAXTREE 10001

int main(){

int L,M;

int tree[MAXM][2];

int hash[MAXTREE];

int count;

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

count=0;

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

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

}

//dealing

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

hash[i]=1;

}

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

for(int j=tree[i][0];j<=tree[i][1];j++){

hash[j]=0;

}

}

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

if(hash[i]==1){

count++;

}

}

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

}

return 0;

}

## 四、排版题

### 题目1432：叠筐

**题目描述：**

把一个个大小差一圈的筐叠上去，使得从上往下看时，边筐花色交错。这个工作现在要让计算机来完成，得看你的了。

**输入：**

输入是一个个的三元组，分别是，外筐尺寸n（n为满足0<n<80的奇整数），中心花色字符，外筐花色字符，后二者都为ASCII可见字符；

**输出：**

输出叠在一起的筐图案，中心花色与外筐花色字符从内层起交错相叠，多筐相叠时，最外筐的角总是被打磨掉。叠筐与叠筐之间应有一行间隔。

**样例输入：**

11 B A

5 @ W

**样例输出：**

AAAAAAAAA

ABBBBBBBBBA

ABAAAAAAABA

ABABBBBBABA

ABABAAABABA

ABABABABABA

ABABAAABABA

ABABBBBBABA

ABAAAAAAABA

ABBBBBBBBBA

AAAAAAAAA

@@@

@WWW@

@W@W@

@WWW@

@@@

#### 评价

1. 字符输入算是有点熟悉了，下次熟悉一下字符串输入

scanf("%d %c %c",&n,&chars[0],&chars[1])，这种输入方式，嗯

scanf("%d%c%c",&n,&chars[0],&chars[1])对比吧

1. 叠筐与叠筐之间应有一行间隔，这个的表达方式注意一下
2. 简单的情况先解决，还是养成这个习惯吧。或者有简单的，一定要先注释一下情况
3. 书上的示例不错，值得一看p28

#### 代码

#include<stdio.h>

#define MAX 80

int main(){

int n;

char basket[MAX][MAX];

int half;

int chars[2];

int is0or1;

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

getchar();

chars[0]=getchar();

getchar();

chars[1]=getchar();

getchar();

half=n/2+1;

if(n==1){

printf("%c\n\n",chars[0]);

continue;

}

//先解决1/4

//first line

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

if(i==0){

basket[0][i]=' ';

}else{

if((n+1)/2%2==1){

basket[0][i]=chars[0];

is0or1=0;

}else{

basket[0][i]=chars[1];

is0or1=1;

}

}

}

//others

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

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

if(j<i){

if(j%2==0)

basket[i][j]=chars[is0or1];

else

basket[i][j]=chars[!is0or1];

}else{

if(i%2==0)

basket[i][j]=chars[is0or1];

else

basket[i][j]=chars[!is0or1];

}

}

}

//解决3/4，第二象限

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

for(int j=half;j<n;j++)

basket[i][j]=basket[i][n-j-1];

}

//解决3/4，第三象限

for(int i=half;i<n;i++){

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

basket[i][j]=basket[n-i-1][j];

}

//解决3/4，第四象限

for(int i=half;i<n;i++){

for(int j=half;j<n;j++)

basket[i][j]=basket[n-i-1][n-j-1];

}

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

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

printf("%c",basket[i][j]);

}

printf("\n");

}

printf("\n");

}

}

### 练习题：（题目1161）Repeater

**题目描述：**

Harmony is indispensible in our daily life and no one can live without it----may be Facer is the only exception. One day it is rumored that repeat painting will create harmony and then hundreds of people started their endless drawing. Their paintings were based on a small template and a simple method of duplicating. Though Facer can easily imagine the style of the whole picture, but he cannot find the essential harmony. Now you need to help Facer by showing the picture on computer.  
You will be given a template containing only one kind of character and spaces, and the template shows how the endless picture is created----use the characters as basic elements and put them in the right position to form a bigger template, and then repeat and repeat doing that. Here is an example.

# #  
 #      <-template  
# #  
So the Level 1 picture will be

# #  
 #  
# #  
Level 2 picture will be

# #     # #  
 #         #  
# #     # #  
     # #     
      #      
     # #     
# #    # #  
 #        #   
# #    # #

**输入：**

The input contains multiple test cases.  
The first line of each case is an integer N, representing the size of the template is N\*N (N could only be 3, 4 or 5).  
Next N lines describe the template.  
The following line contains an integer Q, which is the Scale Level of the picture.  
Input is ended with a case of N=0.  
It is guaranteed that the size of one picture will not exceed 3000\*3000.

**输出：**

For each test case, just print the Level Q picture by using the given template.

**样例输入：**

3

# #

#

# #

1

3

# #

#

# #

3

4

OO

O O

O O

OO

2

0

**样例输出：**

# #

#

# #

# # # # # # # #

# # # #

# # # # # # # #

# # # #

# #

# # # #

# # # # # # # #

# # # #

# # # # # # # #

# # # #

# #

# # # #

# #

#

# #

# # # #

# #

# # # #

# # # # # # # #

# # # #

# # # # # # # #

# # # #

# #

# # # #

# # # # # # # #

# # # #

# # # # # # # #

OO OO

O OO O

O OO O

OO OO

OO OO

O O O O

O O O O

OO OO

OO OO

O O O O

O O O O

OO OO

OO OO

O OO O

O OO O

OO OO

#### 评价

中规中矩用递归的思想

|  |  |  |  |
| --- | --- | --- | --- |
| Accepted | 9896 *kb* | 180 *ms* | 100/100 |

有大牛 2712kb 50ms

#### 代码

#include<stdio.h>

#include<math.h>

#define MAX\_PICTURE\_BORDER 3000

char picture[MAX\_PICTURE\_BORDER][MAX\_PICTURE\_BORDER];

char tmplate[5][5];

int N;//only be 3,4,5

int Q;

void fillBlank(int x1,int y1,int x2,int y2){

for(int i=x1;i<x2;i++){

for(int j=y1;j<y2;j++){

picture[i][j]=' ';

}

}

}

void fillTmplate(int x1,int y1,int x2,int y2){

for(int i=x1;i<x2;i++){

for(int j=y1;j<y2;j++){

picture[i][j]=tmplate[i-x1][j-y1];

}

}

}

void paintPicture(int N,int Q,int x1,int y1,int x2,int y2){

int partLen=pow((double)N,Q-1);

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

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

if(tmplate[i][j]==' '){

fillBlank(x1+i\*partLen,y1+j\*partLen,x1+(i+1)\*partLen,y1+(j+1)\*partLen);

}else{

if(Q==2){

fillTmplate(x1+i\*partLen,y1+j\*partLen,x1+(i+1)\*partLen,y1+(j+1)\*partLen);

}else{

paintPicture(N,Q-1,x1+i\*partLen,y1+j\*partLen,x1+(i+1)\*partLen,y1+(j+1)\*partLen);

}

}

}

}

}

void printPicture(){

int len=pow((double)N,Q);

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

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

printf("%c",picture[i][j]);

}

printf("\n");

}

}

int main(){

while(scanf("%d",&N)!=EOF&&N!=0){

getchar();

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

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

scanf("%c",&tmplate[i][j]);

}

getchar();

}

scanf("%d",&Q);

if(Q==1){

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

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

printf("%c",tmplate[i][j]);

}

printf("\n");

}

}else{

paintPicture(N,Q,0,0,pow((double)N,Q),pow((double)N,Q));

printPicture();

}

}

return 0;

}

## 五、查找

### 题目1069：查找学生信息

**题目描述：**

 输入N个学生的信息，然后进行查询。

**输入：**

 输入的第一行为N，即学生的个数(N<=1000)

接下来的N行包括N个学生的信息，信息格式如下：

01 李江 男 21

02 刘唐 男 23

03 张军 男 19

04 王娜 女 19

然后输入一个M(M<=10000),接下来会有M行，代表M次查询，每行输入一个学号，格式如下：

02

03

01

04

**输出：**

 输出M行，每行包括一个对应于查询的学生的信息。

如果没有对应的学生信息，则输出“No Answer!”

**样例输入：**

4

01 李江 男 21

02 刘唐 男 23

03 张军 男 19

04 王娜 女 19

5

02

03

01

04

03

**样例输出：**

02 刘唐 男 23

03 张军 男 19

01 李江 男 21

04 王娜 女 19

03 张军 男 19

#### 评价

这题好醉，name[20]不行，网上查了别人50也不行，非得开到100，哪这么长的名字。。。（要开的大一点。。。）

1172kb 20ms，好像跟别人的比我很好诶

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAXSTU 1000

#define MAXM 10000

typedef struct Student{

int No;

char name[100];

char sex[3];

int age;

bool operator < (const Student &s)const{

return No<s.No;

}

};

Student stu[MAXSTU];

int N;

int M;

int M\_num[MAXM];

int halfSearch(int No,int start,int end){

int half=(start+end)/2;

if(start<end){

if(No<stu[half].No){

return halfSearch(No,start,half-1);

}else if(No>stu[half].No){

return halfSearch(No,half+1,end);

}else{

return half;

}

}else{

if(No!=stu[half].No){

return -1;

}else{

return half;

}

}

}

int main(){

int tmp;

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

if(N==0)

continue;

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

scanf("%d %s %s %d",&stu[i].No,stu[i].name,stu[i].sex,&stu[i].age);

}

scanf("%d",&M);

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

scanf("%d",&M\_num[i]);

}

//sort

sort(stu,stu+N);

//search

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

tmp=halfSearch(M\_num[i],0,N-1);

if(tmp!=-1){

printf("%02d %s %s %d\n",stu[tmp].No,stu[tmp].name,stu[tmp].sex,stu[tmp].age);//a while

}else

printf("No Answer!\n");

}

}

return 0;

}

### 练习题：（题目1126）打印极值点下标

**题目描述：**

在一个整数数组上，对于下标为i的整数，如果它大于所有它相邻的整数，  
或者小于所有它相邻的整数，则称为该整数为一个极值点，极值点的下标就是i。

**输入：**

每个案例的输入如下：

有2×n+1行输入：第一行是要处理的数组的个数n；  
对其余2×n行，第一行是此数组的元素个数k(4<k<80)，第二行是k个整数，每两个整数之间用空格分隔。

**输出：**

每个案例输出为n行：每行对应于相应数组的所有极值点下标值，下标值之间用空格分隔。

**样例输入：**

3

10

10 12 12 11 11 12 23 24 12 12

15

12 12 122 112 222 211 222 221 76 36 31 234 256 76 76

15

12 14 122 112 222 222 222 221 76 36 31 234 256 76 73

**样例输出：**

0 7

2 3 4 5 6 10 12

0 2 3 10 12 14

#### 评价

根本不需要脑子

Scanf()!=eof可以，多重输入

#### 代码

#include<stdio.h>

#define MAX 80

#define MAXN 1000

int main(){

int n;

int k[MAXN];

int arrays[MAXN][MAX];//源数组，从0开始

int result[MAXN][MAX];//结果数组，从1开始，0下标储存结果总数

int max,min;

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

//input

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

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

for(int j=0;j<k[i];j++){

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

}

}

//dealing

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

int tmp=1;

if(arrays[i][0]!=arrays[i][1]){

result[i][tmp++]=0;

}

for(int j=1;j<k[i]-1;j++){

if(arrays[i][j-1]>arrays[i][j]&&arrays[i][j+1]>arrays[i][j])

result[i][tmp++]=j;

else if(arrays[i][j-1]<arrays[i][j]&&arrays[i][j+1]<arrays[i][j])

result[i][tmp++]=j;

}

if(arrays[i][k[i]-2]!=arrays[i][k[i]-1]){

result[i][tmp++]=k[i]-1;

}

result[i][0]=tmp-1;

}

//output

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

printf("%d",result[i][1]);

for(int j=2;j<=result[i][0];j++){

printf(" %d",result[i][j]);

}

printf("\n");

}

}

return 0;

}

## 六、贪心算法

### 练习题：（题目1435）迷瘴

**题目描述：**

通过悬崖的yifenfei，又面临着幽谷的考验——  
幽谷周围瘴气弥漫，静的可怕，隐约可见地上堆满了骷髅。由于此处长年不见天日，导致空气中布满了毒素，一旦吸入体内，便会全身溃烂而死。  
幸好yifenfei早有防备，提前备好了解药材料（各种浓度的万能药水）。现在只需按照配置成不同比例的浓度。  
现已知yifenfei随身携带有n种浓度的万能药水，体积V都相同，浓度则分别为Pi%。并且知道，针对当时幽谷的瘴气情况，只需选择部分或者全部的万能药水，然后配置出浓度不大于 W%的药水即可解毒。  
现在的问题是：如何配置此药，能得到最大体积的当前可用的解药呢？  
特别说明：由于幽谷内设备的限制,只允许把一种已有的药全部混入另一种之中（即：不能出现对一种药只取它的一部分这样的操作）。

**输入：**

输入数据的第一行是一个整数C，表示测试数据的组数；  
每组测试数据包含2行，首先一行给出三个正整数n,V,W(1<=n,V,W<=100)；  
接着一行是n个整数，表示n种药水的浓度Pi%(1<=Pi<=100)。

**输出：**

对于每组测试数据，请输出一个整数和一个浮点数；  
其中整数表示解药的最大体积，浮点数表示解药的浓度(四舍五入保留2位小数)；  
如果不能配出满足要求的的解药，则请输出0 0.00。

**样例输入：**

3

1 100 10

100

2 100 24

20 30

3 100 24

20 20 30

**样例输出：**

0 0.00

100 0.20

300 0.23

#### 评价

做这个没有亏，起码让我知道了还要看四舍五入。（具体一点，转化成整数的时候是直接截断，转化成浮点数是就近舍入。就近，加入4.5，舍入到最近的偶数）

我现在看题真是不太小心啊。

有点不错，我看到很多accepted的都是20ms，而我是0ms，不错，也不知道那些人都是怎么写的

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAXC 100

#define MAXN 100

int C;

int n[MAXC],V[MAXC],w[MAXC];

int w\_level[MAXC][MAXN];

int main(){

int count\_V=0;

int count\_w=0;

double avg\_w=0;

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

//input

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

scanf("%d%d%d",&n[i],&V[i],&w[i]);

for(int j=0;j<n[i];j++){

scanf("%d",&w\_level[i][j]);

}

}

//dealing

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

double tmp;

count\_V=0;

count\_w=0;

avg\_w=0;

sort(w\_level[i],w\_level[i]+n[i]);

for(int j=0;j<n[i];j++){

tmp=(double)(count\_w+w\_level[i][j])/(count\_V+1);

if(tmp<=w[i]){

count\_V++;

count\_w+=w\_level[i][j];

avg\_w=tmp;

}else{

printf("%d %.2f\n",count\_V\*V[i],(int)(avg\_w+0.5)>(int)avg\_w?(avg\_w+0.5)/100:avg\_w/100);

break;

}

}

if(count\_V==n[i])//全都可以

printf("%d %.2f\n",count\_V\*V[i],(int)(avg\_w+0.5)>(int)avg\_w?(avg\_w+0.5)/100:avg\_w/100);

}

}

return 0;

}

### 练习题：（题目1436）Repair the Wall

**题目描述：**

Long time ago , Kitty lived in a small village. The air was fresh and the scenery was very beautiful. The only thing that troubled her is the typhoon.

When the typhoon came, everything is terrible. It kept blowing and raining for a long time. And what made the situation worse was that all of Kitty's walls were made of wood.

One day, Kitty found that there was a crack in the wall. The shape of the crack is   
a rectangle with the size of 1×L (in inch). Luckly Kitty got N blocks and a saw(锯子) from her neighbors.  
The shape of the blocks were rectangle too, and the width of all blocks were 1 inch. So, with the help of saw, Kitty could cut down some of the blocks(of course she could use it directly without cutting) and put them in the crack, and the wall may be repaired perfectly, without any gap.

Now, Kitty knew the size of each blocks, and wanted to use as fewer as possible of the blocks to repair the wall, could you help her ?

**输入：**

The problem contains many test cases, please process to the end of file( EOF ).  
Each test case contains two lines.  
In the first line, there are two integers L(0<L<1000000000) and N(0<=N<600) which  
mentioned above.  
In the second line, there are N positive integers. The ith integer Ai(0<Ai<1000000000 ) means that the ith block has the size of 1×Ai (in inch).

**输出：**

For each test case , print an integer which represents the minimal number of blocks are needed.  
If Kitty could not repair the wall, just print "impossible" instead.

**样例输入：**

5 3

3 2 1

5 2

2 1

**样例输出：**

2

impossible

#### 评价

Int 的范围在2\*10^9范围内都没问题

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAXN 600

bool cmp(int x,int y){

return x>y;

}

int main(){

int L,N;

int arr[MAXN];

int cnt\_L;

int cnt\_num;

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

cnt\_L=0;

cnt\_num=0;

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

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

}

//dealing

sort(arr,arr+N,cmp);

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

if(cnt\_L+arr[i]<L){

cnt\_L+=arr[i];

cnt\_num++;

}else{

cnt\_num++;

cnt\_L+=arr[i];

printf("%d\n",cnt\_num);

break;

}

}

if(cnt\_num==N&&cnt\_L<L){

printf("impossible\n");

}

}

return 0;

}

### 练习题：（题目1437）To Fill or Not to Fill

**题目描述：**

With highways available, driving a car from Hangzhou to any other city is easy. But since the tank capacity of a car is limited, we have to find gas stations on the way from time to time. Different gas station may give different price. You are asked to carefully design the cheapest route to go.

**输入：**

For each case, the first line contains 4 positive numbers: Cmax (<= 100), the maximum capacity of the tank; D (<=30000), the distance between Hangzhou and the destination city; Davg (<=20), the average distance per unit gas that the car can run; and N (<= 500), the total number of gas stations. Then N lines follow, each contains a pair of non-negative numbers: Pi, the unit gas price, and Di (<=D), the distance between this station and Hangzhou, for i=1,...N. All the numbers in a line are separated by a space.

**输出：**

For each test case, print the cheapest price in a line, accurate up to 2 decimal places. It is assumed that the tank is empty at the beginning. If it is impossible to reach the destination, print "The maximum travel distance = X" where X is the maximum possible distance the car can run, accurate up to 2 decimal places.

**样例输入：**

50 1300 12 8

6.00 1250

7.00 600

7.00 150

7.10 0

7.20 200

7.50 400

7.30 1000

6.85 300

50 1300 12 2

7.10 0

7.00 600

**样例输出：**

749.17

The maximum travel distance = 1200.00

#### 评价

五颗星

做了3遍，终于对了

1. 发现visual stdio的命令行可以粘贴，太感动了
2. 先要把思路在纸上写清楚再做题
3. 此题把终点也算作一站，但是油价要无穷大，这样能简单一点

思路：

（主）

能否到达下一站

否，加满油，能走多远是多远

能，有更便宜的吗

有，就是下一站（或一样便宜）

否，终点在里程范围内吗

否，加满油，里程内相对最便宜的站为下一站

是，加油到刚够终点，终点就是下一站

（辅）

计算sumPrice, sumDistance, gasExist

把主思路完成后再写剩下的，也就是，先确定怎么走，再计算。不要边想怎么走边计算

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

#define MAXN 502

struct Station{

double p;

double d;

};

bool cmp(Station s1,Station s2){return s1.d<s2.d;}

void cntPrice\_Distance\_GasExist(int i,int j,bool isFull);

int Cmax,D,Davg,N;

Station sta[MAXN];

int step;

double sumDistance;

double sumPrice;

int tmpCheapest;

bool ifCheaper;

double gasExist;

int main(){

while(scanf("%d%d%d%d",&Cmax,&D,&Davg,&N)!=EOF){

step=Cmax\*Davg;

sumDistance=0;

sumPrice=0;

gasExist=0;

//input

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

scanf("%lf%lf",&sta[i].p,&sta[i].d);

}

sta[N].p=1000000000;sta[N].d=D;//设置终点

//dealing

sort(sta,sta+N+1,cmp);

//特殊情况

if(sta[0].d!=0.0){printf("The maximum travel distance = 0.00\n");continue;}//起点没有加油站

if(D==0){printf("0.00\n");continue;}//终点为0

//一般情况

for(int i=0;i<N;){

tmpCheapest=-1;

ifCheaper=false;

if(sta[i+1].d-sta[i].d>step){//不能到达下一站

printf("The maximum travel distance = %.2lf\n",sumDistance+step);

break;

}

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

if(sta[j].d-sta[i].d>step){

break;

}

if(sta[j].p<=sta[i].p){

//next station

ifCheaper=true;

cntPrice\_Distance\_GasExist(i,j,false);

i=j;

break;

}else{

if(tmpCheapest==-1)

tmpCheapest=j;

else if(sta[tmpCheapest].p>sta[j].p){

tmpCheapest=j;

}

}

}

if(ifCheaper==false&&tmpCheapest!=-1){

//next station

if(sta[N].d-sta[i].d>step){

cntPrice\_Distance\_GasExist(i,tmpCheapest,true);

i=tmpCheapest;

}else{

cntPrice\_Distance\_GasExist(i,N,false);

i=N;

}

}

if(i==N){

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

break;

}

}

}

return 0;

}

void cntPrice\_Distance\_GasExist(int i,int j,bool isFull){

double iToJGas=(sta[j].d-sta[i].d)/Davg;

if(isFull==false){

if(iToJGas-gasExist>=0){

sumPrice+=(iToJGas-gasExist)\*sta[i].p;

gasExist=0;

}else{

gasExist-=iToJGas;

}

}else{

sumPrice+=(Cmax-gasExist)\*sta[i].p;

gasExist=Cmax-iToJGas;

}

sumDistance=sta[j].d;

//printf("GasExist is %.2lf\nsumPrice is %.2lf\nsumDistance is%.2lf\n\n",gasExist,sumPrice,sumDistance);

}

# 第三章：数据结构

## 一、栈的应用

### 题目1153：括号匹配问题

**题目描述：**

    在某个字符串（长度不超过100）中有左括号、右括号和大小写字母；规定（与常见的算数式子一样）任何一个左括号都从内到外与在它右边且距离最近的右括号匹配。写一个程序，找到无法匹配的左括号和右括号，输出原来字符串，并在下一行标出不能匹配的括号。不能匹配的左括号用"$"标注,不能匹配的右括号用"?"标注.

**输入：**

    输入包括多组数据，每组数据一行，包含一个字符串，只包含左右括号和大小写字母，字符串长度不超过100。  
    注意：cin.getline(str,100)最多只能输入99个字符！

**输出：**

    对每组输出数据，输出两行，第一行包含原始输入字符，第二行由"$","?"和空格组成，"$"和"?"表示与之对应的左括号和右括号不能匹配。

**样例输入：**

)(rttyy())sss)(

**样例输出：**

)(rttyy())sss)(

? ?$

#### 评价

熟悉了stack标准函数库的用法

Puts和gets都是一行的

#### 代码

#include<stdio.h>

#include<stack>

#include<string.h>

#define MAX 101

using namespace std;

int main(){

stack<int> s;

char str[MAX];

char strCpy[MAX];

int len;

while(gets(str)!=NULL){

len=strlen(str);

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

strCpy[i]=' ';

}

strCpy[len]='\0';

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

if(str[i]=='('){

s.push(i);

}else if(str[i]==')'){

if(s.empty()==false){

s.pop();

}else{

strCpy[i]='?';

}

}

}

while(s.empty()!=true){

strCpy[s.top()]='$';

s.pop();

}

puts(str);

puts(strCpy);

}

return 0;

}

### 题目1019：简单计算器

**题目描述：**

    读入一个只包含 +, -, \*, / 的非负整数计算表达式，计算该表达式的值。

**输入：**

    测试输入包含若干测试用例，每个测试用例占一行，每行不超过200个字符，整数和运算符之间用一个空格分隔。没有非法表达式。当一行中只有0时输入结束，相应的结果不要输出。

**输出：**

    对每个测试用例输出1行，即该表达式的值，精确到小数点后2位。

**样例输入：**

1 + 2

4 + 2 \* 5 - 7 / 11

0

**样例输出：**

3.00

13.36

#### 评价

1. 直接输入字符串，然后把字符串变成字符数组，再根据需要转化为double类型数字和char类型运算符号（其中把字符串变成字符数组，有点麻烦的）
2. 网上有在输入上动手脚的（算是神代码了，很简洁）
3. 书上的是通用的该类型的题，此题只有加减乘除，所以看起来复杂，所以书上的值得一看

#### 代码

##### Version1.0

#include<stdio.h>

#include<stack>

#include<string.h>

#define MAX 201

using namespace std;

stack<char> op;

stack<double> num;

char str[MAX];

int len;

char arr[MAX][MAX];

int arr\_i;

int tmp;

char tmpc;

double tmp1,tmp2;

void dividToArr(){

tmp=-1;

arr\_i=0;

for(int i=0;i<len+1;i++){

if(str[i]!=' '&&str[i]!='\0'){

if(str[i]<='9'&&str[i]>='0'){

if(tmp==-1)tmp=i;

}else{

arr[arr\_i][0]=str[i];

arr[arr\_i++][1]='\0';

i++;//因为字母下一个肯定是空格，为避免与整数后的空格混淆

}

}else{

for(int j=tmp;j<i;j++){

arr[arr\_i][j-tmp]=str[j];

}

arr[arr\_i++][i-tmp]='\0';

tmp=-1;

}

}

}

double calculate(double x,double y,char a){

if(a=='+'){

return x+y;

}else if(a=='-'){

return x-y;

}else if(a=='\*'){

return x\*y;

}else if(a=='/'){

return x/y;

}

return -1;

}

double charsToDouble(char a[]){

int len=strlen(a);

double tmp=0;

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

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

}

return tmp;

}

int main(){

while(gets(str)!=NULL){

if(strcmp(str,"0")==0)

break;

len=strlen(str);

dividToArr();

for(int i=0;i<arr\_i;i++){

if(i%2==0){//是数字

num.push(charsToDouble(arr[i]));

if(op.empty()==false){//如果符号栈不是空

while(op.empty()==false){

if((op.top()=='\*'||op.top()=='/')||(arr[i+1][0]=='+'||arr[i+1][0]=='-')||i+1==arr\_i){

tmpc=op.top();op.pop();

tmp2=num.top();num.pop();

tmp1=num.top();num.pop();

tmp1=calculate(tmp1,tmp2,tmpc);

num.push(tmp1);

}else{

break;

}

}

}

}else{

op.push(arr[i][0]);

}

}

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

num.pop();

}

return 0;

}

##### Version 2.0

#include<stdio.h>

#include<stack>

using namespace std;

stack<double> num;

double tmp;

int main(){

int a;//数字

char b;//符号

char c;//空格或回车

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

if(a!=0)

num.push(a);

else

continue;

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

if(b=='+'){

num.push(a);

}else if(b=='-'){

num.push(-1.0\*a);

}else if(b=='\*'){

tmp=num.top()\*a;

num.pop();

num.push(tmp);

}else if(b=='/'){

tmp=num.top()/a;

num.pop();

num.push(tmp);

}

if(c!=' '){

break;

}

}

tmp=0;

while(!num.empty()){

tmp+=num.top();

num.pop();

}

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

}

return 0;

}

### 练习题：（题目1101）计算表达式

**题目描述：**

对于一个不存在括号的表达式进行计算

**输入：**

存在多种数据，每组数据一行，表达式不存在空格

**输出：**

输出结果

**样例输入：**

6/2+3+3\*4

**样例输出：**

18

#### 评价

见鬼了，参考书上的例题代码，我的一直不对

两种同类型的题网上盛行直接在输入上做文章的很简洁的代码

#### 代码

#include<stdio.h>

int main()

{

char ch;

int i,j,temp,a[200];

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

{

i=1;

a[0]=0;

a[1]=temp;

while(scanf("%c",&ch)!=EOF && ch!='\n')

{

scanf("%d",&temp);

if(ch=='-')a[++i]=-temp;

else if(ch=='+')a[++i]=temp;

else if(ch=='\*')a[i]\*=temp;

else if(ch=='/')a[i]/=temp;

}

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

a[0]+=a[j];

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

}

return 0;

}

## 二、哈夫曼树

#### 题目1172：哈夫曼树

**题目描述：**

哈夫曼树，第一行输入一个数n，表示叶结点的个数。需要用这些叶结点生成哈夫曼树，根据哈夫曼树的概念，这些结点有权值，即weight，题目需要输出所有结点的值与权值的乘积之和。

**输入：**

输入有多组数据。  
每组第一行输入一个数n，接着输入n个叶节点（叶节点权值不超过100，2<=n<=1000）。

**输出：**

输出权值。

**样例输入：**

5

1 2 2 5 9

**样例输出：**

37

#### 评价

注意，要> >而不是>>，本地可以，但在九度上会出错

#### 代码

#include<stdio.h>

#include<queue>

using namespace std;

priority\_queue<int,vector<int>,greater<int> > Q;

int main(){

int n;

int cnt;

int tmp1,tmp2,tmp;

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

while(!Q.empty())Q.pop();

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

int x;

scanf("%d",&x);

Q.push(x);

}

cnt=0;

while(Q.size()>1){

tmp1=Q.top();Q.pop();

tmp2=Q.top();Q.pop();

tmp=tmp1+tmp2;

cnt+=tmp;

Q.push(tmp);

}

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

}

return 0;

}

### 题目1107：搬水果

**题目描述：**

    在一个果园里，小明已经将所有的水果打了下来，并按水果的不同种类分成了若干堆，小明决定把所有的水果合成一堆。每一次合并，小明可以把两堆水果合并到一起，消耗的体力等于两堆水果的重量之和。当然经过 n‐1 次合并之后，就变成一堆了。小明在合并水果时总共消耗的体力等于每次合并所耗体力之和。

    假定每个水果重量都为 1，并且已知水果的种类数和每种水果的数目，你的任务是设计出合并的次序方案，使小明耗费的体力最少，并输出这个最小的体力耗费值。例如有 3 种水果，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。然后将新堆与原先的第三堆合并得到新的堆，耗费体力为 12。所以小明总共耗费体力=3+12=15，可以证明 15 为最小的体力耗费值。

**输入：**

    每组数据输入包括两行,第一行是一个整数 n(1<=n<=10000),表示水果的种类数，如果 n 等于 0 表示输入结束，且不用处理。第二行包含 n 个整数，用空格分隔，第 i 个整数(1<=ai<=1000)是第 i 种水果的数目。

**输出：**

对于每组输入，输出一个整数并换行，这个值也就是最小的体力耗费值。输入数据保证这个值小于 2^31。

**样例输入：**

3

9 1 2

0

**样例输出：**

15

#### 评价

跟例题根本一模一样

#### 代码

#include<stdio.h>

#include<queue>

using namespace std;

priority\_queue<int,vector<int>,greater<int> > Q;

int main(){

int n;

int cnt;

int tmp1,tmp2,tmp;

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

while(!Q.empty())Q.pop();

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

int x;

scanf("%d",&x);

Q.push(x);

}

cnt=0;

while(Q.size()>1){

tmp1=Q.top();Q.pop();

tmp2=Q.top();Q.pop();

tmp=tmp1+tmp2;

cnt+=tmp;

Q.push(tmp);

}

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

}

return 0;

}

## 三、二叉树

### 题目1078：二叉树遍历

**题目描述：**

二叉树的前序、中序、后序遍历的定义：

前序遍历：对任一子树，先访问跟，然后遍历其左子树，最后遍历其右子树；

中序遍历：对任一子树，先遍历其左子树，然后访问根，最后遍历其右子树；

后序遍历：对任一子树，先遍历其左子树，然后遍历其右子树，最后访问根。

给定一棵二叉树的前序遍历和中序遍历，求其后序遍历（提示：给定前序遍历与中序遍历能够唯一确定后序遍历）。

**输入：**

两个字符串，其长度n均小于等于26。

第一行为前序遍历，第二行为中序遍历。

二叉树中的结点名称以大写字母表示：A，B，C....最多26个结点。

**输出：**

输入样例可能有多组，对于每组测试样例，

输出一行，为后序遍历的字符串。

**样例输入：**

ABC

BAC

FDXEAG

XDEFAG

**样例输出：**

BCA

XEDGAF

#### 评价

想想递归就觉得很简单

指针方面不太熟，尤其现在，用c++编译，更担心有什么不一样

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 27

struct Node{

Node \*lChild;

Node \*rChild;

char elem;

};

char a[MAX];

char b[MAX];

Node tree[MAX];

int loc=0;

Node \*create(){

tree[loc].lChild=tree[loc].rChild=NULL;

return &tree[loc++];

}

Node \*build(int s1,int e1,int s2,int e2){

Node\* ret=create();

ret->elem=a[s1];

for(int i=s2;i<=e2;i++){

if(a[s1]==b[i]){

if(s2<=i-1)

ret->lChild=build(s1+1,s1+1+(i-1-s2),s2,i-1);//对于a子序列，左边是确定的

if(i+1<=e2)

ret->rChild=build(e1-(e2-i-1),e1,i+1,e2);//对于a子序列，右边是确定的

break;

}

}

return ret;

}

void postOrder(Node \*tree){

if(tree->lChild!=NULL){

postOrder(tree->lChild);

}

if(tree->rChild!=NULL){

postOrder(tree->rChild);

}

printf("%c",tree->elem);

}

int main(){

int len;

Node \*tree;

while(gets(a)){

loc=0;

gets(b);

len=strlen(a);

tree=build(0,len-1,0,len-1);

postOrder(tree);

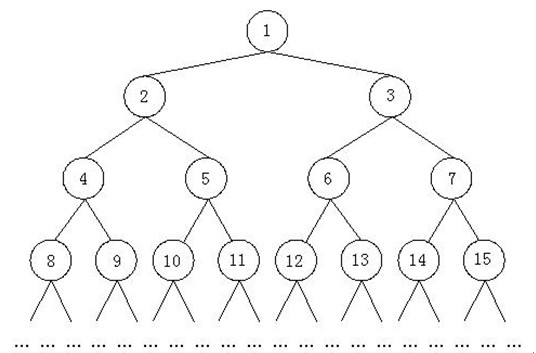
printf("\n");

}

return 0;

}

### 练习题：（题目1113）二叉树



如上所示，由正整数1，2，3……组成了一颗特殊二叉树。我们已知这个二叉树的最后一个结点是n。现在的问题是，结点m所在的子树中一共包括多少个结点。

比如，n = 12，m = 3那么上图中的结点13，14，15以及后面的结点都是不存在的，结点m所在子树中包括的结点有3，6，7，12，因此结点m的所在子树中共有4个结点。

输入：

输入数据包括多行，每行给出一组测试数据，包括两个整数m，n (1 <= m <= n <= 1000000000)。最后一组测试数据中包括两个0，表示输入的结束，这组数据不用处理。

输出：

对于每一组测试数据，输出一行，该行包含一个整数，给出结点m所在子树中包括的结点的数目。

样例输入：

3 12

0 0

样例输出：

4

#### 评价

1. Version 1.0超时，3360ms

Version 2.0 0ms

1. 真奇怪，第一种是指数级的，我就这么做了。一定是太喜欢递归了，代码简洁，走火入魔
2. 本来想，要不用建个二叉树实现一下，麻烦就麻烦点，当复习，看看我的version 1.0，想多了

#### 代码

##### Version 1.0

#include<stdio.h>

void cntNodes(int m,int n,int &cnt){

cnt++;

if(2\*m<=n)

cntNodes(2\*m,n,cnt);

if(2\*m+1<=n)

cntNodes(2\*m+1,n,cnt);

}

int main(){

int m;

int n;

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

if(m==0&&n==0){

break;

}

int cnt=0;

cntNodes(m,n,cnt);

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

}

return 0;

}

##### Version2.0

#include<stdio.h>

#include<math.h>

int main(){

int m;

int n;

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

if(m==0&&n==0){

break;

}

int tmp=n/m;

int cnt=0;

while(tmp/2!=0){

tmp/=2;

cnt++;

}

if(n>=(m+1)\*pow(2.0,cnt)){

cnt=pow(2.0,cnt+1)-1;

}else{

cnt=pow(2.0,cnt);

cnt=n-m\*cnt+cnt;

}

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

}

return 0;

}

### 练习题：（题目1176）树查找

**题目描述：**

有一棵树，输出某一深度的所有节点，有则输出这些节点，无则输出EMPTY。该树是完全二叉树。

**输入：**

输入有多组数据。  
每组输入一个n(1<=n<=1000)，然后将树中的这n个节点依次输入，再输入一个d代表深度。

**输出：**

输出该树中第d层得所有节点，节点间用空格隔开，最后一个节点后没有空格。

**样例输入：**

4

1 2 3 4

2

**样例输出：**

2 3

#### 评价

跟上题差不多，挺简单的

#### 代码

#include<stdio.h>

#include<math.h>

#define MAX 1001

int main(){

int n;

int a[MAX];

int d;

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

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

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

}

scanf("%d",&d);

int start=pow(2.0,d-1);

int end=pow(2.0,d)-1;

if(end>n)end=n;

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

for(int i=start+1;i<=end;i++){

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

}

printf("\n");

}

return 0;

}

## 四、二叉排序树

### 题目1201：二叉排序树

题目描述：

输入一系列整数，建立二叉排序数，并进行前序，中序，后序遍历。

输入：

输入第一行包括一个整数n(1<=n<=100)。

接下来的一行包括n个整数。

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排序树，并对二叉排序树进行前序、中序和后序遍历。

每种遍历结果输出一行。每行最后一个数据之后有一个空格。

样例输入：

5

1 6 5 9 8

样例输出：

1 6 5 9 8

1 5 6 8 9

5 8 9 6 1

提示：

输入中可能有重复元素，但是输出的二叉树遍历序列中重复元素不用输出。

#### 评价

我写的不太好。但是我这样算复习了指针相关的内容，又有点会用了。还是尽量不用指针

Version 2.0 是网上的版本，更简洁

#### 代码

##### Version 1.0

#include<stdio.h>

#define MAX 100

struct Node{

Node \*lchild;

Node \*rchild;

int elem;

}tree[101];

int loc;

Node\* create(){

tree[loc].lchild=tree[loc].rchild=NULL;

return &tree[loc++];

}

void addNode(int elem,Node \*tree){

Node \*p=tree;

while(tree->elem!=elem){

if(elem<tree->elem){

if(tree->lchild==NULL){

tree->lchild=create();

tree=tree->lchild;

tree->elem=elem;

tree=p;

break;

}

tree=tree->lchild;

}else{

if(tree->rchild==NULL){

tree->rchild=create();

tree=tree->rchild;

tree->elem=elem;

tree=p;

break;

}

tree=tree->rchild;

}

}

}

Node\* build(int a[],int n){

Node\* ret=create();

ret->elem=a[0];

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

addNode(a[i],ret);

}

return ret;

}

void preOrder(Node \*tree){

printf("%d ",tree->elem);

if(tree->lchild!=NULL)

preOrder(tree->lchild);

if(tree->rchild!=NULL)

preOrder(tree->rchild);

}

void inOrder(Node \*tree){

if(tree->lchild!=NULL)

inOrder(tree->lchild);

printf("%d ",tree->elem);

if(tree->rchild!=NULL)

inOrder(tree->rchild);

}

void postOrder(Node \*tree){

if(tree->lchild!=NULL)

postOrder(tree->lchild);

if(tree->rchild!=NULL)

postOrder(tree->rchild);

printf("%d ",tree->elem);

}

int main(){

int n;

int a[MAX];

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

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

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

}

Node \*tree;

loc=0;

tree=build(a,n);

preOrder(tree);

printf("\n");

inOrder(tree);

printf("\n");

postOrder(tree);

printf("\n");

}

return 0;

}

##### Version 2.0

#include <stdio.h>

struct Node{

Node \*lchild;

Node \*rchild;

int c;

}Tree[150];

int loc;

Node \*creat(){

Tree[loc].lchild = Tree[loc].rchild = NULL;

return &Tree[loc++];

}

void preOrder(Node \*T){

printf("%d ", T->c);

if (T->lchild != NULL)

preOrder(T->lchild);

if (T->rchild != NULL)

preOrder(T->rchild);

}

void inOrder(Node \*T){

if (T->lchild != NULL)

inOrder(T->lchild);

printf("%d ", T->c);

if (T->rchild != NULL)

inOrder(T->rchild);

}

void postOrder(Node \*T){

if (T->lchild != NULL)

postOrder(T->lchild);

if (T->rchild != NULL)

postOrder(T->rchild);

printf("%d ", T->c);

}

Node \*insert(Node \*T, int x){

if (T == NULL){

T = creat();

T->c = x;

return T;

}

else if (x < T->c)

T->lchild = insert(T->lchild, x);

else if (x > T->c)

T->rchild = insert(T->rchild, x);

return T;

}

int main(){

int n;

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

int tmp;

loc = 0;

Node \*T = NULL;

while (n--){

scanf("%d", &tmp);

T = insert(T, tmp);

}

preOrder(T);

printf("\n");

inOrder(T);

printf("\n");

postOrder(T);

printf("\n");

}

return 0;

}

### 题目1009：二叉搜索树

题目描述：

判断两序列是否为同一二叉搜索树序列

输入：

开始一个数n，(1<=n<=20) 表示有n个需要判断，n= 0 的时候输入结束。

接下去一行是一个序列，序列长度小于10，包含(0~9)的数字，没有重复数字，根据这个序列可以构造出一颗二叉搜索树。

接下去的n行有n个序列，每个序列格式跟第一个序列一样，请判断这两个序列是否能组成同一颗二叉搜索树。

输出：

如果序列相同则输出YES，否则输出NO

样例输入：

2

567432

543267

576342

0

样例输出：

YES

NO

#### 评价

1. 指针用的不是很熟练。

T在刚开始时一定要置为NULL,不然未定，会出错。或者使用双重指针。

而且，指针用来引用，注意是否要是NULL很重要

建树前，要看树是不是空，如果不是空，就是在该树基础上继续建了

1. 另外，写的不太漂亮

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 30

struct Node{

Node \*lchild;

Node \*rchild;

int elem;

}T[10];

int str1[10],str2[10];

int str3[10],str4[10];

int loc;

int loc1;

Node\* create(){

T[loc1].lchild=T[loc1].rchild=NULL;

return &T[loc1++];

}

Node\* insert(Node\* T,int a){

if(T==NULL){

T=create();

T->elem=a;

return T;

}else if(a<T->elem){

T->lchild=insert(T->lchild,a);

}else if(a>T->elem){

T->rchild=insert(T->rchild,a);

}

return T;

}

void preOrder(Node \*T,int flag){

if(flag==1)

str1[loc++]=T->elem;

else

str3[loc++]=T->elem;

if(T->lchild!=NULL)preOrder(T->lchild,flag);

if(T->rchild!=NULL)preOrder(T->rchild,flag);

}

void inOrder(Node \*T,int flag){

if(T->lchild!=NULL)inOrder(T->lchild,flag);

if(flag==1)

str2[loc++]=T->elem;

else

str4[loc++]=T->elem;

if(T->rchild!=NULL)inOrder(T->rchild,flag);

}

int main(){

int n;

char a[MAX][10];

int len1,len2;

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

Node \*T1=NULL,\*T2;

getchar();

gets(a[0]);

len1=strlen(a[0]);

loc1=0;

for(int i=0;i<len1;i++){

T1=insert(T1,a[0][i]-'0');

}

loc=0;

preOrder(T1,1);

loc=0;

inOrder(T1,1);

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

bool isSame=true;

gets(a[i]);

len2=strlen(a[i]);

if(len1!=len2)printf("NO\n");

else{

T2=NULL;

loc1=0;

for(int j=0;j<len2;j++){

T2=insert(T2,a[i][j]-'0');

}

loc=0;

preOrder(T2,2);

loc=0;

inOrder(T2,2);

for(int i=0;i<len1;i++){

if(str1[i]!=str3[i]||str2[i]!=str4[i]){

printf("NO\n");

isSame=false;

break;

}

}

if(isSame)

printf("YES\n");

}

}

}

}

# 第四章：数学问题

## 一、%运算符

### 练习题：（题目1015）还是A+B

题目描述：

读入两个小于10000的正整数A和B，计算A+B。需要注意的是：如果A和B的末尾K（不超过8）位数字相同，请直接输出-1。

输入：

测试输入包含若干测试用例，每个测试用例占一行，格式为"A B K"，相邻两数字有一个空格间隔。当A和B同时为0时输入结束，相应的结果不要输出。

输出：

对每个测试用例输出1行，即A+B的值或者是-1。

样例输入：

1 2 1

11 21 1

108 8 2

36 64 3

0 0 1

样例输出：

3

-1

-1

100

#### 评价

简单

#### 代码

#include<stdio.h>

#include<math.h>

int main(){

int a,b,k;

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

if(a==0&&b==0)

break;

int tmp=pow(10.0,k);

if(a%tmp==b%tmp){

printf("-1\n");

}else{

printf("%d\n",a+b);

}

}

return 0;

}

#### 练习题：（题目1183）守形数

**题目描述：**

守形数是这样一种整数，它的平方的低位部分等于它本身。  
比如25的平方是625，低位部分是25，因此25是一个守形数。  
编一个程序，判断N是否为守形数。

**输入：**

输入包括1个整数N，2<=N<100。

**输出：**

可能有多组测试数据，对于每组数据，  
输出"Yes!”表示N是守形数。  
输出"No!”表示N不是守形数。

**样例输入：**

25

4

**样例输出：**

Yes!

No!

#### 评价

简单

#### 代码

#include<stdio.h>

#include<math.h>

int main(){

int a;

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

int cnt=1;

int tmp=a;

while(tmp/10!=0){

tmp/=10;

cnt++;

}

tmp=pow(10.0,cnt);

if(a\*a%tmp==a)printf("Yes!\n");

else printf("No!\n");

}

return 0;

}

## 二、数位拆解

### 题目1083：特殊乘法

**题目描述：**

写个算法，对2个小于1000000000的输入，求结果。

特殊乘法举例：123 \* 45 = 1\*4 +1\*5 +2\*4 +2\*5 +3\*4+3\*5

**输入：**

 两个小于1000000000的数

**输出：**

 输入可能有多组数据，对于每一组数据，输出Input中的两个数按照题目要求的方法进行运算后得到的结果。

**样例输入：**

123 45

**样例输出：**

54

#### 评价

简单

#### 代码

#include<stdio.h>

#include<math.h>

int main(){

int a,b;

int tmp;

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

tmp=0;

int tmp1=a,tmp2=b;

int cnt1=0,cnt2=0;

while(tmp1!=0){

tmp1/=10;

cnt1++;

}

while(tmp2!=0){

tmp2/=10;

cnt2++;

}

tmp1=a;tmp2=b;

for(int i=0;i<cnt1;i++){

tmp2=b;

for(int j=0;j<cnt2;j++){

tmp+=(tmp1%10)\*(tmp2%10);

tmp2/=10;

}

tmp1/=10;

}

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

}

return 0;

}

## 七、分解素因数

### 题目1104：整除问题

**题目描述：**

给定n，a求最大的k，使n！可以被a^k整除但不能被a^(k+1)整除。

**输入：**

两个整数n(2<=n<=1000)，a(2<=a<=1000)

**输出：**

一个整数.

**样例输入：**

6 10

**样例输出：**

1

#### 评价

竟然忘了(a\*b)%c=(a%c)\*(b%c)，这样可以很简单

#### 代码

##### Version 1.0

#include<stdio.h>

#include<math.h>

#define MAX 10000

int a[MAX]={0};

int Prime[MAX];

int idx=0;

void init(){

for(int i=2;i<MAX;i++){

if(a[0]==0){

Prime[idx++]=i;

for(int j=i\*i;j<MAX;j+=i){

a[j]=1;

}

}else{

continue;

}

}

}

int divid[MAX][2]={0};

int main(){

int n,a;

init();

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

int idx1=0;

int idx2=0;

int idx3=0;

int tmp=a;

while(tmp!=1){

while(tmp%Prime[idx1]==0){//如果能一直整除就继续

tmp/=Prime[idx1];

idx3++;

}

if(idx3!=0){

divid[idx2][0]=Prime[idx1];

divid[idx2++][1]=idx3;

idx3=0;

}

idx1++;

}

int k=100000;

int cnt=0;

for(int i=0;i<idx2;i++){

tmp=n;

while(tmp/divid[i][0]!=0){

tmp/=divid[i][0];

cnt+=tmp;

}

cnt/=divid[i][1];

if(cnt<k)k=cnt;

cnt=0;

}

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

}

return 0;

}

##### Version 2.0

#include<stdio.h>

int main(){

int n,a;

int sum=1;

int k=0;

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

for(int i=n;i>=1;i--){

sum\*=i;

while(sum%a==0){sum/=a;k++;}

sum%=a;

}

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

sum=1;

k=0;

}

return 0;

}

### 练习题：（题目1087）约数的个数

**题目描述：**

输入n个整数,依次输出每个数的约数的个数

**输入：**

输入的第一行为N，即数组的个数(N<=1000)  
接下来的1行包括N个整数，其中每个数的范围为(1<=Num<=1000000000)  
当N=0时输入结束。

**输出：**

可能有多组输入数据，对于每组输入数据，  
输出N行，其中每一行对应上面的一个数的约数的个数。

**样例输入：**

5

1 3 4 6 12

**样例输出：**

1

2

3

4

6

#### 评价

用约数个数定理

最后看看是不是一个大素数很重要

#### 代码

#include<stdio.h>

int mark[100000]={0};

int Prime[100000];

int idx=0;

void init(){

for(int i=2;i<33000;i++){

if(mark[i]==1)continue;

Prime[idx++]=i;

for(int j=i\*i;j<33000;j+=i){

mark[j]=1;

}

}

}

int main(){

int n;

int a[1000];

init();

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

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

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

}

int cnt=0;

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

int ansidx=0;

int sum=1;

while(a[i]!=1){

while(a[i]%Prime[ansidx]==0){

a[i]/=Prime[ansidx];

cnt++;

}

if(cnt!=0){

sum\*=cnt+1;

cnt=0;

}

if(cnt==0&&ansidx==idx-1){

sum\*=2;

break;

}

ansidx++;

}

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

}

}

return 0;

}

## 八、二分求幂

### 题目1441：人见人爱 A ^ B

**题目描述：**

求A^B的最后三位数表示的整数。说明：A^B的含义是“A的B次方”

**输入：**

输入数据包含多个测试实例，每个实例占一行，由两个正整数A和B组成（1<=A,B<=10000），如果A=0, B=0，则表示输入数据的结束，不做处理。

**输出：**

对于每个测试实例，请输出A^B的最后三位表示的整数，每个输出占一行。

**样例输入：**

2 3

12 6

6789 10000

0 0

**样例输出：**

8

984

1

#### 评价

主要永远后3位计算，我写的总是麻烦点

#### 代码

##### Version 1.0

#include<stdio.h>

#include<math.h>

#define MAX 20

int main(){

int a,b,c;

int arr[MAX];

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

if(a==0&&b==0)break;

int i=0;

for(;b!=0;i++){

arr[i]=b%2;

b/=2;

}

int tmp=a;

c=1;

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

if(arr[j]==1){

c\*=tmp;

c%=1000;

}

tmp=tmp\*tmp%1000;

}

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

}

return 0;

}

##### Version 2.0

#include<stdio.h>

#include<math.h>

#define MAX 20

int main(){

int a,b,c;

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

if(a==0&&b==0)break;

c=1;

while(b!=0){

if(b%2==1){

c\*=a;

c%=1000;

}

b/=2;

a=a\*a%1000;

}

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

}

return 0;

}

### 练习题（题目1442）A sequence of numbers

**题目描述：**

Xinlv wrote some sequences on the paper a long time ago, they might be arithmetic or geometric sequences. The numbers are not very clear now, and only the first three numbers of each sequence are recognizable. Xinlv wants to know some numbers in these sequences, and he needs your help.

**输入：**

The first line contains an integer N, indicting that there are N sequences. Each of the following N lines contain four integers. The first three indicating the first three numbers of the sequence, and the last one is K, indicating that we want to know the K-th numbers of the sequence.

You can assume 0 < K <= 10^9, and the other three numbers are in the range [0, 2^63). All the numbers of the sequences are integers. And the sequences are non-decreasing.

**输出：**

Output one line for each test case, that is, the K-th number module (%) 200907.

**样例输入：**

2

1 2 3 5

1 2 4 5

**样例输出：**

5

16

#### 评价

遇到这种题，不要手软，把全部整数类型都变成long long

还有，每一步运算都可以取余

#### 代码

#include<stdio.h>

#define M 200907

int main(){

long long n;

long long a,b,c;

long long k;

long long ans;

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

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

scanf("%lld%lld%lld%lld",&a,&b,&c,&k);

if(b-a==c-b){

long long step=(b-a)%M;

ans=a%M+step\*((k-1)%M)%M;

ans%=M;

}else{

long long step=(b/a)%M;

k=k-1;

ans=a%M;

while(k!=0){

if(k%2==1){

ans\*=step;

ans%=M;

}

k/=2;

step=step\*step%M;

}

}

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

}

}

return 0;

}

### 练习题：（题目1443）Tr A

**题目描述：**

A为一个方阵，则Tr A表示A的迹（就是主对角线上各项的和），现要求Tr(A^k)%9973。

**输入：**

数据的第一行是一个T，表示有T组数据。  
每组数据的第一行有n(2 <= n <= 10)和k(2 <= k < 10^9)两个数据。接下来有n行，每行有n个数据，每个数据的范围是[0,9]，表示方阵A的内容。

**输出：**

对应每组数据，输出Tr(A^k)%9973。

**样例输入：**

2

2 2

1 0

0 1

3 99999999

1 2 3

4 5 6

7 8 9

**样例输出：**

2

2686

#### 评价

熟悉有关矩阵乘法

结构体，赋值引用方便，不错

#### 代码

#include<stdio.h>

#define M 9973

struct Matrix{

int mat[10][10];

};

Matrix mulMatrix(Matrix a,Matrix b,int n){

Matrix c;

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

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

c.mat[i][j]=0;

for(int k=0;k<n;k++){

c.mat[i][j]+=a.mat[k][j]\*b.mat[i][k];

}

c.mat[i][j]%=M;

}

}

return c;

}

int main(){

int T;

int n,k;

Matrix m,ans;

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

while(T--!=0){

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

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

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

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

ans.mat[i][j]=0;

}

}

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

ans.mat[i][i]=1;

}

//deal

while(k!=0){

if(k%2==1){

ans=mulMatrix(ans,m,n);

}

k/=2;

m=mulMatrix(m,m,n);

}

int sum=0;

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

sum+=ans.mat[i][i];

}

sum%=M;

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

}

}

return 0;

}

## 九、高精度整数

### 题目1198：a+b

**题目描述：**

实现一个加法器，使其能够输出a+b的值。

**输入：**

输入包括两个数a和b，其中a和b的位数不超过1000位。

**输出：**

可能有多组测试数据，对于每组数据，  
输出a+b的值。

**样例输入：**

2 6

10000000000000000000 10000000000000000000000000000000

**样例输出：**

8

10000000000010000000000000000000

#### 评价

只有version 3.0是用c++写的，但重点也是这个

#### 代码

##### Version 1.0

import java.math.BigInteger;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner cin = new Scanner(System.in);

while (cin.hasNextBigInteger()) {

BigInteger a = cin.nextBigInteger();

BigInteger b = cin.nextBigInteger();

System.out.println(a.add(b));

}

}

}

##### Version 2.0

import java.math.BigDecimal;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

while(s.hasNext()){

String a = s.next();

String b = s.next();

BigDecimal big1 = new BigDecimal(a);

BigDecimal big2 = new BigDecimal(b);

System.out.println(big1.add(big2).toString());

}

}

}

##### Version 3.0

#include<stdio.h>

#include<string.h>

#define MAX 1010

struct BigInteger{

int digit[MAX];

int size;

void init(){

for(int i=0;i<MAX;i++)digit[i]=0;

size=0;

}

void set(char str[]){

init();

int len=strlen(str);

int tmp=0;

int c=1;//权重

int m=0;//取余

for(int i=len-1;i>=0;i--){//数组最低位，digit最高位

tmp+=(str[i]-'0')\*c;

c\*=10;

m++;

if(m%4==0||i==0){

digit[size++]=tmp;

c=1;

m=0;

tmp=0;

}

}

}

BigInteger operator +(const BigInteger &B)const{

BigInteger c;

c.init();

int carry=0;

int tmp;

int i;

for(i=0;i<size||i<B.size;i++){//取两数最大位数

tmp=digit[i]+B.digit[i]+carry;

carry=tmp/10000;

tmp=tmp%10000;

c.digit[i]=tmp;

}

if(carry!=0){

c.digit[i++]=carry;

}

c.size=i;

return c;

}

void print(){

for(int i=size-1;i>=0;i--){

if(i!=size-1){

printf("%04d",digit[i]);

}else{

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

}

}

printf("\n");

}

}a,b,c;

int main(){

char str1[MAX],str2[MAX];

while(scanf("%s %s",str1,str2)!=EOF){

a.set(str1);

b.set(str2);

c=a+b;

c.print();

}

return 0;

}

### 题目1076：N的阶乘

**题目描述：**

 输入一个正整数N，输出N的阶乘。

**输入：**

正整数N(0<=N<=1000)

**输出：**

 输入可能包括多组数据，对于每一组输入数据，输出N的阶乘

**样例输入：**

4

5

15

**样例输出：**

24

120

1307674368000

#### 评价

750ms，好长

跟加法差不多

#### 代码

#include<stdio.h>

#define MAX 1010

struct BigInteger{

int digit[MAX];

int size;

void init(){

for(int i=0;i<MAX;i++)digit[i]=0;

size=0;

}

void set(int n){

init();

digit[size++]=n;

}

BigInteger operator \* (int x)const{

BigInteger c;

c.init();

int carry=0;

int tmp;

int i;

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

tmp=digit[i]\*x+carry;

carry=tmp/10000;

tmp=tmp%10000;

c.digit[i]=tmp;

}

if(carry!=0){

c.digit[i++]=carry;

}

c.size=i;

return c;

}

void print(){

for(int i=size-1;i>=0;i--){

if(i!=size-1){

printf("%04d",digit[i]);

}else{

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

}

}

printf("\n");

}

}a;

int main(){

int n;

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

if(n==0){printf("1\n");continue;}

a.set(n);

while(n--!=1){

a=a\*n;

}

a.print();

}

return 0;

}

### 题目1080：进制转换

**题目描述：**

将M进制的数X转换为N进制的数输出。

**输入：**

输入的第一行包括两个整数：M和N(2<=M,N<=36)。  
下面的一行输入一个数X，X是M进制的数，现在要求你将M进制的数X转换成N进制的数输出。

**输出：**

输出X的N进制表示的数。

**样例输入：**

16 10

F

**样例输出：**

15

**提示：**

输入时字母部分为大写，输出时为小写，并且有大数据。

#### 评价

大数与普通数的除法，取余

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 100

struct BigInteger{

int digit[MAX];

int size;

void init(){

for(int i=0;i<MAX;i++)digit[i]=0;

size=0;

}

void set(int x){

init();

digit[0]=x;

size++;

}

BigInteger operator +(const BigInteger &b)const{

BigInteger ret;

ret.size=0;

ret.init();

int carry=0;

int tmp;

int r;

for(int i=0;i<size||i<b.size;i++){

tmp=digit[i]+b.digit[i]+carry;

carry=tmp/10000;

r=tmp%10000;

ret.digit[ret.size++]=r;

}

if(carry!=0){

ret.digit[ret.size++]=carry;

}

return ret;

}

BigInteger operator \*(int b)const{

BigInteger ret;

ret.size=0;

ret.init();

int carry=0;

int tmp;

int r;

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

tmp=digit[i]\*b+carry;

carry=tmp/10000;

r=tmp%10000;

ret.digit[ret.size++]=r;

}

if(carry!=0){

ret.digit[ret.size++]=carry;

}

return ret;

}

BigInteger operator /(int b)const{

BigInteger ret;

ret.init();

int tmp=0;

int carry=0;

for(int i=size-1;i>=0;i--){

tmp=(carry\*10000+digit[i])/b;

carry=(carry\*10000+digit[i])%b;

ret.digit[i]=tmp;

}

ret.size=0;

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

if(ret.digit[i]!=0)ret.size=i;

}

ret.size++;

return ret;

}

int operator %(int b)const{

int carry=0;

for(int i=size-1;i>=0;i--){

carry=(carry\*10000+digit[i])%b;

}

return carry;

}

}a,b;

int main(){

int M,N;

char X[10000];

int len;

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

scanf("%s",X);

len=strlen(X);

a.set(0);

b.set(1);

int t;

for(int i=len-1;i>=0;i--){

if(X[i]>='0'&&X[i]<='9')t=X[i]-'0';

else t=X[i]-'A'+10;

a=a+b\*t;

b=b\*M;

}

char s[100000];

int idx=0;

do{

t=a%N;

if(t<10)s[idx++]=t+'0';

else s[idx++]=t+'a'-10;

a=a/N;

}while(a.size!=1||a.digit[0]!=0);

for(int i=idx-1;i>=0;i--)printf("%c",s[i]);

printf("\n");

}

return 0;

}

### 练习题：（题目1137）浮点数加法

**题目描述：**

求2个浮点数相加的和  
题目中输入输出中出现浮点数都有如下的形式：  
P1P2...Pi.Q1Q2...Qj  
对于整数部分，P1P2...Pi是一个非负整数  
对于小数部分，Qj不等于0

**输入：**

对于每组案例，第1行是测试数据的组数n，每组测试数据占2行，分别是两个加数。  
每组测试数据之间有一个空行，每行数据不超过100个字符

**输出：**

每组案例是n行，每组测试数据有一行输出是相应的和。  
输出保证一定是一个小数部分不为0的浮点数

**样例输入：**

2

0.111111111111111111111111111111

0.111111111111111111111111111111

10000000.655555555555555555555555555555

1.444444444444444444444444444445

**样例输出：**

0.222222222222222222222222222222

10000002.1

#### 评价

巨坑

Scanf和gets()????????????????????

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 110

struct flt{

int p[MAX];

int q[MAX];

int p\_size;

int q\_size;

void init(){

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

p[i]=0;

q[i]=0;

}

p\_size=0;

q\_size=0;

}

void set(char str[]){

init();

int len=strlen(str);

int t=0;

while(str[t]!='.')t++;//小数点的位置

//整数部分

int c=1;

int tmp=0;

int m=0;

for(int i=t-1;i>=0;i--){

tmp+=(str[i]-'0')\*c;

m++;

c\*=10;

if(m%4==0||i==0){

p[p\_size++]=tmp;

m=0;

c=1;

tmp=0;

}

}

//小数部分

c=1000;

tmp=0;

m=0;

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

tmp+=(str[i]-'0')\*c;

m++;

c/=10;

if(m%4==0||i==len-1){

q[q\_size++]=tmp;

m=0;

tmp=0;

c=1000;

}

}

}

flt operator +(const flt &b)const{

flt c;

c.init();

//小数部分

int max=q\_size>b.q\_size?q\_size:b.q\_size;

int carry=0;

int t;

c.q\_size=max;

for(int i=max-1;i>=0;i--){

t=q[i]+b.q[i]+carry;

carry=t/10000;

t%=10000;

c.q[i]=t;

}

//整数部分

for(int i=0;i<p\_size||i<b.p\_size;i++){

t=p[i]+b.p[i]+carry;

carry=t/10000;

t%=10000;

c.p[c.p\_size++]=t;

}

if(carry!=0){

c.p[c.p\_size++]=carry;

}

return c;

}

void output(){

int tmp=p\_size;

p\_size=0;

for(int i=tmp-1;i>=0;i--){

if(p[i]==0)continue;

else{

p\_size=i;

break;

}

}

p\_size++;

tmp=q\_size;

q\_size=0;

for(int i=tmp-1;i>=0;i--){

if(q[i]==0)continue;

else{

q\_size=i;

break;

}

}

q\_size++;

for(int i=p\_size-1;i>=0;i--){

if(i!=p\_size-1)printf("%04d",p[i]);

else printf("%d",p[i]);

}

printf(".");

for(int i=0;i<q\_size;i++){

if(i!=q\_size-1)printf("%04d",q[i]);

else if(q[i]%1000==0){

printf("%d",q[i]/1000);

}else if(q[i]%100==0){

printf("%02d",q[i]/100);

}else if(q[i]%10==0){

printf("%03d",q[i]/10);

}else{

printf("%04d",q[i]);

}

}

printf("\n");

}

}a,b,c;

int main(){

char s1[MAX],s2[MAX];

int n;

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

// getchar();

while(n--!=0){

scanf("%s%s",s1,s2);

// gets(s1);gets(s2);

a.set(s1);b.set(s2);

c=a+b;

c.output();

}

}

return 0;

}

### 练习题：（题目1190）大整数排序

**题目描述：**

对N个长度最长可达到1000的数进行排序。

**输入：**

输入第一行为一个整数N，(1<=N<=100)。  
接下来的N行每行有一个数，数的长度范围为1<=len<=1000。  
每个数都是一个正数，并且保证不包含前缀零。

**输出：**

可能有多组测试数据，对于每组数据，将给出的N个数从小到大进行排序，输出排序后的结果，每个数占一行。

**样例输入：**

3

11111111111111111111111111111

2222222222222222222222222222222222

33333333

**样例输出：**

33333333

11111111111111111111111111111

2222222222222222222222222222222222

#### 评价

char s[MAX][MAX];如果这个不是设置成全局变量，它也不是在一开始就退出

#### 代码

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

#define MAX 1010

struct BigInteger{

int digit[MAX];

int size;

void init(){

for(int i=0;i<MAX;i++)digit[i]=0;

size=0;

}

void set(char s[]){

init();

int len=strlen(s);

int t=0;

int carry=0;

int c=1;

int m=0;

for(int i=len-1;i>=0;i--){

t+=(s[i]-'0')\*c+carry;

c\*=10;

carry=t/10000;

t%=10000;

m++;

if(m==4||i==0){

digit[size++]=t;

m=0;

c=1;

t=0;

}

}

if(carry!=0){

digit[size++]=carry;

}

}

void output(){

for(int i=size-1;i>=0;i--){

if(i!=size-1)printf("%04d",digit[i]);

else printf("%d",digit[i]);

}

printf("\n");

}

bool operator <(const BigInteger &b)const{

if(size!=b.size)return size<b.size;

else{

for(int i=size-1;i>=0;i--){

if(digit[i]==b.digit[i])continue;

else{

return digit[i]<b.digit[i];

}

}

return true;

}

}

};

BigInteger bigint[MAX];

char s[MAX][MAX];

int main(){

int n;

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

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

scanf("%s",s[i]);

bigint[i].set(s[i]);

}

sort(bigint,bigint+n);

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

bigint[i].output();

}

}

return 0;

}

### 练习题：（题目1208）10进制 VS 2进制

**题目描述：**

    对于一个十进制数A，将A转换为二进制数，然后按位逆序排列，再转换为十进制数B，我们乘B为A的二进制逆序数。  
    例如对于十进制数173，它的二进制形式为10101101，逆序排列得到10110101，其十进制数为181，181即为173的二进制逆序数。

**输入：**

    一个1000位(即10^999)以内的十进制数。

**输出：**

    输入的十进制数的二进制逆序数。

**样例输入：**

173

**样例输出：**

181

#### 评价

都差不多

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 1010

struct BigInteger{

int digit[MAX];

int size;

void init(){

for(int i=0;i<MAX;i++)digit[i]=0;

size=0;

}

void set(char s[]){

init();

int len=strlen(s);

int t=0;

int carry=0;

int c=1;

int m=0;

for(int i=len-1;i>=0;i--){

t+=(s[i]-'0')\*c+carry;

c\*=10;

carry=t/10000;

t%=10000;

m++;

if(m==4||i==0){

digit[size++]=t;

m=0;

c=1;

t=0;

}

}

if(carry!=0){

digit[size++]=carry;

}

}

void output(){

for(int i=size-1;i>=0;i--){

if(i!=size-1)printf("%04d",digit[i]);

else printf("%d",digit[i]);

}

printf("\n");

}

int operator % (int b)const{

int r=0;

for(int i=size-1;i>=0;i--){

r=(digit[i]+r\*10000)%b;

}

return r;

}

BigInteger operator + (const BigInteger &b)const{

BigInteger ret;

ret.init();

int carry=0;

int t;

for(int i=0;i<size||i<b.size;i++){

t=digit[i]+b.digit[i]+carry;

carry=t/10000;

t%=10000;

ret.digit[ret.size++]=t;

}

if(carry!=0){

ret.digit[ret.size++]=carry;

}

return ret;

}

BigInteger operator \* (int b)const{

BigInteger ret;

ret.init();

int carry=0;

int t;

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

t=digit[i]\*b+carry;

carry=t/10000;

t%=10000;

ret.digit[ret.size++]=t;

}

if(carry!=0){

ret.digit[ret.size++]=carry;

}

return ret;

}

BigInteger operator / (int b)const{

BigInteger c;

c.init();

int t;

int r=0;

int max=size;

for(int i=size-1;i>=0;i--){

t=(digit[i]+r\*10000)/b;

r=(digit[i]+r\*10000)%b;

c.digit[i]=t;

}

c.size=0;

for(int i=max-1;i>=0;i--){

if(c.digit[i]==0)continue;

else{

c.size=i;

break;

}

}

c.size++;

return c;

}

};

BigInteger a,t,r;

char s[MAX];

int main(){

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

a.set(s);

r.set("0");//表结果

t.set("1");

int tmp=0;

while(a.size!=1||a.digit[0]!=0){

tmp=a%2;

if(tmp==1){

r=t+r\*2;

}else{

r=r\*2;

}

a=a/2;

}

r.output();

}

return 0;

}

# 第五章：图论

## 二、并查集

### 题目1012：畅通工程

**题目描述：**

    某省调查城镇交通状况，得到现有城镇道路统计表，表中列出了每条道路直接连通的城镇。省政府“畅通工程”的目标是使全省任何两个城镇间都可以实现交通（但不一定有直接的道路相连，只要互相间接通过道路可达即可）。问最少还需要建设多少条道路？

**输入：**

    测试输入包含若干测试用例。每个测试用例的第1行给出两个正整数，分别是城镇数目N ( < 1000 )和道路数目M；随后的M行对应M条道路，每行给出一对正整数，分别是该条道路直接连通的两个城镇的编号。为简单起见，城镇从1到N编号。   
    注意:两个城市之间可以有多条道路相通,也就是说  
    3 3  
    1 2  
    1 2  
    2 1  
    这种输入也是合法的  
    当N为0时，输入结束，该用例不被处理。

**输出：**

    对每个测试用例，在1行里输出最少还需要建设的道路数目。

**样例输入：**

4 2

1 3

4 3

3 3

1 2

1 3

2 3

5 2

1 2

3 5

999 0

0

**样例输出：**

1

0

2

998

#### 评价

并查集、并查集

#### 代码

#include<stdio.h>

#define N 1010

int tree[N];

int findRoot(int x){

if(tree[x]==-1)return x;

// else return findRoot(tree[x]);

else{

int root=findRoot(tree[x]);

tree[x]=root;

return root;

}

}

int main(){

int n,m;

int t1,t2;

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

scanf("%d",&m);

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

tree[i]=-1;

}

int r1,r2;

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

scanf("%d%d",&t1,&t2);

r1=findRoot(t1);

r2=findRoot(t2);

if(r1!=r2)tree[r1]=r2;

}

int cnt=0;

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

// printf("tree is %d\n",tree[i]);

if(tree[i]==-1)cnt++;

}

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

}

return 0;

}

## 题目1444：More is better

**题目描述：**

Mr Wang wants some boys to help him with a project. Because the project is rather complex, the more boys come, the better it will be. Of course there are certain requirements.Mr Wang selected a room big enough to hold the boys. The boy who are not been chosen has to leave the room immediately. There are 10000000 boys in the room numbered from 1 to 10000000 at the very beginning. After Mr Wang's selection any two of them who are still in this room should be friends (direct or indirect), or there is only one boy left. Given all the direct friend-pairs, you should decide the best way.

**输入：**

The first line of the input contains an integer n (0 ≤ n ≤ 100 000) - the number of direct friend-pairs. The following n lines each contains a pair of numbers A and B separated by a single space that suggests A and B are direct friends. (A ≠ B, 1 ≤ A, B ≤ 10000000)

**输出：**

The output in one line contains exactly one integer equals to the maximum number of boys Mr Wang may keep.

**样例输入：**

4

1 2

3 4

5 6

1 6

4

1 2

3 4

5 6

7 8

**样例输出：**

4

2

#### 评价

我原本想，都千万数量级了，有没有简单点的，看示例也是这样，算了

我把每个根的值都弄成负的，其绝对值就是此树的数目

看了几面，就一个人比我快100ms，其他人至少比我慢600ms。我310ms

#### 代码

#include<stdio.h>

#define N 10000001

int tree[N];

int findRoot(int x){

if(tree[x]<0)return x;

else{

int root=findRoot(tree[x]);

tree[x]=root;

return root;

}

}

int main(){

int n;

int t1,t2;

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

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

tree[i]=-1;

}

int r1,r2,max=1;

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

scanf("%d%d",&t1,&t2);

r1=findRoot(t1);

r2=findRoot(t2);

if(r1!=r2){

tree[r2]+=tree[r1];

tree[r1]=r2;

if(-1\*tree[r2]>max){

max=-1\*tree[r2];

}

}

}

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

}

return 0;

}

### 练习题：（题目1109）连通图

**题目描述：**

    给定一个无向图和其中的所有边，判断这个图是否所有顶点都是连通的。

**输入：**

    每组数据的第一行是两个整数 n 和 m（0<=n<=1000）。n 表示图的顶点数目，m 表示图中边的数目。如果 n 为 0 表示输入结束。随后有 m 行数据，每行有两个值 x 和 y（0<x, y <=n），表示顶点 x 和 y 相连，顶点的编号从 1 开始计算。输入不保证这些边是否重复。

**输出：**

    对于每组输入数据，如果所有顶点都是连通的，输出"YES"，否则输出"NO"。

**样例输入：**

4 3

1 2

2 3

3 2

3 2

1 2

2 3

0 0

**样例输出：**

NO

YES

#### 评价

简单

#### 代码

#include<stdio.h>

#define N 1001

int tree[N];

int findRoot(int x){

if(tree[x]<0)return x;

else{

int root=findRoot(tree[x]);

tree[x]=root;

return root;

}

}

int main(){

int n,m;

int t1,t2;

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

scanf("%d",&m);

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

tree[i]=-1;

}

int r1,r2;

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

scanf("%d%d",&t1,&t2);

r1=findRoot(t1);

r2=findRoot(t2);

if(r1!=r2){

tree[r1]=r2;

}

}

int cnt=0;

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

if(tree[i]==-1)cnt++;

}

if(cnt==1)printf("YES\n");

else printf("NO\n");

}

return 0;

}

### 练习题：（题目1446）Head of a Gang

**题目描述：**

One way that the police finds the head of a gang is to check people's phone calls. If there is a phone call between A and B, we say that A and B is related. The weight of a relation is defined to be the total time length of all the phone calls made between the two persons. A "Gang" is a cluster of more than 2 persons who are related to each other with total relation weight being greater than a given threthold K. In each gang, the one with maximum total weight is the head. Now given a list of phone calls, you are supposed to find the gangs and the heads.

**输入：**

For each case, the first line contains two positive numbers N and K (both less than or equal to 1000), the number of phone calls and the weight threthold, respectively. Then N lines follow, each in the following format:

Name1 Name2 Time

where Name1 and Name2 are the names of people at the two ends of the call, and Time is the length of the call. A name is a string of three capital letters chosen from A-Z. A time length is a positive integer which is no more than 1000 minutes.

**输出：**

For each test case, first print in a line the total number of gangs. Then for each gang, print in a line the name of the head and the total number of the members. It is guaranteed that the head is unique for each gang. The output must be sorted according to the alphabetical order of the names of the heads.

**样例输入：**

8 59

AAA BBB 10

BBB AAA 20

AAA CCC 40

DDD EEE 5

EEE DDD 70

FFF GGG 30

GGG HHH 20

HHH FFF 10

8 70

AAA BBB 10

BBB AAA 20

AAA CCC 40

DDD EEE 5

EEE DDD 70

FFF GGG 30

GGG HHH 20

HHH FFF 10

**样例输出：**

2

AAA 3

GGG 3

0

#### 评价

开始想不出怎么用并查集做，最后才明白，并查集干不了所有事，只能确定该组的人数及某人是不是组内的。并查集有结构体表示，这样信息会多一点

结构体Each代表人，each代表所有人，orderHead代表左右的Head

结构体Node代表组

#### 代码

#include<stdio.h>

#include<string.h>

#include<algorithm>

using namespace std;

#define MAX 1010

struct Each{

char name[5];

int w;

int size;

bool operator <(const Each &e)const{

int t=strcmp(name,e.name);

return t<0;

}

}each[MAX],orderHead[MAX];

struct Node{

int size;

int w;

int value;//指向

}node[MAX];//并查集

int findRoot(int x){

if(node[x].value==-1)return x;

else{

int root=findRoot(node[x].value);

node[x].value=root;

return root;

}

}

void init(int n){

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

node[i].size=1;

node[i].w=0;

node[i].value=-1;

each[i].w=0;

}

}

int main(){

int n,k;

char n1[5],n2[5];

int t;

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

int idx=0;

init(2\*n);

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

scanf("%s %s %d",n1,n2,&t);

//找字符串

int tmp1=-1,tmp2=-1;

for(int i=0,cnt=0;i<idx;i++){

if(strcmp(n1,each[i].name)==0){

tmp1=i;

cnt++;

}

if(strcmp(n2,each[i].name)==0){

tmp2=i;

cnt++;

}

if(cnt==2)break;

}

if(tmp1==-1){

strcpy(each[idx].name,n1);

tmp1=idx++;

}

if(tmp2==-1){

strcpy(each[idx].name,n2);

tmp2=idx++;

}

each[tmp1].w+=t;

each[tmp2].w+=t;

//弄并查集了

int r1=findRoot(tmp1);

int r2=findRoot(tmp2);

if(r1!=r2){

node[r1].value=r2;

node[r2].size+=node[r1].size;

node[r2].w+=node[r1].w+t;

}else{

node[r1].w+=t;

}

}

//3人以上，k以上的组

//组里w最高的成员

int group[MAX];

int iGroup=0;

int max[MAX]={0};//head的w

int order[MAX];//head的序号

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

if(node[i].value==-1&&node[i].size>2&&node[i].w>k){

group[iGroup++]=i;

}

}

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

int r=findRoot(i);

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

if(r==group[j]){

each[i].size=node[r].size;

if(max[j]<each[i].w){

max[j]=each[i].w;

order[j]=i;

break;

}

}

}

}

//所有head排序

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

orderHead[i]=each[order[i]];

}

sort(orderHead,orderHead+iGroup);

//output

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

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

printf("%s %d\n",orderHead[i].name,orderHead[i].size);

}

}

return 0;

}

## 三、最小生成树

### 题目1017：还是畅通工程

**题目描述：**

    某省调查乡村交通状况，得到的统计表中列出了任意两村庄间的距离。省政府“畅通工程”的目标是使全省任何两个村庄间都可以实现公路交通（但不一定有直接的公路相连，只要能间接通过公路可达即可），并要求铺设的公路总长度为最小。请计算最小的公路总长度。

**输入：**

    测试输入包含若干测试用例。每个测试用例的第1行给出村庄数目N ( < 100 )；随后的N(N-1)/2行对应村庄间的距离，每行给出一对正整数，分别是两个村庄的编号，以及此两村庄间的距离。为简单起见，村庄从1到N编号。  
    当N为0时，输入结束，该用例不被处理。

**输出：**

    对每个测试用例，在1行里输出最小的公路总长度。

**样例输入：**

3

1 2 1

1 3 2

2 3 4

4

1 2 1

1 3 4

1 4 1

2 3 3

2 4 2

3 4 5

0

**样例输出：**

3

5

#### 评价

受贱贱影响，用特别弱智的方法，反正也能过

核心思想是分为两个集合，并查集挺好的

Version 1.0用矩阵

Version 2.0用并查集

#### 代码

##### Version 1.0

#include<stdio.h>

#define MAX 110

#define MAXNUM 999999999

struct Node{

bool set;

int d;

}m[MAX][MAX];

void init(int n){

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

m[i][0].set=false;

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

m[i][j].d=MAXNUM;

}

}

}

int main(){

int n;

int n1,n2,d;

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

init(n);

int min=MAXNUM;

int s,e;

for(int i=0,tmp=n\*(n-1)/2;i<tmp;i++){

scanf("%d%d%d",&n1,&n2,&d);

m[n1][n2].d=d;

m[n2][n1].d=d;

if(min>d){

min=d;

s=n1;e=n2;

}

}

m[s][0].set=true;m[e][0].set=true;

int cnt=min;

int tmp=n-2;

while(tmp--!=0){

min=MAXNUM;

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

if(m[i][0].set==true){

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

if(m[j][0].set==false&&min>m[i][j].d){

min=m[i][j].d;

e=j;

}

}

}

}

m[e][0].set=true;

cnt+=min;

}

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

}

return 0;

}

##### Version 2.0

#include <cstdio>

#include <algorithm>

using namespace std;

#define N 100

int Tree[N];

int findRoot(int x){

if (Tree[x] == -1) return x;

else{

int tmp = findRoot(Tree[x]);

Tree[x] = tmp;

return tmp;

}

}

struct Edge{

int a, b;

int cost;

bool operator < (const Edge &A) const{

return cost < A.cost;

}

}edge[5500];

int main(){

int n;

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

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

Tree[i] = -1;

}

for (int i = 1; i <= n \* (n - 1) / 2; i++){

scanf("%d%d%d", &edge[i].a, &edge[i].b, &edge[i].cost);

}

sort(edge+1, edge+1+n\*(n-1)/2);

int ans = 0;

for (int i = 1; i <= n\*(n - 1) / 2; i++){

int a = findRoot(edge[i].a);

int b = findRoot(edge[i].b);

if (a != b){

Tree[a] = b;

ans += edge[i].cost;

}

}

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

}

return 0;

}

### 题目1144：Freckles

**题目描述：**

    In an episode of the Dick Van Dyke show, little Richie connects the freckles on his Dad's back to form a picture of the Liberty Bell. Alas, one of the freckles turns out to be a scar, so his Ripley's engagement falls through.   
    Consider Dick's back to be a plane with freckles at various (x,y) locations. Your job is to tell Richie how to connect the dots so as to minimize the amount of ink used. Richie connects the dots by drawing straight lines between pairs, possibly lifting the pen between lines. When Richie is done there must be a sequence of connected lines from any freckle to any other freckle.

**输入：**

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

**输出：**

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

**样例输入：**

3

1.0 1.0

2.0 2.0

2.0 4.0

**样例输出：**

3.41

#### 评价

中规中矩，用并查集写

Edge要大一点

#### 代码

#include<stdio.h>

#include<math.h>

#include<algorithm>

using namespace std;

#define MAX 110

struct Node{

double x,y;

}node[MAX];

struct Edge{

double d;

int a,b;

bool operator <(const Edge &e)const{

return d<e.d;

}

}edge[MAX\*MAX];

int s[MAX];

void init(int n){for(int i=0;i<n;i++)s[i]=-1;}

int findRoot(int x){

if(s[x]==-1)return x;

else{

int r=findRoot(s[x]);

s[x]=r;

return r;

}

}

int main(){

int n;

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

init(n);

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

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

}

int idx=0;

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

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

edge[idx].d=sqrt(pow((node[i].x-node[j].x),2)+pow((node[i].y-node[j].y),2));

edge[idx].a=i;

edge[idx++].b=j;

}

}

sort(edge,edge+idx);

int r1,r2;

double ans=0;

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

r1=findRoot(edge[i].a);

r2=findRoot(edge[i].b);

if(r1!=r2){

s[r1]=r2;

ans+=edge[i].d;

}

}

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

}

return 0;

}

## 四、最短路径

### 题目1447：最短路

**题目描述：**

在每年的校赛里，所有进入决赛的同学都会获得一件很漂亮的t-shirt。但是每当我们的工作人员把上百件的衣服从商店运回到赛场的时候，却是非常累的！所以现在他们想要寻找最短的从商店到赛场的路线，你可以帮助他们吗？

**输入：**

输入包括多组数据。每组数据第一行是两个整数N、M（N<=100，M<=10000），N表示成都的大街上有几个路口，标号为1的路口是商店所在地，标号为N的路口是赛场所在地，M则表示在成都有几条路。N=M=0表示输入结束。接下来M行，每行包括3个整数A，B，C（1<=A,B<=N,1<=C<=1000）,表示在路口A与路口B之间有一条路，我们的工作人员需要C分钟的时间走过这条路。输入保证至少存在1条商店到赛场的路线。  
当输入为两个0时，输入结束。

**输出：**

对于每组输入，输出一行，表示工作人员从商店走到赛场的最短时间。

**样例输入：**

2 1

1 2 3

3 3

1 2 5

2 3 5

3 1 2

0 0

**样例输出：**

3

2

#### 评价

**Version 1.0:**

Floyd算法

我好像init()经常忘掉

写的可以紧凑一点

**Version 2.0**

迪杰斯特拉算法

#### 代码

##### Version 1.0

#include<stdio.h>

#define MAX 110

#define INFINITY 999999999

int mat[MAX][MAX];

void init(int n){for(int i=1;i<=n;i++)for(int j=1;j<=n;j++)mat[i][j]=INFINITY;}

int main(){

int n,m;

int t1,t2,t;

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

init(n);

if(n==0&&m==0)break;

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

scanf("%d%d%d",&t1,&t2,&t);

mat[t1][t2]=mat[t2][t1]=t;

}

//dealing

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

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

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

if(mat[i][k]==INFINITY||mat[k][j]==INFINITY)continue;

if(mat[i][k]+mat[k][j]<mat[i][j])mat[i][j]=mat[i][k]+mat[k][j];

}

}

}

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

}

return 0;

}

##### Version 2.0

#include<stdio.h>

#include<vector>

#define INFINITY 999999999

using namespace std;

struct Edge{

int next;

int cost;

}tmp;

vector<Edge> edge[110];

bool mark[110];

int dis[110];

void init(int n){

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

edge[i].clear();

mark[i]=false;

dis[i]=-1;

}

}

int main(){

int n,m;

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

if(n==0&&m==0)break;

init(n);

int t1,t2,t;

while(m--!=0){

scanf("%d%d%d",&t1,&t2,&t);

tmp.cost=t;

tmp.next=t2;

edge[t1].push\_back(tmp);

tmp.next=t1;

edge[t2].push\_back(tmp);

}

int nextP=1;

dis[1]=0;

mark[1]=true;

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

for(int j=0;j<edge[nextP].size();j++){

int t=edge[nextP][j].next;

int c=edge[nextP][j].cost;

if(mark[t]==true)continue;

if(dis[t]==-1||dis[nextP]+c<dis[t])dis[t]=dis[nextP]+c;

}

int min=INFINITY;

for(int j=2;j<=n;j++){

if(mark[j]==true||dis[j]==-1)continue;

if(dis[j]<min){min=dis[j];nextP=j;}

}

mark[nextP]=true;

}

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

}

return 0;

}

### 题目1008：最短路径问题

**题目描述：**

给你n个点，m条无向边，每条边都有长度d和花费p，给你起点s终点t，要求输出起点到终点的最短距离及其花费，如果最短距离有多条路线，则输出花费最少的。

**输入：**

输入n,m，点的编号是1~n,然后是m行，每行4个数 a,b,d,p，表示a和b之间有一条边，且其长度为d，花费为p。最后一行是两个数 s,t;起点s，终点t。n和m为0时输入结束。  
(1<n<=1000, 0<m<100000, s != t)

**输出：**

输出 一行有两个数， 最短距离及其花费。

**样例输入：**

3 2

1 2 5 6

2 3 4 5

1 3

0 0

**样例输出：**

9 11

##### 评价

打//的两行是我犯过错的地方

小错误

##### 代码

#include<stdio.h>

#include<vector>

#define MAX 1001

#define INFINITY 999999999

using namespace std;

struct Edge{

int p;

int d;

int next;

}tmp;

vector<Edge> edge[MAX];

int mark[MAX];

int dis[MAX];

int price[MAX];

void init(int n){

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

edge[i].clear();mark[i]=false;dis[i]=-1;price[i]=-1;

}

}

int main(){

int n,m;

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

if(n==0&&m==0)break;

init(n);

int a,b,d,p;

while(m--){

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

tmp.d=d;tmp.p=p;tmp.next=b;

edge[a].push\_back(tmp);

tmp.next=a;

edge[b].push\_back(tmp);

}

int s,t;

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

int nextP=s;

mark[s]=true;

dis[s]=0;

price[s]=0;

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

for(int j=0;j<edge[nextP].size();j++){

int t=edge[nextP][j].next;

int d=edge[nextP][j].d;

int p=edge[nextP][j].p;

if(mark[t]==true)continue;

if(dis[t]==-1||dis[nextP]+d<=dis[t]){

if(dis[t]==-1||dis[nextP]+d<dis[t]){//

dis[t]=dis[nextP]+d;

price[t]=price[nextP]+p;

}else if(price[t]>price[nextP]+p){

price[t]=price[nextP]+p;

}

}

}

int min=INFINITY;

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

if(mark[j]==true||dis[j]==-1)continue;

if(dis[j]<min){

min=dis[j];nextP=j;

}

}

mark[nextP]=true;

}

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

}

return 0;

}

### 练习题：（题目1100）最短路径

**题目描述：**

N个城市，标号从0到N-1，M条道路，第K条道路（K从0开始）的长度为2^K，求编号为0的城市到其他城市的最短距离

**输入：**

第一行两个正整数N（2<=N<=100）M(M<=500),表示有N个城市，M条道路  
接下来M行两个整数，表示相连的两个城市的编号

**输出：**

N-1行，表示0号城市到其他城市的最短路，如果无法到达，输出-1，数值太大的以MOD 100000 的结果输出。

**样例输入：**

4 4

1 2

2 3

1 3

0 1

**样例输出：**

8

9

11

##### 评价

Version 1.0 数据类型全用大整数，内存大，时间长，9兆，360ms

Version 2.0 网上找的。最小生成树思想。由于路径长度的特殊性，更简单的办法是求最小生成树，求的过程中顺便求得最短路径。

##### 代码

##### Version 1.0

#include<stdio.h>

#include<math.h>

#define MAX 101

#define M 100000

struct BigInteger{

int digit[200];

int size;

void init(){

for(int i=0;i<200;i++)digit[i]=0;

size=0;

}

void set(int x){init();digit[0]=x;size=1;}

BigInteger operator \*(int x)const{

BigInteger ret;

ret.init();

int carry=0;

int t;

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

t=digit[i]\*x+carry;

carry=t/10000;

t%=10000;

ret.digit[ret.size++]=t;

}

if(carry!=0)ret.digit[ret.size++]=carry;

return ret;

}

BigInteger operator +(const BigInteger &x)const{

BigInteger ret;

ret.init();

int carry=0;

int t;

for(int i=0;i<size||i<x.size;i++){

t=digit[i]+x.digit[i]+carry;

carry=t/10000;

t%=10000;

ret.digit[ret.size++]=t;

}

if(carry!=0)ret.digit[ret.size++]=carry;

return ret;

}

int operator %(int x)const{

int carry=0;

for(int i=size-1;i>=0;i--){

carry=(carry\*10000+digit[i])%x;

}

return carry;

}

bool operator <(const BigInteger &b)const{

if(size!=b.size)return size<b.size;

for(int i=size-1;i>=0;i--){

if(digit[i]!=b.digit[i])return digit[i]<b.digit[i];

}

return true;

}

}mat[MAX][MAX];

BigInteger mulCnt(int cnt){

BigInteger ret;

ret.set(1);

while(cnt--!=0){

ret=ret\*2;

}

return ret;

}

void init(int n){

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

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

mat[i][j].set(-1);

}

}

}

int main(){

int n,m;

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

init(n);

int t1,t2,cnt=0;

while(m--){

scanf("%d%d",&t1,&t2);

mat[t1][t2]=mat[t2][t1]=mulCnt(cnt++);

}

for(int k=0;k<n;k++){

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

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

if(mat[i][k].size==1&&mat[i][k].digit[0]==-1||mat[k][j].size==1&&mat[k][j].digit[0]==-1)continue;

if(mat[i][j].size==1&&mat[i][j].digit[0]==-1||mat[i][k]+mat[k][j]<mat[i][j])mat[i][j]=mat[i][k]+mat[k][j];

}

}

}

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

if(mat[0][i].size!=1||mat[0][i].digit[0]!=-1)printf("%d\n",mat[0][i]%M);

else printf("-1\n");

}

}

return 0;

}

##### Version 2.0

#include <stdio.h>

#include <stdlib.h>

int rank[100] ;//记录每个树的深度

int pre[100];//记录每个点的父节点

int d[100][100];//记录各对间的距离

void initSet(int n)//初始化，将每个点的父节点设为自己，深度为1

{

int i;

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

{

rank[i] = 1;

pre[i] = i;

d[i][i] = 0;

}

}

int findSet(int x)//找到每个点的父节点，并将这个点的父节点设置为数的根节点

{

if(x != pre[x])

pre[x] = findSet(pre[x]);

return pre[x];

}

void unionSet(int a, int b)//合并树，

{

int x = findSet(a);

int y = findSet(b);

if(x == y)//如果两个节点的父节点（树的根节点）是同一个，无需合并，直接跳过

return;

if(rank[x] >= rank[y])

{

rank[x] += rank[y];

pre[y] = x;

}

else

{

rank[y] += rank[y];

pre[x] = y;

}//不是同一个树的的节点，小树合并到大树

}

int mod(int a, int b)//取模

{

int ret = 1;

while(b--)

ret = (ret\*a)%100000;

return ret;

}

int main()

{

int n, m, dist;

int x, y, a, b;

int i, j, k;

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

{

initSet(n);

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

{

scanf("%d%d", &x, &y);

a = findSet(x);

b = findSet(y);

if(a == b)//二者已在同一个连通分量，距离定是最小了

continue;

dist = mod(2, i);//取模

for(j=0; j<n; j++)//更新两个连通分量的各对经过中间对的距离

{

if(a != findSet(j))

continue;

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

{

if(b != findSet(k))

continue;

d[j][k] = d[k][j] = (d[j][x]+dist+d[y][k])%100000;

}

}

unionSet(x, y);

}

x = findSet(0);

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

if(findSet(i) != x)

printf("-1\n");

else

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

}

}

## 五、拓扑排序

### 题目1448：Legal or Not

**题目描述：**

ACM-DIY is a large QQ group where many excellent acmers get together. It is so harmonious that just like a big family. Every day,many "holy cows" like HH, hh, AC, ZT, lcc, BF, Qinz and so on chat on-line to exchange their ideas. When someone has questions, many warm-hearted cows like Lost will come to help. Then the one being helped will call Lost "master", and Lost will have a nice "prentice". By and by, there are many pairs of "master and prentice". But then problem occurs: there are too many masters and too many prentices, how can we know whether it is legal or not?We all know a master can have many prentices and a prentice may have a lot of masters too, it's legal. Nevertheless，some cows are not so honest, they hold illegal relationship. Take HH and 3xian for instant, HH is 3xian's master and, at the same time, 3xian is HH's master,which is quite illegal! To avoid this,please help us to judge whether their relationship is legal or not. Please note that the "master and prentice" relation is transitive. It means that if A is B's master ans B is C's master, then A is C's master.

**输入：**

The input consists of several test cases. For each case, the first line contains two integers, N (members to be tested) and M (relationships to be tested)(2 <= N, M <= 100). Then M lines follow, each contains a pair of (x, y) which means x is y's master and y is x's prentice. The input is terminated by N = 0.TO MAKE IT SIMPLE, we give every one a number (0, 1, 2,..., N-1). We use their numbers instead of their names.

**输出：**

For each test case, print in one line the judgement of the messy relationship.If it is legal, output "YES", otherwise "NO".

**样例输入：**

3 2

0 1

1 2

2 2

0 1

1 0

0 0

**样例输出：**

YES

NO

#### 评价

注意点就是入度和邻接矩阵

#### 代码

#include<stdio.h>

#include<queue>

#include<vector>

#define MAX 100

using namespace std;

int inDegree[MAX];

vector<int> edge[MAX];

queue<int> Q;

void init(int n){

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

inDegree[i]=0;

edge[i].clear();

}

}

int main(){

int n,m;

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

init(n);

int x,y;

while(m--){

scanf("%d%d",&x,&y);

inDegree[y]++;

edge[x].push\_back(y);

}

while(!Q.empty())Q.pop();

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

if(inDegree[i]==0)Q.push(i);

}

int cnt=0;

while(!Q.empty()){

int x=Q.front();

Q.pop();

cnt++;

for(int i=0;i<edge[x].size();i++){

int tmp=edge[x][i];

inDegree[tmp]--;

if(inDegree[tmp]==0)Q.push(tmp);

}

}

if(cnt!=n)printf("NO\n");

else printf("YES\n");

}

return 0;

}

### 练习题：（题目1449）确定比赛名次

**题目描述：**

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

**输入：**

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

**输出：**

给出一个符合要求的排名。输出时队伍号之间有空格，最后一名后面没有空格。

其他说明：符合条件的排名可能不是唯一的，此时要求输出时编号小的队伍在前；输入数据保证是正确的，即输入数据确保一定能有一个符合要求的排名。

**样例输入：**

4 3

1 2

2 3

4 3

**样例输出：**

1 2 4 3

#### 评价

要求输出编号小的队伍在前，优先队列

#### 代码

#include<stdio.h>

#include<vector>

#include<queue>

#define MAX 501

using namespace std;

vector<int> edge[MAX];

priority\_queue<int,vector<int>,greater<int> > Q;

int inDegree[MAX];

void init(int n){

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

edge[i].clear();

inDegree[i]=0;

}

}

int main(){

int n,m;

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

init(n);

int a,b;

while(m--){

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

edge[a].push\_back(b);

inDegree[b]++;

}

while(!Q.empty())Q.pop();

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

if(inDegree[i]==0)Q.push(i);

}

int cnt=0;

int arr[MAX];

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

int x=Q.top();

Q.pop();

arr[cnt++]=x;

for(int j=0;j<edge[x].size();j++){

int y=edge[x][j];

if(--inDegree[y]==0)Q.push(y);

}

}

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

if(i!=0)printf(" %d",arr[i]);

else printf("%d",arr[i]);

}

printf("\n");

}

return 0;

}

### 练习题：（题目1450）产生冠军

**题目描述：**

有一群人，打乒乓球比赛，两两捉对撕杀，每两个人之间最多打一场比赛。  
球赛的规则如下：  
如果A打败了B，B又打败了C，而A与C之间没有进行过比赛，那么就认定，A一定能打败C。  
如果A打败了B，B又打败了C，而且，C又打败了A，那么A、B、C三者都不可能成为冠军。  
根据这个规则，无需循环较量，或许就能确定冠军。你的任务就是面对一群比赛选手，在经过了若干场撕杀之后，确定是否已经实际上产生了冠军。

**输入：**

输入含有一些选手群，每群选手都以一个整数n(n<1000)开头，后跟n对选手的比赛结果，比赛结果以一对选手名字（中间隔一空格）表示，前者战胜后者。如果n为0，则表示输入结束。

**输出：**

对于每个选手群，若你判断出产生了冠军，则在一行中输出“Yes”，否则在一行中输出“No”。

**样例输入：**

3

Alice Bob

Smith John

Alice Smith

5

a c

c d

d e

b e

a d

0

**样例输出：**

Yes

No

#### 评价

没好好看题目，YES和NO与Yes和No

不需要邻接矩阵

While(n--)，n变了，我下面还接着用。。。，然后出错

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 2020

int inDegree[MAX];

char name[MAX][100];

void init(int n){

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

inDegree[i]=0;

}

}

int main(){

int n;

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

char a[100],b[100];

int idx=0;

init(2\*n);

while(n--){

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

int t1=-1,t2=-1,cnt=0;

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

if(strcmp(a,name[i])==0){t1=-2;cnt++;}

if(strcmp(b,name[i])==0){t2=i;cnt++;}

if(cnt==2)break;

}

if(t1==-1){strcpy(name[idx],a);t1=idx++;}

if(t2==-1){strcpy(name[idx],b);t2=idx++;}

inDegree[t2]++;

}

int cnt=0;

for(int i=0;i<idx;i++)if(inDegree[i]==0)cnt++;

if(cnt==1)printf("Yes\n");

else printf("No\n");

}

return 0;

}

# 第六章：搜索

## 一、枚举

## 题目1045：百鸡问题

**题目描述：**

    用小于等于n元去买100只鸡，大鸡5元/只，小鸡3元/只,还有1/3元每只的一种小鸡，分别记为x只,y只,z只。编程求解x,y,z所有可能解。

**输入：**

    测试数据有多组，输入n。

**输出：**

    对于每组输入,请输出x,y,z所有可行解，按照x，y，z依次增大的顺序输出。

**样例输入：**

40

**样例输出：**

x=0,y=0,z=100

x=0,y=1,z=99

x=0,y=2,z=98

x=1,y=0,z=99

#### 评价

这么简单，也能写的更好一点

#### 代码

#include<stdio.h>

int main(){

int n;

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

int idx=0;

for(int x=0;5\*x<=n;x++){

for(int y=0;3\*y<=n;y++){

for(int z=0;z<=n\*3;z++){

if(15\*x+9\*y+z<=3\*n&&x+y+z==100){

printf("x=%d,y=%d,z=%d\n",x,y,z);

}

}

}

}

}

return 0;

}

### 练习题：（题目1036）Old Bill

**题目描述：**

    Among grandfather's papers a bill was found.  
    72 turkeys $\_679\_  
    The first and the last digits of the number that obviously represented the total price of those turkeys are replaced here by blanks (denoted \_), for they are faded and are illegible. What are the two faded digits and what was the price of one turkey?  
    We want to write a program that solves a general version of the above problem.  
    N turkeys $\_XYZ\_  
    The total number of turkeys, N, is between 1 and 99, including both. The total price originally consisted of five digits, but we can see only the three digits in the middle. We assume that the first digit is nonzero, that the price of one turkeys is an integer number of dollars, and that all the  
turkeys cost the same price.  
    Given N, X, Y, and Z, write a program that guesses the two faded digits and the original price. In case that there is more than one candidate for the original price, the output should be the most expensive one. That is, the program is to report the two faded digits and the maximum price per turkey for the turkeys.

**输入：**

    The first line of the input file contains an integer N (0<N<100), which represents the number of turkeys. In the following line, there are the three decimal digits X, Y, and Z., separated by a space, of the original price $\_XYZ\_.

**输出：**

    For each case, output the two faded digits and the maximum price per turkey for the turkeys.

**样例输入：**

72

6 7 9

5

2 3 7

78

0 0 5

**样例输出：**

3 2 511

9 5 18475

0

#### 评价

这题目真是很奇怪

output the two faded digits and the maximum price per turkey for the turkeys.

我只能理解成前者后者都要最贵的

#### 代码

#include<stdio.h>

int main(){

int n,a,b,c;

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

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

int max=100000/n;

int cmp=(a\*100+b\*10+c);

int t;

int ans1=0,ans2=0,ans3=0;

for(int i=100;i<max;i++){

t=i\*n;

if(t/10%1000==cmp){

ans1=t/10000;ans2=t%10;ans3=i;

}

}

if(ans1!=0)printf("%d %d %d\n",ans1,ans2,ans3);else printf("0\n");

}

return 0;

}

## 二、广度优先搜索（BFS）

### 题目1456：胜利大逃亡

**题目描述：**

Ignatius被魔王抓走了,有一天魔王出差去了,这可是Ignatius逃亡的好机会.魔王住在一个城堡里,城堡是一个A\*B\*C的立方体,可以被表示成A个B\*C的矩阵,刚开始Ignatius被关在(0,0,0)的位置,离开城堡的门在(A-1,B-1,C-1)的位置,现在知道魔王将在T分钟后回到城堡,Ignatius每分钟能从一个坐标走到相邻的六个坐标中的其中一个.现在给你城堡的地图,请你计算出Ignatius能否在魔王回来前离开城堡(只要走到出口就算离开城堡,如果走到出口的时候魔王刚好回来也算逃亡成功),如果可以请输出需要多少分钟才能离开,如果不能则输出-1.

**输入：**

输入数据的第一行是一个正整数K,表明测试数据的数量.每组测试数据的第一行是四个正整数A,B,C和T(1<=A,B,C<=50,1<=T<=1000),它们分别代表城堡的大小和魔王回来的时间.然后是A块输入数据(先是第0块,然后是第1块,第2块......),每块输入数据有B行,每行有C个正整数,代表迷宫的布局,其中0代表路,1代表墙。

**输出：**

对于每组测试数据,如果Ignatius能够在魔王回来前离开城堡,那么请输出他最少需要多少分钟,否则输出-1.

**样例输入：**

1

3 3 4 20

0 1 1 1

0 0 1 1

0 1 1 1

1 1 1 1

1 0 0 1

0 1 1 1

0 0 0 0

0 1 1 0

0 1 1 0

**样例输出：**

11

#### 评价

疲劳驾驶

Mark[][][]走过的块，maze[][][]迷宫

广度优先搜索，队列

#### 代码

#include<stdio.h>

#include<queue>

using namespace std;

#define MAX 50

struct N{

int x,y,z;

int t;

};

int maze[MAX][MAX][MAX];

bool mark[MAX][MAX][MAX];

queue<N> Q;

int go[][3]={

1,0,0,

0,1,0,

0,0,1,

-1,0,0,

0,-1,0,

0,0,-1

};

int BFS(int a,int b,int c){

while(!Q.empty()){

N tmp=Q.front();

Q.pop();

for(int i=0;i<6;i++){

int nx=tmp.x+go[i][0],ny=tmp.y+go[i][1],nz=tmp.z+go[i][2];

if(nx<0||ny<0||nz<0||nx>=a||ny>=b||nz>=c)continue;

if(maze[nx][ny][nz]==1)continue;

if(mark[nx][ny][nz]==true)continue;

if(nx==a-1&&ny==b-1&&nz==c-1)return tmp.t+1;

N nt;

nt.x=nx;nt.y=ny;nt.z=nz;nt.t=tmp.t+1;

Q.push(nt);

mark[nx][ny][nz]=true;

}

}

return -1;

}

int main(){

int k;

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

int a,b,c,t;

while(k--){

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

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

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

for(int k=0;k<c;k++){

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

mark[i][j][k]=false;

}

}

}

if(a==1&&b==1&&c==1){printf("0\n");continue;}

while(!Q.empty())Q.pop();

mark[0][0][0]=true;

N tmp;

tmp.x=tmp.y=tmp.z=tmp.t=0;

Q.push(tmp);

int ans=BFS(a,b,c);

if(ans<=t)printf("%d\n",ans);else printf("-1\n");

}

}

return 0;

}

### 题目1457：非常可乐

**题目描述：**

大家一定觉的运动以后喝可乐是一件很惬意的事情，但是seeyou却不这么认为。因为每次当seeyou买了可乐以后，阿牛就要求和seeyou一起分享这一瓶可乐，而且一定要喝的和seeyou一样多。但seeyou的手中只有两个杯子，它们的容量分别是N 毫升和M 毫升 可乐的体积为S （S<101）毫升(正好装满一瓶) ，它们三个之间可以相互倒可乐 (都是没有刻度的，且 S==N+M，101＞S＞0，N＞0，M＞0) 。聪明的ACMER你们说他们能平分吗？如果能请输出倒可乐的最少的次数，如果不能输出"NO"。

**输入：**

三个整数 : S 可乐的体积 , N 和 M是两个杯子的容量，以"0 0 0"结束。

**输出：**

如果能平分的话请输出最少要倒的次数，否则输出"NO"。

**样例输入：**

7 4 3

4 1 3

0 0 0

**样例输出：**

NO

3

#### 评价

细节

还有，平分有歧义。题目意思是两个容易都有一半。

我理解的是一个有一半，另两个合一起

#### 代码

#include<stdio.h>

#include<queue>

using namespace std;

#define MAX 110

struct B{

int s,n,m;

int t;

};

queue<B> Q;

bool mark[MAX][MAX][MAX];

void init(int s,int n,int m){

for(int i=0;i<=s;i++)for(int j=0;j<=n;j++)for(int k=0;k<=m;k++)mark[i][j][k]=false;

}

void aToB(int sb,int &a,int &b){

if(a>sb-b){

a-=sb-b;

b=sb;

}else{

b+=a;

a=0;

}

}

int BFS(int half,int s,int n,int m){

while(!Q.empty()){

B tmp,b=Q.front();

Q.pop();

//s to n

tmp=b;

if(tmp.s!=0){

aToB(n,tmp.s,tmp.n);

if(mark[tmp.s][tmp.n][tmp.m]==false){

tmp.t++;

if(tmp.s==half&&tmp.n==half)return tmp.t;

if(tmp.s==half&&tmp.m==half)return tmp.t;

if(tmp.n==half&&tmp.m==half)return tmp.t;

Q.push(tmp);

mark[tmp.s][tmp.n][tmp.m]=true;

}

}

//s to m

tmp=b;

if(tmp.s!=0){

aToB(m,tmp.s,tmp.m);

if(mark[tmp.s][tmp.n][tmp.m]==false){

tmp.t++;

if(tmp.s==half&&tmp.n==half)return tmp.t;

if(tmp.s==half&&tmp.m==half)return tmp.t;

if(tmp.n==half&&tmp.m==half)return tmp.t;

Q.push(tmp);

mark[tmp.s][tmp.n][tmp.m]=true;

}

}

//n to s

tmp=b;

if(tmp.n!=0){

aToB(s,tmp.n,tmp.s);

if(mark[tmp.s][tmp.n][tmp.m]==false){

tmp.t++;

if(tmp.s==half&&tmp.n==half)return tmp.t;

if(tmp.s==half&&tmp.m==half)return tmp.t;

if(tmp.n==half&&tmp.m==half)return tmp.t;

Q.push(tmp);

mark[tmp.s][tmp.n][tmp.m]=true;

}

}

//n to m

tmp=b;

if(tmp.n!=0){

aToB(m,tmp.n,tmp.m);

if(mark[tmp.s][tmp.n][tmp.m]==false){

tmp.t++;

if(tmp.s==half&&tmp.n==half)return tmp.t;

if(tmp.s==half&&tmp.m==half)return tmp.t;

if(tmp.n==half&&tmp.m==half)return tmp.t;

Q.push(tmp);

mark[tmp.s][tmp.n][tmp.m]=true;

}

}

//m to s

tmp=b;

if(tmp.m!=0){

aToB(s,tmp.m,tmp.s);

if(mark[tmp.s][tmp.n][tmp.m]==false){

tmp.t++;

if(tmp.s==half&&tmp.n==half)return tmp.t;

if(tmp.s==half&&tmp.m==half)return tmp.t;

if(tmp.n==half&&tmp.m==half)return tmp.t;

Q.push(tmp);

mark[tmp.s][tmp.n][tmp.m]=true;

}

}

//m to n

tmp=b;

if(tmp.m!=0){

aToB(n,tmp.m,tmp.n);

if(mark[tmp.s][tmp.n][tmp.m]==false){

tmp.t++;

if(tmp.s==half&&tmp.n==half)return tmp.t;

if(tmp.s==half&&tmp.m==half)return tmp.t;

if(tmp.n==half&&tmp.m==half)return tmp.t;

Q.push(tmp);

mark[tmp.s][tmp.n][tmp.m]=true;

}

}

}

return -1;

}

int main(){

int s,n,m;

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

if(s==0)break;

if(s%2==1){printf("NO\n");continue;}

init(s,n,m);

while(!Q.empty())Q.pop();

B b;

b.s=s;

b.n=b.m=b.t=0;

Q.push(b);

mark[s][0][0]=true;

int ans=BFS(s/2,s,n,m);

if(ans!=-1)printf("%d\n",ans);else printf("NO\n");

}

return 0;

}

## 三、递归

### 题目1458：汉诺塔III

**题目描述：**

约19世纪末，在欧州的商店中出售一种智力玩具，在一块铜板上有三根杆，最左边的杆上自上而下、由小到大顺序串着由64个圆盘构成的塔。目的是将最左边杆上的盘全部移到右边的杆上，条件是一次只能移动一个盘，且不允许大盘放在小盘的上面。现在我们改变游戏的玩法，不允许直接从最左(右)边移到最右(左)边(每次移动一定是移到中间杆或从中间移出)，也不允许大盘放到下盘的上面。Daisy已经做过原来的汉诺塔问题和汉诺塔II，但碰到这个问题时，她想了很久都不能解决，现在请你帮助她。现在有N个圆盘，她至少多少次移动才能把这些圆盘从最左边移到最右边？

**输入：**

包含多组数据，每次输入一个N值(1<=N=35)。

**输出：**

对于每组数据，输出移动最小的次数。

**样例输入：**

1

3

12

**样例输出：**

2

26

531440

#### 评价

汉诺塔见一次傻一次，还好，我总算是想出来了

看N值最大35，int要溢出，所以long long

思路：(version 1.0)

每走动一个，底下大的动一下，小的动3下

书上的思路：(version 2.0)

每走两格算一下。即从一端到另一端，底下大的动2下，小的动6下，6下是3个来回。

#### 代码

##### Version 1.0

#include<stdio.h>

long long fun(long long n){

if(n==1)return 1;

else {

return 1+3\*fun(n-1);

}

}

int main(){

long long n;

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

long long cnt=2\*fun(n);

printf("%lld\n",cnt);

}

return 0;

}

##### Version 2.0

#include<stdio.h>

long long fun(long long n){

if(n==1)return 2;

else {

return 2+3\*fun(n-1);

}

}

int main(){

long long n;

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

long long cnt=fun(n);

printf("%lld\n",cnt);

}

return 0;

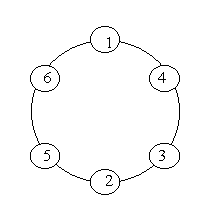
}

## 四、递归的应用

### 题目1459：Prime ring problem

**题目描述：**

A ring is compose of n circles as shown in diagram. Put natural number 1, 2, ..., n into each circle separately, and the sum of numbers in two adjacent circles should be a prime.  
Note: the number of first circle should always be 1.



**输入：**

n (1 < n < 17).

**输出：**

The output format is shown as sample below. Each row represents a series of circle numbers in the ring beginning from 1 clockwisely and anticlockwisely. The order of numbers must satisfy the above requirements. Print solutions in lexicographical order.  
You are to write a program that completes above process.  
Print a blank line after each case.

**样例输入：**

6

8

**样例输出：**

Case 1:

1 4 3 2 5 6

1 6 5 2 3 4

Case 2:

1 2 3 8 5 6 7 4

1 2 5 8 3 4 7 6

1 4 7 6 5 8 3 2

1 6 7 4 3 8 5 2

**提示：**

用printf打印输出。

#### 评价

630ms，大家都差不多

递归——回溯法枚举

#### 代码

#include<stdio.h>

int ans[20];

int Prime[13]={2,3,5,7,11,13,17,19,23,29,31,37};

int n;

int hash[20];

bool isPrime(int n){

for(int i=0;i<13;i++)if(Prime[i]==n)return true;

return false;

}

void check(){

if(isPrime(ans[1]+ans[n])==false)return;

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

if(i!=1)printf(" ");

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

}

printf("\n");

}

void DFS(int num){

if(num>1)if(isPrime(ans[num]+ans[num-1])==false)return;

if(num==n)check();

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

if(hash[i]==1)continue;

hash[i]=1;

ans[num+1]=i;

DFS(num+1);

hash[i]=0;

}

}

int main(){

int Case=0;

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

Case++;

for(int i=0;i<=n;i++)hash[i]=0;

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

hash[1]=1;

ans[1]=1;

DFS(1);

printf("\n");

}

return 0;

}

### 题目1460：Oil Deposit

题目描述：

The GeoSurvComp geologic survey company is responsible for detecting underground oil deposits. GeoSurvComp works with one large rectangular region of land at a time, and creates a grid that divides the land into numerous square plots. It then analyzes each plot separately, using sensing equipment to determine whether or not the plot contains oil. A plot containing oil is called a pocket. If two pockets are adjacent, then they are part of the same oil deposit. Oil deposits can be quite large and may contain numerous pockets. Your job is to determine how many different oil deposits are contained in a grid.

输入：

The input file contains one or more grids. Each grid begins with a line containing m and n, the number of rows and columns in the grid, separated by a single space. If m = 0 it signals the end of the input; otherwise 1 <= m <= 100 and 1 <= n <= 100. Following this are m lines of n characters each (not counting the end-of-line characters). Each character corresponds to one plot, and is either `\*', representing the absence of oil, or `@', representing an oil pocket.

输出：

For each grid, output the number of distinct oil deposits. Two different pockets are part of the same oil deposit if they are adjacent horizontally, vertically, or diagonally. An oil deposit will not contain more than 100 pockets.

样例输入：

1 1

\*

3 5

\*@\*@\*

\*\*@\*\*

\*@\*@\*

1 8

@@\*\*\*\*@\*

5 5

\*\*\*\*@

\*@@\*@

\*@\*\*@

@@@\*@

@@\*\*@

0 0

样例输出：

0

1

2

2

#### 评价

九度系统跟字符有关的，无力吐槽

发现scanf("%s",&maze[i]);前面放不放1个getchar()没区别

Version 1.0并查集

Version 2.0 DFS

#### 代码

##### Version 1.0

#include<stdio.h>

#define MAX 110

int pocket[MAX\*MAX];

int findRoot(int x){

if(pocket[x]==-1)return x;

else {

int root=findRoot(pocket[x]);

pocket[x]=root;

return root;

}

}

int main(){

int m,n;

char c;

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

if(m==0)break;

getchar();

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

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

scanf("%c",&c);

if(c=='@'){

pocket[i\*n+j]=-1;

}else{

pocket[i\*n+j]=-2;

}

}

getchar();

}

int square=m\*n;

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

if(pocket[i]>=-1){

if(i%n<=n-2)//右边

if(pocket[i+1]>=-1){

int r1=findRoot(i);

int r2=findRoot(i+1);

if(r1!=r2)pocket[r1]=r2;

}

if(i+n-1<square&&i%n>=1)//左下

if(pocket[i+n-1]>=-1){

int r1=findRoot(i);

int r2=findRoot(i+n-1);

if(r1!=r2)pocket[r1]=r2;

}

if(i+n<square&&pocket[i+n]>=-1){//下

int r1=findRoot(i);

int r2=findRoot(i+n);

if(r1!=r2)pocket[r1]=r2;

}

if(i+n+1<square&&i%n<=n-2&&pocket[i+n+1]>=-1){//右下

int r1=findRoot(i);

int r2=findRoot(i+n+1);

if(r1!=r2)pocket[r1]=r2;

}

}

}

int cnt=0;

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

// printf("%d ",pocket[i]);

if(pocket[i]==-1)cnt++;

}

// printf("\n");

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

}

return 0;

}

##### Version 2.0

#include<stdio.h>

#define MAX 100

char maze[MAX][MAX];

bool mark[MAX][MAX];

int m,n;

int go[8][2]={

1,0,

-1,0,

0,1,

0,-1,

1,1,

1,-1,

-1,1,

-1,-1

};

void DFS(int x,int y){

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

int nx=x+go[i][0],ny=y+go[i][1];

if(nx<0||ny<0||nx>=m||ny>=n)continue;

if(maze[nx][ny]!='@')continue;

if(mark[nx][ny]==true)continue;

mark[nx][ny]=true;

DFS(nx,ny);

}

}

int main(){

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

if(m==0)break;

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

scanf("%s",&maze[i]);

}

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

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

mark[i][j]=false;

}

}

int ans=0;

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

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

if(maze[i][j]=='\*')continue;

if(mark[i][j]==true)continue;

mark[i][j]=true;

DFS(i,j);

ans++;

}

}

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

}

return 0;

}

### 练习题：（题目1120）全排列

**题目描述：**

给定一个由不同的小写字母组成的字符串，输出这个字符串的所有全排列。  
我们假设对于小写字母有'a' < 'b' < ... < 'y' < 'z'，而且给定的字符串中的字母已经按照从小到大的顺序排列。

**输入：**

输入只有一行，是一个由不同的小写字母组成的字符串，已知字符串的长度在1到6之间。

**输出：**

输出这个字符串的所有排列方式，每行一个排列。要求字母序比较小的排列在前面。字母序如下定义：  
已知S = s1s2...sk , T = t1t2...tk，则S < T 等价于，存在p (1 <= p <= k)，使得  
s1 = t1, s2 = t2, ..., sp - 1 = tp - 1, sp < tp成立。

**样例输入：**

abc

**样例输出：**

abc

acb

bac

bca

cab

cba

**提示：**

每组样例输出结束后要再输出一个回车。

#### 评价

#### 代码

#include<stdio.h>

#include<string.h>

#define M 10

bool mark[M];

int len;

char a[M];

char b[M];

void DFS(int num){

if(num==len){

b[num]=0;

printf("%s\n",b);

return;

}

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

if(mark[i]==true)continue;

b[num]=a[i];

mark[i]=true;

DFS(num+1);

mark[i]=false;

}

}

int main(){

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

len=strlen(a);

for(int i=0;i<len;i++)mark[i]=false;

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

b[0]=a[i];

mark[i]=true;

DFS(1);

mark[i]=false;

}

printf("\n");

}

return 0;

}

### 五、深度优先搜索（DFS）

**题目描述：**

The doggie found a bone in an ancient maze, which fascinated him a lot. However, when he picked it up, the maze began to shake, and the doggie could feel the ground sinking. He realized that the bone was a trap, and he tried desperately to get out of this maze.  
The maze was a rectangle with sizes N by M. There was a door in the maze. At the beginning, the door was closed and it would open at the T-th second for a short period of time (less than 1 second). Therefore the doggie had to arrive at the door on exactly the T-th second. In every second, he could move one block to one of the upper, lower, left and right neighboring blocks. Once he entered a block, the ground of this block would start to sink and disappear in the next second. He could not stay at one block for more than one second, nor could he move into a visited block. Can the poor doggie survive? Please help him.

**输入：**

The input consists of multiple test cases. The first line of each test case contains three integers N, M, and T (1 < N, M < 7; 0 < T < 50), which denote the sizes of the maze and the time at which the door will open, respectively. The next N lines give the maze layout, with each line containing M characters. A character is one of the following:  
'X': a block of wall, which the doggie cannot enter;   
'S': the start point of the doggie;   
'D': the Door; or  
'.': an empty block.  
The input is terminated with three 0's. This test case is not to be processed.

**输出：**

For each test case, print in one line "YES" if the doggie can survive, or "NO" otherwise.

**样例输入：**

4 4 5

S.X.

..X.

..XD

....

3 4 5

S.X.

..X.

...D

0 0 0

**样例输出：**

NO

YES

**提示：**

 用scanf读取输入。

#### 评价

跟之前的差不多

#### 代码

#include<stdio.h>

#define MAX 10

int go[][2]={

1,0,

-1,0,

0,1,

0,-1

};

char maze[MAX][MAX];

bool mark[MAX][MAX];

int n,m,t;

int DFS(int x,int y,int time){

if(maze[x][y]=='D'&&time==t){printf("YES\n");return 1;}

if(time==t&&maze[x][y]!='D')return 0;

if(maze[x][y]=='D'&&time!=t)return 0;

for(int i=0;i<4;i++){

int nx=x+go[i][0],ny=y+go[i][1];

if(nx<0||ny<0||nx>=n||ny>=m)continue;

if(mark[nx][ny]==true)continue;

if(maze[nx][ny]=='X')continue;

mark[nx][ny]=true;

if(DFS(nx,ny,time+1)==1)return 1;

mark[nx][ny]=false;

}

return 0;

}

int main(){

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

if(n==0)break;

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

scanf("%s",maze[i]);

}

int t1=-1,t2=-1;

int ti,tj;

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

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

mark[i][j]=false;

if(maze[i][j]=='S'){t1=i+j;ti=i;tj=j;}

if(maze[i][j]=='D')t2=i+j;

}

}

if(t1%2!=t2%2&&t%2==0||t1%2==t2%2&&t%2==1){printf("NO\n");continue;}

mark[ti][tj]=true;

if(DFS(ti,tj,0)==0)printf("NO\n");

}

return 0;

}

# 第七章：动态规划

## 一、递推求解

### 题目1205：N阶楼梯上楼问题

**题目描述：**

N阶楼梯上楼问题：一次可以走两阶或一阶，问有多少种上楼方式。（要求采用非递归）

**输入：**

输入包括一个整数N,(1<=N<90)。

**输出：**

可能有多组测试数据，对于每组数据，  
输出当楼梯阶数是N时的上楼方式个数。

**样例输入：**

4

**样例输出：**

5

##### 评价

到89时int溢出，要long long。要用大的试一下

#### 代码

#include<stdio.h>

#define MAX 90

long long ans[MAX];

void init(){

ans[1]=1;ans[2]=2;

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

ans[i]=ans[i-1]+ans[i-2];

}

}

int main(){

int n;

init();

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

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

}

return 0;

}

### 题目1451：不容易系列之一

**题目描述：**

大家常常感慨，要做好一件事情真的不容易，确实，失败比成功容易多了！  
做好“一件”事情尚且不易，若想永远成功而总从不失败，那更是难上加难了，就像花钱总是比挣钱容易的道理一样。  
话虽这样说，我还是要告诉大家，要想失败到一定程度也是不容易的。比如，我高中的时候，就有一个神奇的女生，在英语考试的时候，竟然把40个单项选择题全部做错了！大家都学过概率论，应该知道出现这种情况的概率，所以至今我都觉得这是一件神奇的事情。如果套用一句经典的评语，我们可以这样总结：一个人做错一道选择题并不难，难的是全部做错，一个不对。

不幸的是，这种小概率事件又发生了，而且就在我们身边：  
事情是这样的——HDU有个网名叫做8006的男性同学，结交网友无数，最近该同学玩起了浪漫，同时给n个网友每人写了一封信，这都没什么，要命的是，他竟然把所有的信都装错了信封！注意了，是全部装错哟！

现在的问题是：请大家帮可怜的8006同学计算一下，一共有多少种可能的错误方式呢？

**输入：**

输入数据包含多个多个测试实例，每个测试实例占用一行，每行包含一个正整数n（1<n<=20），n表示8006的网友的人数。

**输出：**

对于每行输入请输出可能的错误方式的数量，每个实例的输出占用一行。

**样例输入：**

2

3

**样例输出：**

1

2

#### 评价

没想出来

排列组合本来就差。。。

#### 代码

#include<stdio.h>

#define MAX 21

int main(){

int n;

long long ans[MAX];

ans[1]=0;ans[2]=1;

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

ans[i]=(i-1)\*(ans[i-1]+ans[i-2]);

}

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

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

}

return 0;

}

## 二、最长递增子序列

### 题目1112：拦截导弹

**题目描述：**

某国为了防御敌国的导弹袭击，开发出一种导弹拦截系统。但是这种导弹拦截系统有一个缺陷：虽然它的第一发炮弹能够到达任意的高度，但是以后每一发炮弹都不能高于前一发的高度。某天，雷达捕捉到敌国的导弹来袭，并观测到导弹依次飞来的高度，请计算这套系统最多能拦截多少导弹。拦截来袭导弹时，必须按来袭导弹袭击的时间顺序，不允许先拦截后面的导弹，再拦截前面的导弹。

**输入：**

每组输入有两行，

第一行，输入雷达捕捉到的敌国导弹的数量k（k<=25），

第二行，输入k个正整数，表示k枚导弹的高度，按来袭导弹的袭击时间顺序给出，以空格分隔。

**输出：**

每组输出只有一行，包含一个整数，表示最多能拦截多少枚导弹。

**样例输入：**

8

300 207 155 300 299 170 158 65

**样例输出：**

6

#### 评价

复杂度O(n^2)

#### 代码

#include<stdio.h>

#define MAX 26

int main(){

int k,num[MAX];

int a[MAX];

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

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

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

}

int max=1;

a[0]=1;

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

a[i]=1;

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

if(num[i]<=num[j]){

if(a[i]<a[j]+1){

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

if(max<a[i])

max=a[i];

}

}

}

}

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

}

return 0;

}

### 练习题：（题目1131）合唱队形

**题目描述：**

N位同学站成一排，音乐老师要请其中的(N-K)位同学出列，使得剩下的K位同学不交换位置就能排成合唱队形。  
合唱队形是指这样的一种队形：设K位同学从左到右依次编号为1, 2, …, K，他们的身高分别为T1, T2, …, TK，  
则他们的身高满足T1 < T2 < … < Ti , Ti > Ti+1 > … > TK (1 <= i <= K)。  
你的任务是，已知所有N位同学的身高，计算最少需要几位同学出列，可以使得剩下的同学排成合唱队形。

**输入：**

输入的第一行是一个整数N（2 <= N <= 100），表示同学的总数。  
第一行有n个整数，用空格分隔，第i个整数Ti（130 <= Ti <= 230）是第i位同学的身高（厘米）。

**输出：**

可能包括多组测试数据，对于每组数据，  
输出包括一行，这一行只包含一个整数，就是最少需要几位同学出列。

**样例输入：**

8

186 186 150 200 160 130 197 220

**样例输出：**

4

#### 评价

700ms ……

#### 代码

#include<stdio.h>

#define MAX 110

int main(){

int list[MAX];

int a[MAX],b[MAX];

int n;

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

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

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

}

a[0]=1;

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

a[i]=1;

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

if(list[i]>list[j]){

if(a[i]<a[j]+1)a[i]=a[j]+1;

}

}

}

b[n-1]=1;

for(int i=n-2;i>=0;i--){

b[i]=1;

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

if(list[i]>list[j]){

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

}

}

}

int min=999999999;

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

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

a[i]=n-a[i];

if(a[i]<min)min=a[i];

}

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

}

return 0;

}

## 三、最长公共子序列

### 题目1042：Coincidence

**题目描述：**

Find a longest common subsequence of two strings.

**输入：**

First and second line of each input case contain two strings of lowercase character a…z. There are no spaces before, inside or after the strings. Lengths of strings do not exceed 100.

**输出：**

For each case, output k – the length of a longest common subsequence in one line.

**样例输入：**

abcd

cxbydz

**样例输出：**

2

#### 评价

C[i][j]中i，j指的是第i，j个数，c[i][j]的值是s1的前i个字符和s2的前j个字符的最长递增子序列的值

#### 代码

#include<stdio.h>

#include<string.h>

#define MAX 101

char a[MAX],b[MAX];

int c[MAX][MAX];

int max(int x,int y){return x>y?x:y;}

int main(){

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

int lena=strlen(a),lenb=strlen(b);

for(int i=0;i<lenb;i++)c[0][MAX]=0;

for(int i=0;i<lena;i++)c[MAX][0]=0;

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

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

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

c[i+1][j+1]=c[i][j]+1;

}else c[i+1][j+1]=max(c[i][j+1],c[i+1][j]);

}

}

printf("%d\n",c[lena][lenb]);

}

return 0;

}

## 五、动态规划问题分析举例

### 题目1452：搬寝室

**题目描述：**

搬寝室是很累的,xhd深有体会.时间追述2006年7月9号,那天xhd迫于无奈要从27号楼搬到3号楼,因为10号要封楼了.看着寝室里的n件物品,xhd开始发呆,因为n是一个小于2000的整数,实在是太多了,于是xhd决定随便搬2\*k件过去就行了.但还是会很累,因为2\*k也不小是一个不大于n的整数.幸运的是xhd根据多年的搬东西的经验发现每搬一次的疲劳度是和左右手的物品的重量差的平方成正比(这里补充一句,xhd每次搬两件东西,左手一件右手一件).例如xhd左手拿重量为3的物品,右手拿重量为6的物品,则他搬完这次的疲劳度为(6-3)^2 = 9.现在可怜的xhd希望知道搬完这2\*k件物品后的最佳状态是怎样的(也就是最低的疲劳度),请告诉他吧。

**输入：**

每组输入数据有两行,第一行有两个数n,k(2<=2\*k<=n<2000).第二行有n个整数分别表示n件物品的重量(重量是一个小于2^15的正整数).

**输出：**

对应每组输入数据,输出数据只有一个表示他的最少的疲劳度,每个一行.

**样例输入：**

2 1

1 3

**样例输出：**

4

#### 评价

首先要想到

#### 代码

#include<stdio.h>

#include<algorithm>

using namespace std;

int a[2002];

long long ans[1001][2002];

long long min(int x,int y){

return x<y?x:y;

}

int main(){

int n,k;

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

for(int i=1;i<=n;i++)scanf("%d",&a[i]);

sort(a+1,a+n+1);

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

ans[0][n]=0;

}

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

ans[i][2\*i]=ans[i-1][2\*i-2]+(a[i\*2]-a[i\*2-1])\*(a[i\*2]-a[i\*2-1]);

}

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

for(int j=2\*i+1;j<=n;j++){

ans[i][j]=min(ans[i][j-1],ans[i-1][j-2]+(a[j]-a[j-1])\*(a[j]-a[j-1]));

}

}

printf("%lld\n",ans[k][n]);

}

return 0;

}

### 题目1453：Greedy Tino

**题目描述：**

 Tino wrote a long long story. BUT! in Chinese...  
So I have to tell you the problem directly and discard his long long story. That is tino want to carry some oranges with "Carrying pole", and he must make two side of the Carrying pole are the same weight. Each orange have its' weight. So greedy tino want to know the maximum weight he can carry.

**输入：**

The first line of input contains a number t, which means there are t cases of the test data.  
for each test case, the first line contain a number n, indicate the number of oranges.  
the second line contains n numbers, Wi, indicate the weight of each orange  
n is between 1 and 100, inclusive. Wi is between 0 and 2000, inclusive. the sum of Wi is equal or less than 2000.

**输出：**

For each test case, output the maximum weight in one side of Carrying pole. If you can't carry any orange, output -1. Output format is shown in Sample Output.

**样例输入：**

1

5

1 2 3 4 5

**样例输出：**

Case 1: 7

#### 评价

得再做一遍

Define的不要太多才好

一直都用熟悉的变量不要临时改

OFFSET写着真不习惯

#### 代码

#include<stdio.h>

#define OFFSET 2000

#define INFINITY 999999999

int dp[101][4001];

int w[101];

int main(){

int t;

int cas=0;

scanf("%d",&t);

while(t--){

int n;

scanf("%d",&n);

bool hasZero=false;

int cnt=0;

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

scanf("%d",&w[++cnt]);

if(w[cnt]==0){cnt--;hasZero=true;}

}

for(int i=-2000;i<=2000;i++)dp[0][i+OFFSET]=-INFINITY;

dp[0][0+OFFSET]=0;

n=cnt;

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

for(int j=-2000;j<=2000;j++){

int tmp1=-INFINITY,tmp2=-INFINITY;

if(j+w[i]<=2000&&dp[i-1][j+w[i]+OFFSET]!=-INFINITY){

tmp1=dp[i-1][j+w[i]+OFFSET]+w[i];

}

if(j-w[i]>=-2000&&dp[i-1][j-w[i]+OFFSET]!=-INFINITY){

tmp2=dp[i-1][j-w[i]+OFFSET]+w[i];

}

if(tmp1<tmp2)tmp1=tmp2;

if(tmp1<dp[i-1][j+OFFSET])tmp1=dp[i-1][j+OFFSET];

dp[i][j+OFFSET]=tmp1;

}

}

printf("Case %d: ",++cas);

if(dp[n][0+OFFSET]!=0)printf("%d\n",dp[n][OFFSET]/2);

else if(hasZero==true)printf("0\n");

else printf("-1\n");

}

return 0;

}

## 六、背包

### 题目1123：采药

**题目描述：**

辰辰是个很有潜能、天资聪颖的孩子，他的梦想是称为世界上最伟大的医师。  
为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。  
医师把他带到个到处都是草药的山洞里对他说：  
“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。  
我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”  
如果你是辰辰，你能完成这个任务吗？

**输入：**

输入的第一行有两个整数T（1 <= T <= 1000）和M（1 <= M <= 100），T代表总共能够用来采药的时间，M代表山洞里的草药的数目。  
接下来的M行每行包括两个在1到100之间（包括1和100）的的整数，分别表示采摘某株草药的时间和这株草药的价值。

**输出：**

可能有多组测试数据，对于每组数据，  
输出只包括一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

**样例输入：**

70 3

71 100

69 1

1 2

**样例输出：**

3

#### 评价

Version 1.0 二维数组

Version 2.0 一维数组

很久以后重做

#### 代码

##### Version 1.0

#include<stdio.h>

int dp[101][1001];

struct Herb{

int time;

int value;

}herb[101];

int max(int x,int y){return x>y?x:y;}

int main(){

int t,m;

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

for(int i=1;i<=m;i++)scanf("%d%d",&herb[i].time,&herb[i].value);

for(int i=0;i<=t;i++)dp[0][i]=0;

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

for(int j=t;j>=herb[i].time;j--){

dp[i][j]=max(dp[i-1][j-herb[i].time]+herb[i].value,dp[i-1][j]);

}

for(int j=herb[i].time-1;j>=0;j--){

dp[i][j]=dp[i-1][j];

}

}

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

}

return 0;

}

##### Version 2.0

#include<stdio.h>

int dp[1001];

struct Herb{

int time;

int value;

}herb[101];

int max(int x,int y){return x>y?x:y;}

int main(){

int t,m;

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

for(int i=1;i<=m;i++)scanf("%d%d",&herb[i].time,&herb[i].value);

for(int i=0;i<=t;i++)dp[i]=0;

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

for(int j=t;j>=herb[i].time;j--){

dp[j]=max(dp[j-herb[i].time]+herb[i].value,dp[j]);

}

}

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

}

return 0;

}

### 题目1454：Piggy-Bank

**题目描述：**

Before ACM can do anything, a budget must be prepared and the necessary financial support obtained. The main income for this action comes from Irreversibly Bound Money (IBM). The idea behind is simple. Whenever some ACM member has any small money, he takes all the coins and throws them into a piggy-bank. You know that this process is irreversible, the coins cannot be removed without breaking the pig. After a sufficiently long time, there should be enough cash in the piggy-bank to pay everything that needs to be paid.   
But there is a big problem with piggy-banks. It is not possible to determine how much money is inside. So we might break the pig into pieces only to find out that there is not enough money. Clearly, we want to avoid this unpleasant situation. The only possibility is to weigh the piggy-bank and try to guess how many coins are inside. Assume that we are able to determine the weight of the pig exactly and that we know the weights of all coins of a given currency. Then there is some minimum amount of money in the piggy-bank that we can guarantee. Your task is to find out this worst case and determine the minimum amount of cash inside the piggy-bank. We need your help. No more prematurely broken pigs!

**输入：**

The input consists of T test cases. The number of them (T) is given on the first line of the input file. Each test case begins with a line containing two integers E and F. They indicate the weight of an empty pig and of the pig filled with coins. Both weights are given in grams. No pig will weigh more than 10 kg, that means 1 <= E <= F <= 10000. On the second line of each test case, there is an integer number N (1 <= N <= 500) that gives the number of various coins used in the given currency. Following this are exactly N lines, each specifying one coin type. These lines contain two integers each, Pand W (1 <= P <= 50000, 1 <= W <=10000). P is the value of the coin in monetary units, W is it's weight in grams.

**输出：**

Print exactly one line of output for each test case. The line must contain the sentence "The minimum amount of money in the piggy-bank is X." where X is the minimum amount of money that can be achieved using coins with the given total weight. If the weight cannot be reached exactly, print a line "This is impossible.".

**样例输入：**

3

10 110

2

1 1

30 50

10 110

2

1 1

50 30

1 6

2

10 3

20 4

**样例输出：**

The minimum amount of money in the piggy-bank is 60.

The minimum amount of money in the piggy-bank is 100.

This is impossible.

#### 评价

#### 代码

#include<stdio.h>

#define INF 999999999

struct Coin{

int w;

int p;

}coin[501];

int dp[10001];

int min(int x,int y){return x<y?x:y;}

int main(){

int t;

scanf("%d",&t);

while(t--){

int e,f;

scanf("%d%d",&e,&f);

f-=e;

int n;

scanf("%d",&n);

for(int i=1;i<=n;i++)scanf("%d%d",&coin[i].p,&coin[i].w);

dp[0]=0;

for(int i=1;i<=f;i++)dp[i]=INF;

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

for(int j=coin[i].w;j<=f;j++){

if(dp[j-coin[i].w]!=INF)dp[j]=min(dp[j],dp[j-coin[i].w]+coin[i].p);

}

}

if(dp[f]!=INF)printf("The minimum amount of money in the piggy-bank is %d.\n",dp[f]);

else printf("This is impossible.\n");

}

return 0;

}

### 题目1455：珍惜现在，感恩生活

**题目描述：**

为了挽救灾区同胞的生命，心系灾区同胞的你准备自己采购一些粮食支援灾区，现在假设你一共有资金n元，而市场有m种大米，每种大米都是袋装产品，其价格不等，并且只能整袋购买。请问：你用有限的资金最多能采购多少公斤粮食呢？

**输入：**

输入数据首先包含一个正整数C，表示有C组测试用例，每组测试用例的第一行是两个整数n和m(1<=n<=100, 1<=m<=100),分别表示经费的金额和大米的种类，然后是m行数据，每行包含3个数p，h和c(1<=p<=20,1<=h<=200,1<=c<=20)，分别表示每袋的价格、每袋的重量以及对应种类大米的袋数。

**输出：**

对于每组测试数据，请输出能够购买大米的最多重量，你可以假设经费买不光所有的大米，并且经费你可以不用完。每个实例的输出占一行。

**样例输入：**

1

8 2

2 100 4

4 100 2

**样例输出：**

400

#### 评价

#### 代码

#include<stdio.h>

struct N{

int p;

int w;

}rice[2001];

int dp[101];

int max(int x,int y){return x>y?x:y;}

int main(){

int c;

scanf("%d",&c);

while(c--){

int n,m;

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

int p,w,num;

int idx=0;

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

scanf("%d%d%d",&p,&w,&num);

int q=1;

while(num-q>=0){

rice[++idx].p=q\*p;

rice[idx].w=q\*w;

num-=q;

q\*=2;

}

if(num!=0){

rice[++idx].p=num\*p;

rice[idx].w=num\*w;

}

}

for(int i=0;i<=n;i++)dp[i]=0;

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

for(int j=n;j>=rice[i].p;j--){

dp[j]=max(dp[j],dp[j-rice[i].p]+rice[i].w);

}

}

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

}

return 0;

}

# 第八章、其他技巧

## 一、标准模板库（STL）

### 题目1168：字符串的查找删除

**题目描述：**

给定一个短字符串（不含空格），再给定若干字符串，在这些字符串中删除所含有的短字符串。

**输入：**

输入只有1组数据。  
输入一个短字符串（不含空格），再输入若干字符串直到文件结束为止。

**输出：**

删除输入的短字符串(不区分大小写)并去掉空格,输出。

**样例输入：**

in

#include

int main()

{

printf(" Hi ");

}

**样例输出：**

#clude

tma()

{

prtf("Hi");

}

**提示：**

注:将字符串中的In、IN、iN、in删除。

#### 评价

#include<string>

#include<iostream>

#include<ctype.h>

Cout<<c<<endl;

#### 代码

#include<stdio.h>

#include<string>

#include<iostream>

#include<ctype.h>

using namespace std;

int main(){

char str[101];

gets(str);

string a=str;

for(int i=0;i<a.size();i++)a[i]=tolower(a[i]);

// if(a[i]>='A'&&a[i]<='Z')a[i]+=32;

while(gets(str)){

string b=str,c=b;

for(int i=0;i<b.size();i++)b[i]=tolower(b[i]);

// if(b[i]>='A'&&b[i]<='Z')b[i]+=32;

int t=b.find(a,0);

while(t!=string::npos){

b.erase(t,a.size());

c.erase(t,a.size());

t=b.find(a,t);

}

t=c.find(' ',0);

while(t!=string::npos){

c.erase(t,1);

t=c.find(' ',t);

}

cout<<c<<endl;

}

return 0;

}

### 题目1450：产生冠军

**题目描述：**

有一群人，打乒乓球比赛，两两捉对撕杀，每两个人之间最多打一场比赛。  
球赛的规则如下：  
如果A打败了B，B又打败了C，而A与C之间没有进行过比赛，那么就认定，A一定能打败C。  
如果A打败了B，B又打败了C，而且，C又打败了A，那么A、B、C三者都不可能成为冠军。  
根据这个规则，无需循环较量，或许就能确定冠军。你的任务就是面对一群比赛选手，在经过了若干场撕杀之后，确定是否已经实际上产生了冠军。

**输入：**

输入含有一些选手群，每群选手都以一个整数n(n<1000)开头，后跟n对选手的比赛结果，比赛结果以一对选手名字（中间隔一空格）表示，前者战胜后者。如果n为0，则表示输入结束。

**输出：**

对于每个选手群，若你判断出产生了冠军，则在一行中输出“Yes”，否则在一行中输出“No”。

**样例输入：**

3

Alice Bob

Smith John

Alice Smith

5

a c

c d

d e

b e

a d

0

**样例输出：**

Yes

No

#### 评价

#include<map>

#### 代码

#include<stdio.h>

#include<string>

#include<map>

using namespace std;

map<string,int> M;

int inDegree[2002];

int main(){

int n;

int idx;

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

M.clear();

idx=0;

int t=2\*n;

for(int i=0;i<t;i++)inDegree[i]=0;

char name1[50],name2[50];

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

scanf("%s%s",name1,name2);

string a=name1,b=name2;

int tmp1,tmp2;

if(M.find(a)==M.end()){

tmp1=idx;

M[a]=idx++;

}else tmp1=M[a];

if(M.find(b)==M.end()){

tmp2=idx;

M[b]=idx++;

}else tmp2=M[b];

inDegree[tmp2]++;

}

int cnt=0;

for(int i=0;i<idx;i++)if(inDegree[i]==0)cnt++;

if(cnt==1)printf("Yes\n");

else printf("No\n");

}

return 0;

}

### 练习题：（题目1111）单词替换

**题目描述：**

输入一个字符串，以回车结束（字符串长度<=100）。该字符串由若干个单词组成，单词之间用一个空格隔开，所有单词区分大小写。现需要将其中的某个单词替换成另一个单词，并输出替换之后的字符串。

**输入：**

多组数据。每组数据输入包括3行，

第1行是包含多个单词的字符串 s，

第2行是待替换的单词a，(长度<=100)

第3行是a将被替换的单词b。(长度<=100)

s, a, b 最前面和最后面都没有空格.

**输出：**

每个测试数据输出只有 1 行，

将s中所有单词a替换成b之后的字符串。

**样例输入：**

You want someone to help you

You

I

**样例输出：**

I want someone to help you

#### 评价

注意S.find(A,t1)的含义

#### 代码

#include<stdio.h>

#include<iostream>

#include<string>

using namespace std;

int main(){

char s[101],a[101],b[101];

string S,A;

while(gets(s)){

gets(a);

gets(b);

S=s;A=a;

int t1=S.find(A,0),cnt1=0;

while(t1!=string::npos){

if((t1==0||S[t1-1]==' ')&&(t1+A.size()==S.size()||S[t1+A.size()]==' ')){//单词的开头结尾

S.erase(t1,A.size());

S.insert(t1,b);

t1=S.find(A,t1);

}else t1=S.find(A,t1+1);

}

cout<<S<<endl;

}

return 0;

}

### 练习题：（题目1049）字符串去特定字符

**题目描述：**

输入字符串s和字符c，要求去掉s中所有的c字符，并输出结果。

**输入：**

测试数据有多组，每组输入字符串s和字符c。

**输出：**

对于每组输入,输出去除c字符后的结果。

**样例输入：**

heallo

a

**样例输出：**

hello

#### 评价

Getchar()只接收一个字符，不把后面的回车去掉

Gets()把后面的回车去掉了

Scanf();没把后面的回车去掉

#### 代码

#include<stdio.h>

#include<iostream>

#include<string>

using namespace std;

int main(){

char s[101],c;

while(gets(s)){

string S=s;

c=getchar();

int t1=S.find(c,0);

while(t1!=string::npos){

S.erase(t1,1);

t1=S.find(c,t1);

}

cout<<S<<endl;

getchar();

}

return 0;

}