### 注意点

1.数据量小，并且麻烦，直接暴力。

2.小心中间过程计算过程（比如涉及到乘）可能会超出范围，类型不能光考虑结果，另外别用long，和int范围一样，用long long

3.long long极大值设0x7fffffffffffffff（不会求和或乘）

4.double极小值eps设1e-8

5.对时间有把握时，sqrt()可以转化为i\*i<?的形式，无误差

6.输出格式要注意，不要多了空格或换行，比如输出

1 2 3 4

形式，有些题目很严格，最后一个数字后不能有空格

7.一定要小心运算过程中的溢出问题，特别是乘法！乘法可通过判断(x\*y)/y是否等于x来判断溢出

8.memset只能一个个字节设置相同的值，不是什么值都能设！

9.有时循环读入，读入到某种数据可略去剩余数据时，也应继续读完而不是直接break，否则产生脏数据

10.判断奇偶不要用%2，用&1，因为负奇数取余得到的-1，用与运算就没问题

11.for(i=0;i<n;i++)的形式，执行完i会比“期望值”多1，必须注意，有时可以把i++这种对i改变的语句放在循环内部

12.int强制转换是直接截取小数部分，返回值为int型，int(-1.9)返回-1

floor是向下取整，返回值为double型floor(-1.9)返回-2

与floor相对的是ceil

13.<<运算符优先级比较低，使用时用括号括起来

14.cin与cout对不同平台的兼容性强很多，有时scanf与printf可能会错

15.大的数组放在main外面，作为全局变量，否则可能栈溢出

16.逻辑运算有先后顺序并且得到答案后不会继续算下去，所以不要把希望执行的函数放在逻辑运算中

如(flag||dfs());

当flag为1时，不会执行dfs

17.减法取余，先加MOD，如(x-1+MOD)%MOD

18.结构体或类中有指针或数组时，不要直接赋值，否则为浅拷贝，会相互影响

19.浮点型不想四舍五入（类似%.3f控制符会四舍五入），假如要保留后三位，可以先乘1000，转为整型，再除以1000，转为浮点型

20.格式控制一定要对，不然可能出现极其诡异的错误

21.特别注意有符号数和无符号数不要混用计算，strlen和各种size函数返回的都是无符号数

不要写i<x.size()-1这种表达式，因为有符号数为0时再减一，是一个非常大的正数而不是负数

应该先将size返回值存于有符号变量中

22.对multiset这种树形结构指针不能进行减运算

23.不要将函数调用写在形如max/min的形参里，因为若为宏定义，函数会被执行2次！

24.重载<运算符，形参与函数都加const

不要省事用while(cin>>x[num++]);的形式来读入！因为即使是EOF，这部分代码也会执行，最后num会更大！

Java

java.math.\*

BigInteger

BigDecimal

重定向输入输出流

freopen("文件名","r",stdin);

freopen("文件名","w",stdout);

#include<algorithm>

next\_permutation(首指针，末尾指针+1);

pre\_permutation(首指针，末尾指针+1);//皆返回bool，还能排则true

升序排列时：

lower\_bound(x,x+10,i)，返回x中第一个大于等于i的元素地址

upper\_bound(x,x+10,i)，返回x中第一个大于i的元素地址

减去x，得到数组下标

如{1,2,2,3}，upper\_bound查找2，下标为3，lower\_bound下标为1

找不到，返回的都是恰好越界的下标

一般用upper\_bound-lower\_bound快速得到指定元素的个数

lower\_bound左边（若存在）一定是第一个比i小的元素

upper\_bound（若存在）一定是第一个比i大的元素

需要稳定排序时用stable\_sort。

实际上在CMP函数中判断相等情况即可

要让某些元素不多不少，可以为每个元素对应一个二进制位，状压判断

区间交：最右的左端点L，最左的右端点R，交集为[L,R]，L>R表示交集为空集

priority\_queue，保证队头元素最大（默认最大，也可最小）

priority\_queue<int,vector<int>,less<int> >//降序，首元素最大

priority\_queue<int,vector<int>,greater<int> >//升序，首元素最大

#include<string.h>

memset(sum,0,sizeof(sum));//用0初始化数组

字符串与数字互转

sprintf(被写入字符串,"%d",n);

sscanf(读入字符串,"%d",&n);

对于有顺序的模拟，如果正着不好搞，试试反着来

卡输入输出的题，可以将getchar和putchar改造

结构体置于需要排序的容器前需要先重载<

仅有<，能比较等于的原理是，若a<b和b<a都不成立，那么a==b（就等价于a>=b且b>=a）

所以在重载<时，内部元素比较不要出现<=

互质的几个性质

a与b互质，则a与b+ka互质

a与b互质，则ka与kb的gcd为k

scanf(" %c",&c);略过空格

多用全局变量简化形参

可以用0x即16进制来定义某些数据类型的最大最小值

想递归得到返回值并进行其他操作

i=dg();

//other operations

移位操作最好只对无符号数进行

若右移，负数左边会补1，可能出现错误结果

1.最大与次大

if(tem>t2)

{

t1=t2;

t2=tem;

}

else if(tem>t1)

{

t1=tem;

}

2.sort与qsort

bool cmp(int x,int y)

{

return x<=y;

}

sort(首指针，末尾指针+1);

int cmp(const void \* x,const void \* y)

{

return \*((int \*)x)-\*((int \*)y);

}

qsort(首指针,元素个数,sizeof(int),cmp);

3.前缀和，用来解决连续子串之和的问题

Sn=A1+A2+…+An

涉及到区间，就应想想前缀和

4.位操作

状态压缩，每个元素只有两个状态的总状态用一个二进制数表示，每一位表示一个元素的状态

子集枚举

for(int i=x;i;)

{

i=(i-1)&x;

}

全集S，子集S0

S0补集为S^S0

统计二进制数中1的个数

不断右移>>，末尾&1，判断是否为1

5.

求组合数有时会太大需要取模，使用类似com[i][j]=(com[i-1][j]+com[i-1][j-1])%1000007来递推求解

0.调试类内代码方法为单步进入。

1.带默认参数的函数，默认参数写在声明而不是实现。

2.记得自己写的头文件包含用""。

3.类中成员函数声明后一定要定义完，定义一半就编译会报错。

4.

printf(“格式控制字符串”，输出表列)

格式控制字符串 为 %- 0 m.n l/h 格式字符

% 为格式说明的引导符号

- 为指定左对起输出

0 为 指定空位填0

m.n 指定输出域宽及精度

l/h 输出长度的修正

格式字符 为指定输出的数据类型

5.浮点型不直接用“=”和“！=”与其他数比较，化为不等号比较。

6.布尔型变量在if等语句中不要加入0、1，直接留下或加上！

7.输入问题：

Windows下，回车是\r\n，但回车，缓冲区中只有\n；

getchar()、cin.get()可接收回车，得'\n'；

gets、cin.getline()不接收回车，但回车会当一行的结束符（注意使用前检查缓冲区是否有多余回车），且读完一次回车会排出缓冲区；

其他类型输入，一般读完回车、空格还在缓冲区，注意下次是用什么函数读取，若为1、2所示，需清除缓冲区的多余字符！

8.输出问题

若题目中让两组输出之间需要间隔一行，那么最后一组注意不能有空行；

若题目让一行输入两个数据之间间隔一个空格，同样要注意最后一个数据后不能有空格；

若T组输出，每组数据占一行，最后一组这时候也是需要换行符号的。

防止最后出现多余空格或换行，可设置如下判断：

bool flag=false;

if(flag)

{

cout<<'\n'<<i;

}

else

{

cout<<i;

}

flag=true;//即第一次运行程序后，flag变成true，之后都先输换行再输值。

9.

再次注意强制转换问题，如求某倒数数列之和，不要写1/i，应为1.0/i。

10.大数乘法：

假设乘a，从低位向高位乘，进位为d，起初d=0，乘到某位w，则该位变成(w\*a+d)%10，产生的新的进位为(w\*a+d)/10。

11.大数除法：

假设除以a，从高位向低位除以，借位为d，起初d=0，除以到某位w，则该位变成(w+10\*d)/a，产生的新的借位为(w+10\*d)%a。

12.同一个常量字符串，地址一样。

13.当以控制台输入作为输入流时，以Ctrl+Z作为结束符，此时cin.eof()要读完结束符才返回1，小心多循环一次。

14.用数组存储下标和元素的对应关系时，每个元素若有多个属性，存为类或结构体。

15.形参需要用到指针时，明确该函数是改变指针变量本身还是改变指针所指空间，若为前者，需传引用。

16.数组需要首尾相接循环使用时，假设长度为n，以a为起点，则

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

{

x[i%n];

}

17.数组越界运行时不一定会报错，一定要随时防止数组越界，比如循环中要比较x[i]和x[i+1]，就必须时i从零到n-2而不是到n-1。

18.清除输入流

cin.clear();

cin.sync();

19.

需要稳定排序时用stable\_sort。

20.scanf时区分%f和%lf，printf时只有%f，acm里%lf可能会错

21.next\_permutation(s,s+n)，prev\_permutation(s,s+n)，返回值bool，自动为s生成下一个/上一个排列

22.

升序排列时：

lower\_bound(x,x+10,i)，返回x中第一个大于等于i的元素地址

upper\_bound(x,x+10,i)，返回x中第一个大于i的元素地址

23.指针运算

指针+/-整型

指针-指针

结果会根据指针类型进行字节调整

24.

acm里，整型与浮点比较，最好把整型转换为浮点型，因为浮点型无法精确表示，转换成整型可能差1

比较时也不要太绝对，可用一个极小值来辅助，比如1e-8

### STL

#### algorithm等杂项

lower\_bound(x,x+10,i)，返回x中第一个大于等于i的元素地址

upper\_bound(x,x+10,i)，返回x中第一个大于i的元素地址

next\_permutation(首指针，末尾指针+1);

pre\_permutation(首指针，末尾指针+1);//皆返回bool，还能排则true

set\_ intersection (s1.begin(),s1.end(),s2.begin(),s2.end(),inserter(s3,s3.begin()));//s1与s2的交集存于s3

set\_intersection 获得两个集合的交集

set\_union 获得两个集合的并集

set\_difference 获得两个集合的差集

set\_symmetric\_difference 获得两个集合的对称差

includes(s1.begin(),s1.end(),s2.begin(),s2.end());//判断s1是否包含s2

sscanf();//第一个参数为char数组，其他和scanf一样

sprintf;// 第一个参数为char数组，其他和printf一样

streamstring包含于sstream包中，用法类比cin、cout

清空：

ss.str(“”);

ss.clear();

ceil浮点数向上取整

floor浮点数向下取整

int cmp(const void \* x,const void \* y)

{

return \*((int \*)x)-\*((int \*)y);

}

qsort(首指针,元素个数,sizeof(int),cmp);

#### string

已重载 + , -, =, !=, <, >等运算符

reserve预分配空间

c\_str得到char\*风格字符串

s.insert(0,1,'2');//0位置插入1个’2’字符

s.insert(0,”2”);//0位置插入字符串

s.push\_back(‘2’);//末尾添加单个字符

erase删除子串

find一系列函数，找不到返回string::npos

s.substr(0,2);//获取从下标0开始的长度为2的字符串

#### queue

queue，最普通的

priority\_queue

默认队头最大，反着改为

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

结构体，需重载<

deque双端队列

#### list

reverse() 倒转

sort()排序

a.splice(a.end(),b,b.begin(),b.end());//将b连接至a尾，b清空

merge用法类似，但要求两个链表都有序

#### set和multiset

set不可重复，multiset可以，都为有序容器，结构体需重载<运算符

insert插入

erase删除，需注意会删除所有值相同的元素，可以先find再erase

count返回元素个数

lower\_bound、upper\_bound返回迭代器

存在unordered\_set

#### map

按关键字有序，若为结构体需重载<运算符

已经重载[]、=可当“数组”用，插入前先find判断之前有没有

unordered\_map，查询O(1)，位于unordered\_map包中，需C++11标准支持

#### bitset

bitset<100> b;//长度设为100

可当bool数组使用，也可看作二进制数

bitset间已重载&、|、^等逻辑运算

与整型间重载了&=、|=、<<、>>等

size：返回二进制位数

count：返回1的个数

any：是否有1