CSP-J 复赛笔记

1.考试必备：

(1) Noi Linux系统安装/使用/编译器使用

环境安装：http://acm.club/discussion-detail/215

使用：https://www.bilibili.com/video/BV1Ld4y1M7UQ/?spm_id_from=333.337.search-card.all.click&vd_source=0ae5f1593252bfc7bf6cbcd67ceda2de

编译器：推荐Codeblock和Geany，详细使用方式参考

复赛须知：http://acm.club/discussion-detail/55

注意！！！CodeBlocks编译器在NOI LINUX中不能在包含中文的路径下进行编译，”桌面“、”下载“也不行。

最好存到主目录（也称Home）中。

(2) 代码模板格式

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    freopen("test.in","r",stdin);//test.in和test.out名称根据给出题目要求敲上
    freopen("test.out","w",stdout);
    /*此处书写代码省略*/
    return 0;
}//记住freopen的两个语句必须在主函数内的开头敲上，不能在大括号外面！！！就算主函数里面的输入语句，也要放到这两行后面
```

(3)注意：看数据范围，十年OI一场空，不开long long 见祖宗

2.超时检测

（1）时间复杂度

时间复杂度	n最大限制的范围
O(logn)	long long范围内
O(n)	10^7
O(nlogn)	10^5~5*10^5
O(n^2)	1000~5000
O(n^3)	200~500
O(n !)	12

时间复杂度	n最大限制的范围
O(1)	几乎无限

（2）代码计时（不推荐，最好通过计算时间复杂度的方式判断）

因为Noi Linux系统中的所有编译器编译的结果是不会像DevC++那样会显示程序运行时间的，所以为了能展示出运行时间，以做参考，便于对程序进行检查，可使用如下模板代码：

```
//此代码需要使用C++ 11的编译器才可以正常运行。
#include <iostream>
#include <chrono>

int main() {
    // 开始计时
    auto start = std::chrono::high_resolution_clock::now();

    // 这里放置你的代码，以记录其运行时间
    // 例如：
    for (int i = 0; i < 1000000; ++i) {
        // 模拟耗时操作
    }

    // 结束计时
    auto finish = std::chrono::high_resolution_clock::now();

    // 计算运行时间
    std::chrono::duration<double> elapsed = finish - start;
    std::cout << "Elapsed time: " << elapsed.count() << "s\n";

    return 0;
}
```

（3）思路：

逻辑错误类超时：如类似死循环等会让程序一直无法满足某个条件，因此程序将不再有返回值，也就是没有输出。

繁琐循环类超时：循环语句嵌套过多（遇到二维数组时使用2层循环，其余情况尽可能用1层）或循环语句的终止条件过大导致，可使用数学方法或者减少循环语句，用变量做模拟的方式处理。

代码过于屎山类超时：想更好的方法。

3.数据结构

STL详情见郭老师的STL讲义

（1）常见方式：int,long long,char,string,float,double,bool等。

（2）标记物：

1.和循环搭配使用，可实现下述代码的效果。

2.标记某个或某些量的状态，便于后需重新利用时直接就可以查看到状态，无需重复运算。

这里效果可以参考数独代码：http://acm.club/discussion-detail/512

标记物例子代码：

```
//判断回文字符串，虽然不是最好的方式，而且相当的屎山，但起码道理可以理解。
#include<bits/stdc++.h>
using namespace std;
int main(){
    string s;cin>>s;
    bool f=0;//标记物f，f=1表示不是回文数，f若一直是0则表示是回文数
    int i=0,j=s.size()-1;
    while(i<j){
        if(s[i]!=s[j]){//当左侧不等于右侧时说明不是回文数
            f=1;
            break;
        }
        i++;
        j--;
    }
    if(f==1) cout<<"No"<<endl;
    else cout<<"Yes"<<endl;//如果不写标记物，用程序的方式在循环外就不能判断出是回文数。
}
```

（3）数组：一维数组二维数组。

（4）尽量让所有使用的变量设为全局变量，如需对某个变量进行反复输入并运算，直接cin>>n;就可以了。

（5）Vector

vector的数组为边长数组，我们以前学习的是定长数组，其各自优点缺点如下可见：

```
int a[10001];//定长数组,在这个程序中被设为全局
int main(){
    /*代码：略*/
    return 0;
}
//设为全局的定长数组优点：是不占用栈空间，定义效率高。
//设为全局的定长数组缺点：需要在最开始就设定数组大小，对于数组大小的数据量过大情况，如int a[1e9]，效率并不乐观，无法实现在程序计算中不断调整数组的空间大小、以提高效率的效果。
```

```
vector<int> a;//变长数组
/*
优点：不强制需要定义空间大小，其空间大小可根据数组存元素量进行动态改变，无论是数据量小的情况，还是a[1e9]的数据量大的情况，都可以通过计算时的动态空间大小的特点解决。
缺点：同学们不常用
*/
```

Vector数组相关的语法

语句	说明
vector v;	创建空vector
v.push_back(x)	向尾部增加一个元素x
v.insert(pos,x)	向pos地址指向元素前增加一个元素x
v[i]	访问 i 位置的元素
v.pop_back()	删除向量中最后一个元素
v.clear()	清空向量中所有元素
v.empty()	判断向量是否为空
v.size()	返回向量中元素个数
v.begin()	返回向量头指针（迭代器），指向第一个元素
v.end()	返回向量尾指针（迭代器），指向最后一个元素+1位置
v.erase(v.begin()+i)	删除第i位位置的元素

```
vector<int> a; //定义了一个名为a的一维数组,数组存储int类型数据
vector<double> b;//定义了一个名为b的一维数组，数组存储double类型数据
vector<node> c;//定义了一个名为c的一维数组，数组存储结构体类型数据，node是结构体类型

vector<int> v(n);// 定义一个长度为n的数组，初始值默认为0，下标范围[0, n - 1]
vector<int> v(n, 1);// v[0] 到 v[n - 1]所有的元素初始值均为1
//注意：指定数组长度之后（指定长度后的数组就相当于正常的数组了）

vector<int> a{1, 2, 3, 4, 5};//数组a中有五个元素，数组长度就为5

vector<int> a(n + 1, 0);
vector<int> b(a); // 两个数组中的类型必须相同,a和b都是长度为n+1，初始值都为0的数组
vector<int> c = a; // 也是拷贝初始化,c和a是完全一样的数组
--------------------------------------------------------------------------------
----------
vector<int> a(n + 1);
sort(a.begin() + 1, a.end()); // 对[1, n]区间进行从小到大排序
```

(6) 栈

原理：先进后出。

应用：括号匹配等类似栈原理的问题。

```
s.empty();          //如果栈为空则返回true，否则返回false;
s.size();           //返回栈中元素的个数
s.top();            //返回栈顶元素，但不删除该元素
s.pop();            //弹出栈顶元素，但不返回其值
s.push();           //将元素压入栈顶
```

(7) 链表

由某一元素指向另一个元素的连接方式，组成一个和数组功能相似的东西。

最大优点是，易删除。

数组

1 2 3 4 5 6 7 8 9 10

如果我想删除元素 3，第一种方式是给 3赋值 0.那么就会是如下情况。

1 2 0 4 5 6 7 8 9 10

后续遍历时只需要判断元素不是 0 就输出 即可实现“懒删除”。

1 2 3 4 5 6 7 8 9 10

还是这样的数组，如果我想删除元素 3，第二种方式是循环到最后，把后一位的元素赋值给前一位元素，即a[i]=a[i+1];直到元素 9 变成了元素 10 。也就是 1 2 4 5 6 7 8 9 10 。虽然实现了删除元素，但是需要循环，会造成程序效率可能出现问题。为了解决这个问题，就要引入链表了。

链表

链表是一个元素链接另一个元素，从元素 1 开始。即为

1-->2-->3-->4-->5-->6-->7-->8-->9-->10;

如果我们要删除元素 3 ，那么就让 2 和 3 的链接断开， 3 和 4 链接断开，改为 2 链接 4，最后就是。

1-->2-->4-->5-->6-->7-->8-->9-->10 这样实现了删数的操作，还保证效率上不会出现问题。

链表有2种，一种是模拟链表，另一种是c++自带的 list ；

用法、用法

```
list<T> lst; //list采用采用模板类实现,对象的默认构造形式,"T“表示数据类型，如list<int>
lst;
list(beg,end); //构造函数将[beg, end)区间中的元素拷贝给本身。
list(n,elem); //构造函数将n个elem拷贝给本身。
list(const list &lst); //拷贝构造函数。
assign(beg, end); //将[beg, end)区间中的数据拷贝赋值给本身。
assign(n, elem); //将n个elem拷贝赋值给本身。
list& operator=(const list &lst); //重载等号操作符
swap(lst); //将lst与本身的元素互换。
size(); //返回容器中元素的个数
empty(); //判断容器是否为空

resize(num); //重新指定容器的长度为num，若容器变长，则以默认值填充新位置。
```

```
//如果容器变短，则末尾超出容器长度的元素被删除。

resize(num, elem);重新指定容器的长度为num，若容器变长，则以elem值填充新位置。
如果容器变短，则末尾超出容器长度的元素被删除。

push_back(elem);//在容器尾部加入一个元素
pop_back();//删除容器中最后一个元素
push_front(elem);//在容器开头插入一个元素
pop_front();//从容器开头移除第一个元素
insert(pos,elem);//在pos位置插elem元素的拷贝，返回新数据的位置。
insert(pos,n,elem);//在pos位置插入n个elem数据，无返回值。
insert(pos,beg,end);//在pos位置插入[beg,end)区间的数据，无返回值。
clear();//移除容器的所有数据
erase(beg,end);//删除[beg,end)区间的数据，返回下一个数据的位置。
erase(pos);//删除pos位置的数据，返回下一个数据的位置。
remove(elem);//删除容器中所有与elem值匹配的元素。

front(); //返回第一个元素
back(); //返回最后一个元素
```

(8) 字符（char s，不是char s[10]，也不是string s）

函数语句	功能	例子
islower(x)	判断x是否为小写字母	islower('a')==true;
isupper(x)	判断x是否为大写	isupper('A')==true;isupper('1')==false;
isdigit(x)	判断x是否为数字	isdigit('9')==1;isdigit('A')==0;
isalpha(x)	判断x是否为字母	isalpha('a')==2;isalpha('A')==1;isalpha('9')==0;
char()	ASCII码转字符	char(97)=='a';
int()	字符转ASCII码	int('a')==97;

(9) 数学函数

abs(x) 绝对值　max(x,y) 最大值　min(x,y) 最小值

swap(x,y) 交换　int(x) 取整数部分　pow(x,y) 指数函数

floor(x) 向下取整 ceil(x) 向上取整　round(x) 四舍五入

sqrt(x) 平方根函数

```
int main() {
    int t=INT_MAX;//开一个特别大的整数
    return 0;
}
```

（10）string字符串函数

```
string s;
cin>>s;//不包含空格输入
getline(s);//包含空格输入
s.substr(i,x);//返回从第i位开始长度为 x 的子字符串
s.find(x);//返回字符串内找到的第一个字符的位置，没找到目标就返回-1
s.erase(i,x);//从i位开始往后删除 x 个
s.insert(i,str);//在字符串中间的 i 位置插入字符串str，插入在其后面

……+=…… //在尾部添加字符
……+……//串联字符串
--------------------------------------------------------
find函数重点注意
string s="IAAA";
int f=s.find("D");
cout<<f<<endl;
//输出前写一个变量存一下，这样如果找到就返回第一个字符的位置，找不到就会正常返回-1，
```

4.算法

（1）贪心算法：无具体模板代码，但是贪心算法讲究策略，通常用来求最优解。

（2）二分算法：

```
#include<bits/stdc++.h>
using namespace std;
int mid,a[1001],l,r,n,f=0,x;
int main(){
    cin>>n;
    cin>>x;
    for(int i=1;i<=n;i++) cin>>a[i];
    l=1;
    r=n;
    while(l<=r){
        mid=(l+r)/2;
        if(a[mid]>x) r=mid-1;
        else if(x>a[mid]) l=mid+1;
        else{
            f=1;
            break;
        }
    }
    if(f==1) cout<<"yes";
    else cout<<"no";
}
```

（3）快速幂算法

```
//原始代码，求x^p
#include<iostream>
using namespace std;
int x, p, m, i, result;
int main()
```

```
{   cin >> x >> p ;
    result = 1;
    while (p)
    {  if (p % 2 == 1)
            result = result*x;
        p /= 2;
        x = x*x;
    }
    cout << result << endl;
    return 0;
}
//变式：求x ^ p % m
#include<iostream>
using namespace std;
int x, p, m, i, result;
int main()
{   cin >> x >> p >> m;
    result = 1;
    while (p)
    {  if (p % 2 == 1)
            result = result * x % m;
        p /= 2;
        x = x * x % m;
    }
    cout << result << endl;
    return 0;
}


//整合函数：
#include<iostream>
using namespace std;
long long qpow(int x,int p){
    long long result = 1;
    while (p)
    {  if (p % 2 == 1)
            result = result * x;
        p /= 2;
        x = x * x;
    }
    return result;
}
int main()
{
    int x,p;
    cin >> x >> p;
    cout<<qpow(x,p);
    return 0;
}
//自带函数
pow(x,p);
```

(4) 前缀和、差分

```
//前缀和：求区间和
for (int i = 1; i <= n; i++) cin >> a[i]; //a数组为原始输入
for (int i = 1; i <= n; i++) prefix[i] = prefix[i - 1] + a[i]; //prefix数组为前缀和存储
int l,r;cin>>l>>r;
cout << prefix[r] - prefix[l-1]<<endl;//输出a数组[l,r]区间的和
```

```
//差分：通常用于对于数组中某个区间的所有元素进行统一的加减运算。
    int n,a[10001],diff[10001];cin >> n;
    for (int i = 1; i <= n; i++) cin >> a[i];
    diff[0] = a[0];
    for (int i = 1; i <= n; i++) {
        diff[i] = a[i] - a[i - 1];
    }
    int l, r;
    cin >> l >> r;
    diff[l]++;
    diff[r + 1]--;// 求修改后的原数组，即等价于求前缀和
    int sum = 0;
    for (int i = l; i <= r; i++) {
        sum += diff[i];
    }
    cout << sum << endl;
```

(5) GCD/LCM + 素数判断

```
int gcd(int a,int b)
{
    if(b==0) return a;
    else return gcd(b,a%b);
}
int lcm(int a,int b){
    return a*b/gcd(a,b);
}
```

```
bool check(int num){
    if(num < 2){
       return false;
    }
    for(int i=2;i*i<=num;i++){
        if(num%i==0){
            return false;
        }
    }
    return true;
}
```

(6) 数位分离

基本代码：，若输入num为 123，则输出为 3 2 1.

```
    long long num;
    cin >> num;
    while (num!= 0) {
        cout << num % 10 << endl;//每一位输出(倒着)
        num /= 10;
    }
```

同理数位相加,例如输入num为 123，则输出为 6，也就是 1+2+3=6

```
while (num != 0) {
    sum += num % 10;
    num /= 10;
}
```

统计数中包含多少个数字？
如num输入100，x输入1，则输出100里面有1个1；若x=0，则输出100里面有2个0；若x=2，则输出100里面有0个2。

```
    long long num;
    int x;
    cin >> num >> x;
    int sum=0;
    do{
        if(num%10==x) sum++;
        num/=10;
    }while(num);
    cout<<sum<<endl;
```

(7) 常用排序算法+排序思想

常用排序算法：

优先桶排序，效率最高。其次快速排序。

桶排序代码

```
int n,a[100001]={0},v;
int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        cin>>v;
        a[v]++;
    }
    for(int i=1;i<=n;i++){//此时为升序,降序为int i=n;i>=1;i--
        if(a[i]>0){
            int t=a[i];
            while(t--) cout<<i<<" ";//处理相等元素排序
        }
    }
    return 0;

}
```

快速排序代码

```
sort(a,a+n,cmp);//从第0位
sort(a+1,a+1+n,cmp);//从第一位开始排序
```

排序思想：冒泡思想。

思想：只要a[i]>ans大，就ans=a[i]。最终得到的ans是a数组中的最大元素。

下述为运用此思想的复赛集训模拟题示范：

```
//题目：求排好升序后数组相邻元素之差的最大值。如1 3 4 5
//3-1=2,4-3=1,2+1=3，因此最大值为3.
cin>>n;
for(int i=1;i<=n;i++) cin>>a[i];
sort(a+1,a+1+n);
for(int i=2;i<n;i++){//因为是求相邻元素，所以第一位和最后一位无需遍历。
    tmp=max(abs(a[i]-a[i-1]),abs(a[i+1]-a[i]));//求左右相邻差值的最大值
    if(tmp>ans) ans=tmp;//冒泡思想
}
cout<<ans<<endl;
```

（8）动态规划、DFS（T3、T4常考，选看）

动态规划(DP)：https://blog.csdn.net/m0_64036070/article/details/128723056

DFS:https://blog.csdn.net/Arabot_/article/details/129702049

（9）数学方法：

纯数学类题：

必会公式：

$$a^2-b^2=(a+b)(a-b)$$

$$a^2+2ab+b^2=(a+b)^2$$

$$a^2-2ab+b^2=(a-b)^2$$

$$1^2+2^2+3^2+\ldots\ldots+n^2=\frac{n(n+1)(2n+1)}{6}$$

必会知识点：一元一次方程、二元一次方程、三元一次方程(导成2个二元一次方程)、

情境题：

方法：找规律。

5.考前必须记住的30句话

1.比赛前一天晚上请准备好你的各种证件，事先查好去往考场的路线

2.比赛之前请先调整你的屏幕分辨率到你喜欢的大小

3.比赛之前请把编译器的字体调为你平时惯用的字体，尤其是注意这种字体中的逗号，点，1，l这种易混淆的字是不是区分明显

4.在不影响视野的情况下，请将字号尽可能调大，方便查错

5.请将题目通读完以后，再开始深入思考你认为最容易的一道题

6.即使这道题再容易，也不要着急写代码，请先明确自己每一步要干什么后，再开始写，轻敌会是你最大的错误

7.即使这道题看起来再没法做，也不要提早放弃，这个时候纸和笔会是你最好的朋友，自己尝试几个例子，也许你就会找到答案

8.请一定先明确自己要干什么之后再写程序，不要走一步想一步

9.如果这是一道动态规划题，请先把转移方程写在纸上再编程

10.涉及到边界处理、加一减一之类的问题，请在纸上举个例子，标上下标以后，在编程时参照纸上的下标写

11.如果思考30分钟仍一头雾水，没有可以实现的算法，请你果断屏蔽掉100%的那一栏数据，开始写60%，50%乃至30%的算法——在CSP里面，30分绝不是小数目

12.几个常用的复杂度参考：100以下——可能是搜索；100~500——N^3，1000~5000——N^2，100000~500000——NlogN，500000以上——N或1

13.如果你发现你旁边的人写得很快，请你放心，他的算法十有八九是错的， 调整好自己的心态很重要

14.虽然1s+128MB内存 (这是以前的了，现在应该是 1s + 256MB) 是标准配置，不过也不是每道题都是这样的，还是请认真阅读试卷首页的试题说明

15.计算内存的方法：数组大小*类型长度/1000 / 1000=所占内存MB数，int类型长度是4, long long =8

16.记不住的话，记住int型数组在128MB内存下最大开到2500 0000是比较保险的（占100MB内存）

17.写完程序之后，请一定不要忙着编译，请一定要将你的代码从头到尾通读一遍，也就是静态查错，这是整个编程过程中最重要的步骤，有的变量重复调用问题调试的话，一个小时也看不出来，静态查错可以一下指出错误

18.静态查错请注意以下方面：
（1）是否写上了using namespace std?
（2）数组开得是否够大?
（3）变量类型是否正确?
（4）memset时，所填的sizeof（XX）的XX是不是匹配？大小是不是正确？（Pascal 是 fillchar)
（5）外层循环与内层循环的i，j是不是混用了?
（6）循环之前，i，j是否定义了?
（7）输入数据都输入了吗?
（8）这个程序是在执行你想让它执行的步骤吗?

19.通过样例后，请你一定不要放松警惕，因为样例并不能覆盖所有的情况，请自己设计几组数据，争取卡死你的程序

20.如果出现问题，请你调试你的程序，请一定要分模块调试，不要从头跟到尾

21.如果你已经设计不出能卡住你的程序的数据，恭喜你可以做下一题了

22.return 0;是否写上了

23.在内存允许的情况下，能开普通队列就不要用循环队列，能开下普通数组就不要用滚动数组

24.在时间允许的情况下，能暴力就暴力，高精度能不压位就不压位，优化不需要的就不要

25.总之，在不超限制的前提下，能不优化就不优化，以减少代码量和出错概率为第一原则

26.当比赛还剩下5~15分钟的时候，请不要再改动你的程序，即使你怀疑它对你的一个输入给出了错误答案，因为你自己算出的结果也有可能是错的

27.这个时候请你检查是否注释掉了该注释掉的东西，文件名是否写对，文件夹是否建对，请一定反复检查！

28.今年的更改，没有人知道究竟会变成什么样，所以，与其瞻前顾后，不如集中精力做出你眼前的题目来的实际

29.请记住，CSP不怕暴力，怕瞎算，不怕不会，怕不敢

30.放平心态才能使写代码时候的思路达到最好状态。