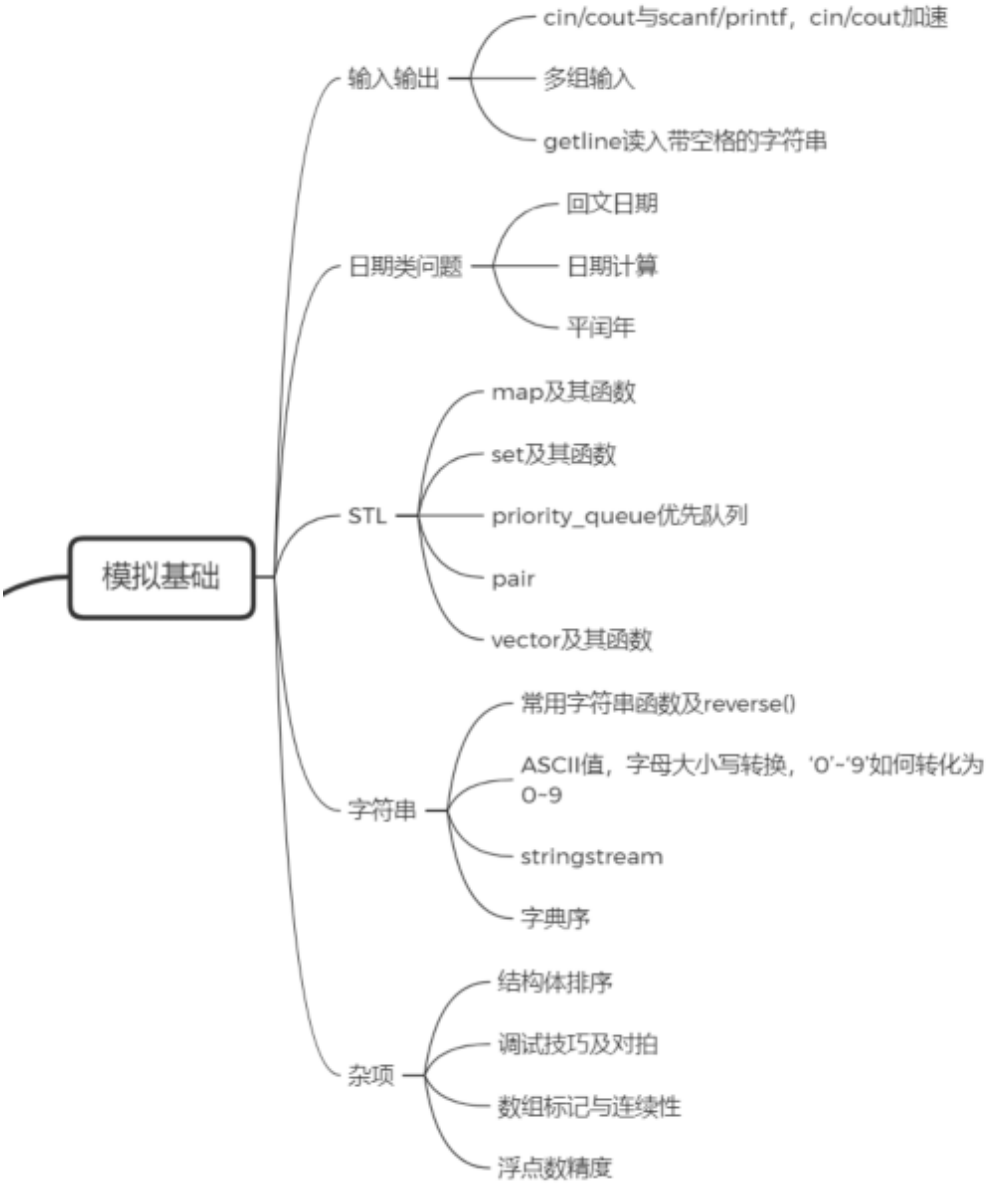
蓝桥杯十天冲刺省一


模拟基础
输入输出
cin/cout与scanf/printf，cin/cout加速
多组输入
getline读入带空格的字符串
日期类问题
回文日期
日期计算
平闰年
STL
map及其函数
set及其函数
priority_queue优先队列
pair
vector及其函数
字符串
常用字符串函数及reverse()
ASCII值，字母大小写转换，'0'~'9'如何转化为0~9
stringstream
字典序
杂项
结构体排序
调试技巧及对拍
数组标记与连续性
浮点数精度

Day-1 模拟基础

输入输出

cin/cout与scanf/printf

- 万能头文件 `#include<bits/stdc++.h>`
- `cin` 与 `cout` 是 C++ 提供的函数输入输出方便但速度较慢，所以需要用指令进行输入输出加速，**切记使用加速命令后不要同时使用cin/cout与scanf/printf**

```
#include <bits/stdc++.h>

using namespace std;

int main() {
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
  int x, y;                          // 声明变量
  cin >> x >> y;                     // 读入 x 和 y
  cout << y << endl << x;            // 输出 y，换行，再输出 x
  return 0;                          // 结束主函数
}
```

- `scanf` 与 `printf` 其实是 C 语言提供的函数。大多数情况下，它们的速度比 `cin` 和 `cout` 更快，并且能够控制输入输出格式。

1. `%s` 表示字符串。
2. `%c` 表示字符。
3. `%lf` 表示双精度浮点数（`double`）。
4. `%lld` 表示长整型（`long long`）。

5. `%llu` 表示无符号长整型（`unsigned long long`），无符号整数不能读入负数。

多组输入

我们在输入输出的时候要遵守题目的输入输出格式规范常见的有以下几种输入方式

有条件的多组输入

奇偶统计

- 题目描述

> *给你若干个数字，最后一个数字是0，让你统计这些数字中有有多少个偶数，和所有奇数的和。*

- 输入格式

> *一行，若干个数字，最后一个数字是0*

- 输出格式

> *第一行是这些数字中的偶数的个数*
> *第二行是这些数字中奇数的总和*

- 样例

输入

```
12 53 72 3 9 94 36 54 28 99 93 36 6 0
```

输出

```
8
257
```

- 题目思路与代码

利用while循环加if分支语句进行判断当输入的数字为 `0` 的时候break跳出循环

```
#include <bits/stdc++.h>
```

```
using namespace std;
int main()
{
    int n,num_even = 0,sum_odd = 0;
    while(cin >> n)
    {
        if(n == 0) break;//0的时候结束循环
        else
        {
            if(n % 2 == 0) num_even++;
            else sum_odd += n;
        }
    }
    cout << num_even << endl << sum_odd;
}
```

无条件的多组输入

奇偶统计改

题目同上，删去了以 `0` 作为循环结束的条件

- 题目思路与代码

利用while循环进行读入，但输入的时候你可能发现程序无法退出，这时候需要输入 `ctrl + z` 再按下回车就可以正常结束程序。

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int n,num_even = 0,sum_odd = 0;
    while(cin >> n)
    {
        if(n % 2 == 0) num_even++;
        else sum_odd += n;
    }
    cout << num_even << endl << sum_odd;
}
```

getline读入带空格的字符串

我们在读入字符串的时候如果用 `cin` 或 `scanf` 会无法读入空格，因为 `cin` 或 `scanf` 读入空格后会认为字符串读入已经结束。所以我们需要用 `getline` 读入带空格的字符串。

作文标题

- 题目思路

利用 `getline` 读入空格，利用字符串的 `size()` 函数求出字符串长度，for 循环遍历非空格的字符

- 代码

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    string str;
    getline(cin,str);
    int cnt = 0;
    int n = str.length();
    for(int i = 0;i < n;i++) {
        if(str[i] != ' ') {
            cnt++;
```

```
        }
    }
    cout << cnt << endl;
}
```

- 但有的时候我们也可以利用 `cin` 与 `scanf` 根据空格判定字符串读入结束的特性方便地解决一些问题

拓拓在打字

解法1

- 题目思路

利用 `getline`，输入整个字符串，然后只要不是连续出现的空格或者非空格字符，就输出

- 代码

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
    string s;
    getline(cin,s);
    for(int i = 0;i < s.size();i++)
    {
        if((s[i] == ' ' && s[i+1] != ' ') || s[i] != ' ')
            cout<<s[i];
    }
}
```

解法2

- 题目思路

利用 `cin` 不能读入空格的特性，结合 `while` 循环读入，可以巧妙忽略空格。

- 代码

```
#include<bits/stdc++.h>

using namespace std;

int main()
{
    string str;
    while(cin >> str) {
        cout << str << " ";
    }
}
```

getline与cin混用产生的问题及字符串与数字的转换

简单来讲就是 cin 会剩一个换行符， getline 会把这个换行符读进来导致直接结束字符串读入。

所以说我们在使用的时候，尽量避免 getline 与 cin 混用。

作文标题改

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    string t;
    string str;
    getline(cin, t);
    getline(cin,str);
    int cnt = 0;
    int n = stoi(t);//将字符串转化为数字
    for(int i = 0;i < n;i++) {
        if(str[i] != ' ') {
            cnt++;
        }
    }
    cout << cnt << endl;
```

```
}
```

在上面的代码中我们注意到我们使用了 `stoi` 函数将字符串转化为了数字，如果想将数字转化为字符串可以使用 `to_string()` 指令

```
#include<bits/stdc++.h>

using namespace std;

int main() {
    int n;
    string t;
    cin >> n;
    t = to_string(n);
    cout << t << endl;
}
```

日期问题

基础循环遍历模板

对于蓝桥杯所有的日期问题遍历，都可以使用的上

```
for(int year=2000;year<=2022;year++)
for(int month=1;month<=12;month++)
for(int day=1;day<=31;day++)
{
    if(month == 1 || month == 3 || month == 5 || month == 7 ||
month == 8 || month == 10 || month == 12);
    else if(month == 2)
    {
        if((year%4==0&&year%100!=0) || year % 400 == 0)
        {
            if(day > 29) break;
        }
        else
        {
            if(day > 28) break;
```

```
        }
    }
    else
    {
        if(day > 30) break;
    }
}
```

给定日期问过多少天后日期是多少

星系炸弹 （蓝桥杯C++2015B组省赛）

- 题目描述

在 X 星系的广袤空间中漂浮着许多 X 星人造“炸弹”，用来作为宇宙中的路标。每个炸弹都可以设定多少天之后爆炸。

比如：阿尔法炸弹 2015 年 1 月 1 日放置，定时为 15 天，则它在 2015 年 1 月 16 日爆炸。
有一个贝塔炸弹，2014 年 11 月 9 日放置，定时为 1000 天，请你计算它爆炸的准确日期。

请填写该日期，格式为 *yyyy-mm-dd* 即 4 位年份 2 位月份 2 位日期。比如：2015-02-19。

- 题目答案：2017-08-05
- 题目思路

直接模拟日期的计算与进位，我们假设年份用 *year* 表示，月份用 *month* 表示，日期用 *day* 表示

1.月份进位

(1)小月满31天进1月

(2)大月满32天进1月

(3)2月要分平年闰年，闰年满30天进1月，平年满29天进1月

(4)月份进1之后记得将天数 *day* 归1

2.年份进位

月份满13进1年，月份 *month* 归1

- 题目代码

```
#include<bits/stdc++.h>

using namespace std;

int m1[] = {0, 1, 3, 5, 7, 8, 10, 12};//数组m1存储大月
int m2[] = {0, 4, 6, 9, 11};//数组m2存储小月

int main() {
    int year = 2014, month = 11, day = 9;
    for(int i = 1;i <= 1000;i++) {//加1000天
        day++;
        for(int j = 1;j <= 7;j++) {//判断是否是大月
            if(month == m1[j] && day == 32) {//大月满32天进一月
                month++;
                day = 1;//天数重新回到1号
                break;
            }
        }
        for(int j = 1;j <= 4;j++) {//判断是否是小月
            if(month == m2[j] && day == 31) {//小月满31天进一月
                month++;
                day = 1;
                break;
            }
        }
        if(month == 2) {//2月单独判断
            if((year % 4 == 0 && year % 100 != 0) || year % 400
== 0) {//判断是否是闰年
                if(day == 30) {//闰年满30进一月
                    month++;
                    day = 1;
                }
            }
            else {
                if(day == 29) {//平年满29进一月
```

```
                    month++;
                    day = 1;
                }
            }
        }
        if(month == 13) {//满13月进1年
            month = 1;//月份回到1月重新计数
            year++;
        }
    }
    cout << year << "-" << month << "-" << day;
}
```

日期与日期之间有多少天

第几天 （蓝桥杯C++2018B组省赛）

- 题目描述

2000 年的 1 月 1 日，是那一年的第 1 天。那么，2000 年的 5 月 4 日，是那一年的第几天?

- 题目答案：125
- 题目思路

如果只是为了应付这道题目计算器或者手算一下都可以，但如果把题目扩展为任意的两个日期之间间隔多少天我们又该如何去做呢。

其实思路也很简单从开始日期一直加1，直到与最终日期相等为止，在上一题的代码基础上略作修改即可

- 代码

```
#include<bits/stdc++.h>

using namespace std;

int m1[] = {0, 1, 3, 5, 7, 8, 10, 12};//数组m1存储大月
int m2[] = {0, 4, 6, 9, 11};//数组m2存储小月
```

```
int main() {
    int num = 0;
    int year = 2000, month = 1, day = 1;
    int year2 = 2000, month2 = 5, day2 = 4;
    while(1) {
        num++;
        day++;
        for(int j = 1;j <= 7;j++) {//判断是否是大月
            if(month == m1[j] && day == 32) {//大月满32天进一月
                month++;
                day = 1;//天数重新回到1号
                break;
            }
        }
        for(int j = 1;j <= 4;j++) {//判断是否是小月
            if(month == m2[j] && day == 31) {//小月满31天进一月
                month++;
                day = 1;
                break;
            }
        }
        if(month == 2) {//2月单独判断
            if((year % 4 == 0 && year % 100 != 0) || year % 400
== 0) {//判断是否是闰年
                if(day == 30) {//闰年满30进一月
                    month++;
                    day = 1;
                }
            }
            else {
                if(day == 29) {//平年满29进一月
                    month++;
                    day = 1;
                }
            }
        }
        if(month == 13) {//满13月进1年
            month = 1;//月份回到1月重新计数
            year++;
```

```
        }
        if(year == year2 && month == month2 && day == day2) {//
到目标日期后跳出循环
            break;
        }
    }
    cout << num << endl;
}
```

日期与分钟之间的转换

纪念日（蓝桥杯C/C++2020B组省赛第一场）

- 题目描述

2020 年 7 月 1 日是×××××成立 99 周年纪念日。

×××××成立于 1921 年 7 月 23 日。

请问从 1921 年 7 月 23 日中午 12 时到 2020 年 7 月 1 日中午 12 时一共包含多少分钟?

- 题目答案：52038720
- 题目思路

我们此时计算的不是天数，但我们知道一天有多少分钟，所以累加的时候换成 1440 分钟即可。

- 代码

```
#include<bits/stdc++.h>

using namespace std;

int m1[] = {0, 1, 3, 5, 7, 8, 10, 12};//数组m1存储大月
int m2[] = {0, 4, 6, 9, 11};//数组m2存储小月

int main() {
    int num = 0;
    int year = 1921, month = 7, day = 23;
    int year2 = 2020, month2 = 7, day2 = 1;
```

```
    while(1) {
        num += 24 * 60;
        day++;
        for(int j = 1;j <= 7;j++) {//判断是否是大月
            if(month == m1[j] && day == 32) {//大月满32天进一月
                month++;
                day = 1;//天数重新回到1号
                break;
            }
        }
        for(int j = 1;j <= 4;j++) {//判断是否是小月
            if(month == m2[j] && day == 31) {//小月满31天进一月
                month++;
                day = 1;
                break;
            }
        }
        if(month == 2) {//2月单独判断
            if((year % 4 == 0 && year % 100 != 0) || year % 400 == 0) {//判断是否是闰年
                if(day == 30) {//闰年满30进一月
                    month++;
                    day = 1;
                }
            }
            else {
                if(day == 29) {//平年满29进一月
                    month++;
                    day = 1;
                }
            }
        }
        if(month == 13) {//满13月进1年
            month = 1;//月份回到1月重新计数
            year++;
        }
```

```
            if(year == year2 && month == month2 && day ==
day2) {
                break;
            }
        }
        cout << num << endl;
}
```

星期与周期性问题

星期计算（蓝桥杯Java2022B组省赛）

- 题目描述

  已知今天是星期六，请问 20^{22} 天后是星期几？

  注意用数字 1 到 7 表示星期一到星期日。

- 题目答案：7

- 题目思路

  题目难度不大，我们只需要看一下 $20^{22}\ mod\ 7$ 之后还剩几天，注意取mod即可。

- 代码

```
public class Main {
    public static void main(String[] args) {
        int currentDay = 6;
        int ans = 1;
        for (int i = 1; i <= 22; i++) {
            ans = (ans % 7) * 20 % 7;
        }
        int nextDay = (currentDay + ans - 1) % 7 + 1;
        System.out.println(nextDay);
    }
}
```

时间转换

时间显示（蓝桥杯C/C++2021B组省赛）

- 题目思路

只需要利用取余运算即可，我们假设题目输入为 n 毫秒

1.首先将 ms 转化为 s，$1s = 1000ms$ 所以开始先除以 1000，$n = n/1000$。

2.转化成秒之后，先需要对一天有多少秒进行取余，这样剩下的时间肯定不到一天才能进行时间计算，也就是 mod 86400。

3.对一天有多少秒取余后，我们先算还剩下多少小时也就是 $n/3600$, 然后再 mod 3600 剩下的秒数计算分钟

4.取余后算一下还有多少分钟也就是 $n/60$，最后再 mod 60 输出还剩下多少秒。

- 代码

```
#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

int main()
{
    int h,m,s;
    ll n;
    cin >> n;
    n = n / 1000 % 86400; // 毫秒化秒，并且保留最后一天天数
    h = n / 3600;  //求得最后一天的小时
    n = n % 3600;
    m = n / 60;    //分钟
    s = n % 60;  //秒数
    printf("%02d:%02d:%02d",h,m,s); //02d的意思是如果不足俩位数，前补0
    return 0;//蓝桥杯编程题最后一定要记得加return 0;
```

```
}
```

回文日期（蓝桥杯C/C++2020B组省赛第二场）

- 题目思路

1.首先要用 `for` 循环从题目要求的日期开始往后遍历，假设日期为 $date$,把日期先当作数字也就是从 $date+1$ 开始往后遍历。

2.然后我们要判断当前日期是否合法，先利用取余与除法分离出年/月/日

(1) $year = date/10000$ 、 $month = date\%10000/100$ 、$day = date\%100$

(2) 月份和天数必须要符号日期格式，$1 \leq month \leq 12$、天数 day 要根据大小月和 2 月平闰年判断合法性

- 代码

```
#include<bits/stdc++.h>

using namespace std;

int months[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};

bool check(int date) {
    int year = date / 10000;//从date中分离出年
    int month = date % 10000 / 100;//从date中分离出月
    int day = date % 100;//从date中分离出日

    if (!day || month <= 0 || month > 12) return false;//月份和
日期为0不合法
    if (month != 2 && day > months[month]) return false;//日期
不合法
    if (month == 2) {//2月单独判断日期是否合法
        if ((year % 4 == 0 && year % 100 != 0) || (year % 400
== 0)) {//判断平闰年
            if (day > 29) return false;//闰年日期不合法
        } else {
```

```
            if (day > 28) return false;//平年日期不合法
        }
    }
    return true;
}

bool check1(string s) {//判断是否是回文数
    int len = s.size();
    for (int i = 0, j = len - 1; i < j; i++, j--) {//两个变量当两个指针，一个从头一个从尾用if判断
        if (s[i] != s[j]) return false;
    }
    return true;
}

bool check2(string s) {
    if (check1(s)) {//先确定是否是回文数
        if (s[0] != s[2] || s[1] != s[3] || s[0] == s[1]) return false;//判断是否是ABABBABA型
        return true;
    }
    return false;
}

int main() {
    int date, flag = 0;
    cin >> date;
    for (int i = date + 1;; i++) {
        if (check(i)) {//判断日期是否合法
            string s = to_string(i);//判断完后将年月日从数字转化为字符串，方便进行回文判断
            if (check1(s) && !flag) {//flag标记避免重复输出
                cout << i << endl;
                flag = 1;
            }
            if (check2(s)) {//判断是否是ABABBABA型
                cout << i << endl;
                return 0;
            }
```

```
        }
    }
    return 0;
}
```

练习题

日期问题（蓝桥杯C/C++2017B组省赛）

跑步锻炼（蓝桥杯C/C++2020B组省赛第二场）

STL

map及其函数

`map<key,value>` 提供一对一的数据处理能力，由于这个特性，它完成有可能在我们处理一对一数据的时候，在编程上提供快速通道。map 中的第一个值称为关键字(key)，每个关键字只能在 map 中出现一次，第二个称为该关键字的值(value)，可以重复。

```
// 定义，以 string, int 对为例
#include <map>
map<string, int> mp;     // 底层是红黑树，元素可比较大小，key 的结构体要重载 < 运算

// 2- 插入
mp.insert(make_pair("zhs", 18));         // 插入一个键值对，若原先存在该 key，则无法插入和覆盖
mp.insert(pair<string,int>("zhs", 18)); // 插入一个键值对，若原先存在该 key，则无法插入和覆盖
mp["zhs"] = 18;                          // 甚至可以类似数组下标去访问 key 对应的 value，若原先不存在该 key，则创建，若原先存在，则覆盖以前该关键字对应的值
mp.emplace("zhs", 18);                   // 插入效果跟 insert 效果完全一样

// 3- 删除
```

```
mp.erase(key);      // 删除值为 key 的元素
mp.erase(iter);     // 删除迭代器 iter 指向的元素，例如
mp.erase(mp.begin());
mp.erase(iter1, iter2); // 删除区间 [iter1, iter2) 的所有元素，例
如 mp.erase(mp.begin(), mp.end());
mp.clear();         // 清空集合

// 3- 求大小
int siz = mp.size();          // 求集合大小
bool flag = mp.empty();       // 集合判空

// 4-查询
if(mp.find(key) != mp.end())            // find 函数返回一个指向
被查找到元素的迭代器
    cout << mp[key] << endl;
if(mp.count(key))                 // count 返回某个值元素的个数
    cout << mp[key] << endl;
auto iter = mp.lower_bound(key);    // 求 key 的下界，返回指向大于
等于某值的第一个元素的迭代器
auto iter = mp.upper_bound(key);    // 求 key 的上界，返回大于某个
值元素的迭代器

// 5-遍历
map<string, int>::iterator iter;              // 正向遍历
for(iter=mp.begin(); iter!=mp.end(); iter++)
{
    cout << iter->first << " " << iter->second << endl;
    // 或者
    cout << (*iter).first << " " << (*iter).second << endl;
}
map<int>::reverse_iterator riter;     // 反向遍历
for(riter=mp.rbegin(); riter!=mp.rend(); riter++)
{
    // 遍历的同时修改
    iter->second += 10;
    cout << iter->first << " " << iter->second << endl;
}
for (auto x : mp){                // C++ 11 auto 遍历
    cout << x.first << " " << x.second << endl;
```

```
}
for (auto& x : mp){                // C++ 11 auto 遍历
    x.second += 10;   // 遍历并修改
    cout << x.first << " " << x.second << endl;
}

// 6- 求最值
map<string, int>::iterator it = mp.begin();    // 最小值
cout << *it << endl;
map<string, int>::iterator it = mp.end();      // 最大值
cout << *(--it) << endl;

/*
1. map 和 set 一样，其中的元素必须支持 < 运算符
2. 在插入时，使用 insert 和 用数组方式去插入，在原先存在要插入的键值时是有区别的，insert不会插入，用数组方式插入则会直接覆盖原先数据
3. map 的迭代器支持 ++、--、==、!=、(*iter).first、(*iter).second、iter->first、 iter->second 等操作
4. map 的迭代器不支持 +、-、 +=、-= 等算术操作符，也不支持 >、<、>=、<= 等比较操作符
*/
```

map的插入与遍历

```
#include<bits/stdc++.h>

using namespace std;

map<string,int> m;

int main(){
    int n,age;
    string s;
    cin >> n;
    for(int i = 1;i <= n;i++){
        cin >> s >> age;
        m[s] = age;
    }
```

```
    map<string,int>::iterator it;
    for(it = m.begin();it != m.end();it++){
        cout << it->first<< " "<< it->second<<endl;
    }
}
```

map的查询

```
#include<bits/stdc++.h>
using namespace std;
map<string,string> ma;
pair<string,string> p;
int main(){
    int m,n,a;
    string s,t,str;
    set<string,string>::iterator it;
    cin >> n >> m;
    for(int i=1;i<=n;i++){
        cin >> s >> t;

        p.first = s;
        p.second = t;
        ma.insert(p);
    }
    for(int i=1;i<=m;i++){
        cin >> a>> str;
        if(a==1){
            if(ma.find(str)!=ma.end()){
                cout << ma[str];
            }else{
                cout << "NO";
            }
        }
        if(a==2){
            if(str <= (ma.begin()->first))
            {
                cout<<"NO";
            }
            else{
```

```
                auto it = ma.lower_bound(str);
                it--;
                cout<< it->second;
            }
        }
        if(a==3){
            if(str >= ((--ma.end())->first))
            {
                cout<<"NO";
            }
            else{
                auto it = ma.upper_bound(str);
                cout<<it->second;
            }
        }
        cout << endl;
    }
}
```

set及其函数

set 的含义是集合，它是一个 **有序** 的容器，里面的元素都是排序好的，支持插入、删除、查找等操作，就像一个集合。所有的操作的都是严格在 $O(logn)$ 时间之内完成，效率非常高。 set 和 multiset 的区别是：set插入的元素不能相同，但是 multiset 可以相同。

```
// 1- 定义
#include <set>
set<int> s;     // 元素必须可比较大小，元素类型必须要支持 < 运算，结构体需要重载 <

// 2- 插入
s.insert(key);    // 插入

// 3- 删除
s.erase(key);     // 删除值为 key 的元素
s.erase(iter);    // 删除迭代器 iter 指向的元素，例如 s.erase(s.begin());
```

```
s.erase(iter1, iter2); // 删除区间 [iter1, iter2) 的所有元素，例如 s.erase(s.begin(), s.end());
s.clear();         // 清空集合

// 4- 求大小
int siz = s.size();          // 求集合大小
bool flag = s.empty();       // 集合判空

// 5-查询
if(s.find(key) != s.end())           // find 函数返回一个指向被查找到元素的迭代器
    cout << "exist" << endl;
if(s.count(key) == 1)                // count 返回某个值元素的个数
    cout << "exist" << endl;


set<int>::iterator iter = s.lower_bound(key);   // 求 key 的下界，返回指向大于等于某值的第一个元素的迭代器
set<int>::iterator iter = s.upper_bound(key);   // 求 key 的上界，返回大于某个值元素的迭代器

// auto 类型推断关键字 在NOI 系列比赛中也可以使用了
auto iter = s.lower_bound(key);   // 求 key 的下界，返回指向大于等于某值的第一个元素的迭代器
auto iter = s.upper_bound(key);   // 求 key 的上界，返回大于某个值元素的迭代器

// 6-遍历
set<int>::iterator iter;             // 正向遍历
for(iter=s.begin(); iter!=s.end(); iter++)
{
    cout<<*iter<<endl;
}
set<int>::reverse_iterator riter;    // 反向遍历，不重要
for(riter=s.rbegin(); riter!=s.rend(); riter++)
{
    cout<<*riter<<endl;
}
```

```
// 7- 求最值
set<int>::iterator it = s.begin();     // 最小值
cout << *it << endl;
set<int>::iterator it = s.end();       // 最大值
cout << *(--it) << endl;

/*
注意:
1. set 和优先队列一样，其中的元素必须支持 < 运算符
2. set 的迭代器支持 ++、--、==、!=、*iter 等操作
3. set 的迭代器不支持 +、-、 +=、-= 等算术操作符，也不支持 >、<、
>=、<= 等比较操作符
*/
```

set的插入与遍历

```
#include<bits/stdc++.h>
using namespace std;
int main(){
    set<int> s;//set可以自动去重和排序，默认升序
    int n,t;
    cin >> n;
    for(int i=1;i<=n;i++){
        cin >> t;
        s.insert(t);//set没有push_back操作
    }
    set<int>::iterator it;//set需要使用迭代器遍历数据
    for(it=s.begin();it!=s.end();it++){//set支持双向迭代器
        cout << *it << " ";
    }
}
```

set的查询

```
#include<bits/stdc++.h>
using namespace std;
int n, m, c, x;
set<int>a;
set <int, greater<int>> b;
```

```
int main() {
    cin >> n >> m;
    for (int i = 1; i <= n; i++) {
        int t;
        cin >> t;
        a.insert(t);
        b.insert(t);
    }

    for (int i = 1; i <= m; i++) {
        cin >> c >> x;
        if (c == 1) {
            if (a.find(x) != a.end())cout << x << "\n";
            else cout << "NO\n";
        }
        if (c == 2) {
            set<int>::iterator it;
            it = b.upper_bound(x);//二分查找，找第一个大于x的数的
地址
            if (it != b.end()) {
                cout << *it << "\n";
            } else cout << "NO\n";
        }
        if (c == 3) {
            set<int>::iterator it;
            it = a.upper_bound(x);
            if (it != a.end()) {
                cout << *it << "\n";
            } else cout << "NO\n";
        }
    }
}
```

priority_queue优先队列

因为堆的作用主要是用来获取最大/最小值，类似队列的取最值操作，因此堆有一个别名叫优先队列。在 STL 中提供了"优先队列"的模板，即 `priority_queue` 。关于优先队列具体的应用会在二叉堆与对顶堆中详细讲解。

- 常见使用

```
#include <queue>
using namespace std;
// using std::priority_queue;
priority_queue<int> q;    // 定义一个空的优先队列，默认是大根堆

void test()
{

    int x = 5;
    q.push(x);        // 将 5 插入堆，这时堆顶是 5
    q.push(3);        // 将 3 插入堆，这时堆顶是 5
    q.push(2*x);      // 将 10 插入堆，这时堆顶是 10
    int k = q.top(); // 取堆顶元素，应该是 10
    q.pop();          // 删除堆顶元素，删除了 10，这时堆顶为 5
    int s = q.size();// 求堆中元素个数，应为 2
    // 清空优先队列的一种方法
    while(!q.empty())
        q.pop();
    // 清空优先队列的第二种方法
    q = priority_queue<int>();
}
```

- 实现小根堆

```
// 大根堆改成小根堆，其中 vector<Type> 为内部存储容器，
greater<Type> 为比较方式
priority_queue<Type,vector<Type>,greater<Type>> q;

// 常为 int 类型
priority_queue<int,vector<int>,greater<int>> q;
```

- 优先队列与结构体重载运算符

```
// 只能用于大根堆的结构体
struct node{
    int val;
    string info;
```

```
    ...
    // 大根堆，重载小于运算符
    bool operator <(const node& other) const {
        return val < other.val;
    }
};
// 定义一个大根堆（默认）
priority_queue<node> q;

// 只能用于小根堆的结构体
struct node{
    int val;
    string info;
    ...
    // 小根堆，重载大于运算符
    bool operator >(const node& other) const {
        return val > other.val;
    }
};
// 定义一个小根堆（默认）
priority_queue<node,vector<node>,greater<node>> q;
```

练习题

最大开支（蓝桥杯Java2023B组省赛）

pair

将两个变量成对组合打包在一起的数据类型，可以理解为C++内置的保存两个元素(类型可以自定义)的结构体常见的应用就是保存坐标等
pair 支持比较运算符。其比较规则是:先比较第一个元素，如果第一个元素值相等，再比较第二个元素。

```
pair<int,int> pos;
pos.first = 1;
pos.second = 1;
a = make_pair(1,1);
a = pair<int,int>(1,1);
```

练习题

扫雷（蓝桥杯C/C++2022B组省赛）

vector及其函数

问题	普通数组	vector 数组
定义数组	int a[N];int a[N] = {0}; // 全初始为 0	vector<int> a;vector<int> a(N); vector<int> a(N, 1);// 全赋值为 1
输入元素	cin >> a[i];	cin>>t; a.push_back(t);
比较两数组相等	for(int i = 0; i < n; i++){ if(a[i] == b[i]) ..... }	if(a == b)
删除最后一个元素	len--;	a.pop_back();
删除下标 i 位置的元素	for(int j = i+1; j < n; j++){ a[j-1] = a[j]; }	a.erase(a.begin()+i, a.begin()+i+1);
删除下标 t1~t2的所有元素	。。。。。。	a.erase(a.begin()+t1, a.begin()+t2+1);
在下标 i 处插入一个元素	for(int j = n+1; j >= i+1; j--){ a[j] = a[j-1]; }a[i] = t;	a.insert(a.begin()+i, t);
翻转数组	reverse(a, a+n);	reverse(a.begin(), a.end());
排序	sort(a, a+n);	sort(a.begin(), a.end());

vector插入

```
#include<bits/stdc++.h>
using namespace std;
int a[15];
int main(){
    int n,x,y;
    cin >> n ;
    for(int i=1;i<=n;i++){
        cin >> a[i];
    }
    cin >> x >> y;
    vector<int> v(a+1,a+n+1);
    v.insert(v.begin()+x-1,y);
    for(int i=0;i<v.size();i++){
        cout << v[i] << endl;
    }
}
```

vector遍历

```
#include<bits/stdc++.h>
using namespace std;
vector<int> a[100100];
int n,m,i,j,x,y;
int main(){
    cin >> n>> m;
    for(i=1;i<=m;i++){
        cin>> x>> y;
        a[x-1].push_back(y);//这里使用a[x]也是可以的
    }
    for(i=0;i<n;i++){
        cout << a[i].size();
        sort(a[i].begin(),a[i].end());//对每一个向量排序
        for(j=0;j<a[i].size();j++){
            cout << " "<<a[i][j];
        }
        cout<< endl;
    }
    return 0;
}
```

vector排序

```
#include<bits/stdc++.h>
using namespace std;
vector<int> a[1005];//定义向量数组
int main(){
    int n,c,x;
    cin >> n;
    for(int i=1;i<=n;i++){
        cin >> c;
        for(int j=1;j<=c;j++){
            cin>> x;
            a[i].push_back(x);
        }
        sort(a[i].begin(),a[i].end());
    }
```

```
    sort(a+1,a+n+1);
    for(int i=1;i<=n;i++){
        for(int j=0;j<a[i].size();j++){
            cout << a[i][j]<< " ";
        }
        cout << endl;
    }
}
```

字符串

常见字符串函数与reverse

```
// 最常用的操作
str.size();//返回字符串长度
str.length();//返回字符串长度
str.empty();//检查 str 是否为空，为空返回 1，否则返回 0
str[n];//存取 str 第 n + 1 个字符
str.at(n);//存取 str 第 n + 1 个字符（如果溢出会抛出异常）
// 反转
reverse(str.begin(), str.end());
// 查找
str.find("ab");//返回字符串 ab 在 str 的位置
str.find("ab", 2);//在 str[2]~str[n-1] 范围内查找并返回字符串 ab
在 str 的位置
str.rfind("ab", 2);//在 str[0]~str[2] 范围内查找并返回字符串 ab
在 str 的位置
if(str.find("ab")!=string::npos)
{    cout << "下标为:" << str.find("ab"); }
// 子串
str.substr(3);  // 返回 [3] 及以后的子串
str.substr(2, 4); // 返回 str[2]~str[2+(4-1)] 子串(即从[2]开始4
个字符组成的字符串)
str.substring(5, 10);  // 返回 str[5]~str[9] 子串(不包含结尾)
// 插入
str.insert(2, "sz");//从 [2] 位置开始添加字符串 "sz"，并返回形成的
新字符串
```

```
str.insert(2, "abcd", 3);//从 [2] 位置开始添加字符串 "abcd" 的前 3 个字符，并返回形成的新字符串
str.insert(2, "abcd", 1, 3);//从 [2] 位置开始添加字符串 "abcd" 的前 [2]~[2+(3-1)] 个字符，并返回形成的新字符串
// 删除
str.erase(3);//删除 [3] 及以后的字符，并返回新字符串
str.erase(3, 5);//删除从 [3] 开始的 5 个字符，并返回新字符串
// 替换
str.replace(2, 4, "sz");//返回把 [2]~[2+(4-1)] 的内容替换为 "sz" 后的新字符串
str.replace(2, 4, "abcd", 3);//返回把 [2]~[2+(4-1)] 的内容替换为 "abcd" 的前3个字符后的新字符串
// 追加
str = str + "abc";
str.push_back('a');//在 str 末尾添加字符'a'
str.append("abc");//在 str 末尾添加字符串"abc"
```

字符串编号（蓝桥杯Java2020B组省赛第一场）

子串

```
//可通过90%数据
#include<bits/stdc++.h>
using namespace std;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    string a,b;
    int s=0,t;
    cin>>a>>b;
    t=a.find(b);
    while(t!=string::npos)
    {
        s++;
        t=a.find(b,t+1);
    }
    cout<<s;
```

```
}
```

删除字符

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string s;
    char c;
    getline(cin,s);
    cin>>c;
    int p=s.find(c);
    while(p!=string::npos)
    {
        s.erase(p,1);
        p=s.find(c);
    }
    cout<<s;
    return 0;
}
```

替换单词

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    string s,w,t;
    int p;
    cin>>s;
    w="black";
    t="block";
    p=s.find(w);
    while(p!=string::npos)//也可改成while(p!=-1)
    {
        s.replace(p,5,t);//替换字符
        p=s.find(w);
    }
```

```
    cout<<s<<endl;
    return 0;
}
```

ASCII码值、字母大小写转换、'0'~'9'

```
// 数字转字符: 'A'(65)  'a'(97)  '0'(48)
char A = char(65);
char a = char(97);
char c = 'a' + 2;       // 'c' = 'a' + 2
char seven = '0' + 7;  // '7' = '0' + 7

// 字符转数字
int a = 'a';    // a: 97
int A = 'A';    // A: 65
int t = 't' - 'a';     // 计算字母间差值
int seven = '7' - '0';
```

stringstream

`stringstream` 主要是用在字串分割，可以先用 `clear()` 以及 `str()` 将字符串读入并进行分割，再用 >> 把内容输出，例如：

```
string s;
stringstream ss;
int a, b, c;
getline(cin, s);
ss.clear();
ss.str(s);
ss >> a >> b >> c;
```

stringstream练习

- 代码

```
#include <bits/stdc++.h>
```

```
using namespace std;

int main() {
    string s;
    stringstream ss;
    int n, i, sum, a;
    cin >> n;
    getline(cin, s);
    for(i = 1;i <= n;i++){
        getline(cin, s);
        ss.clear();
        ss.str(s);
        sum = 0;
        while(1) {
            ss >> a;
            if (ss.fail()) break;
            sum += a;
        }
        cout << sum << endl;
    }
}
```

sprintf

sprintf是一个比较冷门的函数，但是作用十分的强大，可以直接把相应的整数拼接组合成我们规定的格式，这个在蓝桥杯的日期问题中也非常的好用。

顺子日期（蓝桥杯C/C++2022年B组省赛）

```
#include <iostream>

using namespace std;

char str[15];

int main()
{
    int cnt = 0,flag = 0,ans=0;
```

```
    for(int year=2022;year<=2022;year++)
    for(int month=1;month<=12;month++)
    for(int day=1;day<=31;day++)
    {
        if(month == 1 || month == 3 || month == 5 || month == 7
|| month == 8 || month == 10 || month == 12);
        else if(month == 2)
        {
            if((year%4==0&&year%100!=0) || year % 400 == 0)
            {
                if(day > 29) break;
            }
            else
            {
                if(day > 28) break;
            }
        }
        else
        {
            if(day > 30) break;
        }

        //转换格式
        sprintf(str+1,"%d%02d%02d",year,month,day);

        //判断顺子
        for(int k=4;k+2<=8;k++)
        {
            if(str[k+2]-str[k+1]==1 && str[k+1] - str[k] ==1)
            {
                puts(str+1);
                ans++;
                break;
            }
        }

    }
    cout<<ans;
}
```

练习题

日期统计（蓝桥杯C/C++2023年B组省赛）

字典序

什么是字典序：所谓字典序就是以ASCII码排序。

（1）比如两个字符串 `abcd` 和 `abdd` 比较大小。从第一个字符开始逐位比较，第一个字符不相等，谁的ASCII码值小谁的字典序就小。若第一个相等，继续逐位比较后续字符。比较到字母c < 字母d，所以第一个字符串 `abcd` 字典序较小。

（2）再比如 `hist` 和 `history` 比较大小。若逐位比较都相等，但其中一个没有后续的字符了，则较短的串 `hist` 字典序较小。

（3）使用 `sort()` 可以对字符串进行字典序排序，字符按ASCII码值由小到大排列

```
string str = "agfdvdsgds";
sort(str.begin(),str.end());
```

拓拓拼字符

```
#include<bits/stdc++.h>

using namespace std;

string s[100];

int main() {
    int n,l;
    cin >> n >> l;
    for (int i = 0; i < n; i++) cin >> s[i];
    sort(s, s + n);
    for (int i = 0; i < n; i++) cout << s[i];
    cout << endl;
    return 0;
}
```

排列字母（蓝桥杯Python2022B组省赛）

杂项

结构体排序

结构体是 **自定义类型** ，它就像一个 **能装不同数据类型** 的“包裹”。当你声明一个结构体后，需要赋予这个结构体一个名字（即 **结构体名** ）。而 **定义具体变量时，也可以定义结构体数组** 。这相当于告诉计算机，你一口气定义了许多“包裹”，每一个“包裹”里面都可以装入许多不同或相同数据类型的变量。

1.结构体的声明

使用结构体前，需要先声明一个结构体类型，再进行定义和使用。结构体类型的声明格式为：

```
struct 结构体类型名{      //其中，sruct是关键字，在c++中表示结构体
    数据类型 成员变量1;     //多个成员变量可以具有不同的数据类型
    数据类型 成员变量2;
    ......
};
```

声明之后，就可以定义结构体变量了，格式如下：

```
结构体类型名 变量名；（struct 可省略）
```

例如

我们需要表示一个学生的信息，首先我们声明学生的结构体：

```
struct student{   //student表示结构体的类型名
    string name;
    char sex;
    int age;
    float height,score;
};
```

然后定义一个具体的学生：

```
student zhangsan;//声明一个student的结构体变量叫zhangsan
```

当然我们如果需要很多学生，可以声明为结构体数组：

```
student a[1001];//创建结构体数组a
//a数组中的每一个元素都是一个结构体类型student的变量
```

另外

为方便起见，我们一般直接在声明结构体的同时定义变量，格式如下：

```
struct 结构体类型名{      //其中，sruct是关键字，在c++中表示结构体
    数据类型成员变量1;  //多个成员变量可以具有不同的数据类型
    数据类型成员变量2;
    ......
}结构体变量表;
```

比如：

```
struct student{
    string name;
    char sex;
    int age;
    float height,score;
}a[1001];
```

2.结构体的使用

（1）将结构体变量视为一个整体进行操作，例如：

```
swap(a[1], a[2]);
```

（2）使用符号“.”对结构体变量的成员进行操作，“.”的含义可以理解为中文的“的”，格式为：

```
结构体名.成员变量名
```

比如：

```
student  a;//李四
cin >> a.name;//输入赋值李四的名字
a.age = 23;//直接赋值给李四的年龄为23
```

3.结构体的初始化

（1）集合形式，比如：

```
student  a = {"tuotuo",'F',12,168,100};
```

（2）逐一赋值，比如：

```
student  a;
cin >> a.name >> a.sex;
a.age = 12;
```

4.结构体中的sort()函数

现在给你一个结构体，现在有这样一个要求：按照年龄从小到大输出。

那我们就照结构体中成员变量age（也就是年龄）进行从小到大的排序，代码如下：

```
struct student {
    string name;
    int age;
    float height;
}a[10];
bool cmp(student a, student b){
    return  a.age < b.age;
}
int main(){
    ...//输出
        sort(a, a + n, cmp);//按照年龄从小到大排序
    ...//输出
}
```

阿里巴巴与金币

```cpp
#include<bits/stdc++.h>
using namespace std;

struct Node {
    double m,v;
    double avg;
};

Node a[110];

bool cmp(Node x,Node y) {
    return x.avg > y.avg;
}

int main()
{
    int n,t;
    cin >> n >> t;
    for(int i = 1;i <= n;i++) {
        cin >> a[i].m >> a[i].v;
        a[i].avg = a[i].v / a[i].m;
    }
    sort(a + 1,a + 1 + n,cmp);
    double sum = 0;
    for(int i = 1;i <= n;i++) {
        if(t >= a[i].m) {
            sum = sum + a[i].v;
            t = t - a[i].m;
        }
        else {
            sum = sum + t * 1.0 * a[i].avg;
            break;
        }
    }
    cout << fixed << setprecision(2) << sum;
    return 0;
}
```

数组标记与数组连续性

字符统计（蓝桥杯Java 2022B组省赛 | 数组标记）

- 题目思路

数组标记是指用数组去存储问题中的输入数据或者中间数据，比如计数、判断有无，本质上是用空间换时间，最常见的好处就是降低代码的时间复杂度，不容易出现超时问题。同时这种标记法也可以用上我们之前的 `map` 。

本题只需要用数组去标记出现字符出现的次数即可，需要结合前面字符ASCII值的知识。

```
// 标记数组
int flag[N];  // flag[i] 用于标记状态或者计数

flag[i] = 1;  // 数组标记状态，选择 / 不选择

flag[i]++;    // 数组标记计数，记录i出现的次数
```

- 代码

```
#include<bits/stdc++.h>

using namespace std;

string s;
int b[26];

int main() {
    cin >> s;
    for (int i = 0; i < s.size(); i ++)
        b[s[i] - 'A'] ++;
    int maxx = 0;
    for (int i = 0; i < 26; i ++)
        maxx = max(maxx, b[i]);
    for (int i = 0; i < 26; i ++)
```

```
        if (b[i] == maxx) cout << char(i + 'A');
    return 0;
}
```

单词分析（蓝桥杯Java2020B组省赛第二场 | 数组标记）

数组连续性

```
// 连续最长相同、最长上升或下降：  相邻两项比较，并讨论即可
int a[N], s = 0;
int ans = 0;

for(int i = 1; i <= n; i++) {
    if(a[i] == a[i-1]) {  // a[i] < a[i-1]  或 a[i] > a[i-1]
        s++;
        ans = max(ans, s);
    } else {
        s = 1;
    }
}
cout << ans;
```

拓拓的字符串压缩

- 题目思路

先求出字符串长度，然后从下标 0 开始每次和后面一个字符比较

（1）如果和后面一个字符相等，计数的变量加1

（2）如果和后面一个字符不相等，输出字符以及计数的变量，并将计数的变量清空为1

最后需要在循环外，再输出最后一个字符以及计数变量。为什么呢？我们考虑一下 `aabbccc` 最后一个字符 `c` 是不是后面没有不同的字符，我们发现我们只会在字符串不相等的时候才输出，所以最后一个字符 `c` 和它出现的次数没有输出。

- 代码

```
#include<bits/stdc++.h>
```

```cpp
using namespace std;

int main()
{
    string str;
    cin >> str;
    int n = str.size();
    int c = 1;
    for(int i = 0;i < n - 1;i++) {
        if(str[i] == str[i + 1]) {//相等计数加1
            c++;
        }
        else {
            cout << str[i] << c;
            c = 1;//清空为1不是0，因为这个字符出现了代表至少有1个
        }
    }
    cout << str[n - 1] << c;//可能最后几个字符连续相等需要在循环外补充输出
    return 0;
}
```

调试技巧与对拍

调试中常见错误

1.注意 数据范围!!!定义单个变量能用 `long long` 就不要用 `int`，能定义为 `double` 就不要用 `float`。
2. `maxn`、 `minn`(最大值、最小值)或者 `cnt`(计数器)忘记 赋初始值。
3.忘记删除用来检查代码的语句，比如添加了多余的打印操作。
4.没理解题目的意思，就开始做题。要结合样例去理解题目。
5.注意题目数据范围很大的时候， 考虑优化。
6.数组开太大。比如: `int arr[10000][10000]`，则 arr 的空间大小约为 400M，远超题目的空间限制。
7.边界条件考虑不充分。
8.变量命名与c++自带的名称冲突(因此，慎用万能头文件)。如:int time; int max; int min等等。

9.写了初始化函数，没有调用。
10.使用函数时，参数传错。
11.用了一个新的知识点，但自己不太清楚，就用了。要注意，写的代码最好是你完全能明白的。
12.if里面的判断条件没想清楚，还继续往下写。
13.循环里面条件写反，比如: `for(int i = n;i > 0;i+ +)`
14.双重循环里面 i 和 i 写反，或者两个循环变量都写成 i。
15.字符串 `string` 中，遍历 `string` 的操作最好用 `for(int i = 0;i < s.size(); i++)`，不要用小于等于 `i<= s.size() - 1`

对拍

- 先写出保证正确性的暴力代码，假设叫 `baoli.cpp`，编译运行得到 `baoli.exe`
- 再写出不确定正确性的正解，假设叫 `zhengjie.cpp`，编译运行得到 `zhengjie.cpp`
- 最后写出数据生成器，假设叫 `generator.cpp`，编译运行得到 `generator.exe`
    - 生成随机数组

```
int main(){
    freopen("data.in", "w", stdout);
    srand(time(0));  // 初始化随机种子
    int n = rand();
    cout << n <<endl;
    for(int i = 1; i <= n; i++) {
        cout << rand() << " ";
    }
    return 0;
}
```

    - 生成 1 ~ n 的随机排列

```
int a[N];
int main(){
    freopen("data.in", "w", stdout);
    srand(time(0));  // 初始化随机种子
    int n = rand() % 100 + 1;
    cout << n <<endl;
```

```
    for(int i = 1; i <= n; i++) {
        a[i] = i;
    }
    random_shuffle(a+1, a+n+1);
    for(int i = 1; i <= n; i++) {
        cout << a[i] << ' ';
    }
    return 0;
}
```

- 生成普通树

```
int fa[N];
int main(){
    freopen("data.in", "w", stdout);
    srand(time(0));  // 初始化随机种子
    int n = rand() % 100 + 1;
    cout << n <<endl;
    for(int i = 2; i <= n; i++) {
        fa[i] = rand()% (i-1) + 1;  // fa[i]: 1~i-1
    }
    // 按照格式要求去输出树的信息
    // 1: 输入的是每个结点的父节点信息
    for(int i = 1; i <= n; i++) {
        cout << fa[i] << ' ';
    }
    // 2: 输入 x, y
    for(int i = 1; i <= n; i++) {
        cout << i << " " << fa[i] << '\n';
    }
    return 0;
}
```

- 生成二叉树

```
int fa[N], cnt[N], l[N], r[N];
int main(){
    freopen("data.in", "w", stdout);
    srand(time(0));  // 初始化随机种子
```

```
    int n = rand() % 10 + 1;
    cout << n <<endl;
    for(int i = 2; i <= n; i++) {
        fa[i] = rand()% (i-1) + 1;  // fa[i]: 1~i-1
        while(cnt[fa[i]] >= 2) {
            fa[i] = rand()% (i-1) + 1;
        }
        cnt[fa[i]]++;
        if(l[fa[i]] == 0) l[fa[i]] = i;
        else r[fa[i]] = i;
    }
    // 按照格式要求去输出树的信息
    for(int i = 1; i <= n; i++) {
        cout << l[i] << " " << r[i] << '\n';
    }
    return 0;
}
```

- 生成一个不以1为根的树

```
int fa[N], a[N];
int main(){
    freopen("data.in", "w", stdout);
    srand(time(0));  // 初始化随机种子
    int n = rand() % 8 + 2;
    for(int i = 2; i <= n; i++) {
        fa[i] = rand()% (i-1) + 1;  // fa[i]: 1~i-1
    }
    // 按照格式要求去输出树的信息
    // 希望 1 不一定是根
    for(int i = 1; i <= n; i++) {
        a[i] = i;
    }
    random_shuffle(a+1, a+n+1);
    cout << n <<endl;

    for(int i = 1; i <= n; i++) {
        cout << a[i] << " " << a[fa[i]] << '\n';
    }
```

```
    return 0;
}
```

- 生成一个随机简单图，可能不连通

```
int fa[N], a[N];
set<pair<int, int> > st;
int main(){
    freopen("data.in", "w", stdout);
    srand(time(0));  // 初始化随机种子
    int n = rand() % 8 + 2;
    int m = rand() % (n*(n-1)/2) + 1;
    int cnt = 0;
    while(cnt < m) {
        int x = rand()%n + 1;  // 1~n
        int y = rand()%n + 1;  // 1~n
        if(x == y) continue;
        if(st.count(make_pair(x,y)) == 0) {
            cout << x << " " <<y << '\n';
            st.insert(make_pair(x, y));
        }
    }
    return 0;
}
```

- 运行以下代码即可进行对拍，弹出提示错误后去 `data.in` 查看错误数据

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int t = 30;   // 比较的数据组数，可以写得很大，让它慢慢跑，跑的时候，写后面的题目
    while(t--) {
        // 1. 利用数据生成程序，生成 .in 文件
        system("generator.exe");
        // 2. 运行暴力代码程序，得到 data.out
        system("baoli.exe");
        // 3. 运行正解代码程序，得到 data2.out
```

```
        system("zhengjie.exe");
        // 4. 对比输出文件，如果不一致，输出提示信息，否则继续对拍
        if(system("fc data.out data2.out")) {  // windows 下用
fc，linux 下 fc 要替换为 diff
            cout << "Not correct!";
            return 0;
        }
    }
    cout << "AC!";
    system("pause");
    return 0;
}
```

浮点数精度

浮点数精度问题其实很简单就是尽量变除法为乘法。把小数转换成整数。

卷王

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e4+5;
long long a[4][N];
int n;
int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    cin>>n;
    for(int i=1;i<=3;i++){
        for(int j=1;j<=n;j++){
            cin>>a[i][j];
        }
    }
    long long s1=0,s2=0;
    for(int i=2;i<=3;i++){
        for(int j=1;j<=n;j++){
            if(i==2){
                s1=s1+a[i][j]*a[1][j];
```

```
            }
            else{
                s2=s2+a[i][j]*a[1][j];
            }
        }
    }
    if(s1>s2){
        cout<<"ke";
    }else if(s1<s2){
        cout<<"do";
    }else if(s1==s2){
        cout<<"same";
    }
}
```

浮点数比较

因为C/C++内置的double类型也是由相应的二进制存储的，所以double在计算的时候是会可能丢失精度的，最后进行浮点数比较的时候要注意做一个 fabs

```
//比较两个浮点数是否相等
bool compareDouble(double a,double b)
{
    double eps = 1e-6 //注意精度，默认是1e-6，有时候精度要求不高可以降低精度要求
    if(fabs(a-b)<eps) return true;
    return false;
}
```

01 串的熵（蓝桥杯C/C++2023B组省赛）