> 作者： 李春阳
>
> 仅供学习交流

目录

实验一

问题 A: 排列问题

- 题目描述
- 输入
- 输出
- 样例输入
- 样例输出
- 思路
- 代码

问题 B: 快速幂

- 题目描述
- 输入
- 输出
- 样例输入
- 样例输出
- 思路
- 代码

问题 C: 求第k小

- 题目描述
- 输入
- 输出
- 样例输入
- 样例输出
- 思路
- 代码
  - 方法一 使用qsort
  - 方法二 手写sort
  - 方法三 发现数组sort能过

问题 D: 内部收益率

- 题目描述
- 输入
- 输出
- 样例输入
- 样例输出
- 思路
- 代码

问题 E: 跳台阶

- 题目描述
- 输入
- 输出
- 样例输入
- 样例输出
- 思路
- 代码
  - 方法一 直接dp存储
  - 方法二 改进只需要维护3个dp就行了

实验二

问题 A: 沙子的质量

- 题目描述
- 输入
- 输出
- 样例输入
- 样例输出
- 思路

代码
问题 B: 最长公共子序列
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 C: 三角形的路径权
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 D: 跳跃游戏二
题目描述
输入
输出
样例输入
样例输出
思路
代码
方法一 贪心算法
问题 E: 字母排序
题目描述
输入
输出
样例输入
样例输出
思路
代码
实验三
问题 A: Homework
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 B: 区间包含问题
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 C: 最长子序列
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 D: 三值排序

题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 E: 法师康的工人
题目描述
输入
输出
样例输入
样例输出
思路
代码
作业一
问题 A: 进制转换
题目描述
输入
输出
样例输入
样例输出
思路
代码
方法一 c语言%o强转
方法二 %/存储
问题 B: 排列问题
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 C: 快速幂
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 D: 求第k小
题目描述
输入
输出
样例输入
样例输出
代码
问题 E: 沙子的质量
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 F: 最长公共子序列
题目描述
输入

输出
样例输入
样例输出
思路
代码
问题 G: sort
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 H: Joseph
题目描述
输入
输出
样例输入
样例输出
思路
问题 I: Factstone Benchmark
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 J: Ants
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 K: Matches Game
题目描述
输入
输出
样例输入
样例输出
思路
代码
c++过不了
c就能过
问题 L: sort2
题目描述
输入
输出
样例输入
样例输出
思路
代码
直接sort超时
map超时
方法三
作业二
问题 A: 单词排序
题目描述

输入
输出
样例输入
样例输出
思路
代码
问题 B: 求数组的最长递减子序列
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 C: 矩形滑雪场
题目描述
输入
输出
样例输入
样例输出
思路
代码
动态规划
问题 D: Homework
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 E: 区间包含问题
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 F: 最长子序列
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 G: 元素整除问题
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 H: 渊子赛马
题目描述
输入
输出
样例输入

样例输出
思路
代码
问题 I: The Hardest Problem Ever
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 J: Rock-Paper-Scissors Tournament
题目描述
输入
输出
样例输入
样例输出
思路
代码
问题 K: Balloon Robot
题目描述
输入
输出
样例输入
样例输出
思路

实验一

问题 A: 排列问题

题目描述

输入一个可能含有重复字符的字符串，打印出该字符串中所有字符的全排列。

输入

单组测试数据，输入数据是一个长度不超过10个字符的字符串，以逗号结尾。

输出

打印出该字符串中所有字符的全排列。以字典序顺序输出，用空格分隔。

样例输入

```
abc,
```

样例输出

```
abc acb bac bca cab cba
```

思路

回溯法，这里采用回溯进行遍历枚举。

代码

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

string path;
vector<string> result;
vector<int> used;

void backtrack(string str){
    if(path.size() == str.size()){
        result.push_back(path);
        return;
    }
    for(int i=0;i<str.size();i++){
        if(used[i]){
            continue;
        }
        used[i] = 1;
        path.push_back(str[i]);
        backtrack(str);
        path.pop_back();
        used[i] = 0;
    }
}

int main(){
    string str;
    cin >> str;
    str.pop_back();
    sort(str.begin(),str.end());
    used.resize(str.size(),0);
    backtrack(str);
    for(auto s : result){
        cout << s << " ";
    }
    cout << endl;
    return 0;
}

```

问题 B: 快速幂

时间限制: 1 Sec

内存限制: 128 MB

题目描述

输入

多组测试样例，最多50组。每组测试样例给定一个整数x(1<=x<=25000)

输出

对每个样例，输出一行，代表f(x)对100000007取余的结果。

样例输入

```
3
4
5
```

样例输出

```
33
289
3414
```

思路

这里是快速幂的技巧，采用化二进制判断迭代。

这里代码思路没问题，但得用long long类型

代码

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

const long long mod = 100000007;

long long mypow(long long n,long long m){
    long long ans = 1;
    while(m){
        if(m & 1){
            ans = ans * n % mod;
        }
        m = m >> 1;
        n = (n * n) % mod;
    }
```

```
    return ans;
}


int main(){
    int n;
    while(cin >> n){
        long long ans = 0;
        for(int i=1;i<=n;i++){
            ans = (ans + mypow(i,i)) % mod;
        }
        cout << ans+1 << endl;
    }
    return 0;
}
```

问题 C: 求第k小

时间限制: 1 Sec

内存限制: 128 MB

题目描述

给定n(1<=n<=1000000)个元素，求第k小数(1<=k<=n)。

输入

一组样例。第一行输入两个整数n和k。第二行输入n个不同的int范围内的数。

输出

输出一行，输出第k小数。

样例输入

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

样例输出

```
2
```

思路

https://blog.51cto.com/svenman/1851716

qsort排序 https://blog.csdn.net/weixin_41096569/article/details/104771864

不知道为什么c++算法的sort会超时，用c的就可以 过

这里有一点qsort使用配合数组，

```
void qsort (void* base, size_t num, size_t size,
            int (*compar)(const void*,const void*));
```

代码

方法一 使用qsort

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

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}
int nums[1000011];
int main(){
    int n,k;
    cin >> n >> k;
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    //sort(nums.begin(),nums.end());
    qsort(nums,n,sizeof(int),compare);
    cout << nums[k-1] << endl;
    return 0;
}
```

方法二 手写sort

这里输入cin会超时（离谱）就这一点点差距。

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

int a[1000001];

int partition(int a[],int p,int r)
{
    int x=a[r];
    int middle=p;
    int j;
    for(j=p;j<r;j++)
    {
        if(a[j]<x)
        {
            if(j!=middle)
              swap(a[middle],a[j]);
            middle++;
        }
    }
    swap(a[middle],a[j]);
    return middle;
```

```
}

void select(int a[],int p,int r)
{
    if(p<r)
    {
        int q=partition(a,p,r);
        select(a,p,q-1);
        select(a,q+1,r);
    }
}

int main()
{
    int n,k;
    cin>>n>>k;
    for(int i=0;i<n;i++)
        scanf("%d",&a[i]);
    select(a,0,n-1);
    cout<<a[k-1]<<endl;
    return 0;
}
```

方法三 发现数组sort能过

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

int a[1000000];

int main(){
    int n,k;
    cin>>n>>k;
    for(int i=0;i<n;i++){
        cin >> a[i];
    }
    sort(a,a+n);
    cout << a[k-1] << endl;
    return 0;
}
```

问题 D: 内部收益率

时间限制: 1 Sec

内存限制: 128 MB

题目描述

输入

输出

对于每组数据，输出仅一行，即项目的IRR，四舍五入保留小数点后两位。

样例输入

```
1
-1 2
2
-8 6 9
0
```

样例输出

```
1.00
0.50
```

思路

二分搜索，模拟逼近

代码

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

int main(){
    int n;
    while((cin >> n) && n){
        vector<int> cf(n+1,0);
        for(int i=0;i<= n;i++){
            cin >> cf[i];
        }
        double min, max, ans, mid;
        min = -1.0;
        max = 1000000;
        while(max - min > 1e-6){
            ans = cf[0];
            mid = (max-min) / 2 + min;
            for(int i=1;i<=n;i++){
                ans += cf[i] / pow(1 + mid, i);
            }
            if(ans > 0)
                min = mid;
            else
                max = mid;
        }
        printf("%.2lf\n",mid);
    }
    return 0;
}

```

问题 E: 跳台阶

> 时间限制: 1 Sec
>
> 内存限制: 128 MB

题目描述

一只青蛙一次可以跳上1级台阶，也可以跳上2级。求该青蛙跳上一个n级的台阶总共有多少种跳法。

输入

多组测试样例。每组测试样例包含一个整数n。(1<=n<=100)

输出

每组测试样例输出一行，表示青蛙跳上n级台阶的跳法数量.

所得到的结果模1000000007

样例输入

```
3
4
```

样例输出

```
3
5
```

思路

可以看成初等的动态规划

dp数组存储

代码

方法一 直接dp存储

```cpp
#include <bits/stdc++.h>
using namespace std;
const long long mod = 1000000007;
int main(){
    int n;
    while(cin >> n){
        if(n <= 1){
            cout << n << endl;
            continue;
        }
        vector<long long> dp(n+1,0);
        dp[0] = 1;
        dp[1] = 1;
        for(int i=2;i<=n;i++){
            dp[i] = (dp[i-1] + dp[i-2]) % mod;
        }
        cout << dp[n] << endl;
    }
    return 0;
```

```
}
```

方法二 改进只需要维护3个dp就行了

```
#include <bits/stdc++.h>
using namespace std;
const long long mod = 1000000007;

int main(){
    int n;
    while(cin >> n){
        if(n <= 1){
            cout << n << endl;
            continue;
        }
        int dp[3];
        dp[0] = 1;
        dp[1] = 1;
        for(int i=2;i<=n;i++){
            dp[2] = (dp[0] + dp[1]) % mod;
            dp[0] = dp[1];
            dp[1] = dp[2];
        }
        cout << dp[2] << endl;
    }
    return 0;
}
```

实验二

问题 A: 沙子的质量

时间限制: 1 Sec

内存限制: 128 MB

题目描述

设有N堆沙子排成一排，其编号为1，2，3，…，N（N< =300）。每堆沙子有一定的数量，可以用一个整数来描述，现在要将N堆沙子合并成为一堆，每次只能合并相邻的两堆，合并的代价为这两堆沙子的数量之和，合并后与这两堆沙子相邻的沙子将和新堆相邻，合并时由于选择的顺序不同，合并的总代价也不相同，如有4堆沙子分别为1 3 5 2我们可以先合并1、2堆，代价为4，得到4 5 2又合并1，2堆，代价为9，得到9 2，再合并得到11，总代价为4+9+11=24，如果第二步是先合并2，3堆，则代价为7，得到4 7，最后一次合并代价为11，总代价为4+7+11=22；问题是：找出一种合理的方法，使总的代价最小。输出最小代价。

输入

第一行一个数N表示沙子的堆数N。 第二行N个数，表示每堆沙子的质量。 a[i]< =1000。

输出

合并的最小代价。

样例输入

```
4
1 3 5 2
```

样例输出

```
22
```

思路

这里和矩阵连乘差不多

这里cost（）也就是合并有`sum[j]-sum[i]`给出

这里外层遍历是长度，合并的长度

然后计算出i，j也就是合并区间

k是中间循环查找min

这里记得一个是sum初始化，一个是dp `i==j`时`dp=0`初始化`dp[i][j] = INT_MAX`

参考

代码

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

int main(){
    int n;
    cin >> n;
    vector<int> nums(n,0);
    vector<int> sum(n+1,0);
    vector<vector<int>> dp(n,vector<int>(n,0));
    //先输入数组
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    //初始化sum，方便求i,j之间的代价
    for(int i=0;i<n;i++){
        sum[i+1] = sum[i] + nums[i];
    }
    //这是是遍历长度，2开始，1为0；
    for(int len = 2;len<=n;len++){
        //这里初始化i，j,这里要右端点小于n，防止越界，斜方向遍历
        for(int i=0;i+len-1 < n;i++){
            int j = i+len-1;
            dp[i][j] = INT_MAX;
            for(int k=i;k<j;k++){
                dp[i][j] = min(dp[i][j],dp[i][k]+dp[k+1][j]+sum[j+1]-sum[i]);
            }
        }
    }
    cout << dp[0][n-1] <<endl;
    return 0;
}

```

问题 B: 最长公共子序列

时间限制: 1 Sec 内存限制: 128 MB

题目描述

一个字符串A的子串被定义成从A中顺次选出若干个字符构成的串。如A=“cdaad”,顺次选1，3，5个字符就构成子串”cad”,现给定两个字符串，求它们的最长共公子串。

输入

第一行两个字符串用空格分开。两个串的长度均小于2000 。

输出

最长子串的长度。

样例输入

```
abccd aecd
```

样例输出

```
3
```

思路

代码

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

int main(){
    string str1,str2;
    cin >> str1 >> str2;
    int m = str1.size();
    int n = str2.size();
    vector<vector<int>> dp(m+1,vector<int>(n+1,0));
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(str1[i-1] == str2[j-1]){
                dp[i][j] = dp[i-1][j-1] + 1;
            }
            else{
                dp[i][j] = max(dp[i-1][j],dp[i][j-1]);
            }
        }
    }
    cout << dp[m][n] << endl;
    return 0;
}
```

问题 C: 三角形的路径权

时间限制: 1 Sec 内存限制: 128 MB

题目描述

如输入样例所示出了一个数字三角形。请编一个程序计算从顶至底的某处的一条路径，使该路径所经过的数字的总和最大。每一步可沿左斜线向下或右斜线向下走；1< 三角形行数< 25；三角形中的数字为整数< 1000；

输入

输入第一行为N，表示有N行 后面N行表示三角形每条路的路径权。

输出

输出路径所经过的数字的总和最大的答案。

样例输入

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

样例输出

```
30
```

思路

逆向，自底向上

i == N 时 `dp[i][j] = mp[i][j]`

其他： `dp[i][j]=max(dp[i+1][j],dp[i+1][j+1])+mp[i][j]`

代码

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

int main(){
    int n;
    cin >> n;
    vector<vector<int>> mp(n,vector<int>(n,-1));
    for(int i=0;i<n;i++){
        for(int j=0;j<=i;j++){
            cin >> mp[i][j];
        }
    }
    vector<vector<int>> dp(n+1,vector<int>(n+1,0));
    for(int i=n-1;i>=0;i--){
        for(int j=0;j<=i;j++){
            if(i == n-1){
```

```
                dp[i][j] = mp[i][j];
            }
            else{
                dp[i][j]=max(dp[i+1][j],dp[i+1][j+1])+mp[i][j];
            }
        }
    }
    cout << dp[0][0] << endl;
    return 0;
}
```

问题 D: 跳跃游戏二

时间限制: 1 Sec

内存限制: 128 MB

题目描述

给定一个非负整数数组，假定你的初始位置为数组第一个下标。数组中的每个元素代表你在那个位置能够跳跃的最大长度。你的目标是到达最后一个下标，并且使用最少的跳跃次数。例如：A = [2,3,1,1,4]，到达最后一个下标的最少跳跃次数为 2。（先跳跃11步，从下标0到1，然后跳跃3步，到达最后一个下标。一共两次）

输入

第一行输入一个正整数n(1≤n≤100)，接下来的一行，输入n个整数，表示数组A。

输出

最后输出最少的跳跃次数。

样例输入

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

样例输出

```
2
```

思路

这里可以拿dp动态规划

但我选择更简单的贪心算法，求最大覆盖范围

看几次就可以覆盖终点

代码

方法一 贪心算法

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

int main(){
    int n;
    cin >> n;
    vector<int> nums(n,0);
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    int curDistance = 0;    // 当前覆盖的最远距离下标
    int ans = 0;            // 记录走的最大步数
    int nextDistance = 0;   // 下一步覆盖的最远距离下标
    for(int i=0;i<n-1;i++){
        nextDistance = max(nums[i]+i,nextDistance);
        if(i == curDistance){
            curDistance = nextDistance;
            ans++;
        }
    }
    cout << ans << endl;
    return 0;
}
```

问题 E: 字母排序

时间限制: 1 Sec

内存限制: 128 MB

题目描述

XXXX年突然有外星人造访，但大家语言不通，不过科学家们经过研究发现外星人用26个英文字母组成的单词中最长不降子序列的长度来表述数字，且英文字母的排列顺序不同，现给出其排列顺序，再给出外星人说的每个数字（其实是每个英文单词，用空格隔开），翻译出外星人所说的数字（连续输出，最后加回车）。(因为是最长不降子序列，所以数字中没有0，也就是说外星人的数字是大于0的数字)。例如，我们正常的字母排列顺序是abcdefg.....xyz，代表a< b< c< .....< x< y< z abcd efg hhh ihg四个字符串的最长不降子序列的长度分别为4 3 3 1。

输入

第1，2行为字符串 含义如题描述。1≤第二行长度≤255。

输出

输出答案，含义如题描述

样例输入

```
abcdefghijklmnopqrstuvwxyz
abcd efg hhh ihg
```

样例输出

```
4331
```

思路

代码

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

int main(){
    string str;
    cin >> str;
    //这里map映射，因为是26的字母不分大小写，如果混合使用，统一转成小写就好了，map用红黑树，unordered_map应用哈希
    unordered_map<char,int> map;
    for(int i=0;i<str.size();i++){
        map[str[i]] = i;
    }
    while(cin >> str){
        //里面就是最简单最大子序列
        //这里初始化为1
        vector<int> dp(str.size(),1);
        int ans = 0;
        for(int i=0;i<str.size();i++){
            for(int j=0;j<i;j++){
                if(map[str[i]] >= map[str[j]]){
                    dp[i] = max(dp[i],dp[j]+1);
                }
                if(ans < dp[i]){
                    ans = dp[i];
                }
            }
        }
        cout << ans;
    }
    cout << endl;
    return 0;
}
```

实验三

问题 A: Homework

时间限制: 1 Sec

内存限制: 128 MB

题目描述

临近开学了，大家都忙着收拾行李准 备返校，但 I_Love_C 却不为此担心! 因为他的心思全在暑假作业上：目前为止还未开动。

暑假作业是很多张试卷，我们这些从试卷里爬出来的人都知道，卷子上的题目有选择题、填空题、简答题、证明题等。而做选择题的好处就在于工作量很少，但又因为选择题题目都普遍很长。如果有 5 张试卷，其中 4 张是选择题，最后一张是填空题，很明显做最后一张所花的时间要比前 4 张长很多。但如果你只做了选择题，虽然工作量很少，但表面上看起来也已经做了4/5的作业了。

I_Love_C决定就用这样的方法来蒙混过关，他统计出了做完每一张试卷所需的时间以及它做完后能得到的价值（按上面的原理，选择题越多价值当然就越高咯）。

现在就请你帮他安排一下，用他仅剩的一点时间来做最有价值的作业。

输入

测试数据包括多组。每组测试数据以两个整数 M,N(1<M<20,1<N<10000) 开头，分别表示试卷的数目和 I_Love_C 剩下的时间。接下来有 M 行，每行包括两个整数 T,V(1<T<N,1<V<10000)分别表示做完这张试卷所需的时间以及做完后能得到的价值，输入以 0 0 结束。

输出

对应每组测试数据输出 I_Love_C 能获得的最大价值。保留小数点 2 位

提示：float 的精度可能不够，你应该使用 double 类型。

样例输入

```
4 20
4 10
5 22
10 3
1 2
0 0
```

样例输出

```
37.00
```

思路

贪心算法，寻找性价比最高的

代码

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

bool cmp(vector<double>a,vector<double> b){
    return a[2] > b[2];
}

int main(){
    int m,n;
    while(cin >> m >> n){
        if(m == 0 && n ==0){
            break;
        }
        vector<vector<double>> homework(m,vector<double>(3,0));
        for(int i=0;i<m;i++){
            cin >> homework[i][0] >> homework[i][1];
            homework[i][2] = homework[i][1] / homework[i][0];
        }
        sort(homework.begin(),homework.end(),cmp);
        double ans = 0;
```

```
        for(int i=0;i<n;i++){
            if(n > homework[i][0]){
                ans += homework[i][1];
                n -= homework[i][0];
            }
            else{
                ans += homework[i][2] * n;
                break;
            }
        }
        printf("%.2lf\n",ans);

    }
    return 0;
}

```

问题 B: 区间包含问题

时间限制: 1 Sec

内存限制: 128 MB

题目描述

已知 n 个左闭右开区间 [a,b)，对其进行 m 次询问，求区间[l,r]最多可以包含 n 个区间中的多少个区间，并且被包含的所有区间都不相交。

输入

输入包含多组测试数据，对于每组测试数据：

第一行包含两个整数 n ,m (1≤n≤100000,1≤m≤100) 。

接下来 n 行每行包含两个整数 a ,b (0≤a<b≤10^9) 。

接下来 m 行每行包含两个整数 l ,r (0≤l<r≤10^9) 。

输出

对于每组测试数据，输出 m 行，每行包含一个整数。

数据过大请使用快速输入输出。

样例输入

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

样例输出

```
1
1
2
```

思路

就是右端点小排序

优先选取满足小区间

代码

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

bool cmp(vector<int> a, vector<int> b){
    return a[1] < b[1];
}

int main(){
    int n, m;
    while(cin >> n >> m){
        vector<vector<int>> point(n,vector<int>(2,0));
        for(int i=0;i<n;i++){
            cin >> point[i][0] >> point[i][1];
        }
        sort(point.begin(),point.end(),cmp);
        while(m--){
            int left,right;
            cin >> left >> right;
            int ans = 0;
            for(int i=0;i<n;i++){
                if(point[i][1] > right){
                    break;
                }
                if(left <= point[i][0]){
                    left = point[i][1];
                    ans++;
                }
            }
            cout << ans << endl;
        }
    }
    return 0;
}

```

问题 C: 最长子序列

时间限制: 1 Sec 内存限制: 128 MB

题目描述

在一个数组中找出和最大的连续几个数。（至少包含一个数）

例如：

数组A[] = [-2,1,-3,4,-1,2,1,-5,4]，则连续的子序列[4,-1,2,1]有最大的和6.

输入

第一行输入一个不超过1000的整数n。

第二行输入n个整数A[i]。

输出

输出一个整数，表示最大的和。

样例输入

```
3
1 1 -2
```

样例输出

```
2
```

思路

代码

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

int main(){
    int n;
    cin >> n;
    vector<int> nums(n,0);
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    int sum = 0;
    int ans = INT_MIN;
    for(int i=0;i<n;i++){
        sum += nums[i];
        if(sum > ans){
            ans = sum;
        }
        if(sum < 0){
            sum = 0;
        }
    }
    cout << ans << endl;
    return 0;
}
```

问题 D: 三值排序

时间限制: 1 Sec

内存限制: 128 MB

题目描述

排序是一种很频繁的计算任务。一个实际的例子是，当我们给某项竞赛的优胜者按金银铜牌排序的时候。在这个任务中可能的值只有三种1，2和3。我们用交换的方法把他排成升序的。

写一个程序计算出，计算出的一个包括1、2、3三种值的数字序列，排成升序所需的最少交换次数。

输入

输入第1行为类别的数量N（1≤N≤1000）

输入第2行到第N+1行，每行包括一个数字（1或2或3）。

输出

输出包含一行，为排成升序所需的最少交换次数。

样例输入

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

样例输出

```
4
```

思路

交换次序的，

这个不太懂，背吧

代码

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

int main(){
    int n;
    cin >> n;
    int sum[4] = {0,0,0,0};
    vector<int> nums(n,0);
    for(int i=0;i<n;i++){
```

```
        cin >> nums[i];
        sum[nums[i]]++;
    }
    int x = 0,y = 0,z=0;
    for(int i=0;i<sum[1];i++){
        if(nums[i] != 1){
            x++;
        }
    }
    for(int i=sum[1];i<sum[1]+sum[2];i++){
        if(nums[i] == 3){
            y++;
        }
    }
    for(int i=sum[1]+sum[2];i<n;i++){
        if(nums[i] == 2){
            z++;
        }
    }
    cout << x + max(y,z) << endl;
    return 0;
}
```

问题 E: 法师康的工人

时间限制: 1 Sec

内存限制: 128 MB

题目描述

三个法师康的工人每天早上6点到工厂开始到三条产品生产线上组装桔子手机。第一个工人在200时刻开始（从6点开始计时，以秒作为单位）在生产线上开始生产，一直到1000时刻。第二个工人，在700时刻开始，在1100时刻结束。第三个工人从1500时刻工作到2100时刻。期间最长至少有一个工人在生产线上工作的连续时间为900秒（从200时刻到1100时刻），而最长的无人生产的连续时间（从生产开始到生产结束）为400时刻（1100时刻到1500时刻）。

你的任务是用一个程序衡量N个工人在N条产品线上的工作时间列表（1≤N≤5000，以秒为单位）。

·最长的至少有一个工人在工作的时间段

·最长的无人工作的时间段（从有人工作开始计）

输入

输入第1行为一个整数N，第2-N+1行每行包括两个均小于1000000的非负整数数据，表示其中一个工人的生产开始时间与结束时间。

输出

输出为一行，用空格分隔开两个我们所求的数。

样例输入

```
3
200 1000
700 1100
1500 2100
```

样例输出

```
900 400
```

思路

代码

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

bool cmp(vector<int> a, vector<int> b){
    if(a[0] == b[0])
        return a[1] < b[1];
    return a[0] < b[0];
}

int main(){
    int n;
    cin >> n;
    vector<vector<int>> point(n,vector<int>(2,0));
    for(int i=0;i<n;i++){
        cin >> point[i][0] >> point[i][1];
    }
    sort(point.begin(),point.end(),cmp);
    int maxx = 0;
    int minn = 0;
    int start,end;
    start = point[0][0];
    end = point[0][1];
    for(int i=1;i<n;i++){
        if(point[i][0] <= end){
            end =max(point[i][1],end);
            maxx = max(maxx,end - start);
        }
        else{
            start = point[i][0];
            minn = max(minn,start - end);
            end = point[i][1];
        }
    }
    cout << maxx << " " << minn << endl;
    return 0;
}

```

作业一

问题 A: 进制转换

时间限制: 1 Sec

内存限制: 128 MB

题目描述

输入一个十进制正整数，然后输出它所对应的八进制数。

输入

输入一个十进制正整数n(1≤n≤106) 。

输出

输出n对应的八进制数，输出在一行。

样例输入

```
10
```

样例输出

```
12
```

思路

代码

方法一 c语言%o强转

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

int main(){
    int n;
    cin >> n;
    printf("%o\n",n);
    return 0;
}

```

方法二 %/存储

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

int main(){
    int n;
```

```
    cin >> n;
    vector<int> num;
    while(n){
        num.push_back(n%8);
        n /= 8;
    }
    for(int i=num.size()-1;i>=0;i--){
        cout << num[i];
    }
    cout << endl;
    return 0;
}
```

问题 B: 排列问题

时间限制: 1 Sec

内存限制: 128 MB

题目描述

输入一个可能含有重复字符的字符串，打印出该字符串中所有字符的全排列。

输入

单组测试数据，输入数据是一个长度不超过10个字符的字符串，以逗号结尾。

输出

打印出该字符串中所有字符的全排列。以字典序顺序输出，用空格分隔。

样例输入

```
abc,
```

样例输出

```
abc acb bac bca cab cba
```

思路

代码

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

```
3
4  string path;
5  vector<string> result;
6  vector<int> used;
7
8  void backtrack(string str){
9      if(path.size() == str.size()){
10         result.push_back(path);
11         return;
12     }
13     for(int i=0;i<str.size();i++){
14         if(used[i]){
15             continue;
16         }
17         used[i] = 1;
18         path.push_back(str[i]);
19         backtrack(str);
20         path.pop_back();
21         used[i] = 0;
22     }
23 }
24
25 int main(){
26     string str;
27     cin >> str;
28     str.pop_back();
29     sort(str.begin(),str.end());
30     used.resize(str.size(),0);
31     backtrack(str);
32     for(auto s : result){
33         cout << s << " ";
34     }
35     cout << endl;
36     return 0;
37 }
```

问题 C: 快速幂

时间限制: 1 Sec

内存限制: 128 MB

题目描述

给定x,求$f(x)$对 100000007 取余的结果。$f(x) = \sum_{i=1}^{x} pow(i,i) + 1$

输入

多组测试样例，最多50组。每组测试样例给定一个整数x(1<=x<=25000)

输出

对每个样例，输出一行，代表f(x)对100000007取余的结果。

样例输入

```
3
4
5
```

样例输出

```
33
289
3414
```

思路

代码

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

const long long mod = 100000007;

long long mypow(long long x,long long m){
    long long ans = 1;
    while(m){
        if(m & 1){
            ans = (ans * x) % mod;
        }
        m >>= 1;
        x = (x * x) %mod;
    }
    return ans;
}

int main(){
    int n;
    while(cin >> n){
        long long ans = 1;
        for(int i=1;i<=n;i++){
            ans =(ans + mypow(i,i)) % mod;
        }
        cout << ans <<endl;
    }
    return 0;
}
```

问题 D: 求第k小

时间限制: 1 Sec

内存限制: 128 MB

题目描述

给定n(1<=n<=1000000)个元素，求第k小数(1<=k<=n)。

输入

一组样例。第一行输入两个整数n和k。第二行输入n个不同的int范围内的数。

输出

输出一行，输出第k小数。

样例输入

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

样例输出

```
2
```

代码

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

int nums[1000002];

int main(){
    int n,k;
    cin >> n >> k;
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    sort(nums,nums+n);
    cout << nums[k-1] << endl;
    return 0;
}
```

问题 E: 沙子的质量

时间限制: 1 Sec

内存限制: 128 MB

题目描述

设有N堆沙子排成一排，其编号为1，2，3，…，N（N< =300）。每堆沙子有一定的数量，可以用一个整数来描述，现在要将N堆沙子合并成为一堆，每次只能合并相邻的两堆，合并的代价为这两堆沙子的数量之和，合并后与这两堆沙子相邻的沙子将和新堆相邻，合并时由于选择的顺序不同，合并的总代价也不相同，如有4堆沙子分别为1 3 5 2我们可以先合并1、2堆，代价为4，得到4 5 2又合并1，2堆，代价为9，得到9 2，再合并得到11，总代价为4+9+11=24，如果第二步是先合并2，3堆，则代价为7，得到4 7，最后一次合并代价为11，总代价为4+7+11=22；问题是：找出一种合理的方法，使总的代价最小。输出最小代价。

输入

第一行一个数N表示沙子的堆数N。 第二行N个数，表示每堆沙子的质量。 a[i]< =1000。

输出

合并的最小代价。

样例输入

```
4
1 3 5 2
```

样例输出

```
22
```

思路

代码

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

int main(){
    int n;
    cin >> n;
    vector<int> nums(n,0);
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    vector<int> sum(n+1,0);
    for(int i=0;i<n;i++){
        sum[i+1] = sum[i] + nums[i];
    }
    vector<vector<int>> dp(n,vector<int>(n,INT_MAX));
    for(int i=0;i<n;i++){
        dp[i][i] = 0;
    }
```

```
    for(int len=2;len<=n;len++){
        for(int i=0;i + len -1 < n;i++){
            int j = i + len -1;
            for(int k=i;k<j;k++){
                dp[i][j] = min(dp[i][j],dp[i][k]+dp[k+1][j] + sum[j+1]-sum[i]);
            }
        }
    }
    cout << dp[0][n-1] << endl;
    return 0;
}
```

问题 F: 最长公共子序列

时间限制: 1 Sec

内存限制: 128 MB

题目描述

一个字符串A的子串被定义成从A中顺次选出若干个字符构成的串。如A=“cdaad”,顺次选1，3，5个字符就构成子串”cad”,现给定两个字符串，求它们的最长共公子串。

输入

第一行两个字符串用空格分开。两个串的长度均小于2000 。

输出

最长子串的长度。

样例输入

```
abccd aecd
```

样例输出

```
3
```

思路

代码

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

int main(){
    string str1,str2;
    cin >> str1 >> str2;
    int m = str1.size();
    int n = str2.size();
```

```
    vector<vector<int>> dp(m+1,vector<int>(n+1,0));
    for(int i=1;i<=m;i++){
        for(int j=1;j<=n;j++){
            if(str1[i-1] == str2[j-1]){
                dp[i][j] = dp[i-1][j-1] + 1;
            }
            else{
                dp[i][j] = max(dp[i][j-1],dp[i-1][j]);
            }
        }
    }
    cout << dp[m][n] << endl;
    return 0;
}

```

问题 G: sort

时间限制: 1 Sec

内存限制: 64 MB

题目描述

给你n个整数，请按从大到小的顺序输出其中前m大的数。

输入

每组测试数据有两行，第一行有两个数n,m(0<n,m<1000000)，第二行包含n个各不相同，且都处于区间[-500000,500000]的整数。

输出

对每组测试数据按从大到小的顺序输出前m大的数。

样例输入

```
5 3
3 -35 92 213 -644
```

样例输出

```
213 92 3
```

思路

代码

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

```

```cpp
int nums[1000001];

bool cmp(int a,int b){
    return a > b;
}

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

问题 H: Joseph

时间限制: 1 Sec

内存限制: 32 MB

题目描述

The Joseph's problem is notoriously known. For those who are not familiar with the original problem: from among n people, numbered 1, 2, . . ., n, standing in circle every mth is going to be executed and only the life of the last remaining person will be saved. Joseph was smart enough to choose the position of the last remaining person, thus saving his life to give us the message about the incident. For example when n = 6 and m = 5 then the people will be executed in the order 5, 4, 6, 2, 3 and 1 will be saved.

Suppose that there are k good guys and k bad guys. In the circle the first k are good guys and the last k bad guys. You have to determine such minimal m that all the bad guys will be executed before the first good guy.

约瑟夫问题是臭名昭著的。对于那些不熟悉原问题的人来说：从n个人中，编号为1，2，...，n，每隔m月站成一圈就要被处死，只有最后剩下的人的生命才能得到挽救。约瑟夫很聪明地选择了最后剩下的人的位置，从而保住了他的性命，给我们带来了关于这个事件的信息。例如，当n=6，m=5时，那么人们将按5、4、6、2、3的顺序被处决，1人将获救。

假设有k个好人和k个坏人。在这个圈子里，前k个是好人，后k个是坏人。你必须确定这样一个最小的m，使所有的坏人都在第一个好人之前被处决。

输入

The input file consists of separate lines containing k. The last line in the input file contains 0. You can suppose that 0 < k < 14.

输入文件由包含 k 的单独行组成。输入文件的最后一行包含 0。您可以假设 0 < k < 14。

输出

The output file will consist of separate lines containing m corresponding to k in the input file

输出文件将由包含与输入文件中的 k 对应的 m 的单独行组成

样例输入

```
3
4
0
```

样例输出

```
5
30
```

思路

```

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

bool check(int m,int k){
    int res = 0;
    for(int i=1;i<=k;i++){
        res = (res + m -1) % (2*k-i+1);
        if(res < k){
            return false;
        }
    }
    return true;
}

int main(){
    int k;
    while((cin >> k) && k){
        for(int i=k+1;;i++){
            if(check(i,k) == true){
                cout << i << endl;
                break;
            }
        }
    }
    return 0;
}

```

问题 I: Factstone Benchmark

时间限制: 1 Sec

内存限制: 128 MB

题目描述

Amtel has announced that it will release a 128-bit computer chip by 2010, a 256-bit computer by 2020, and so on, continuing its strategy of doubling the word-size every ten years. (Amtel released a 64-bit computer in 2000, a 32-bit computer in 1990, a 16-bit computer in 1980, an 8-bit computer in 1970, and a 4-bit computer, its first, in 1960.)

Amtel will use a new benchmark - the *Factstone* - to advertise the vastly improved capacity of its new chips. The *Factstone* rating is defined to be the largest integer *n* such that *n!* can be represented as an unsigned integer in a computer word.

Given a year $1960 \le y \le 2160$, what will be the *Factstone* rating of Amtel's most recently released chip?

Amtel公司已经宣布，它将在2010年之前发布128位计算机芯片，在2020年之前发布256位计算机，以此类推，继续其每十年将字数增加一倍的战略。(Amtel在2000年发布了64位计算机，1990年发布了32位计算机，1980年发布了16位计算机，1970年发布了8位计算机，1960年发布了其第一款4位计算机）。

Amtel公司将使用一种新的基准–Factstone–来宣传其新芯片的巨大改进的能力。Factstone评级被定义为最大的整数n，使n！可以在计算机字中表示为一个无符号整数。

考虑到1960≤y≤2160年，Amtel最近发布的芯片的Factstone等级将是多少?

输入

There are several test cases. For each test case, there is one line of input containing *y*. A line containing 0 follows the last test case.

有几个测试用例。对于每个测试用例，有一行包含 y 的输入。包含 0 的行跟随最后一个测试用例

输出

For each test case, output a line giving the Factstone rating.

对于每个测试用例，输出一行给出Factstone等级。

样例输入

```
1960
1981
0
```

样例输出

```
3
8
```

思路

代码

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

int main(){
    int n;
    while((cin >> n) && n){
        double  a = log2(4.0);
        for (int i = 1960; i <= n; i += 10)
            a *= 2;
        double  f = 0;
        int i = 0;
        while(f < a)
        {
            ++i;
```

```
            f += log2(double(i));
        }
        cout << i - 1 << endl;
    }
    return 0;
}
```

问题 J: Ants

时间限制: 1 Sec

内存限制: 128 MB

题目描述

An army of ants walk on a horizontal pole of length *l* cm, each with a constant speed of 1 cm/s. When a walking ant reaches an end of the pole, it immediatelly falls off it. When two ants meet they turn back and start walking in opposite directions. We know the original positions of ants on the pole, unfortunately, we do not know the directions in which the ants are walking. Your task is to compute the earliest and the latest possible times needed for all ants to fall off the pole.

一群蚂蚁走在一根长为 l cm 的水平杆上，每支蚂蚁以 1 cm/s 的恒定速度行走。当一只行走的蚂蚁到达杆子的末端时，它会立即从杆子上掉下来。当两只蚂蚁相遇时，它们会转身向相反的方向走。我们知道蚂蚁在杆子上的原始位置，不幸的是，我们不知道蚂蚁行走的方向。你的任务是计算所有蚂蚁从杆子上掉下来所需的最早和最晚时间。

输入

The first line of input contains one integer giving the number of cases that follow. The data for each case start with two integer numbers: the length of the pole (in cm) and *n*, the number of ants residing on the pole. These two numbers are followed by *n* integers giving the position of each ant on the pole as the distance measured from the left end of the pole, in no particular order. All input integers are not bigger than 1000000 and they are separated by whitespace.

输入的第一行包含一个整数，给出后面的案例数。每个案例的数据都以两个整数开始：杆的长度（以厘米为单位）和 n，杆上的蚂蚁数量。这两个数字后跟 n 个整数，表示每只蚂蚁在杆子上的位置，即从杆子左端测量的距离，没有特定的顺序。所有输入的整数都不大于 1000000，并且它们以空格分隔。

输出

For each case of input, output two numbers separated by a single space. The first number is the earliest possible time when all ants fall off the pole (if the directions of their walks are chosen appropriately) and the second number is the latest possible such time.

对于输入的每种情况，输出由单个空格分隔的两个数字。第一个数字是所有蚂蚁从杆子上掉下来的最早时间（如果它们的行走方向选择得当），第二个数字是最晚的时间。

样例输入

```
2
10 3
2 6 7
214 7
11 12 7 13 176 23 191
```

样例输出

```
4 8
38 207
```

思路

`Min=max(Min,min(a[i],L-a[i]));`

`Max=max(Max,max(a[i],L-a[i]));`

代码

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


int main(){
    int t;
    cin >> t;
    while(t--){
        int l,n;
        cin >> l >> n;
        vector<int> ants(n,0);
        for(int i=0;i<n;i++){
            cin >> ants[i];
        }
        int Max = 0, Min = 0;
        for(int i=0;i<n;i++){
            Min = max(Min,min(ants[i],l-ants[i]));
            Max = max(Max,max(ants[i],l-ants[i]));
        }
        cout << Min << " " << Max << endl;
    }
    return 0;
}
```

问题 K: Matches Game

时间限制: 1 Sec

内存限制: 64 MB

题目描述

Here is a simple game. In this game, there are several piles of matches and two players. The two player play in turn. In each turn, one can choose a pile and take away arbitrary number of matches from the pile (Of course the number of matches, which is taken away, cannot be zero and cannot be larger than the number of matches in the chosen pile). If after a player’s turn, there is no match left, the player is the winner. Suppose that the two players are all very clear. Your job is to tell whether the player who plays first can win the game or not.

这是一个简单的游戏。在这场比赛中，有几堆比赛和两名球员。两个玩家轮流玩。在每一回合中，可以选择一堆并从堆中带走任意数量的火柴（当然，被带走的火柴数量不能为零，也不能大于所选堆中的火柴数量）。如果在轮到玩家之后，没有剩余比赛，则该玩家为赢家。假设两个玩家都非常清楚。你的工作是判断先玩的玩家能否赢得比赛。

输入

The input consists of several lines, and in each line there is a test case. At the beginning of a line, there is an integer M (1 <= M <=20), which is the number of piles. Then comes M positive integers, which are not larger than 10000000. These M integers represent the number of matches in each pile.

输入由几行组成，每行都有一个测试用例。在一行的开头，有一个整数M（1 <= M <=20），就是桩的数量。然后是M个正整数，不大于10000000。这M个整数代表每堆匹配的数量。

输出

For each test case, output “Yes” in a single line, if the player who play first will win, otherwise output “No”.

对于每个测试用例，单行输出“Yes”，如果先玩的玩家获胜，否则输出“No”。

样例输入

```
2 45 45
3 3 6 9
```

样例输出

```
No
Yes
```

思路

这题有问题，我一直感觉学算法，优化应该在熟悉证明上，在算法复杂度上进行优化，而不是在语言效率，底层输入输出上。

代码

c++过不了

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

int main(){
    int m;
    while(~scanf("%d",&m)){
        int flag = 0;
        long long x;
        for(int i=0;i<m;i++){
            cin >> x;
            flag ^= x;
        }
        if(flag)
            cout << "Yes" << endl;
        else
            cout << "No" << endl;
    }
    return 0;
}

```

c就能过

```cpp
#include<stdio.h>
using namespace std;
int main()
{
    int n;
```

```
    while(~scanf("%d",&n))
    {
        int flag=0;
        long long x;
        for(int i=1;i<=n;i++)
        {
            scanf("%lld",&x);
            flag^=x;
        }
        if(flag)
       printf("Yes\n");
        else
        printf("No\n");
    }
    return 0;
}

```

问题 L: sort2

时间限制: 1 Sec

内存限制: 64 MB

题目描述

给你n个整数，请按从大到小的顺序输出其中前m大的数。

输入

每组测试数据有两行，第一行有两个数n,m(0<n,m<1000000)，第二行包含n个都处于区间[-500000,500000]的整数，***整数可能会重复出现***。

输出

对每组测试数据按从大到小的顺序输出前m大的数。

样例输入

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

样例输出

```
9 8 7 7 6
```

思路

代码

直接sort超时

map超时

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

int nums[1000001];


int main(){
    int n,m;
    cin >> n >> m;
    map<int,int> mp;
    for(int i=0;i<n;i++){
        cin >> nums[i];
        if(mp.find(nums[i]) != mp.end()){
            mp[nums[i]]++;
        }
        mp.insert(pair<int,int>(nums[i],1));
    }
    int i = 0;
    for(auto num : mp){
        while(num.second){
            nums[i] = num.first;
            num.second--;
            i++;
        }
    }
    for(int i=n-1;i>n-m-1;i--){
        cout << nums[i] << " ";
    }
    cout << endl;
    return 0;
}
```

方法三

```cpp
#include<bits/stdc++.h>
using namespace std;
const int offset = 500000;
int Hash[1000001] = {0};
int main(){
    int m,n;
    while(cin >> n >> m){
        for(int i=0;i<n;i++){
            int x;
            cin >> x;
            Hash[x+offset]++;
        }
        for(int i=offset;i>=-offset && m > 0;i--){
            while(Hash[i+offset] > 0 && m > 0){
                cout << i << " ";
                Hash[i+offset]--;
```

```
17                  m--;
18              }
19          }
20          cout << endl;
21      }
22      return 0;
23  }
24
```

作业二

问题 A: 单词排序

时间限制: 1 Sec

内存限制: 128 MB

题目描述

小红学会了很多英文单词，妈妈为了帮小红加强记忆，拿出纸、笔，把 N 个单词写在纸上的一行里，小红看了几秒钟后，将这张纸扣在桌子上。妈妈问小红：“你能否将这 N 个单词按照字典排列的顺序，从小到大写出来？”小红按照妈妈的要求写出了答案。现在请你编写程序帮助妈妈检查小红的答案是否正确。注意：所有单词都由小写字母组成，单词两两之间用一个空格分隔。

输入

输入包含两行。

第一行仅包括一个正整数N(0<N≤26)。

第二行包含N个单词，表示妈妈写出的单词，两两之间用一个空格分隔。

单个单词长度不超过1010。

输出

输出仅有一行。针对妈妈写出的单词，按照字典排列的顺序从小到大排列成一行的结果，每个单词后带一个空格。

样例输入

```
1 4
2 city boy tree student
```

样例输出

```
1 boy city student tree
```

思路

代码

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

int main(){
    int n;
    cin >> n;
    vector<string> strs;
    for(int i=0;i<n;i++){
        string str;
        cin >> str;
        strs.push_back(str);
    }
    sort(strs.begin(),strs.end());
    for(int i=0;i<n;i++){
        cout << strs[i] << " ";
    }
    cout << endl;
    return 0;
}

```

问题 B: 求数组的最长递减子序列

时间限制: 1 Sec

内存限制: 128 MB

题目描述

给定一个整数序列，输出它的最长递减（注意不是“不递增”）子序列。

输入

输入包括两行，第一行包括一个正整数N（N<=1000），表示输入的整数序列的长度。第二行包括用空格分隔开的N个整数，整数范围区间为[-30000,30000]。

输出

输出最长递减子序列，数字之间有一个空格。

样例输入

```
8
9 4 3 2 5 4 3 2
```

样例输出

```
9 5 4 3 2
```

思路

求个数很简单，但最后输出的是序列数组，这个比较麻烦。

前面求出dp动态数组和最大值，顺便记录最大值的下表和值

以及每一步的前面的下表，方便后面循环查找。

代码

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

int main(){
    int n;
    cin >> n;
    vector<int> nums(n,0);
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    vector<int> dp(n,1);
    vector<int> track(n,-1);
    int result = 0;
    int rp = -1;
    for(int i=0;i<n;i++){
        for(int j=0;j<i;j++){
            if((nums[i] < nums[j]) && dp[j]+1 > dp[i]){
                dp[i] = dp[j] + 1;
                track[i] = j;
            }
            if(dp[i] > result){
                result = dp[i];
                rp = i;
            }
        }
    }
    vector<int> ans;
    for(int i=result;i>0;i--){
        ans.push_back(nums[rp]);
        if(track[rp] == -1)
            break;
        rp = track[rp];
    }
    for(int i=ans.size()-1;i>=0;i--){
        cout << ans[i] << " ";
    }
    cout << endl;
    return 0;
}
```

问题 C: 矩形滑雪场

时间限制: 1 Sec

内存限制: 128 MB

题目描述

zcb喜欢滑雪。他来到了一个滑雪场，这个滑雪场是一个矩形，为了简便，我们用r行c列的矩阵来表示每块地形。为了得到更快的速度，滑行的路线必须向下倾斜。 例如样例中的那个矩形，可以从某个点滑向上下左右四个相邻的点之一。例如24-17-16-1，其实25-24-23...3-2-1更长，事实上这是最长的一条。

输入

第1行:两个数字r，c(1 ≤ r, c ≤ 100)，表示矩阵的行列。第2..r+1行:每行c个数，表示这个矩阵。

输出

仅一行:输出1个整数，表示可以滑行的最大长度。

样例输入

```
5 5
1 2 3 4 5
16 17 18 19 6
15 24 25 20 7
14 23 22 21 8
13 12 11 10 9
```

样例输出

```
25
```

思路

有点难，dfs搜索应该可以

这里还是那动态规划，把这个看成大型的二维的最长递减序列。

代码

动态规划

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

struct node{
    int x;
    int y;
    int n;
};

bool cmp(node a,node b)
{
    return a.n<b.n;
}

//node nums[100005];

int main(){
    int m,n;
    cin >> m >> n;
    vector<node> nums(m*n);
    int index = 0;
    for(int i=0;i<m;i++){
```

```
        for(int j=0;j<n;j++){
            cin >> nums[index].n;
            nums[index].x = i;
            nums[index].y = j;
            index++;
        }
    }
    sort(nums.begin(),nums.end(),cmp);
    //大型的最长递减子序列
    int result = 0;
    vector<int> dp(index,1);
    for(int i=0;i<index;i++){
        for(int j=0;j<i;j++){
            //这里判断条件改成前后左右
            if(((nums[i].x==nums[j].x && abs(nums[i].y-nums[j].y)==1) || (nums[i].y==nums[j].y &&
abs(nums[i].x-nums[j].x)==1)) && nums[i].n > nums[j].n){
                dp[i] = max(dp[i],dp[j]+1);
            }
            if(dp[i] > result){
                result = dp[i];
            }

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

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

struct node{
    int x,y,h;
};

bool cmp(node a,node b){
    return a.h < b.h;
}

node nums[999999];


int main(){
    int r,c;
    cin >> r >> c;
    int n = 0;
    for(int i=0;i<r;i++){
        for(int j=0;j<c;j++){
            cin >> nums[n].h;
            nums[n].x = i;
            nums[n].y = j;
            n++;
        }
    }
    sort(nums,nums+n,cmp);
    vector<int> dp(n,1);
    int result = 0;
```

```
    for(int i=0;i<n;i++){
        for(int j=0;j<i;j++){
            if((nums[i].h > nums[j].h) && (abs(nums[i].x-nums[j].x) + abs(nums[i].y-nums[j].y)) == 1){
                dp[i] = max(dp[i],dp[j]+1);
            }
            if(dp[i] > result){
                result = dp[i];
            }
        }
    }
    cout << result << endl;
    return 0;
}

```

问题 D: Homework

时间限制: 1 Sec

内存限制: 128 MB

题目描述

临近开学了，大家都忙着收拾行李准 备返校，但 I_Love_C 却不为此担心! 因为他的心思全在暑假作业上：目前为止还未开动。

暑假作业是很多张试卷，我们这些从试卷里爬出来的人都知道，卷子上的题目有选择题、填空题、简答题、证明题等。而做选择题的好处就在于工作量很少，但又因为选择题题目都普遍很长。如果有 5 张试卷，其中 4 张是选择题，最后一张是填空题，很明显做最后一张所花的时间要比前 4 张长很多。但如果你只做了选择题，虽然工作量很少，但表面上看起来也已经做了4/5的作业了。

I_Love_C决定就用这样的方法来蒙混过关，他统计出了做完每一张试卷所需的时间以及它做完后能得到的价值（按上面的原理，选择题越多价值当然就越高咯）。

现在就请你帮他安排一下，用他仅剩的一点时间来做最有价值的作业。

输入

测试数据包括多组。每组测试数据以两个整数 M,N(1<M<20,1<N<10000) 开头，分别表示试卷的数目和 I_Love_C 剩下的时间。接下来有 M 行，每行包括两个整数 T,V(1<T<N,1<V<10000)分别表示做完这张试卷所需的时间以及做完后能得到的价值，输入以 0 0 结束。

输出

对应每组测试数据输出 I_Love_C 能获得的最大价值。保留小数点 2 位

提示：float 的精度可能不够，你应该使用 double 类型。

样例输入

```
4 20
4 10
5 22
10 3
1 2
0 0
```

样例输出

```
37.00
```

思路

代码

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

bool cmp(vector<double>a,vector<double> b){
    return a[2] > b[2];
}

int main(){
    int m,n;
    while(cin >> m >> n){
        if(m == 0 && n ==0){
            break;
        }
        vector<vector<double>> homework(m,vector<double>(3,0));
        for(int i=0;i<m;i++){
            cin >> homework[i][0] >> homework[i][1];
            homework[i][2] = homework[i][1] / homework[i][0];
        }
        sort(homework.begin(),homework.end(),cmp);
        double ans = 0;
        for(int i=0;i<n;i++){
            if(n > homework[i][0]){
                ans += homework[i][1];
                n -= homework[i][0];
            }
            else{
                ans += homework[i][2] * n;
                break;
            }
        }
        printf("%.2lf\n",ans);

    }
    return 0;
}
```

问题 E: 区间包含问题

时间限制: 1 Sec

内存限制: 128 MB

题目描述

已知 n 个左闭右开区间 [a,b)，对其进行 m 次询问，求区间[l,r]最多可以包含 n 个区间中的多少个区间，并且被包含的所有区间都不相交。

输入

输入包含多组测试数据，对于每组测试数据：

第一行包含两个整数 n ,m (1≤n≤100000,1≤m≤100) 。

接下来 n 行每行包含两个整数 a ,b (0≤a<b≤10^9) 。

接下来 m 行每行包含两个整数 l ,r (0≤l<r≤10^9) 。

输出

对于每组测试数据，输出 m 行，每行包含一个整数。

数据过大请使用快速输入输出。

样例输入

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

样例输出

```
1
1
2
```

思路

就是右端点小排序

优先选取满足小区间

代码

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

bool cmp(vector<int> a, vector<int> b){
    return a[1] < b[1];
}

int main(){
    int n, m;
```

```
    while(cin >> n >> m){
        vector<vector<int>> point(n,vector<int>(2,0));
        for(int i=0;i<n;i++){
            cin >> point[i][0] >> point[i][1];
        }
        sort(point.begin(),point.end(),cmp);
        while(m--){
            int left,right;
            cin >> left >> right;
            int ans = 0;
            for(int i=0;i<n;i++){
                if(point[i][1] > right){
                    break;
                }
                if(left <= point[i][0]){
                    left = point[i][1];
                    ans++;
                }
            }
            cout << ans << endl;
        }
    }
    return 0;
}

```

问题 F: 最长子序列

时间限制: 1 Sec 内存限制: 128 MB

题目描述

在一个数组中找出和最大的连续几个数。（至少包含一个数）

例如：

数组A[] = [-2,1,-3,4,-1,2,1,-5,4]，则连续的子序列[4,-1,2,1]有最大的和6.

输入

第一行输入一个不超过1000的整数n。

第二行输入n个整数A[i]。

输出

输出一个整数，表示最大的和。

样例输入

```
3
1 1 -2
```

样例输出

```
2
```

思路

代码

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

int main(){
    int n;
    cin >> n;
    vector<int> nums(n,0);
    for(int i=0;i<n;i++){
        cin >> nums[i];
    }
    int sum = 0;
    int ans = INT_MIN;
    for(int i=0;i<n;i++){
        sum += nums[i];
        if(sum > ans){
            ans = sum;
        }
        if(sum < 0){
            sum = 0;
        }
    }
    cout << ans << endl;
    return 0;
}
```

问题 G: 元素整除问题

时间限制: 1 Sec

内存限制: 128 MB

题目描述

输入20个整数，输出其中能被数组中其它元素整除的那些数组元素。

输入

输入20个整数

输出

按输入顺序输出符合要求的数字，每行输出一个整数。

样例输入

```
2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
```

样例输出

```
4
6
8
9
10
12
14
15
16
18
20
21
```

思路

代码

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

int main(){
    vector<int> nums(20,0);
    for(int i=0;i<20;i++){
        cin >> nums[i];
    }
    for(int i=0;i<20;i++){
        bool flag = false;
        for(int j=0;j<20;j++){
            if(i == j){
                continue;
            }
            if((nums[i] % nums[j]) == 0){
                flag = true;
                break;
            }
        }
        if(flag){
            cout << nums[i] <<endl;
        }
    }
    return 0;
}
```

问题 H: 渊子赛马

时间限制: 1 Sec

内存限制: 128 MB

题目描述

赛马是一古老的游戏，早在公元前四世纪的中国，处在诸侯割据的状态，历史上称为“战国时期”。在魏国作官的孙膑，因为受到同僚庞涓的迫害，被齐国使臣救出后，到达齐国国都。 赛马是当时最受齐国贵族欢迎的娱乐项目。上至国王，下到大臣，常常以赛马取乐，并以重金赌输赢。田忌多次与国王及其他大臣赌输赢，屡赌屡输。一天他赛马又输了，回家后闷闷不乐。孙膑安慰他说：“下次有机会带我到马场看看，也许我能帮你。”孙膑仔细观察后发现，田忌的马和其他人的马相差并不远，只是策略运用不当，以致失败。 比赛前田忌按照孙膑的主意，用上等马鞍将下等马装饰起来，冒充上等马，与齐王的上等马比赛。第二场比赛，还是按照孙膑的安排，田忌用自己的上等马与国王的中等马比赛，在一片喝彩中，只见田忌的马竟然冲到齐王的马前面，赢了第二场。关键的第三场，田忌的中等马和国王的下等马比赛，田忌的马又一次冲到国王的马前面，结果二比一，田忌赢了国王。 就是这么简单，现在渊子也来赛一赛马。假设每匹马都有恒定的速度，所以速度大的马一定比速度小的马先到终点（没有意外！！）。不允许出现平局。最后谁赢的场数多于一半(不包括一半)，谁就是赢家(可能没有赢家)。渊子有 N(1<=n<=1000)匹马参加比赛。对手的马的数量与渊子马的数量一样，并且知道所有的马的速度。聪明的你来预测一下这场世纪之战的结果，看看渊子能否赢得比赛。

输入

输入有多组测试数据。 每组测试数据包括 3 行： 第一行输入 N。表示马的数量。 第二行有 N 个整型数字，即渊子的 N 匹马的速度。 第三行有 N 个整型数字，即对手的 N 匹马的速度。 当 N 为 0 时退出。

输出

若通过聪明的你精心安排，如果渊子能赢得比赛，那么输出YES。 否则输出NO。

样例输入

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

样例输出

```
YES
NO
```

思路

贪心吧

先分别排序，看 `a[i] > b[j]` 如果大于那就赢了一把，敌方换马，如果一直没赢，因为从大到小排序，证明后面也赢不了。

切记记录失败次数。

代码

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

int main(){
    int n;
    while((cin >> n) && n){
```

```
        vector<int> a(n,0);
        vector<int> b(n,0);
        for(int i=0;i<n;i++){
            cin >> a[i];
        }
        for(int i=0;i<n;i++){
            cin >> b[i];
        }
        sort(a.begin(),a.end());
        sort(b.begin(),b.end());
        int cnt1 =0, cnt2 = 0;
        int j = 0;
        for(int i=0;i<n;i++){
            if(a[i] > b[j]){
                cnt1++;
                j++;
            }
            else{
                cnt2++;
            }
            if(cnt1 > cnt2){
                cout << "YES" << endl;
            }
            else{
                cout << "NO" << endl;
            }
        }
        return 0;
}
```

问题 I: The Hardest Problem Ever

时间限制: 1 Sec

内存限制: 32 MB

题目描述

Julius Caesar lived in a time of danger and intrigue. The hardest situation Caesar ever faced was keeping himself alive. In order for him to survive, he decided to create one of the first ciphers. This cipher was so incredibly sound, that no one could figure it out without knowing how it worked.

You are a sub captain of Caesar’s army. It is your job to decipher the messages sent by Caesar and provide to your general. The code is simple. For each letter in a plaintext message, you shift it five places to the right to create the secure message (i.e., if the letter is ‘A’, the cipher text would be ‘F’). Since you are creating plain text out of Caesar’s messages, you will do the opposite:

Cipher text
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Plain text
V W X Y Z A B C D E F G H I J K L M N O P Q R S T U

Only letters are shifted in this cipher. Any non-alphabetical character should remain the same, and all alphabetical characters will be upper case.

朱利叶斯·凯撒生活在一个充满危险和阴谋的时代。凯撒面临的最艰难的情况是让自己活着。为了让他活下来，他决定创造第一个密码。这个密码非常可靠，如果不知道它是如何工作的，就没有人能猜出它。

你是凯撒军队的副队长。你的工作是破译凯撒发送的信息并提供给你的将军。代码很简单。对于明文消息中的每个字母，您将其向右移动五位以创建安全消息（即，如果字母是“A”，则密文将是“F”）。由于您是从 Caesar 的消息中创建纯文本，因此您将执行相反的操作：

密文

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

纯文本

V W X Y Z A B C D E F G H I J K L M N O P Q R S T U

在这个密码中只有字母被移位。任何非字母字符都应保持不变，所有字母字符都将大写。

输入

Input to this problem will consist of a (non-empty) series of up to 100 data sets. Each data set will be formatted according to the following description, and there will be no blank lines separating data sets. All characters will be uppercase.

A single data set has 3 components:

1. Start line - A single line, “START”
2. Cipher message - A single line containing from one to two hundred characters, inclusive, comprising a single message from Caesar
3. End line - A single line, “END”

Following the final data set will be a single line, “ENDOFINPUT”.

此问题的输入将包含最多 100 个数据集的（非空）系列。每个数据集将根据以下描述进行格式化，并且不会有分隔数据集的空行。所有字符都将大写。

单个数据集有 3 个组成部分：

起始行 - 单行，“START”

密码消息 - 一行包含 1 到 200 个字符（含），包含来自 Caesar 的单个消息

结束行 - 单行，“END”

在最终数据集之后将是一行，“ENDOFINPUT”。

输出

For each data set, there will be exactly one line of output. This is the original message by Caesar.

对于每个数据集，只会有一行输出。这是凯撒的原始信息。

样例输入

```
START
NS BFW, JAJSYX TK NRUTWYFSHJ FWJ YMJ WJXZQY TK YWNANFQ HFZXJX
END
START
N BTZQI WFYMJW GJ KNWXY NS F QNYYQJ NGJWNFS ANQQFLJ YMFS XJHTSI NS WTRJ
END
START
IFSLJW PSTBX KZQQ BJQQ YMFY HFJXFW NX RTWJ IFSLJWTZX YMFS MJ
END
ENDOFINPUT
```

样例输出

```
IN WAR, EVENTS OF IMPORTANCE ARE THE RESULT OF TRIVIAL CAUSES
I WOULD RATHER BE FIRST IN A LITTLE IBERIAN VILLAGE THAN SECOND IN ROME
DANGER KNOWS FULL WELL THAT CAESAR IS MORE DANGEROUS THAN HE
```

思路

凯撒加密，难度上没啥，就是控制输入输出得调试。

代码

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

int main(){
    string str;
    while(1){
        cin >> str;
        if(str == "ENDOFINPUT"){
            break;
        }
        else if(str == "START"){
            cin >> str;
            string s;
            getline(cin,s);
            s = str + s;
            string c;
            for(int i=0;i < s.size();i++){
                if(s[i] >= 'A' && s[i] <= 'Z'){
                    c.push_back((s[i] - 'A'+26-5)%26 + 'A');
                }
                else{
                    c.push_back(s[i]);
                }
            }
            cout << c << endl;
        }
        else if(str == "END"){
            continue;
        }
    }
    return 0;
}
```

问题 J: Rock-Paper-Scissors Tournament

时间限制: 3 Sec

内存限制: 64 MB

题目描述

Rock-Paper-Scissors is game for two players, A and B, who each choose, independently of the other, one of rock, paper, or scissors. A player chosing paper wins over a player chosing rock; a player chosing scissors wins over a player chosing paper; a player chosing rock wins over a player chosing scissors. A player chosing the same thing as the other player neither wins nor loses.
A tournament has been organized in which each of n players plays k rock-scissors-paper games with each of the other players - k*n*(n-1)/2 games in total. Your job is to compute the win average for each player, defined as w / (w + l) where w is the number of games won, and l is the number of games lost, by the player.

Rock-Paper-Scissors 是 A 和 B 两个玩家的游戏，他们各自独立地选择石头、纸或剪刀之一。选择纸的玩家胜过选择石头的玩家；选择剪刀的玩家胜过选择纸的玩家；选择石头的玩家胜过选择剪刀的玩家。与其他玩家选择相同事物的玩家既不会赢也不会输。

已经组织了一个锦标赛，其中 n 个玩家中的每一个与其他每个玩家玩 k 个石头剪刀布游戏 - 总共 k*n*(n-1)/2 个游戏。您的工作是计算每个玩家的平均获胜次数，定义为 w / (w + l)，其中 w 是该玩家赢得的游戏数量，l 是该玩家输掉的游戏数量。

输入

Input consists of several test cases. The first line of input for each case contains 1 <= n <= 100 1 <= k <= 100 as defined above. For each game, a line follows containing p1, m1, p2, m2. 1 <= p1 <= n and 1 <= p2 <= n are distinct integers identifying two players; m1 and m2 are their respective moves (“rock”, “scissors”, or “paper”). A line containing 0 follows the last test case.

输入由几个测试用例组成。每个案例的第一行输入包含 1 <= n <= 100 1 <= k <= 100，如上所定义。对于每个游戏，后面有一行包含 p1、m1、p2、m2。 1 <= p1 <= n 和 1 <= p2 <= n 是识别两个玩家的不同整数； m1 和 m2 是它们各自的移动（“石头”、“剪刀”或“纸”）。包含 0 的行跟随最后一个测试用例。

输出

Output one line each for player 1, player 2, and so on, through player n, giving the player’s win average rounded to three decimal places. If the win average is undefined, output “-“. Output an empty line between cases.

为玩家 1、玩家 2 等输出一行，通过玩家 n，将玩家的胜利平均值四舍五入到小数点后三位。如果未定义获胜平均值，则输出“-”。在案例之间输出一个空行。

样例输入

```
2 4
1 rock 2 paper
1 scissors 2 paper
1 rock 2 rock
2 rock 1 scissors
2 1
1 rock 2 paper
0
```

样例输出

```
0.333
0.667

0.000
1.000
```

思路

代码

问题 K: Balloon Robot

时间限制: 1 Sec

内存限制: 64 MB

题目描述

The 2017 China Collegiate Programming Contest Qinhuangdao Site is coming! There will be (n) teams participating in the contest, and the contest will be held on a huge round table with (m) seats numbered from 1 to (m) in clockwise order around it. The (i)-th team will be seated on the (s_i)-th seat.

BaoBao, an enthusiast for competitive programming, has made (p) predictions of the contest result before the contest. Each prediction is in the form of ((a_i,b_i)), which means the (a_i)-th team solves a problem during the (b_i)-th time unit.

As we know, when a team solves a problem, a balloon will be rewarded to that team. The participants will be unhappy if the balloons take almost centuries to come. If a team solves a problem during the (t_a)-th time unit, and the balloon is sent to them during the (t_b)-th time unit, then the unhappiness of the team will increase by (t_b-t_a). In order to give out balloons timely, the organizers of the contest have bought a balloon robot.

At the beginning of the contest (that is to say, at the beginning of the 1st time unit), the robot will be put on the (k)-th seat and begin to move around the table. If the robot moves past a team which has won themselves some balloons after the robot's last visit, it will give all the balloons they deserve to the team. During each unit of time, the following events will happen **in order**:

1. The robot moves to the next seat. That is to say, if the robot is currently on the (i)-th ((1 \le i < m)) seat, it will move to the ((i+1))-th seat; If the robot is currently on the (m)-th seat, it will move to the 1st seat.
2. The participants solve some problems according to BaoBao's prediction.
3. The robot gives out balloons to the team seated on its current position if needed.

BaoBao is interested in minimizing the total unhappiness of all the teams. Your task is to select the starting position (k) of the robot and calculate the minimum total unhappiness of all the teams according to BaoBao's predictions.

输入

There are multiple test cases. The first line of the input contains an integer T, indicating the number of test cases. For each test case:

The first line contains three integers (n), (m) and (p) ((1 \le n \le 10^5), (n \le m \le 10^9), (1 \le p \le 10^5)), indicating the number of participating teams, the number of seats and the number of predictions.

The second line contains (n) integers (s_1, s_2, \dots, s_n) ((1 \le s_i \le m), and (s_i \ne s_j) for all (i \ne j)), indicating the seat number of each team.

The following (p) lines each contains two integers (a_i) and (b_i) ((1 \le a_i \le n), (1 \le b_i \le 10^9)), indicating that the (a_i)-th team solves a problem at time (b_i) according to BaoBao's predictions.

It is guaranteed that neither the sum of (n) nor the sum of (p) over all test cases will exceed (5 \times 10^5).

输出

For each test case output one integer, indicating the minimum total unhappiness of all the teams according to BaoBao's predictions.

样例输入

```
4
2 3 3
```

```
1 2
1 1
2 1
1 4
2 3 5
1 2
1 1
2 1
1 2
1 3
1 4
3 7 5
3 5 7
1 5
2 1
3 3
1 5
2 5
2 100 2
1 51
1 500
2 1000
```

样例输出

```
1
4
5
50
```

思路