2170.同步排列

对于1到n的n个数，有一个排列a[1],a[2],...,a[n]。排列里两个相邻的数字可能是从小变大，也可能是从大变小。请问有多少种排列和a[]里邻居们的相对大小关系吻合? 对这个总数量取模1000000007。

例如：1324 里先变大，再变小，最后再变大。吻合如此变化的排列有5种：

1324

1423

2314

2413

3412

输入文件为sychronization.in

第一行包括一个整数n表示数字个数。第二行为n个正整数代表排列。2<=n<=3000

输出文件为sychronization.out

一个整数。

输入样例

4

1 3 2 4

输出样例

5

输入样例

5

2 1 3 5 4

输出样例

6

说明

21354

21453

31254

31452

32451

41253

41352

42351

51243

51342

52341

2171.分组过桥

今年夏天，你和小伙伴们相约来到一座小岛参加夏令营，必须通过一座小桥才可以进出小岛。突然有一晚，狂风大作，暴雨如注，你们决定连夜逃离小岛回家。可是来到小桥旁，你们发现这座桥已经被大风刮得摇摇欲坠。经过大家评估，保险起见，每次安排过桥的人总体重不能超过W斤。目前共有n个人，第i人单独过桥时间为t[i]分钟，体重为w[i]斤。大家会分批次通过。其中每一批次的通过时间为该批次里最慢的人的通行时间。请问，要所有人安全过桥，最少要几分钟？

输入文件bridge.in 输入第一行为正整数W和n，100<=W<=400，n<=16。接着共n行，每行两个正整数表示t[i]和w[i]。其中t[i]<=50，10<=w[i]<=100。

输出文件bridge.out 输出一个整数。

输入样例：

100 3

24 60

10 40

18 50

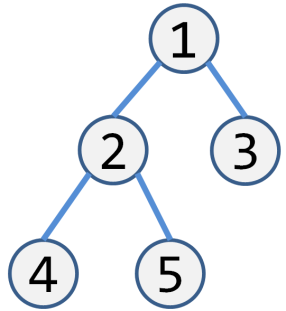
输出样例：

42

2162.欧拉序列2

欧拉序列一共有两种形态，今天我们介绍第二类欧拉序列。

有1棵树，共n个节点编号1到n，根节点是1号。欧拉序列有2种常见定义，今天学习第1种定义：从根节点开始按照DFS顺序访问节点和边，访问包括回溯的步骤。在DFS时若某节点u有多个子节点，优先挑选编号小的子节点访问。对每个节点u，每一次访问u时，我们会将u按照访问顺序依次记录在欧拉序列（第二类）中。这种欧拉序列总长度为2\*n-1，每个节点至少出现1次，也可能出现多次。



如图，第二类欧拉序列为124252131。

根据欧拉序列，我们顺便整理出以下实用信息：

T[u].L代表节点u第一次访问是欧拉序列里的第几个。

T[u].R代表节点u最后一次访问是欧拉序列里的第几个。

这样的话，节点u为根的子树就对应序列中的一个区间[T[u].L,T[u].R]。

在图中例子中，

T[1].L=1, T[1].R=9,

T[2].L=2, T[2].R=6,

T[3].L=8, T[3].R=8,

T[4].L=3, T[4].R=3,

T[5].L=5, T[5].R=5,

输入文件euler.in 输入第一行为正整数n，n<=1000。接着n-1行为树边的描述，每行两个正整数u,v代表u和v之间有一条边。

输出文件euler.out 输出共三行：第一行为欧拉序列，共2\*n-1个正整数；第二行为T[u].L信息，共n个正整数；第二行为T[u].R信息，共n个正整数。格式要求数字之间由空格隔开，但行末不能有空格。

输入样例：

5

1 2

1 3

2 4

2 5

输出样例：

1 2 4 2 5 2 1 3 1

1 2 8 3 5

9 6 8 3 5

1901. 换根子树（模板题）

一棵n个节点的无根树，有q个问询。每个问询为X Y，表示求以X为根时，子树Y的节点数。

本题强制在线回答，也就是每个问询需要直接给出答案，才能够继续处理下一个问询。你需要使用以下代码模板生成输入数据及计算输出

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | int nextInt(int i,int n){      return abs((i\*i+996703)^(i\*i\*i+167317))%n+1;  }    int main(){      cin>>n>>q>>p>>seed;      for (int i=1;i<n;++i) // 生成树结构          add(i+1, i<=p ? i: nextInt(i,i));      int ans=0;      for (int i=1,X,Y,lastAns=seed;i<=q;i++){ // 生成问询          X=nextInt(lastAns+i,n),Y=nextInt(X+i,n);          ans^=(lastAns=solve(X,Y));      }      cout<<ans<<endl;      return 0;  } |

其中add(u,v)表示u,v间有一条边，add函数具体代码由你自己实现。lastAns表示上一个问询的答案（也就是以X为根时子树Y的节点数，solve函数是由你自己实现的算法逻辑）

输入格式:一行四个正整数n,q,p,seed

输出格式:一个整数，表示所有q个问询对应答案的异或。

输入样例

10 10 3 5

输出样例

11

说明

20%的数据n,q<=10

50%的数据n,q<=1000

70%的数据n,q<=100000

100%的数据n<=100000,q<=2000000,1<=p<=n, 1<=seed<=1000000

总共有40%的数据p<=10