提高组模拟赛

题目名称	灯塔覆盖	剪刀石头布	滑稽果
题目类型	传统型	传统型	传统型
目录	lighting	rsp	energy
可执行文件名	lighting	rsp	energy
输入文件名	lighting.in	rsp.in	energy.in
输出文件名	lighting.out	rsp.out	energy.out
每个测试点时限	1s	3s	2s
内存限制	512M	512M	512M
测试点数目	10	10	20
每个测试点分值	10	10	5

提交源程序文件名：

对于 C++语言	lighting.cpp	rsp.cpp	energy.cpp
对于 C 语言	lighting.c	rsp.c	energy.c
对于 Pascal 语言	lighting.pas	rsp.pas	energy.pas

编译选项:

对于 C++语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 C 语言	-O2 -lm	-O2 -lm	-O2 -lm
对于 Pascal 语言	-O2	-O2	-O2

*评测时不单独设置栈空间限制，为内存限制。

灯塔覆盖(lighting)

【题目描述】

一块大小为 N*N 的场地，场地上有 K 个灯塔。每个灯塔可以照亮一块以它为顶点且平行于坐标轴的矩形。小 L 想知道，是否存在一种灯塔照明方案，使得所整个场地都被照亮。多组数据。

**注意不是每个格点被照亮，而是每个格子都被照亮。*

【输入格式】

从文件 *lighting.in* 中读入数据。

第一行一个正整数 T，表示数据组数。

对于每一组数据：

第一行 2 个整数 K,N，表示场地大小和的灯塔数量。

第二行到第 K+1 行，每行两个整数 X_i,Y_i表示每一个灯塔的坐标。

【输出格式】

输出到文件 *lighting.out* 中。

对于每一组数据，如果灯光可以覆盖所有的场地，输出“yes”，否则输出“no”。

【样例 1 输入】

```
2
2 10
0 0
1 0
2 10
1 2
1 1
```

【样例 1 输出】

```
yes
no
```

【样例 2 输入】

见下发文件 *lighting2.in* 。

【样例 2/3/4 输出】

见下发文件 *lighting2.ans* 。

【子任务】

对于 100%的数据：$1 \leqslant N \leqslant 10^9$; $K \leqslant 100$; $T \leqslant 100$；

详细数据范围见下表：

测试点编号	N	K	测试点编号	N	K
1	≤1000	≤2	6	$\leqslant 10^9$	≤15
2		≤5	7		
3		≤6	8		≤100
4		≤7	9		
5		≤8	10		

石头剪刀布(rsp)

【题目描述】

小 P 最近迷上了石头剪刀布，他观看了一场沙雕石头剪刀布大赛

比赛共有 2^n 个沙雕参加，选手的编号为 0...2^n-1，分为 n 轮，在每轮中，第 0 位选手和第 1 位选手对战，胜者作为新的第 0 位选手，第 2 位和第 3 位对战，胜者作为新的第 1 位选手，以此类推。

小 P 得知，每个沙雕都有一种固定的偏爱决策，每个沙雕在每一次对战中都只会使用他的偏爱决策。

如果一次对战的双方的偏爱决策相同，那么这次对战就永远不会结束，那么作为观众会十分无聊。

现在，小 P 知道了偏爱每种决策的沙雕数目，他想知道一种能够决出最终胜负的初始的次序。

若有多种可能次序，我们设**字典序最小**的为答案。

因为答案可能很长，你只需要输出答案的 hash 值以及第 l 到 r 位。

hash=$\sum S_i \times 233^i \bmod 998244353$ (i=0...2^n-1)

(Si 表示初始标号为 i 的人的偏好决策对应的大写字母 ASCII 码)

【输入格式】

从文件 *rsp.in* 中读入数据。

第一行两个整数 n, op，表示数据规模和数据类型。

当 op = 1 时，不需要输出答案的第 l 到 r 位。

当 op = 2 时，不需要输出 hash 值。

第二行一个大整数，表示偏爱决策为石头 R 的人数。

第三行一个大整数，表示偏爱决策为剪刀 S 的人数。

第四行一个大整数，表示偏爱决策为布 P 的人数。

若 op ≠ 1，第五，六行，两个 n 位二进制数 l,r，表示要求你输出的范围。

【输出格式】

输出到文件 *rsp.out* 中。

若不存在合法初始序列，输出-1，否则：

若 op ≠ 2，输出一个整数，表示最优序列 hash 值。

若 op ≠ 1，输出一个由"R,S,P"构成的字符串，表示最优序列的第 l 到 r 位。

【样例 1 输入】

```
4 3
4
4
8
0000
1111
```

【样例 1 输出】

-1

【样例 2 输入】

1 1
1
1
0

【样例 2 输出】

19421

【样例 3 输入】

3 3
2
3
3
011
110

【样例 3 输出】

879001374
SPSR

【子任务】

对于 100%的数据：1≤n≤300000, 0≤r-l≤300000, R+S+P=2^n。
详细数据范围见下表：

测试点编号	N	r-l	op
1	≤3	≤8	1
2			3
3	≤20	≤1000	1
4			2
5	≤2000	≤2000	1
6			2
7			3
8	≤300000	≤300000	1
9			2
10			3

滑稽果(energy)

【题目描述】

树上现在有着 n 个滑稽果，每个滑稽果有一个能量值 Vi，小 L 将要从树上取下 k 个滑稽果。

小 L 对于每种能量和选取的方案数感兴趣，但由于不同的能量和太多了，所以他只想要知道（能量和*方案数）的异或和。具体来说，他想要知道:

$ans_i*i \bmod 998244353$ 的异或和 （设能量和为 i 的选取方案数为 ans_i）

注意如果选了 2 个滑稽果 a 和 b，(a,b)和(b,a)视为一种方案。选 3 个滑稽果(a,b,c)时，(a,b,c),(b,c,a),(c,a,b),(c,b,a),(b,a,c),(a,c,b)视为一种方案。

【输入格式】

从文件 *energy.in* 中读入数据。

第一行 2 个整数 N,K，表示滑稽果的数量，选取的个数。

第二行 N 个整数 $V_1..V_n$。表示每个滑稽果的能量值。

【输出格式】

输出到文件 *energy.out* 中。

设答案中能量和为 i 的方案数为 ans_i。

输出所有 ans_i 不为 0 的位置中，$ans_i*i \bmod 998244353$ 的异或和。注意最后的异或和<u>不要</u>再一次取模。

【样例 1 输入】

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

【样例 1 输出】

```
28
```

【样例 1 解释】

有 4 种不同的选取方案(1,2,3)(1,2,4)(1,3,4)(2,3,4)，它们的能量和分别为 5,8,8,9。异或和为 5 xor (8*2) xor 9 = 28。

【样例 2/3/4 输入】

见下发文件 *energy2/3/4.in* 。

这三个输入对应了 K=2/3/4 的情况。

【样例 2/3/4 输出】

见下发文件 *energy2/3/4.ans* 。

【子任务】

对于 100%的数据：$1 \leqslant N \leqslant 200000$; $K \leqslant 4$; $1 \leqslant Vi \leqslant 100000$;

详细数据范围见下表：

测试点编号	N	K	Vi
1	≤1000	≤4	≤1000
2			
3	≤200000	=2	
4			
5		=3	
6			
7			
8		=4	
9			
10			
11			
12		=2	≤100000
13			
14		=3	
15			
16			
17		=4	
18			
19			
20			