

NOI2019 湖南省组队选拔

第一试试题

一. 题目概况

题目名称	鱼	JOJO	多边形
目录	fish	jojo	polygon
可执行文件名	fish	jojo	polygon
输入文件名	fish.in	jojo.in	polygon.in
输出文件名	fish.out	jojo.out	polygon.out
每个测试点时限	2 秒	1 秒	1 秒
测试点数目	10	10	20
每个测试点分值	10	10	5
结果比较方式	整数比较, 单行 单个数字比较	整数比较, 多行 单个数字比较	整数比较, 多行 多个数字比较
题目类型	传统	传统	传统
内存上限	512M	512M	512M

二. 提交源程序需加后缀

对于 C++语言	fish.cpp	jojo.cpp	polygon.cpp
对于 C 语言	fish.c	jojo.c	polygon.c
对于 Pascal 语言	fish.pas	jojo.pas	polygon.pas

三. 编译命令

对于 C++语言	g++ -o fish fish.cpp -lm -O2	g++ -o jojo jojo.cpp -lm	g++ -o polygon polygon.cpp -lm
对于 C 语言	gcc -o fish fish.c -lm -O2	gcc -o jojo jojo.c -lm	gcc -o polygon polygon.c -lm
对于 Pascal 语言	fpc fish.pas -O2	fpc jojo.pas	fpc polygon.pas

注意事项:

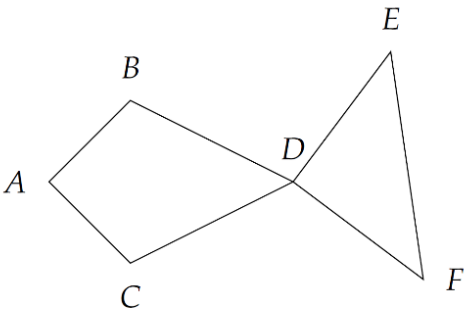
- (1) 选手必须在自己的工作目录下操作, 严禁在其他目录下工作。目录结构请遵从 NOI 规范, 即需要在工作目录下再为每个题目建相应子目录, 子目录名为对应题目的英文名。
- (2) 选手最后提交的源程序 (.pas 或 .c 或 .cpp) 必须在自己的工作目录里对应子目录下, 对于缺少文件者, 不予测试, 该题计零分。
- (3) 子目录名、源程序文件名和输入输出文件名必须使用英文小写。
- (4) 特别提醒: 评测在 NOI Linux 下进行。

第 1 题：鱼(fish)，运行时限 2s，内存上限 512M，100 分。

【问题描述】

在平面坐标系上给定 n 个不同的整点（也即横坐标与纵坐标皆为整数的点）。我们称从这 n 个点中选择6个不同的点所组成的有序六元组 $\langle A,B,C,D,E,F \rangle$ 是一条“鱼”，当且仅当： $AB = AC, BD = CD, DE = DF$ （身形要对称），并且 $\angle BAD, \angle BDA$ 与 $\angle CAD, \angle CDA$ 都是锐角（脑袋和屁股显然不能是凹的）， $\angle ADE, \angle ADF$ 大于 90° （也即为钝角或平角，为了使尾巴不至于翘那么别扭）。

下图就是一个合法的鱼的例子：



其中点的组成相同，但顺序不同的鱼视为不同的鱼，即 $\langle A,B,C,D,E,F \rangle$ 和 $\langle A,C,B,D,E,F \rangle$ 视为不同的两条鱼（毕竟鱼也有背和肚子的两面），同理 $\langle A,B,C,D,E,F \rangle$ 和 $\langle A,B,C,D,F,E \rangle$ 也可以视为不同的两条鱼（假设鱼尾巴可以打结）。

问给定的 n 个点可以构成多少条鱼。特别的，数据保证 n 个点互不重复。

【程序文件名】

源程序文件名为`fish.cpp/c/pas`。

【输入格式】

输入文件名为`fish.in`。

第一行一个正整数 n ，代表平面上点的个数。

接下来 n 行每行两个整数 x,y ，代表点的横纵坐标。

【输出格式】

输出文件名为`fish.out`。

输出一行一个非负整数，代表鱼的个数。

【输入输出样例】

fish.in	fish.out
8 -2 0 -1 0 0 1 0 -1 1 0 2 0 3 1 3 -1	16

【数据范围】

对于前 20%的数据，保证 $n \leq 10$ ， $0 \leq |x|, |y| \leq 5$ 。

对于前 40%的数据，保证 $n \leq 300$ ， $0 \leq |x|, |y| \leq 10^5$ 。

对于另外 20%的数据，保证 $|x|, |y| \leq 20$ 。

对于所有数据，保证 $6 \leq n \leq 1000$ ， $0 \leq |x|, |y| \leq 10^9$ ， n 个点互不重复。

【编译命令】

对于 c++语言： `g++ -o fish fish.cpp -lm -O2`

对于 c 语言： `gcc -o fish fish.c -lm -O2`

对于 pascal 语言： `fpc fish.pas -O2`

第 2 题：JOJO(joyo)，运行时限 1s，内存上限 512M。

【问题描述】

JOJO 的奇幻冒险是一部非常火的漫画。漫画中的男主角经常喜欢连续喊很多的“欧拉”或者“木大”。

为了防止字太多挡住漫画内容，现在打算在新的漫画中用 x 欧拉或者 x 木大表示有 x 个欧拉或者木大。

为了简化内容我们现在用字母表示喊出的话。

我们用数字和字母来表示一个串，例如：2 a 3 b 表示的串就是 aabbb。

一开始漫画中什么话都没有，接下来你需要依次实现 n 个操作，总共只有 2 种操作：

第一种：1 x c 在当前漫画中加入 x 个 c (表示在当前串末尾加入 x 个 c 字符。)(保证当前串是空串或者串尾字符不是 c)

第二种：2 x 觉得漫画没画好将漫画还原到第 x 次操作以后的样子(表示将串复原到第 x 次操作后的样子，如果 $x=0$ 则是将串变成空串。)(如果当前串是 bbaabbb，第 4 次操作后串是 bb，则 2 4 会使 bbaabbb 变成 bb，保证 x 小于当前操作数)

众所周知空条承太郎十分聪明，现在迪奥已经被打败了，他开始考虑自己的漫画中的一些问题：

对于一个串的前缀 A，都有一个最长的比它短的前缀 B 与前缀 A 的一个后缀匹配，设这个最长的前缀 B 的长度为 L。L 为 0 时意味着 B 是一个空串。

每一次操作后，你都需要将当前的串的所有前缀的 L 求和并对 998244353 取模输出告诉空条承太郎，好和他的白金之星算出的答案对比。

比如 bbaaabba 的 L 分别是:0 1 0 0 0 1 2 3，所以对于这个串的答案就是 7。

【程序文件名】

源程序文件名为 jojo.cpp/c/pas。

【输入格式】

输入文件名为 jojo.in。

第一行包括一个正整数 n ，表示操作数量。

接下来 n 行每行包含一个操作，操作格式如题面所示，例如：

1 x c

2 x

保证数据合法。

【输出格式】

输出文件名为jojo.out。

输出文件仅包含 n 行，第 i 行一个整数，表示 i 个操作之后串的答案。

【输入输出样例】

jojo.in	jojo.out
1 1	1
1 2 a	1
1 3 b	4
1 2 a	7
1 1 b	1
2 2	6
1 3 a	13
1 2 b	6
2 6	1
2 5	14
1 7 a	14
1 5 c	

【样例解释】

1:aa 0+1=1

2:aabbb 0+1+0+0+0=1

3:aabbbaa 0+1+0+0+0+1+2=4

4:aabbbaab 0+1+0+0+0+1+2+3=7

5:aabbb 0+1+0+0+0=1

6:aabbbaaa 0+1+0+0+0+1+2+2=6

7:aabbbaaabb 0+1+0+0+0+1+2+2+3+4=13

8:aabbbaaa 0+1+0+0+0+1+2+2=6

9:aabbb 0+1+0+0+0=1

10:aabbbaaaaaa 0+1+0+0+0+1+2+2+2+2+2+2=14

11:aabbbaaaaaacccc 0+1+0+0+0+1+2+2+2+2+2+2+0+0+0+0=14

【数据范围】

20% 的数据满足 $n \leq 300$ 对于每个 1 操作中的 $x \leq 300$;

另有 30% 的数据满足 $n \leq 100000$ 且对于每个 1 操作中的 $x=1$;

另有 30% 的数据满足 $n \leq 100000$ 且不含 2 操作;

100% 的数据满足 $n \leq 100000$ 且每个 1 操作中的 $x \leq 10000$ 。

【编译命令】

对于c++语言: `g++ -o jojo jojo.cpp -lm`

对于c语言: `gcc -o jojo jojo.c -lm`

对于pascal语言: `fpc jojo.pas`

第3题：多边形(polygon)，运行时限 1s，内存上限 512M，100 分。

【问题描述】

小 R 与小 W 在玩游戏。

他们有一个边数为 n 的凸多边形，其顶点沿逆时针方向标号依次为 $1, 2, 3, \dots, n$ 。最开始凸多边形中有 n 条线段，即多边形的 n 条边。这里我们用一个有序数对 (a, b) （其中 $a < b$ ）来表示一条端点分别为顶点 a, b 的线段。

在游戏开始之前，小 W 会进行一些操作。每次操作时，他会选中多边形的两个互异顶点，给它们之间连一条线段，并且所连的线段不会与已存的线段重合、相交（只拥有一个公共端点不算作相交）。他会不断重复这个过程，直到无法继续连线，这样得到了状态 S_0 。 S_0 包含的线段为凸多边形的边与小 W 连上的线段，容易发现这些线段将多边形划分为一个个三角形区域。对于其中任意一个三角形，其三个顶点为 $i, j, k (i < j < k)$ ，我们可以给这个三角形一个标号 j ，这样一来每个三角形都被标上了 $2, 3, \dots, n-1$ 中的一个，且没有标号相同的两个三角形。

小 W 定义了一种“旋转”操作：对于当前状态，选定4个顶点 a, b, c, d ，使其满足 $1 \leq a < b < c < d \leq n$ 且它们两两之间共有5条线段—— $(a, b), (b, c), (c, d), (a, d), (a, c)$ ，然后删去线段 (a, c) ，并连上线段 (b, d) 。那么用有序数对 (a, c) 即可唯一表示该次“旋转”。我们称这次旋转为 (a, c) “旋转”。显然每次进行完“旋转”操作后多边形中依然不存在相交的线段。

当小 W 将一个状态作为游戏初始状态展示给小 R 后，游戏开始。游戏过程中，小 R 每次可以对当前的状态进行“旋转”。在进行有限次“旋转”之后，小 R 一定会得到一个状态，此时无法继续进行“旋转”操作，游戏结束。那么将每一次“旋转”所对应的有序数对按操作顺序写下，得到的序列即为该轮游戏的操作方案。

为了加大难度，小 W 以 S_0 为基础，产生了 m 个新状态。其中第 i 个状态 S_i 为对 S_0 进行一次“旋转”操作后得到的状态。你需要帮助小 R 求出分别以 S_0, S_1, \dots, S_m 作为游戏初始状态时，小 R 完成游戏所用的最少“旋转”次数，并根据小 W 的心情，有时还需求出“旋转”次数最少的不同操作方案数。由于方案数可能很大，输出时请对 $1,000,000,007$ 取模。

【程序文件名】

源程序文件名为 polygon.cpp/c/pas。

【输入格式】

输入文件名为 polygon.in。

第一行一个整数 W ，表示小 W 的心情。若 W 为0则只需求出最少的“旋转”次数，若 W 为1则还需求出“旋转”次数最少时的不同操作方案数。

第二行一个正整数 n ，表示凸多边形的边数。

接下来 $n-3$ 行，每行两个正整数 x, y ，表示小 W 在 S_0 中连的一条线段，端点分别为 x, y 。保证该线段不与已存的线段重合或相交。

接下来一行一个整数 m ，表示小 W 以 S_0 为基础产生的新状态个数。

接下来 m 行，每行两个整数。假设其中第 i 行为 a, b ，表示对 S_0 进行 (a, b) “旋转”后得到 S_i 。

【输出格式】

输出文件名为 polygon.out。

输出共 $m+1$ 行。

若 W 为0则每一行输出一个整数，第 $i(i = 1, 2, \dots, m, m + 1)$ 行输出的整数表示 S_{i-1} 作为初始局面的最少“旋转”次数。

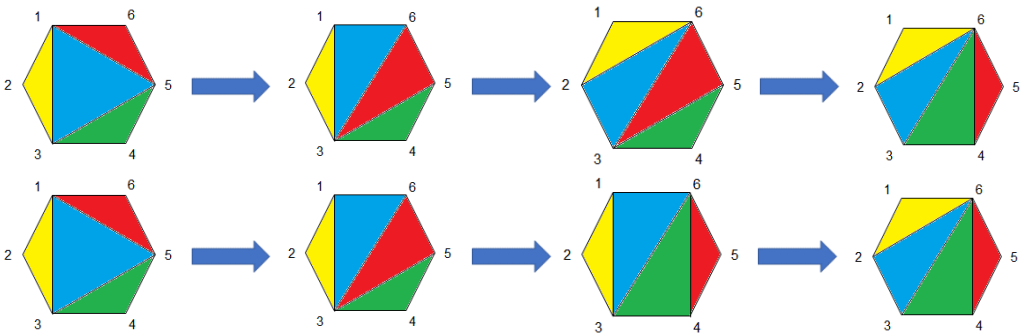
若 W 为1则每一行输出两个整数，第 $i(i = 1, 2, \dots, m, m + 1)$ 行输出的两个整数依次表示 S_{i-1} 作为初始局面的最少“旋转”次数、“旋转”次数最少的不同操作方案数对1,000,000,007取模的结果。

【输入输出样例1】

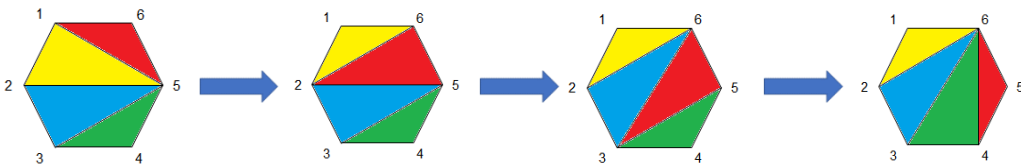
polygon.in	polygon.out
1	3 2
6	3 1
1 3	
1 5	
3 5	
1	
1 3	

【样例解释1】

以 S_0 为初始状态，最少“旋转”次数为3，有2种方案，如下：



以 S_1 为初始状态，最少“旋转”次数为3，有1种方案，如下：



以上图中黄色、蓝色、绿色、红色三角形对应标号分别为2,3,4,5。

【输入输出样例2】

polygon.in	polygon.out
1	8 210
12	7 210
1 10	8 70
1 6	8 105
1 3	8 140
3 6	
3 5	
6 10	
6 8	
8 10	

10 12	
4	
1 10	
1 3	
6 8	
1 6	

【数据范围】

测试点编号 id	W	n	m	测试点编号 id	W	n	m	
1	1	$= id + 8$	$= 0$	11	0	≤ 10000	≤ 100000	
2				12				≤ 1000
3			$\leq n$	13	1		≤ 100000	
4				14				
5		15						
6		16						
7		17						
8		18						
9		≤ 100		19		≤ 100000		
10			$= 0$	20				

对于所有输入数据，保证： $3 \leq n \leq 100000, 0 \leq m \leq 100000$ 。

【编译命令】

对于c++语言： `g++ -o polygon polygon.cpp -lm`

对于c语言： `gcc -o polygon polygon.c -lm`

对于pascal语言： `fpc polygon.pas`