Day 6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 题目名称 | 爱 | 我推的孩子 | 阿库亚 | 露比 |
| 可执行文件名 | love | oshinoko | aqua | ruby |
| 输入文件名 | love.in | oshinoko.in | aqua.in | ruby.in |
| 输出文件名 | love.out | oshinoko.out | aqua.out | ruby.out |
| 每个测试点时限 | 1秒 | 1秒 | 5秒 | 1秒 |
| 内存限制 | 256MB | 256MB | 256MB | 256MB |
| 测试点数目 | 20 | 20 | 20 | 20 |
| 每个测试点分值 | 5 | 5 | 5 | 5 |
| 是否有部分分 | 否 | 否 | 否 | 否 |
| 题目类型 | 传统型 | 传统型 | 传统型 | 传统型 |
| 是否有样例文件 | 否 | 否 | 否 | 否 |
| 是否有附加文件 | 否 | 否 | 否 | 否 |

提交源程序须加后缀

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对于C++ 语言 | love.cpp | oshinoko.cpp | aqua.cpp | ruby.cpp |

编译开关

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 对于C++ 语言 | -O2 -lm | -O2 -lm | -O2 -lm | -O2 -lm |

nodesnodesnodes1.in~nodes10.innodes1.out~nodes10.outnodes1.ans~nodes10.ans

爱(Love)

【问题描述】

我爱你们、这句话绝对不是谎言呢……

星野爱是偶像团体“B小町”毫无疑问的C位。她有一部旧手机和“B小町”成员们秘密通话。

这种算法是要求你用一个 𝑘位的数对一个字符串 𝑠进行加密。加密的算法是我们将这个字符串写下来， 然后将这个 𝑘位的数不断反复写下，然后将对应位相加得到我们的加密串。举个栗子，假设我们用 123对 abcdz进行加密的话，我们首先将它们按照上述的方法写下来：

abcdz

12312

那么我们将对应位加起来，比如a+1=b,z+2=b，就可以得到加密之后的串为：

bdfeb

现在，星野爱清楚想发的串和密码，请问加密串是什么。

【输入格式】

输入文件为***love.in***。

第一行两个整数𝑙,𝑘，代表字符串的长度和数的位数。

第二行一个长度为𝑙的字符串，代表需要被加密的字符串。

第三行一个有𝑘位的数，代表加密所需要用到的数。

【输出格式与部分分】

输出文件为***love .out***。

输出仅一行一个字符串，代表加密后的结果。

【样例1输入】

5 3

abcdz

123

【样例1输出】

bdfeb

【样例2输入】

9 5

hoshinoai

45510

【样例2输出】

ltxiirtfj

【子任务】

对于100%的数据， 1≤𝑙≤100,1≤𝑘≤8，字符串中只会有小写字母，加密

用到的数中只会出现 0−9中的字符。

我推的孩子(Oshinoko)

【问题描述】

考虑一个包含 个节点的树。在这棵树上，我推的孩子星野阿库亚（阿库亚）和星野露比（露比），正在进行一场游戏。一开始，阿库亚位于节点 ，露比位于节点 。然后，他们轮流移动，阿库亚先开始。每一轮中，位于节点 的角色必须选择一个相邻节点 并移动到 。需要注意的是，角色不能移动到另一位角色当前的位置。在完成这个移动后，节点 将变为无效，意味着在接下来的回合中，两位角色都不能移动到该节点。

如果某位角色无法进行有效的移动，他或她将输掉游戏。

请确定星野阿库亚是否有策略可以确保她赢得游戏。

【输入格式】

输入文件为***oshinoko.in***。

输入包含多个测试用例。第一行包含一个整数 - 测试用例的数量。接下来是各个测试用例的描述。

每个测试用例的第一行包含一个整数 。

第二行包含两个整数和。

接下来的行中，每行包含两个整数和，表示和之间存在一条边。。

【输出格式与部分分】

输出文件为 ***oshinoko .out***。

对于每个测试用例，输出一个整数 - 如果星野阿库亚有策略可以确保她赢得游戏，则输出 ，否则输出 。。

【样例1输入】

3

5

2 3

2 5

5 4

5 1

3 4

5

3 5

2 4

1 5

4 3

1 4

5

1 2

3 4

4 2

5 1

4 5

【样例1输出】

1

1

0

【子任务】

所有测试数据的范围和特点如下:

对于 的数据有

对于 的数据有

对于的数据有

阿库亚(Aqua)

【问题背景】

星野阿库亚可是凭借真本事进入的阳东高中，所以区区轻重链剖分加线段树自然是不在话下。

这天，星野阿库亚遇到一道题：

有一棵包含个顶点的根树，根位于顶点，每个顶点都有零个或一个与其相连的重子节点，这些连接构成了一个重链（heavy chain）。一组最大连接的重链构成了一个重链。单个顶点也可以构成一个重链，因此每个顶点都属于且仅属于一个重链。

现在我们想要构建一个搜索树，以维护原始树的一些信息。对于每个包含个顶点的重链，我们构建一个深度为的线段树，其中线段树的每个叶子表示重链上的一个顶点，而线段树根的父节点则是重链顶部顶点的父节点。

需要注意，在这个变种的线段树中，每个叶子的深度都相同。

例如，以下是一棵可能的原始树，注意双斜线表示重边。



在这棵树中有个重链，它们分别是：。

以下是上述原始树的搜索树，注意这里的双斜线没有特殊含义，只是为了使搜索树看起来更美观。

给定这样一棵原始树，你的任务是计算搜索树的深度。根节点的深度为 1。

星野阿库亚当然会做，所以他来考考你会不会。

【输入格式】

输入文件为aqua.in。

输入的第一行包含一个整数 ，表示原始树中顶点的数量。

接下来的行包含个整数，表示每个顶点的父节点。注意，表示没有父节点。

接下来的行包含个整数，表示每个顶点的重子节点。注意，表示没有重子节点。

【输出格式】

输出文件为aqua***.out***。

输出行，表示答案。

【样例1输入】

10

0 1 2 3 4 5 1 7 8 4

2 3 4 5 0 0 8 9 0 0

【样例1输出】

7

【子任务】

所有测试数据的范围和特点如下:

对于的数据有

对于的数据有

对于的数据有

露比(Ruby)

【问题背景】

星野露比现在有 个数组。

为了方便起见，每个数组 长度为，下标从 开始。（直观来说就是第⼀维下标从 开始，第⼆维下标从 开始。）

其中 时时刻刻是 的前缀和。前缀和就是 且。

比如，那么。

此时如果我们修改，得到新的：

你需要支持 个操作。

* 修改操作：输⼊ ，执⾏。
* 询问操作：输⼊ ，返回 的值。

由于结果可能很大，你只需要输出询问的值对 取模的结果。

【输入格式】

输入文件为ruby.in。

第一行三个整数，分别表示数组长度，操作次数，前缀和次数。

接下来行，每行一个操作。

如果第一个数字是，接下来会有个数字表示修改，。

如果第一个数字是1，接下来会有个数字表示询问。

【输出格式】

输出文件为ruby***.out***。

对于每个询问操作，输出询问的值对 取模的结果。

【样例1输入】

4 11 3

0 1 1

0 3 1

1 1

1 2

1 3

1 4

0 3 1

1 1

1 2

1 3

1 4

【样例1输出】

1

3

7

13

1

3

8

16

【子任务】

所有测试数据的范围和特点如下:

对于的数据，满足。

对于的数据，满足。

对于其中的数据，满足。

对于其中的数据，满足。

对于其中的数据，满足。