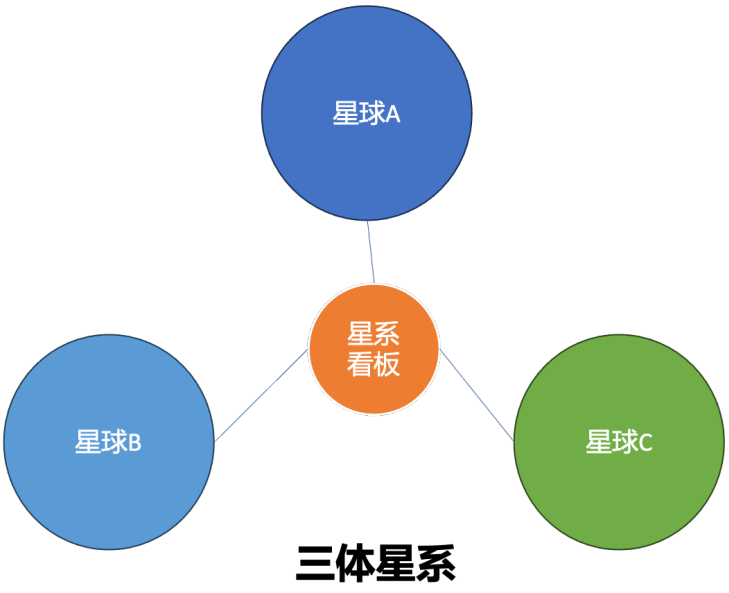
题目：三体星系

Contents

SE@SJTU 2024

1 简介与背景

这里是宇宙中的一个三体星系，包含多个三体星球（在本题目中，始终只需要考虑 3 个星球的情况），例如星球 A、星球 B、星球 C。每个星球内有三体人，他们可以通过脑电波直接交流。但是星球之间的三体人不能直接通过脑电波交流，而需要通过一个全局的星系看板进行信息交流。



为了帮助三体星系构建一个正确高效的信息交流方式，你需要写一段程序完成以下指定的任务。

2 题目要求

2.1 [步骤 1] 加载三体星系信息

函数名：本步骤需要实现为一个名为 `LoadInfo` 的函数。

你需要从标准输入中读取两行文本。第一行文本为“三体星球信息”的路径（路径不含空格）；第二行文本为“三体星人操作信息”的路径（路径不含空格）；

“三体星球信息”的文件中，记录了 3 个三体星球（A、B、C）上的三体人的信息。在开头一行，会包括三个数字，a、b、c，以空格隔开，分别表示 3 个星球的信息的数量。后面会依次给出 a 行、b 行和 c 行的三体人信息。每行包含一个编号（32bit 的整型，表示一个三体人）和一个值（长度不大于 100 的字符串），表示一个三体人的信息。

“三体星人操作信息”的文件中，记录了三个星球上的三体人对于信息的读和写操作。文件中包括若干行，每一行为 3 个或 4 个元素，用空格隔开。

当为三个元素时，表示是一个读取信息的操作，为：星球编号（A 或 B 或 C）三体人 ID（32 位）READ，例如：

A 1002 READ

表示星球 A 上的三体人对标号为 1002 的信息进行了一个读取操作（注意，这里具体是哪一个三体人读取的信息不重要，我们只在意是哪个星球上的人发起的读操作）。

当为四个元素时，表示是一个写信息的操作，为：星球编号（A 或 B 或 C）三体人 ID（32 位）WRITE 值（长度不大于 100 的字符串），例如：

B 1001 WRITE yesSe2024

表示星球 B 上的三体人对标号为 1001 的信息进行了一个写操作（将其值更新为 yesSe2024）。

标准输出：

在本步骤中，你需要按上述方式读取数据，并在标准输出中打印多行信息。

其中，第一行包括四个数字：第一个数字为“三体星球信息”文件中，三个三体星球的信息总数。第二到第四个数字分别为“三体星人操作信息”文件中 A、B、C 三个星球的读操作的数量。数字之间以一个空格隔开。

第二行开始，对于被 Write 修改过的信息，需要打印出该信息的状态变化，打印格式如下：

```
<标号>: (<操作星球的编号>)<新值> -> (<操作星球的编号>)<新值>
```

考虑下面的例子，info.txt 的内容包含：

```
1000 se2024
```

ops.txt 中包含：

```
B 1000 WRITE aaa
A 1000 WRITE bbb
A 1001 WRITE ccc
```

那么，最终的输出应该为：

```
1000: (B)aaa -> (A)bbb
1001: (A)ccc
```

注意：

1. **不需要打印初始状态**，即不需要打印出 info.txt 中的状态
2. 状态变化的顺序是 ops.txt 中的执行顺序，打印的顺序为标号升序

2.2 [步骤 2] 构建星系信息库

在该星系中，应该共需要 4 个信息库，分别表示星球 A、B、C 上的三体人信息以及全局看板上的三体人信息。设计一个数据结构，用于存储和管理三体人的信息。要求该数据结构能够支持：(1) 高效地插入和删除一个三体人的信息；(2) 查询三体人的信息。初始时，三个星球上的信息在步骤 1 的“三体星球信息”中记录，全局看板为空。

函数名：本步骤需要实现为一个名为 `BuildInfoLib` 的函数。

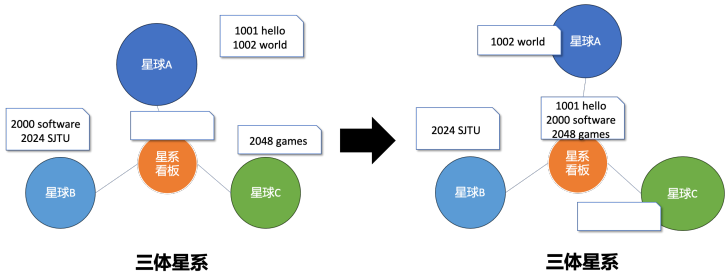
在函数中，基于步骤 1 中加载的信息，使用你的数据结构，构造信息库。并且在完成基于初始数据的信息库构造后，将星球 A、B、C 上面 ID 最小的信息，移动到全局看板中（如果某个星球当前没有信息，则不移动）。注意 (1)，一个 ID 的信息只能出现在 4 个信息库中的一个地方，因此，这 3 个被移出的信息需要从原始的 A、B、C 中删除。注意 (2)，你可以将部分构建信息库的逻辑直接实现在步骤 1 的函数中。

单个三体人的信息可以参考下面的结构：

```cpp
struct ThreeBodyInfo {
    std::string person_info;
    // 按照需求添加其他字段，仅供参考

    ThreeBodyInfo(const std::string& info) : person_info(info) {}
};
```

下图给出了一个移动 3 个 ID 最小信息前后的信息库的状态的示意图：



标准输出：

在本步骤中，你需要在标准输出中打印出经过处理后的全局看板中的三体人信息。每一行是一个 ID 和对应的值（字符串），按照 ID 的升序进行输出。

注意：

1. 全局看板在经过你的简单处理后，应该有不大于 3 个的三体人信息。
2. 请注意在移动过程中，保证全局看板和 3 个星球的信息之间不存在重复的 ID（初始化给的文件中不会出现一个 ID 同时出现在 2 个星球的情况）

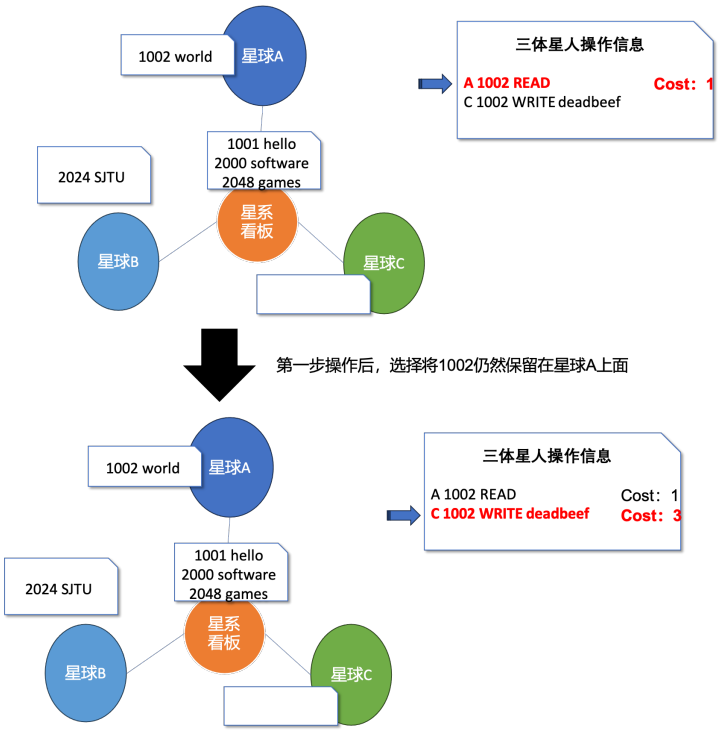
2.3 [步骤 3] 快速信息同步

函数名：本步骤需要实现为一个名为 `FastSync` 的函数。

三个星球上的任意一个三体人都可以读取或者修改任意三体人的信息。读取和修改的路径可以分为三种情况：

1. 假设 A 星球上的人对三体人 x 的数据进行读取或修改，如果这个操作可以在 A 的信息库完成，则它的开销是 1；
2. 如果该操作需要在全局看板完成，则它的开销是 2；
3. 如果这个操作需要在其他星球（比如 B）的信息库上完成，则它的开销是 3。

在完成一次读写操作后，该星球上的三体人可以选择将该数据留在当前的星球（比如 A 的信息库）中，或者留在全局看板（但是不能将数据放在其他星球上），这里不涉及开销。



例如，对于步骤 2 完成后的状态，假设现在有一个操作是：

A 1002 READ

那么，这一个操作的开销是：1（因为当前 1002 对应的信息就在星球 A 上面）。当完成这个操作后，可以选择将 1002 对应的信息仍然保留在 A，或者放到全局看板上（但是不能放在 B 或者 C 上面）。

注意，这里不同的选择可能对后续的开销有显著影响。我们假设下一个操作是：

C 1002 WRITE deadbeef

那么，如果在上一步时，选择了将 1002 的信息仍然保留在 A 上面，此时的开销是 3 (如上图所示)。如果上一步选择将 1002 的信息放在全局看板上，那么此时的开销是 2。

假设已知三个星球上的三体人的所有读写操作，在步骤 1 中读取的“三体星人操作信息”文件中。另外，已知初始时的三体人的信息分布，为步骤二结束后的状态（即最小的 3 个 ID 的信息在全局看板中，其他信息在三个星球上）。

根据上述这些信息，请实现目标函数，使得最终完成这些操作的开销最小。

标准输出：

在本步骤中，你需要在标准输出中打印 2 部分信息：

首先，请打印出每一步操作的开销，和该次操作的选择（保留在本星球写 Local，放在全局写 Global）.

例如，假设“三体星人操作信息”文件包括 4 个操作，你应该输出类似下面的内容（编号，开销和对数据的位置的选择，空格隔开）：

1 1 Local

2 3 Global

3 1 Global

4 2 Local

其次，请打印出最终的你的函数计算出的最小开销，用“Mini cost is:”开头，例如

Mini cost is: 7

注意：不会存在访问一个不存在任何星球的数据的情况。

2.4 [步骤 4] 应对星系看板的崩溃

函数名：本步骤需要实现为一个名为 `DataReOrg` 的函数。

全局星系看板由于被多个星球同时共用，存在一定的概率会崩溃，在星系看板崩溃期间，三体星系仍然允许所有三体人的信息在多个星球间共享。

此时，三个星球上的一个三体人信息读取和修改的路径可以分为 2 种情况：

1. 假设 A 星球上的人对三体人 x 的数据进行读取或修改，如果这个操作可以在 A 的信息库完成，则它的开销是 1；
2. 如果该操作需要在 B 和 C 的信息库中完成，则它的开销是 3。

注意：由于缺乏全局的星系看板，此时数据信息在读写之后，无法进行移动，仍然只能保留在原地。

为了避免在星系看板崩溃期间，三体人读写数据带来太大的开销，星系看板引入了一个瞬时重排的能力（即本步骤要实现的函数），会在崩溃前，将三体星系中的所有数据（包括 3 个星球和全局看板中的）重新分布到三个星球上（不能放在全局看板中，因为看板即将崩溃）。

假设已知：

1. 初始的数据信息和值为步骤 2 之后的状态（注意这里不是步骤 3 之后的状态）；
2. 后续的操作是“三体星人操作信息”文件中的操作。

请设计并实现该函数，使得重新布局后的数据分布，能够使得“三体星人操作信息”文件中的操作的开销最小。

例如，初始的时候，3 个星球上的数据如下：

A: 1002

B: 1003

C: 1004

操作信息的文件中的操作如下：

A 1004 READ

B 1002 READ

那么一个重新布局：

A: 1004

B: 1002

C: 1003

能够使得操作信息的操作开销最小，为：2（两个操作此时均只需要 1 的开销）

标准输出：

在本步骤中，你需要在标准输出中打印 2 部分信息：首先，打印出 3 行，分别表示 3 个星球上你重新分布后的信息的 ID，每行以星球的编号开头，ID 之间以空格隔开。例如：

A: 1004 1002

B: 1005

C: 1003

其次，请打印出最终的你的函数计算出的最小开销，用“Mini cost after reorg is:”开头，例如

Mini cost after reorg is: 5

3 输入输出样例

为了方便大家进行调试，我们在 data 目录中给出了样例（sample）、小规模（small）和大规模（large）数据。

给出的每个测试包括四个文本文件，以 sample 为例：

- sample.stdin.txt 为标准输入的内容
- sample-info.txt sample-ops.txt 为输入文件的内容
- sample.stdout.txt 为正确的标准输出的内容

其中，正确的标准输出内容只有 sample 样例给出。

4 提交要求和考核标准

4.1 编码要求

语言不限，但根据题目选择合适的语言并按照要求进行设计和编码。

4.2 评分标准

设计和实现程序，完成前述功能。如果不能完成全部程序功能，也请不要担心，我们会根据各个方面独立评分。具体标准如下：

步骤	分值
[步骤 1] 加载三体星系信息	15
[步骤 2] 构建星系信息库	20
[步骤 3] 快速信息同步	25
[步骤 4] 应对星系看板的崩溃	20
通过所有测试数据（以 C++ 为基准，运行限时 1 秒）	10
代码规范、命名标准、代码风格、注释和说明等	10

4.3 提交要求

请将程序源代码和说明文档（README 等）使用 7z 格式压缩后命名为“姓名.7z”。说明文档建议覆盖编程语言、运行环境、思路（可选）等内容，如果出现程序运行错误，会结合代码和文档说明进行评分。将其 放置在 **U 盘中**，请确保文件完整性。

重要提醒：请在考试全程使用录屏软件进行录屏，并将录屏文件放入 U 盘中，命名为“姓名.mp4”（或其他常用视频格式也可）录屏不需要放在 7z 的压缩文件内。

压缩包中应仅包含源代码和指定提交的文件，不包含测试用的输入输出、中间文件或可执行文件。压缩包大小原则上不超过 5MB。