评审结论

1、该程序流程图的功能能够满足基本需求，能够找到图书管理系统数据库中编号为X的图书。

2、流程图所展示的算法简洁明了，代码逻辑比较简单，易于理解和维护。但若程序流程需要更改，可能需要大量重构代码。

3、算法复杂度为$O(n)$，即需要遍历整个数据库才能找到所需的图书编号。该算法复杂度较高，效率较低，不适合用于大规模数据的处理，当实际业务场景中面对大量图书会造成搜索耗时大，影响用户体验。

4、不适合用于指导后续编码工作。

修改建议

2.1 关于算法查询速度优化

对于输入的数据组A，可以做一些预处理，以便于提高整体的查询效率

2.1.1 二分查找

一种提高算法效率的方法是使用二分查找。该算法的单次查询时间复杂度为$O(logn)$。可以通过将输入数据排序后储存，对其进行二分查找来找到指定的图书编号。这可以通过添加一个"排序"模块和一个"二分查找"模块来实现。在排序模块中，可以使用快速排序、归并$O(nlogn)$排序等排序算法对数组进行排序，在用户查询时调用二分查找模块。缺点是难以更新维护数据，当添加、删除、修改时都需要重构整个有序数据。

2.1.2 哈希查找

也可以使用哈希查找。哈希查找的时间复杂度通常为$O(1)$。哈希查找的基本思路是将数组中的元素映射到一个哈希表中，然后在哈希表中查找指定的元素。这可以通过添加一个"哈希表"模块和一个"哈希查找"模块来实现。在哈希表模块中，可以选择适当的哈希函数来将数组中的元素映射到哈希表中。在哈希查找模块中，可以根据用户输入的待查询的图书编号X，计算出其哈希值，然后在哈希表中查找该哈希值对应的元。对于哈希函数应当尽量避免冲突，防止某一个哈希值对应元素过多而影响查询效率。

2.1.3 建立数据结构

另一种提高查询效率同时也有利于维护的方法是使用相关数据结构，如B树，以便快速查找指定的图书。B树是一种常用的平衡查找树，其特点是可以在$O(logn)$级别的时间复杂度内完成插入、删除和查找操作。这可以通过添加有关B树的添加、删除以及查询模块模块来实现。对于大规模数据的处理，对于大规模数据的处理。并且其优秀高效的可扩展性，可维护性也更贴合实际使用场景。

2.2 关于编码方式修改

该流程中所展示的过程对于编码几乎没有指导作用，可以按照以下模式修改：

```
图书类型编号-出版商编号-出版时间-图书编号-相同图书序号
```

图书编码更加标准化，同时也包含更多基本信息。数据储存时也可以按照这个编号建立多级索引，进一步提高查询维护的效率以及便利性。

2.3 内容反馈不够完善

未找到时没有反馈的过程，仅仅返回$A(i)$信息较少，可以进一步完善。

2020302131051

冯烨聪