**文件元数据查找优化**

问题背景

文件系统中，使用目录树结构存储文件元数据。系统可以根据目录树查找到你请求的文件的元数据（你可以在Linux中使用stat命令查看文件元数据），进而对文件进行读写操作。

在传统的文件检索中，用户给定文件名，系统在目录树上执行检索。这种方式在现有的基于目录树的文件系统上表现较好。但现有人工智能等应用对文件系统提出了新的查找需求，它们往往需要查找一类元数据相同的文件集合。例如，

* + 查询所有文件后缀名为.png的文件
  + 查询修改时间在xxx之后的所有文件
  + 查询文件大小超过200KB的所有文件

要执行这类查询，你可以尝试使用Linux上的find的命令。但find命令需要遍历整颗目录树以获取查询结果，导致性能很差。

问题描述

为了加速海量文件元数据的查询，我们可以在目录树之外再建立一套索引，专门优化元数据的查询。倒排索引比较适合这一任务。你可以为每一个文件元数据建立一条倒排链，查询时就可以快速获得元数据对应的文件id（或文件名）。一个倒排索引的例子如下：

**| 元数据 | 文件id (文件名)|**

**|-------------|-----------------|**

**| jpg(后缀名) | 1001, 1002, 1008, 1010 |**

**| png(后缀名) | 1003, 1004, 1006|**

**| 211KB(文件大小) | 1001, 1004, 1010 |**

**| 2020/06/07(文件创建时间) | 1010, 1008, 1005 |**

**| user1(文件所有者) | 1001, 1004, 1006, 1008, 1010 |**

利用倒排索引，你可以快速找到文件元数据。例如查询所有后缀名为jpg的文件时，不需要遍历整个目录树，只需要查找jpg对应的一条倒排链，这极大减少了查询时间。

考核要求

你需要实现上述的目录树和倒排索引。简化起见，你不需要在内核文件系统上实现，只需要进行一些模拟。具体的，你可以将目录树和倒排索引都放在内存中，自己设计目录树的存储结构和倒排索引的存储结构，建立目录树和倒排索引，使用自己生成的元数据，对比使用倒排索引前后的查询效率。

为了更真实的模拟文件系统，你设计的文件元数据信息必须包括文件名、文件后缀名、文件大小、文件所有者。

**基本要求：**

在内存中模拟实现目录树和用于加速元数据查询的倒排索引，测试倒排索引的查询加速效果。这一实现可以是单线程的，只支持简单的写入和元数据查询的。

**进阶要求（选做）：**

1. 在多线程读写的情况下，如何保证索引的并发正确性。注意，高并发读写时，你不仅要保证单一索引的并发正确性，还要保证索引之间的正确性。例如，当目录树中文件元数据修改时（例如修改了后缀名），倒排索引中的相应数据也应当修改。
2. 倒排索引虽然改善了元数据查询，但索引自身引入了新的存储空间开销，你你可以思考如何改善空间开销，例如考虑倒排索引的压缩等。

考核提交

你需要提交实现的源代码，以及相应的文档。文档应按照论文格式写作，描述该工作的具体设计、评估结果。你需要尽量全面的评估你的实现，包括各方面的性能对比、实现开销等。本工作对编程语言不做要求。