

**2021** 级

《大数据存储系统与管理》课程

**课 程 报 告**

**姓 名 汪鑫**

**学 号 U202115339**

**班 号 CS2101**

**日 期 2024.04.22**

**目 录**

[一、选题 1](#_Toc164673454)

[二、实验理论 1](#_Toc164673455)

[三、算法设计与实现 2](#_Toc164673456)

[四、测试与分析 3](#_Toc164673457)

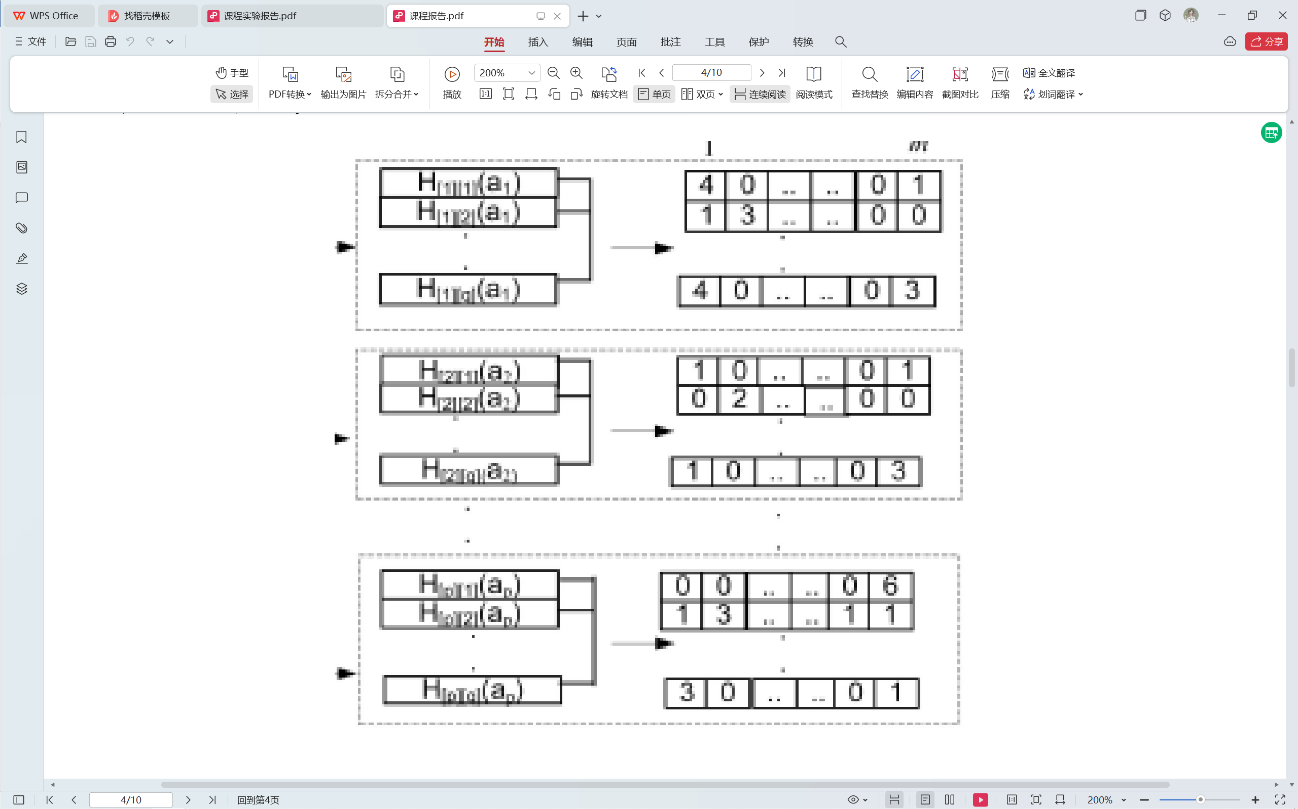
[五、总结 3](#_Toc164673458)

[参考文献 4](#_Toc164673459)

# 一、选题

选题1：基于Bloom Filter的设计。

内容：基于Bloom Filter的多维数据属性表示和索引。



Bloom Filter是一种用于快速检查一个元素是否属于一个集合的概率型数据结构。它具有高效的查询速度和占用较少内存的特点，因此在大数据集的处理中得到了广泛应用。本实验旨在探讨基于Bloom Filter的多维数据属性表示和索引，即将多维数据映射到Bloom Filter中，并利用其进行快速的检索。

# 二、实验理论

首先，选择一个包含多维属性的数据集作为实验对象。假设有一个包含N条记录的数据集，每条记录包含M个属性。为了表示这些多维属性，将采用将每个属性映射到Bloom Filter中的方法。则需要M个Bloom Filter。

针对每个属性，创建一个独立的Bloom Filter。在构建Bloom Filter时，需要确定合适的哈希函数数量和位数组大小，以保证较低的误判率和较小的内存占用。

对于给定的查询，将查询条件中的多维属性映射到各自的Bloom Filter中进行检查。如果所有属性的Bloom Filter均返回True，则认为该记录可能存在于查询结果中。

除此之外，还应该认识到多维数据具有整体性，上述的方法只是分散的检测了每一个属性而并没有考虑到整体性，可能会导致错误。例如在二维数据中有A{1,3},

B{2,4},此时我们查询C{1,2},若没考虑到整体性，则会认为C存在，产生了误判，因此需要在上述的多维Bloom Filter的前提下，在使用一个Bloom Filter来存储所有属性的整体哈希值。如此可以减少误判。

# 三、算法设计与实现

使用C++来实现具体的代码。

首先定义Bloom Filter 的类。存储位数

组使用 C++标准库中的 bitset，计算哈希值使用 C++标准库中提供的 std::hash。如下图3.1所示。

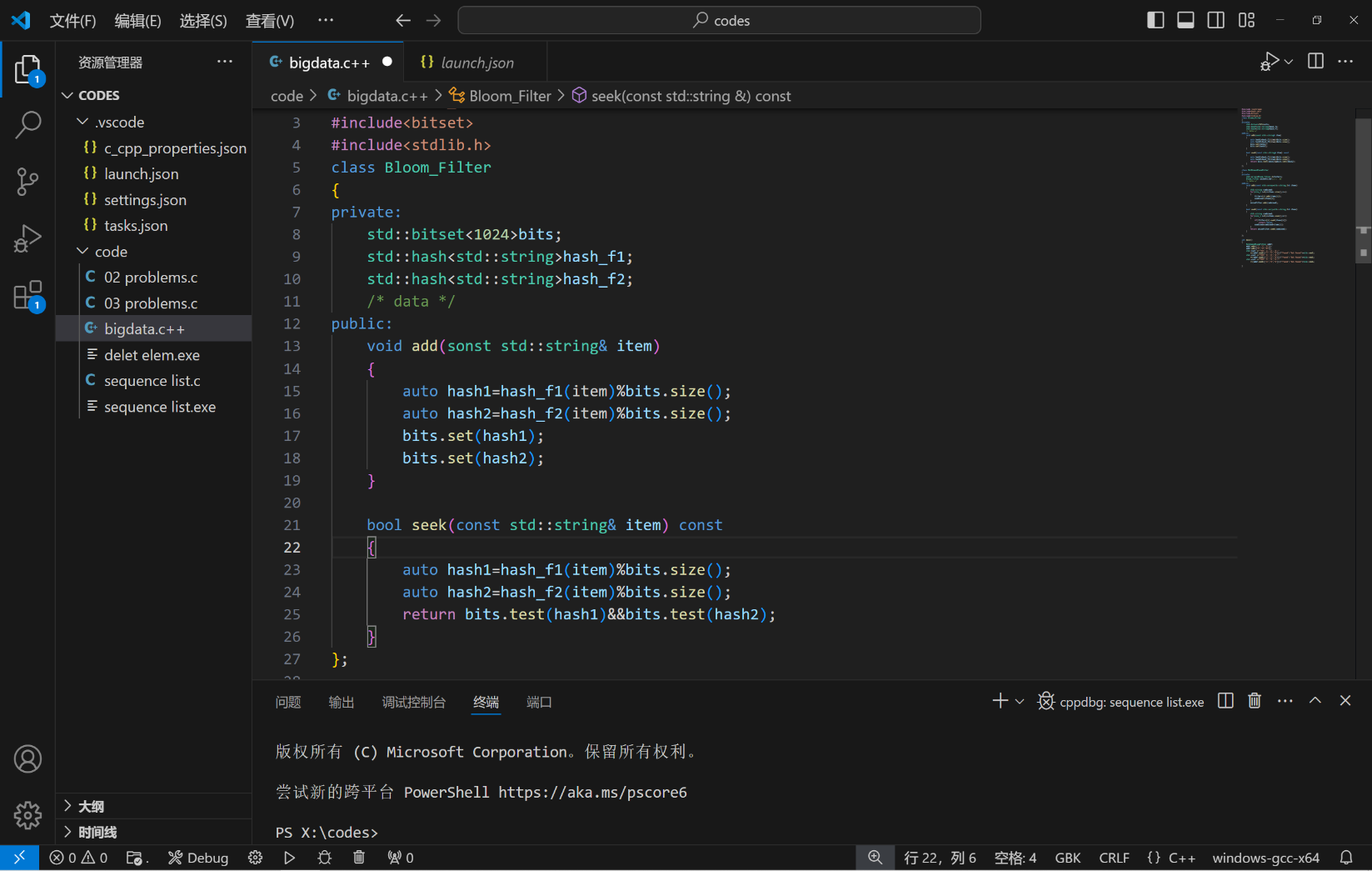


图3.1

为了处理多维数据，需要创建一个包含多个 Bloom Filter 的结构，每个维度都有一个对应的 Bloom Filter。最后再使用一个Bloom Filter用于存储属性的联合值。如下图3.2所示。

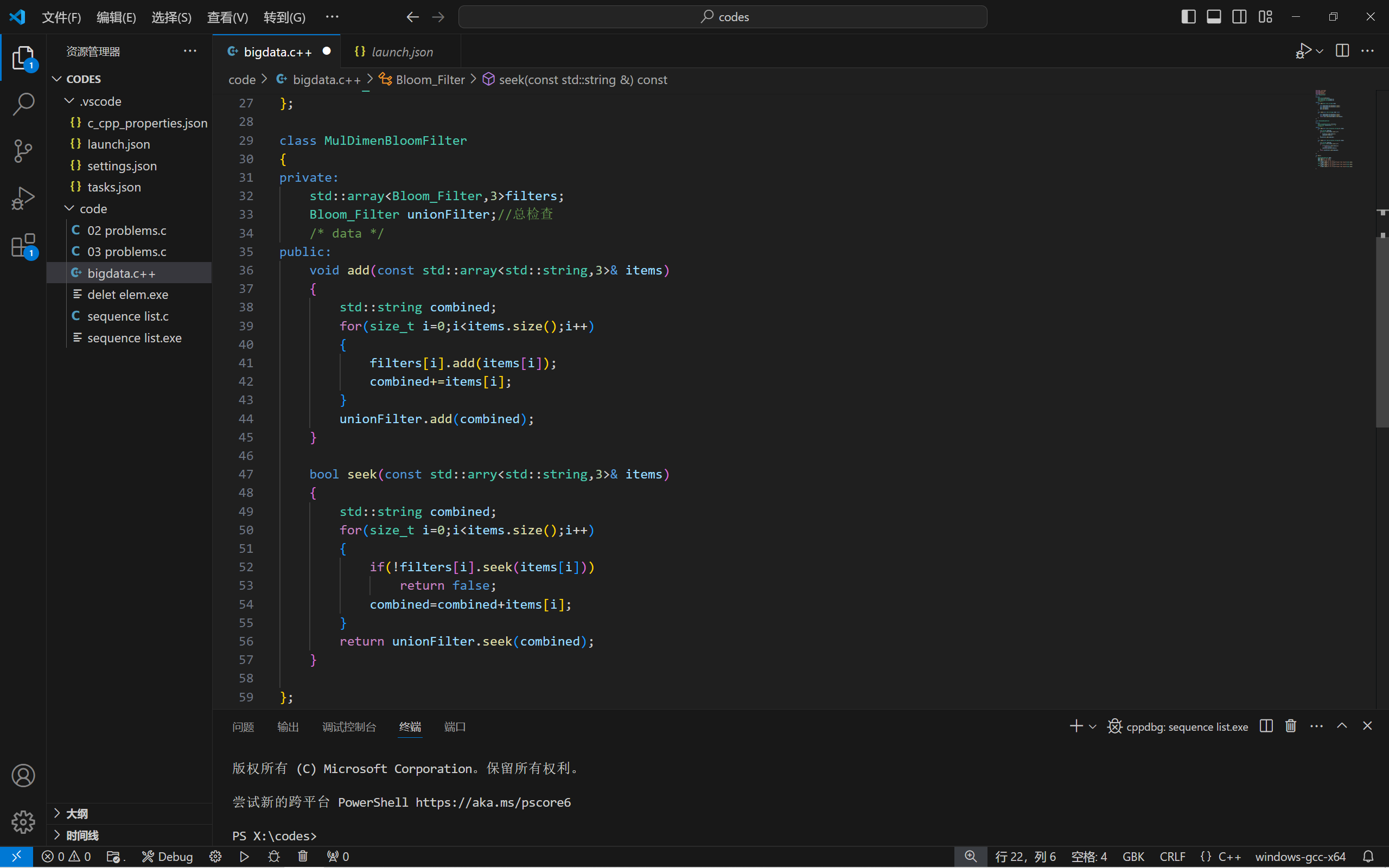


图3.2

简单定义一个主函数来测试上述功能，如下图3.3所示。

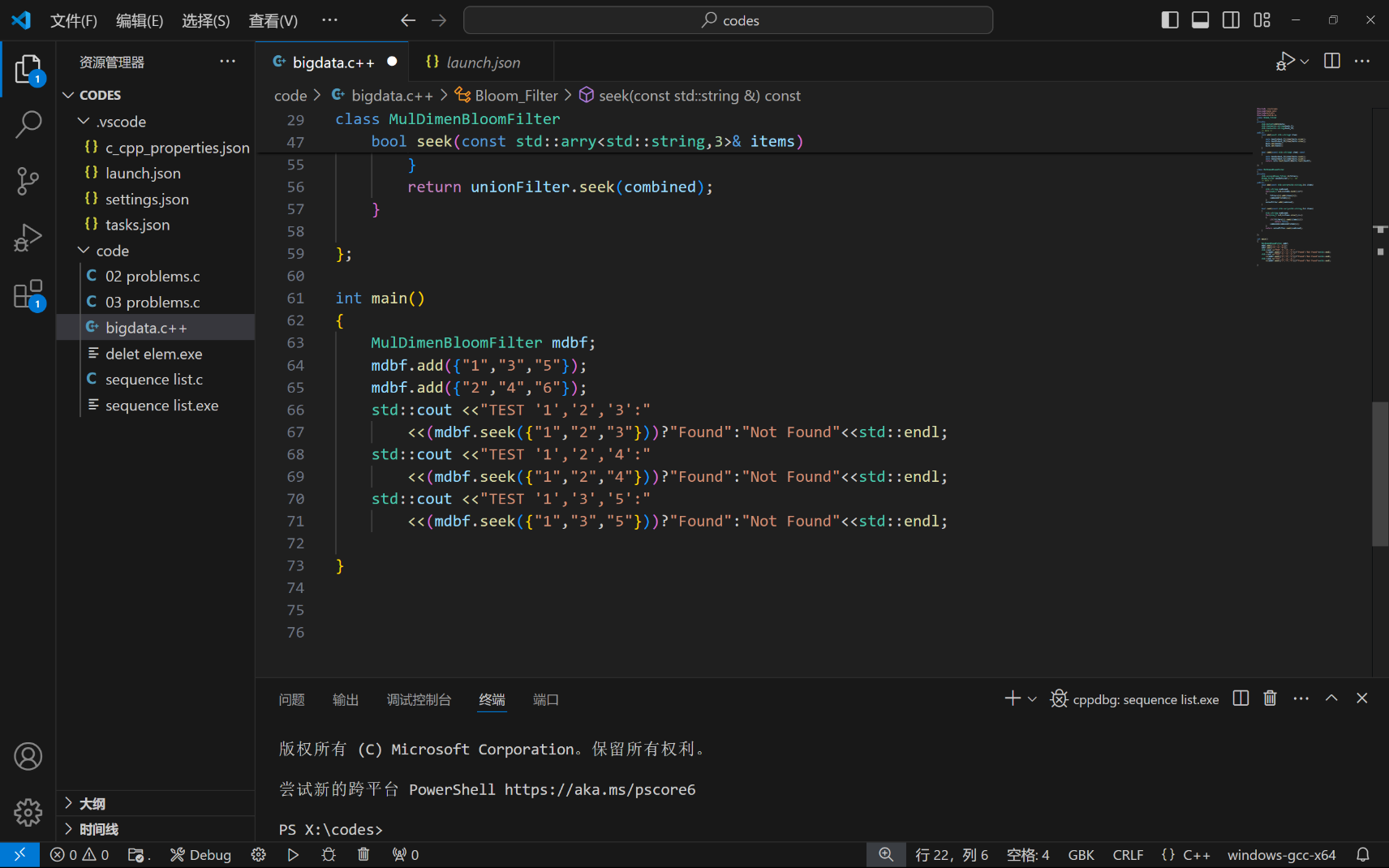


图3.3

# 四、测试与分析

测试集所得到的结果如下图4.1所示。

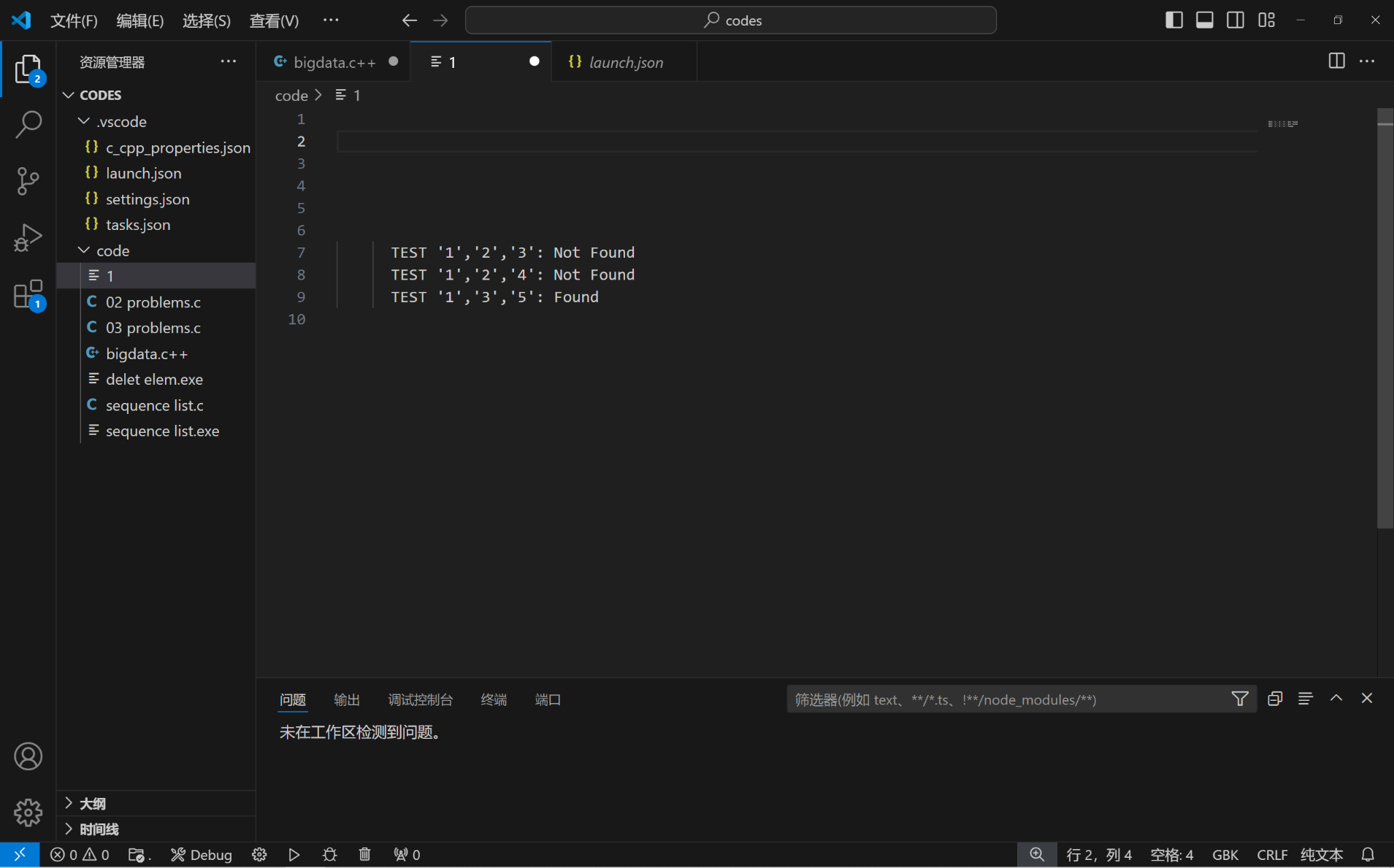


图4.1

正确的得到了相应的结果，由于增加了一个额外的的Bloom Filter，空间开销也会相应的变大。设位数组的长度为L，则所花费的空间开销为(M+1)\*L。另外经查阅资料，所花费的时间与所选取的哈希函数的选择、哈希函数的个数、错误率的要求等也有密不可分的关系。

# 五、总结

基于Bloom Filter的多维数据属性表示和索引是一种高效的数据结构设计，旨在解决大规模多维数据集的快速检索和查询问题。通过将数据对象的多维属性映射到Bloom Filter中，并利用其快速的成员检测特性，我们可以实现快速的数据对象定位和查询操作。

# 参考文献

• F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese, “Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines,” Proc. ACM SIGCOMM, 2006.

• Y. Zhu and H. Jiang, “False Rate Analysis of Bloom Filter Replicas in Distributed Systems,” Proc. Int’l Conf. Parallel Processing (ICPP ’06), pp. 255-262, 2006.

• S. Dharmapurikar, P. Krishnamurthy, and D.E. Taylor, “Longest Prefix

Matching Using Bloom Filters,” Proc. ACM SIGCOMM, pp. 201-212,

2003.

• L. Fan, P. Cao, J. Almeida, and A. Broder, “Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol,” IEEE/ACM Trans. Networking, vol. 8, no. 3, pp. 281-293, June 2000.

• B. Xiao and Y. Hua, “Using Parallel Bloom Filters for Multi- Attribute Representation on Network Services,” IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 1, pp. 20-32, Jan. 2010.

• Y. Hua, Y. Zhu, H. Jiang, D. Feng, and L. Tian, “Scalable and Adaptive Metadata Management in Ultra Large-scale File Systems,” Proc. 28th Int’l Conf. Distributed Computing Systems (ICDCS ’08), pp. 403-410, 2008.

• D. Guo, J. Wu, H. Chen, and X. Luo, “Theory and Network Application of Dynamic Bloom Filters,” Proc. IEEE INFOCOM, 2006.

•【程序员都必须会的技术，面试必备【布隆过滤器详解】，Redis缓存穿透解决方案】 https://www.bilibili.com/video/BV1zK4y1h7pA/?share\_source=copy\_web&vd\_source=c962480b9f97695a0c0bed7f17c46218