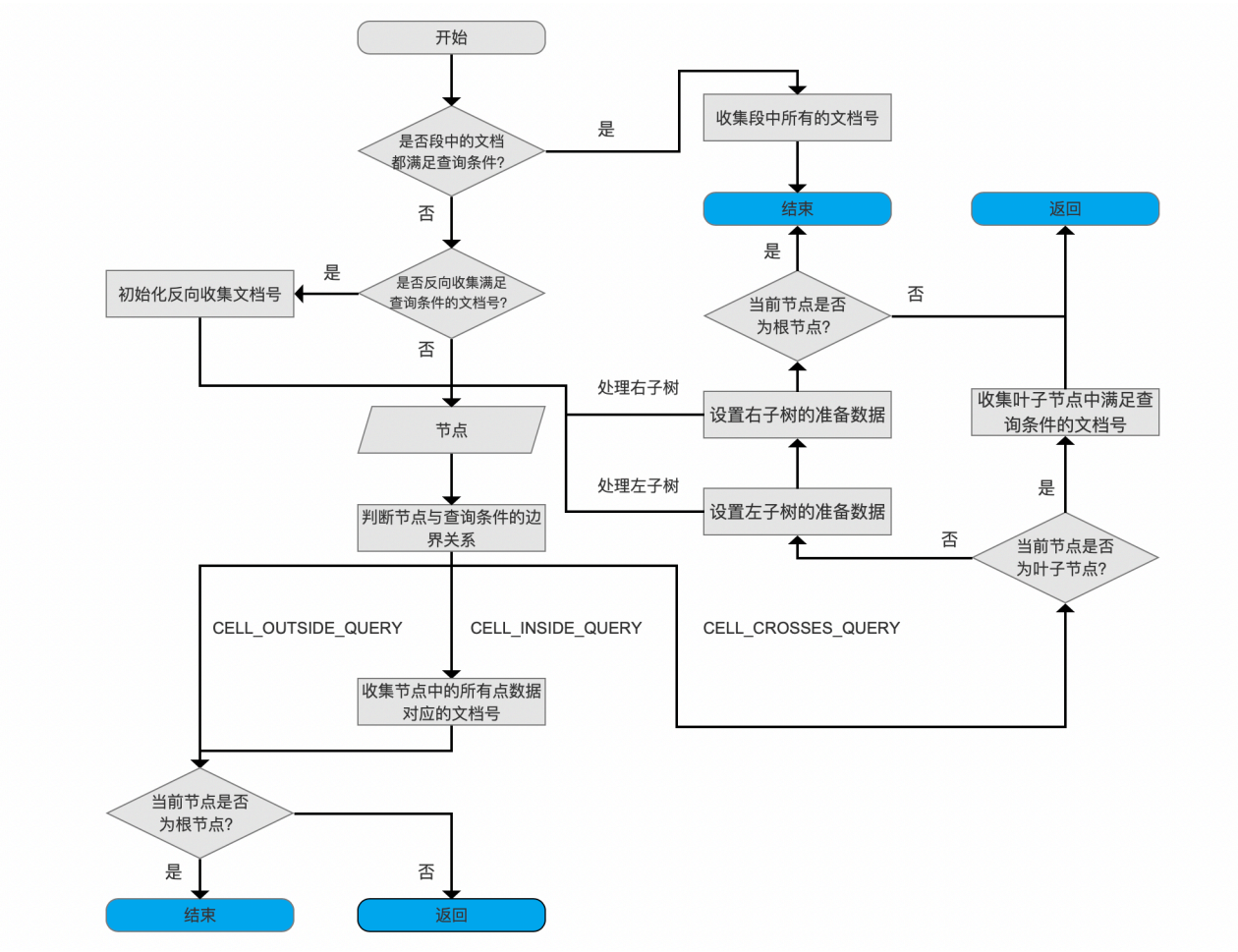


索引文件的读取 (二) (Lucene 8.4.0)

本文承接[索引文件的读取 \(一\)](#)之dim&&dii继续介绍剩余的内容，下面先给出读取索引文件.dim&&dii的流程图：

图1：

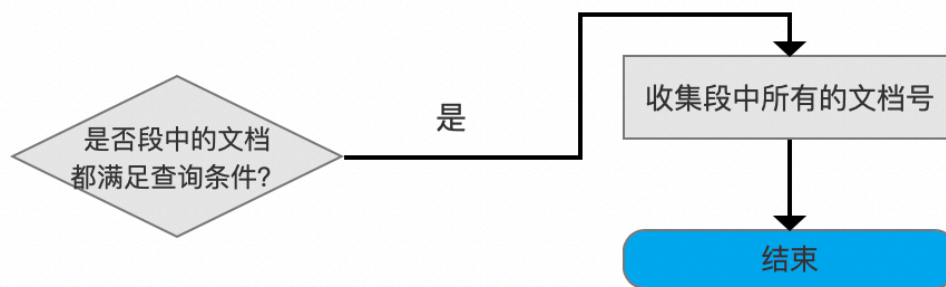


[点击查看大图](#)

读取索引文件.dim&&dii

收集段中所有的文档号

图2：



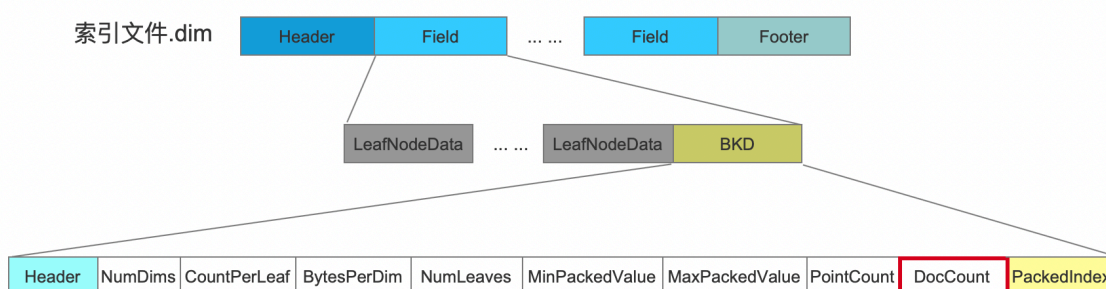
在递归遍历BKD树之前，先判断下是否段中的文档都满足查询条件，如果满足，那么我们就不需要通过读取BKD树的信息来获取文档号，而是根据段中的reader.maxDoc()方法就可以获得满足查询条件的文档号集合。

需要先后同时满足两个条件才能通过reader.maxDoc()获得满足查询条件的文档号集合：

- 条件一：包含当前点数据域的文档数量docCount必须等于段中的文档数量

包含当前点数据域的文档数量可以通过[索引文件.dim](#)的字段获得，如下红框所示所示：

图3：



上图中各个字段的含义见文章[索引文件的读取（一）之dim&&dij](#)。

段中的文档数量通过reader.maxDoc()获取，即读取[索引文件.si](#)中的SegSize字段，如下红框所示所示：

图4：



满足条件一意味着，段中的每篇文档中至少有一个点数据域，这些点数据域对应的域名就是当前流程处理的域。

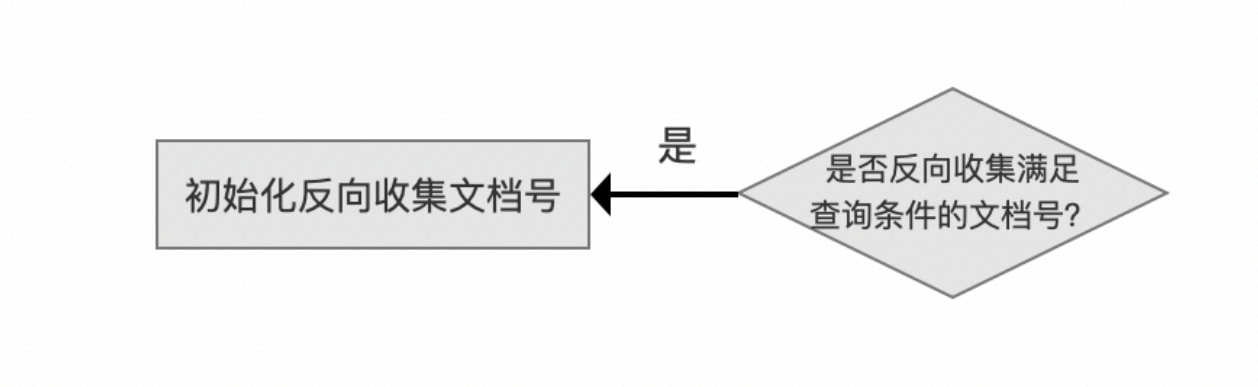
- 条件二：BKD树中点数据的数值范围与查询条件的数值范围的边界关系为CELL_INSIDE_QUERY（见文章[索引文件的读取（一）之dim&&dij](#)的介绍），即查询条件的数值范围包含节点中的所有点数据

如何通过reader.maxDoc()方法获得足查询条件的文档号集合：

由于满足了条件一，那么在后续的Search过程中，只需要从0开始遍历到reader.maxDoc()的值作为文档号传给Collector即可。

反向收集文档号信息

图5：



反向收集文档号意味着存在正向收集文档号，都属于收集文档的方式：

- 正向收集文档号：当某个点数据满足查询条件，那么收集对应的文档号
- 反向收集文档号：先假定段中所有的文档号满足查询条件，在随后的匹配中，当某个点数据不满足查询条件时，那么就移除对应的文档号

为什么要区分收集文档的方式

在回答这个问题前，我们先介绍下满足反向收集文档号的三个条件，这三个条件需要先后同时满足：

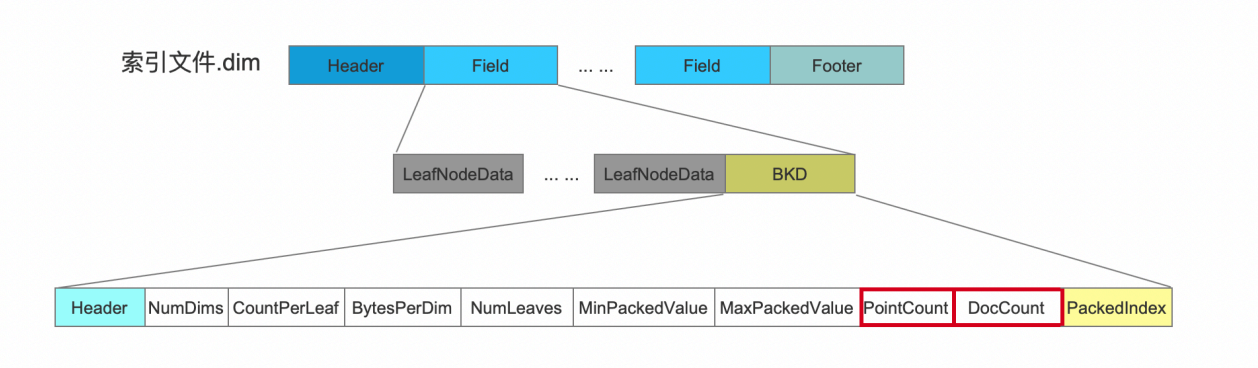
- 条件一：包含当前点数据域的文档数量docCount必须等于段中的文档数量

上文已经介绍，不赘述。

- 条件二：包含当前点数据域的文档数量docCount必须等于BKD树中的点数据数量pointCount

包含当前点数据域的文档数量docCount跟BKD树中的点数据数量pointCount可以通过索引文件dim的字段获得，如下红框所示所示：

图6：



- 条件三：满足查询条件的点数据数量（注意的是这是一个估算值，源码中称之为cost）占段中的文档数量的一半以上（> 50%）

在满足条件一、条件二的前提下，那么通过条件三我们就可以知道，在满足条件三的情况下，不满足查询条件的点数据数量小于满足查询条件的点数据数量，意味着使用反向收集文档号的方式能更快的读取BKD树。

如何估算满足查询条件的点数据数量cost

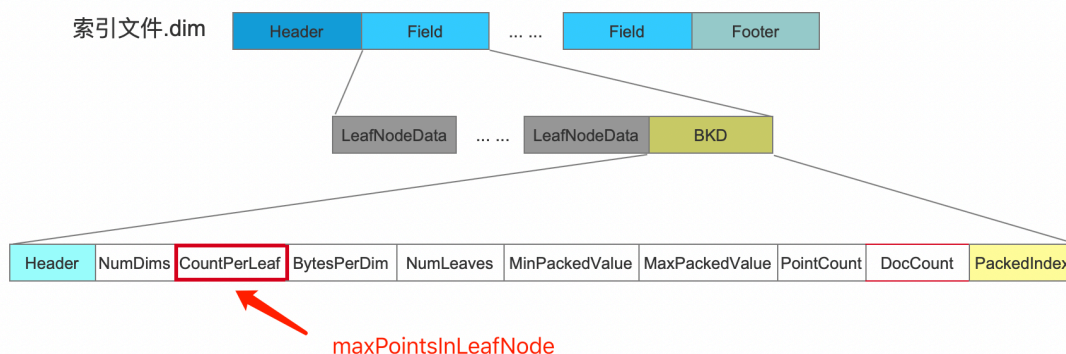
估算满足查询条件的点数据数量cost的过程就是从根节点开始通过深度遍历来计算，详细的过程不详细展开，只介绍计算过程中的一些关键内容。

每次处理一个节点（内部节点（非叶节点）/叶子节点）时，总是用查询条件的数值区间（lowValues、upperValues）跟节点的数值区间（minPackedValue、maxPackedValue）计算边界关系，我们按照边界关系的类型进一步介绍：

- CELL_OUTSIDE_QUERY：查询条件的数值范围跟节点的数值范围没有交集，那么以当前节点为祖先节点的所有叶子节点中的点数据都不满足查询条件，那么就不用再读取当前节点的子节点信息了
- CELL_INSIDE_QUERY：查询条件的数值范围包含节点中的所有点数据，这种情况根据节点的不同类型再做区分：
 - 叶子节点：将maxPointsInLeafNode的值作为当前叶子节点中包含的点数据数量，可见它是一个估算值，最后增量到cost中
 - 内部节点（非叶节点）：计算出以当前内部节点为祖先节点的叶子节点的数量numLeaves，然后通过公式($maxPointsInLeafNode * numLeaves$)计算出的结果作为这些叶子节点中包含的点数据数量总和，可见这同样是一个估算值，最后增量到cost中

maxPointsInLeafNode同样通过读取索引文件.dim获得，如下红框所示：

图7：



- CELL_CROSSES_QUERY：查询条件的数值范围跟节点的数值范围部分重叠（partially overlaps），这种情况同样需要根据节点的不同类型再做区分：
 - 内部节点（非叶节点）：继续递归处理子节点
 - 叶子节点：通过公式($(maxPointsInLeafNode + 1)/2$)的计算结果作为叶子节点中满足查询条件的点数据数量，同样是估算值，最后增量到cost中

从上文的内容可以看出，计算cost可能是个开销昂贵（expensive）的过程，源码中也给出了对应的注释：

```
1 Computing the cost may be expensive, so only do it if necessary
```

结语

基于篇幅，剩余的内容将在下一篇文章中展开。

[点击](#)下载附件