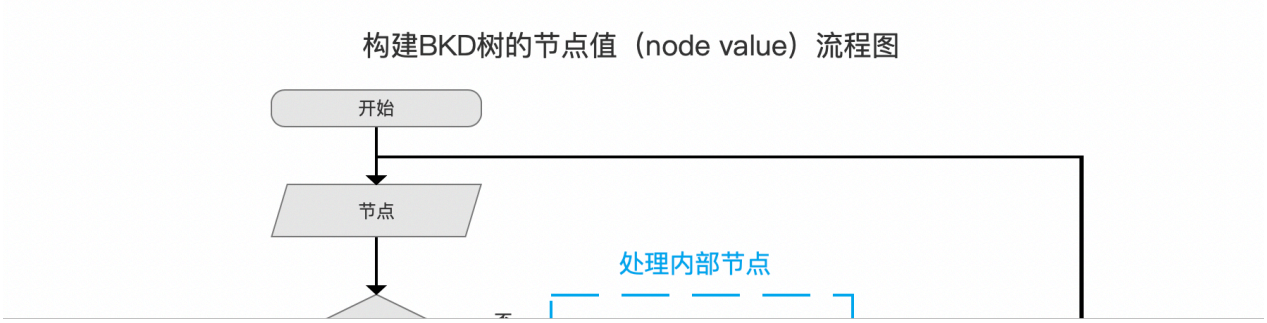
索引文件的生成（十一）（Lucene 8.4.0）

本文承接索引文件的生成（十），继续介绍剩余的内容，为了便于下文的介绍，先给出生成索引文件.dim&&.dii的流程图以及流程点`构建BKD树的节点值（node value）`的流程图：

图1：

图2：



第一次更新parentSplits数组

图3：

parentSplits数组的数组元素数量跟点数据的维度数量相同，下标值为0的数组元素描述的是维度编号（见文章索引文件的生成（十）之dim&&dii）为0的维度值在祖先节点中作为`切分维度`(见文章索引文件的生成（十）之dim&&dii)的次数（累加值）：

图4：

如果某一时刻，parentSplits数组如上述所示，说明当前处理的点数据的维度为3（数组大小），并且维度编号为1（数组下标为1）的维度在祖先节点中已经3次作为切分维度，如果执行完图2的流程点`选出切分维度`后，维度编号1再次作为了切分维度，那么在当前流程点，我们就需要`第一次更新parentSplits数组`，更新后的parentSplits数组如下所示：

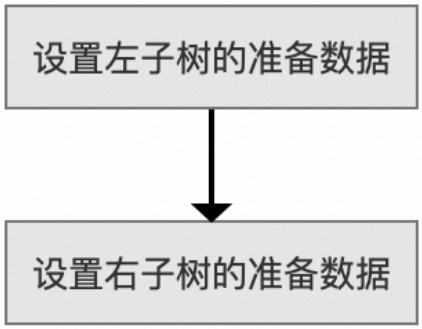
图5：

为什么要更新parentSplits数组

如果当前节点划分后的左右子树还是内部节点，那么左右子树需要根据parentSplits数组提供执行流程点`选出切分维度`的条件判断依据。

设置左子树的准备数据、设置右子树的准备数据

图6：



执行到当前流程点，内部节点的处理工作差不多已经全部完成（除了流程点`第二次更新parentSplits数组`），当前流程点开始为后续的左右子树的处理做一些准备数据，根据当前的节点编号（见文章索引文件的生成（十）之dim&&dii），即将生成的左右子树可能仍然是内部节点，或者是叶子节点。这两种情况对应的准备数据是有差异的，我们会一一介绍。另外根据满二叉树的性质，如果当前节点编号为n，那么左右子树的节点编号必然为2*n、2*n + 1，在文章索引文件的生成（十）之dim&&dii中我们说到，根据节点编号就能判断属于内部节点（非叶节点）还是叶子节点，不赘述。

处理节点（叶子节点或者内部节点）需要的准备数据有好几个，这些准备数据在源码中其实就是 https://github.com/LuXugang/Lucene-7.5.0/blob/master/solr-8.4.0/lucene/core/src/java/org/apache/lucene/util/bkd/BKDWriter.java 类中的build(...)方法的参数：

图7：

```
private void build(int nodeID, int leafNodeOffset,
                   MutablePointValues reader, int from, int to,
                   IndexOutput out,
                   byte[] minPackedValue, byte[] maxPackedValue,
                   int[] parentSplits,
                   byte[] splitPackedValues,
                   long[] leafBlockFPs,
                   int[] spareDocIds) throws IOException {
```

图6中的nodeID参数即节点编号，leafNodeOffset参数即最左叶子节点的节点编号值，其他的参数在下文中会部分介绍。

左右子树为内部节点

如果节点编号nodeId小于最左叶子节点的节点编号leafNodeOffset，那么当前节点是内部节点，对于内部节点的处理，主要关心的参数是minPackedValue、maxPackedValue。

minPackedValue、maxPackedValue

在图2的流程点`选出切分维度`中，我们会使用到minPackedValue、maxPackedValue参与条件判断（见文章索引文件的生成（九）之dim&&dii），我们在处理根节点（内部节点）时，在图1的流程点`MutablePointValues`，，设置了minPackedValue、maxPackedValue的值，为处理根节点做了准备，其初始化的内容见文章索引文件的生成（九）之dim&&dii，而当处理其他内部节点时，minPackedValue、maxPackedValue的值的设置时机点则是在当前流程点。

非根节点的内部节点的minPackedValue、maxPackedValue是如何设置的

我们先回顾下minPackedValue数组的作用是存放每个维度的最小值，maxPackedValue数组的作用是存放每个维度的最大值，比如当前节点中包含的点数据集合如图8所示，那么minPackedValue、maxPackedValue中的内容如下所示：

```
minPackedValue:{2, 1, 2}
maxPackedValue:{9, 9, 9}
```

在图2的执行完流程点`选出切分维度`、`内部节点的排序`之后，我们就获得了当前内部节点划分维度编号n，并且当前节点中点数据集合的每一个点数据按照维度编号n对应的维度值进行了排序（见文章索引文件的生成（十）之dim&&dii），如果我们另排序后的点数据都有一个从0开始递增的序号，假设有N个点数据，那么序号为 0~ (N/2 - 1)的点数据将作为当前节点的左子树的点数据集合，序号为 N/2 ~ N 的点数据将作为当前节点的右子树的点数据集合。

例如当前内部节点有如下的点数据集合，数量为7，并且假设按照维度编号2进行了排序：

图8：

根据图8，当前的minPackedValue、maxPackedValue以及左右子树的minPackedValue、maxPackedValue如下所示：

图9：

从图9可以看出左子树的maxPackedValue跟右子树的minPackedValue的维度编号为2的值被更新为同一个新值，而该值就是序号为(N/2 -1) 的点数据的维度编号为2（父节点的排序维度）对应的维度值，即图8中红色标注的维度值。

对于左右子树来说，如果他们还是内部节点，那么执行图1的`选出切分维度`的流程时就会使用到父节点提供给它们的参数minPackedValue、maxPackedValue。

回看上文中内部节点给左右子树的准备参数minPackedValue、maxPackedValue是有点问题的，比如左子树的点数据集合以及minPackedValue、maxPackedValue如下所示：

图10：

图10中，左子树节点**实际**的minPackedValue、maxPackedValue应该是：

```
minPackedValue:{3, 5, 2}
maxPackedValue:{8, 3, 4}
```

而即将在流程点`选出切分维度`使用的minPackedValue、maxPackedValue是父节点提供的信息，即图10中的值，所以在执行`选出切分维度`的算法时，可能无法能得到最好的一个切分维度。

所以从Lucene 8.4.0版本后，在执行流程点`选出切分维度`会根据一个条件判断是否需要重新计算minPackedValue、maxPackedValue，而不是使用父节点提供的minPackedValue、maxPackedValue，该条件相对简单，直接给出：

```
if (nodeID > 1 && numIndexDims > 2 && Arrays.stream(parentSplits).sum() % SPLITS_BEFORE_EXACT_BOUNDS == 0) {
    // 重新计算minPackedValue、maxPackedValue
        }
```

上述条件中，nodeID为1，即处理根节点的时候不用重新计算，原因是minPackedValue、maxPackedValue总是对的（不明白？见文章索引文件的生成（九）之dim&&dii）；numIndexDims描述的是点数据的维度数量例如图10中，numIndexDims的值为3；Arrays.stream(parentSplits).sum()描述的是每个维度当做切分维度的次数总和，SPLITS_BEFORE_EXACT_BOUNDS的值默认为4。

由于重新计算每个节点的minPackedValue、maxPackedValue的开销是较大的，所以在满足了上述的条件后才会重新计算，下面是源码中的注释：

```
for dimensions > 2 we recompute the bounds for the current inner node to help the algorithm choose best split dimensions. Because it is an expensive operation, the frequency we recompute the bounds is given by SPLITS_BEFORE_EXACT_BOUNDS.
```

上述源码中的algorithm指的就是图2的流程点`选出切分维度`中的划分规则（见文章索引文件的生成（十）之dim&&dii）。

左右子树为叶子节点

在介绍处理叶子节点的时候再展开，留个坑。

第二次更新parentSplits数组

结合图7，第一次更新parentSplits数组时，是为了给当前节点的左右子树在执行流程点`选出切分维度`时提供信息，那么在处理完左右子树后，我们需要恢复parentSplits数组，将parentSplits数组中的信息恢复到执行`第一次更新parentSplits数组`流程前的状态，原因是如果当前节点是左子树，那么当前节点处理结束后，需要处理它的兄弟节点，而兄弟节点（右子树）的parentSplits数组必须是跟左子树一致的。

结语

至此我们介绍完了内部节点的处理流程，在下一篇文章中，我们将继续介绍叶子节点的处理流程，在该流程中，将会把点数据的信息写入到索引文件.dim中。

点击下载附件