索引文件的生成（八）（Lucene 8.4.0）

在前面的文章中，我们介绍了在Lucene7.5.0中索引文件.dim&&.dii的数据结构，从本篇文章开始介绍其生成索引文件.dim&&.dii的内容，注意的是，由于是基于Lucene8.4.0来描述其生成过程，故如果出现跟Lucene7.5.0中不一致的地方会另外指出，最后建议先阅读下文章Bkd-Tree简单的了解下Lucene中点数据的使用。

在文章索引文件的生成（一）之doc&&pay&&pos中，简单的介绍了生成索引文件.dim&&.dii的时机点，为了能更好的理解其生成过程，会首先介绍下在生成索引文件之前，Lucene是如何收集每篇文档的点数据信息（Point Value），随后在flush阶段，会根据收集到的信息生成索引文件.dim&&.dii。

收集文档的点数据信息

在源码中，通过PointValuesWriter对象来实现文档的点数据的收集，并且具有相同域名的点数据使用同一个PointValuesWriter对象来收集，例如下图中添加三篇文档：

图1：

```
46          Document doc;
47          // 文档0
48          doc = new Document();
49          doc.add(new IntPoint( name: "content",  ...point: -3, 5));
50          doc.add(new IntPoint( name: "content",  ...point: -5, 5));
51          doc.add(new IntPoint( name: "title",  ...point: 2, 5));
52          indexWriter.addDocument(doc);
53          // 文档1
54          doc = new Document();
55          doc.add(new IntPoint( name: "content",  ...point: 10, 55));
56          doc.add(new IntPoint( name: "title",  ...point: 3, 55));
57          indexWriter.addDocument(doc);
58          // 文档2
59          doc = new Document();
60          doc.add(new IntPoint( name: "content",  ...point: 100, 65));
61          doc.add(new IntPoint( name: "content",  ...point: 13, 57));
62          indexWriter.addDocument(doc);
```

在图1中，由于第49行、50行、55行。60行、61行，添加了具有相同的域名“content”的点数据，故他们三个的点数据信息会使用同一个PointValuesWriter对象来收集，同理第51行、56行。

PointValuesWriter对象收集的内容主要包括以下信息：

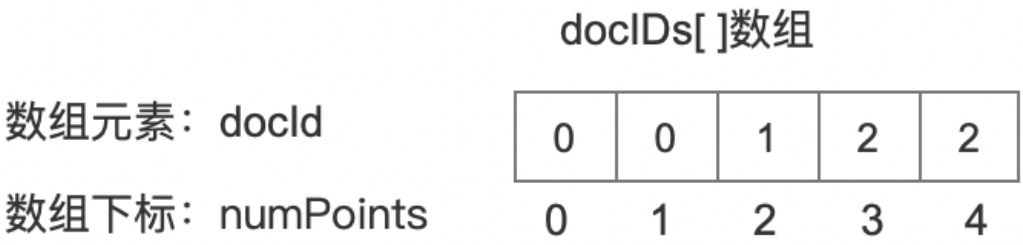
numPoints：

int类型，numPoints是一个从0开始递增的值，可以理解为是每一个点数据的一个唯一编号，并且通过这个编号能映射出该点数据属于哪一个文档(document)（下面会介绍），由于是每一个点数据的唯一编号，所以该值还可以用来统计某个域的点数据的个数，在图1中，域名为"content"的点数据共有5个，那么这么5个点数据的numPoints的值分别为0、1、2、3、4。

docIDs

int类型数组，每添加一条点数据，会将该点数据所属文档号作为数组元素添加到docIDs数组中，并且数组下标为该点数据对应的numPoints，对于域名为"content"的点数据，在完成三篇文档的添加后，docIds数组如下所示：

图2：



numDocs

该值描述的是对于某个点数据域，包含它的文档数量，例如在图1中，域名为"content"的点数据，包含该域的文档有文档0、文档1、文档2，故numDocs的值为3，同理对于域名为"titile"的点数据，numDocs的值为1。

bytes

该值是ByteBlockPool类型，ByteBlockPool为Lucene中的类，在这里我们只需要知道，在这个类中使用了buff(byte[ ]数组)来存储点数据的域值，在Lucene中，数值类型的域值需要通过转化为字节类型来存储，故我们先介绍下在Lucene中数值类型到字节类型的转化实现。

数值类型到字节类型（无符号）的转化

Lucene中的提供了BigInteger、int、long、float、double到字节类型byte的转化，在NumericUtils类中有具体的实现，本文通过例子只介绍下int类型到byte类型的转化。

待转化的数值为3，过程分为两个步骤：

步骤一：将待转化的数值跟0x80000000执行异或操作

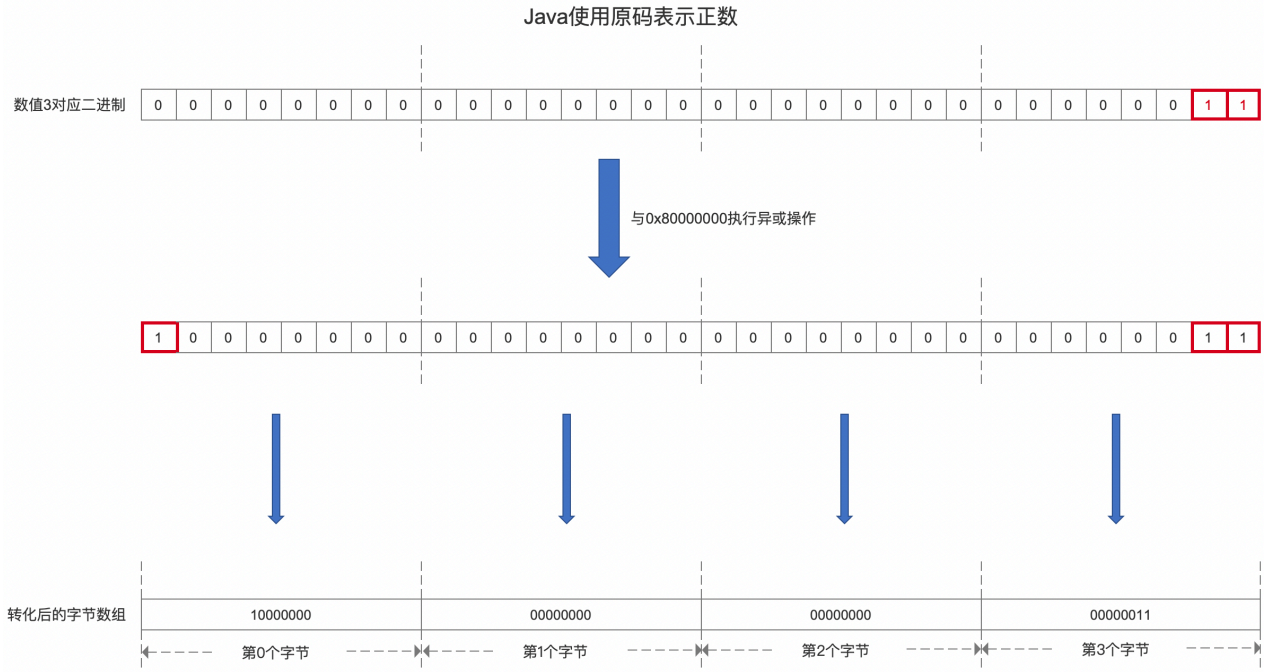
为什么要执行异或操作：

- 等介绍完步骤二再做解释

步骤二：写入到数组大小为4的字节数组中

由于int类型的数值占用4个字节，所以只需要数组大小为4的字节数组存储即可，如下所示：

图3：

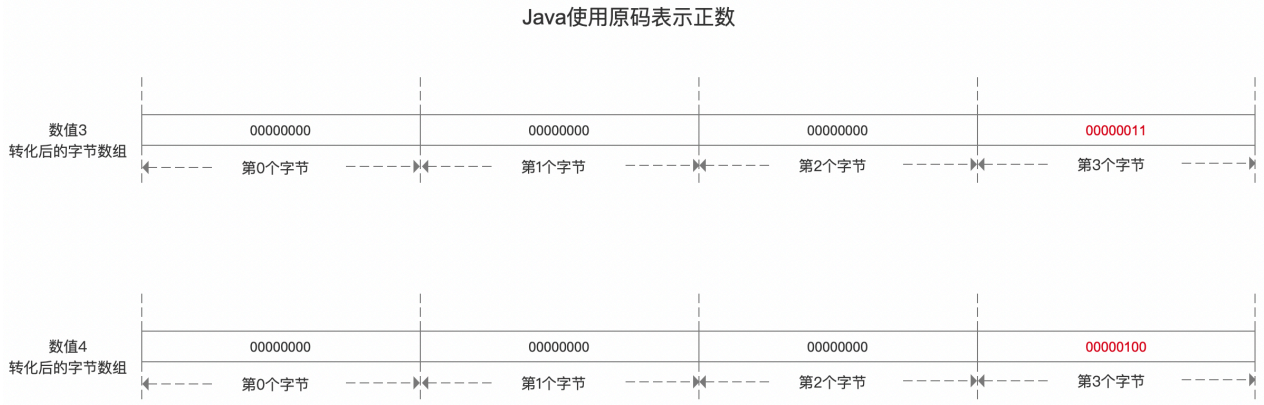


点击查看大图

为什么要执行异或操作：

- 当转化为字节数组后，可以通过比较两个字节数组的相同下标对应的数组元素大小来描述这两个字节数组的大小关系，使得转化后的字节数组依然能具有数值类型的比较功能（大小比较），比如说有两个数值类型3和4，假设不执行上文中的步骤一，即不跟0x80000000执行异或操作，转化后的字节数组如下所示：

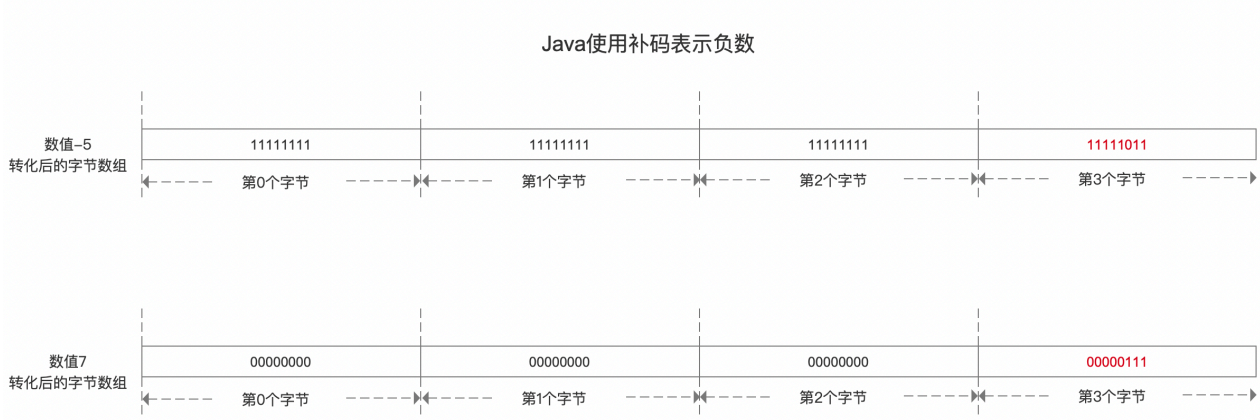
图4：



点击查看大图

图4中，从第0个字节开始，我们依次比较两个字节数组相同下标对应的数组元素，显而易见，第0个、第1个、第2个字节都是相同的，当比较到第3个字节时，能区分出大小关系，如果我们比较的两个数值是-5、7，假设不执行上文中的步骤一，即不跟0x80000000执行异或操作，转化后的字节数组如下所示：

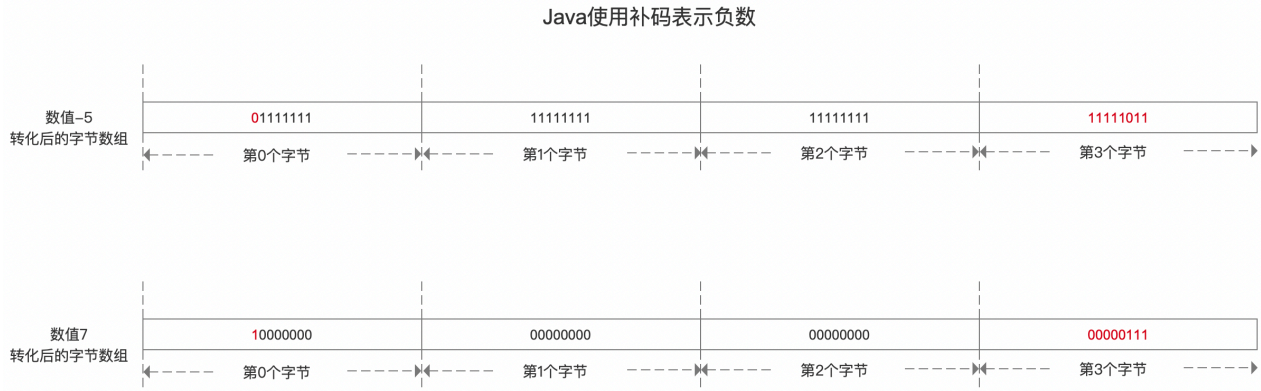
图5：



点击查看大图

从图5可知，由于在Java中使用补码来表示负数，所以当比较第0个字节时，会得出-5比7大的错误结果，如果我们先执行上文中的步骤一，即不跟0x80000000执行异或操作，那么数值-5跟7转化后的字节数组如下所示：

图6：



点击查看大图

由图6可知，可以正确的比较-5与7的大小关系了。

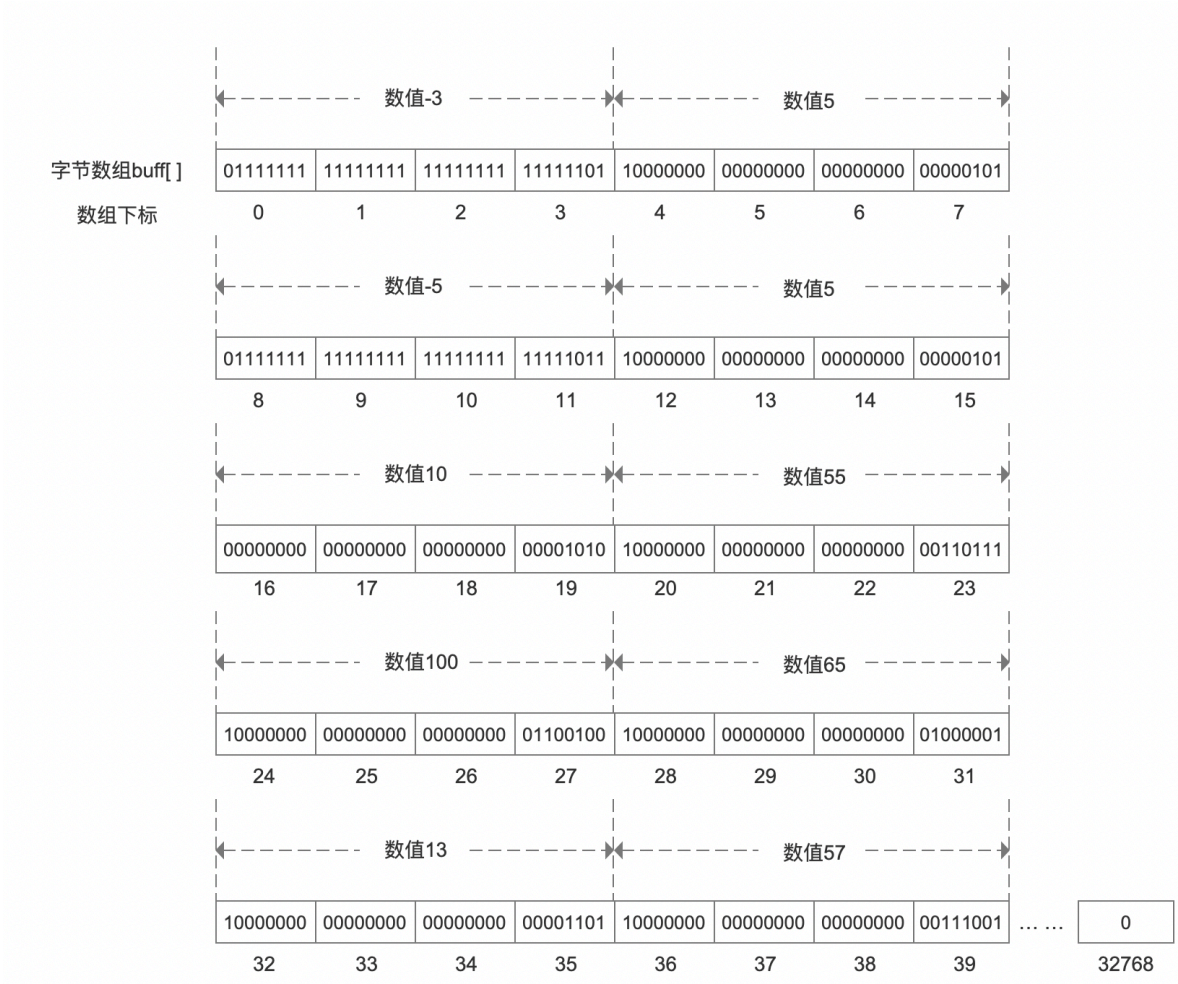
上文中我们说到，ByteBlockPool类型的变量bytes使用字节数组buff来存储点数据的域值，存储的过程十分简单，就是**将点数据的域值转化为字节数组后，拷贝到bytes的字节数组buff中**，例如有下面的例子，同图1：

图7：

```
46          Document doc;
47          // 文档0
48          doc = new Document();
49          doc.add(new IntPoint( name: "content",  ...point: -3, 5));
50          doc.add(new IntPoint( name: "content",  ...point: -5, 5));
51          doc.add(new IntPoint( name: "title",  ...point: 2, 5));
52          indexWriter.addDocument(doc);
53          // 文档1
54          doc = new Document();
55          doc.add(new IntPoint( name: "content",  ...point: 10, 55));
56          doc.add(new IntPoint( name: "title",  ...point: 3, 55));
57          indexWriter.addDocument(doc);
58          // 文档2
59          doc = new Document();
60          doc.add(new IntPoint( name: "content",  ...point: 100, 65));
61          doc.add(new IntPoint( name: "content",  ...point: 13, 57));
62          indexWriter.addDocument(doc);
```

我们将要介绍在添加了3篇文档后，域名为"content"的点数据的域值在字节数组buff中的数据分布，如下所示：

图8：



由图8可以看出，在字节数组buff[ ]中，下标0~7对应的数组元素存储的是图7中代码第49行的点数据的域值，下标24~31对应的数组元素存储的是图7中代码第60行的点数据的域值。

生成索引文件.dim&&.dii阶段，会读取buff中的域值，其读取的过程将会在后面的文章中介绍。

结语

本文介绍了在执行flush之前，Lucene是如何收集点数据的信息，即上文中的numPoints、docIDs、numDocs、bytes，那么在flush阶段，就可以通过这些信息来生成索引文件.dim&&.dii。

点击下载附件