package com.twitter.search.core.earlybird.index.inverted;

import org.apache.lucene.util.BytesRef;

/\*\*

\* Utilities for encoding and decoding BytesRefs into ints. The encoding is:

\* [0..n] n bytes big-endian decoded into integers.

\* n: number of bytes.

\*

\* Example:

\* encode([DE, AD, BE, EF, AB]) => [0xDEADBEEF, 0xAB000000, 5]

\*

\* It's necessary to store the length at the end instead of the start so that we can know how far to

\* jump backward from a skiplist entry. We can't store it after the skip list entry because there

\* can be a variable number of pointers after the skip list entry.

\*

\* An example skip list entry, with labels on the following line:

\* [0xDEADBEEF, 12, 654, 0x877, 0x78879]

\* [ payload, position, docID, level0Pointer, level1Pointer]

\*/

public final class PayloadUtil {

private PayloadUtil() {

}

public static final int[] EMPTY\_PAYLOAD = new int[]{0};

/\*\*

\* Encodes a {@link BytesRef} into an int array (to be inserted into a

\* {@link IntBlockPool}. The encoder considers the input to be big-endian encoded ints.

\*/

public static int[] encodePayload(BytesRef payload) {

if (payload == null) {

return EMPTY\_PAYLOAD;

}

int intsInPayload = intsForBytes(payload.length);

int[] arr = new int[1 + intsInPayload];

for (int i = 0; i < intsInPayload; i++) {

int n = 0;

for (int j = 0; j < 4; j++) {

int index = i \* 4 + j;

int b;

if (index < payload.length) {

// mask off the top bits in case b is negative.

b = payload.bytes[index] & 0xFF;

} else {

b = 0;

}

n = n << 8 | b;

}

arr[i] = n;

}

arr[intsInPayload] = payload.length;

return arr;

}

/\*\*

\* Decodes a {@link IntBlockPool} and position into a {@link BytesRef}. The ints are

\* converted into big-endian encoded bytes.

\*/

public static BytesRef decodePayload(

IntBlockPool b,

int pointer) {

int length = b.get(pointer);

BytesRef bytesRef = new BytesRef(length);

bytesRef.length = length;

int numInts = intsForBytes(length);

for (int i = 0; i < numInts; i++) {

int n = b.get(pointer - numInts + i);

for (int j = 0; j < 4; j++) {

int byteIndex = 4 \* i + j;

if (byteIndex < length) {

bytesRef.bytes[byteIndex] = (byte) (n >> 8 \* (3 - byteIndex % 4));

}

}

}

return bytesRef;

}

private static int intsForBytes(int byteCount) {

return (byteCount + 3) / 4;

}

}