package com.twitter.search.common.search;

import java.io.IOException;

import com.google.common.base.Preconditions;

import org.apache.lucene.search.DocIdSetIterator;

/\*\*

\* Disjunction over 2 DocIdSetIterators. This should be faster than a disjunction over N since there

\* would be no need to adjust the heap.

\*/

public class PairDocIdSetIterator extends DocIdSetIterator {

private final DocIdSetIterator d1;

private final DocIdSetIterator d2;

private int doc = -1;

/\*\* Creates a new PairDocIdSetIterator instance. \*/

public PairDocIdSetIterator(DocIdSetIterator d1, DocIdSetIterator d2) throws IOException {

Preconditions.checkNotNull(d1);

Preconditions.checkNotNull(d2);

this.d1 = d1;

this.d2 = d2;

// position the iterators

this.d1.nextDoc();

this.d2.nextDoc();

}

@Override

public int docID() {

return doc;

}

@Override

public int nextDoc() throws IOException {

int doc1 = d1.docID();

int doc2 = d2.docID();

DocIdSetIterator iter = null;

if (doc1 < doc2) {

doc = doc1;

//d1.nextDoc();

iter = d1;

} else if (doc1 > doc2) {

doc = doc2;

//d2.nextDoc();

iter = d2;

} else {

doc = doc1;

//d1.nextDoc();

//d2.nextDoc();

}

if (doc != NO\_MORE\_DOCS) {

if (iter != null) {

iter.nextDoc();

} else {

d1.nextDoc();

d2.nextDoc();

}

}

return doc;

}

@Override

public int advance(int target) throws IOException {

if (d1.docID() < target) {

d1.advance(target);

}

if (d2.docID() < target) {

d2.advance(target);

}

return (doc != NO\_MORE\_DOCS) ? nextDoc() : doc;

}

@Override

public long cost() {

// very coarse estimate

return d1.cost() + d2.cost();

}

}