

<pre>private Book[] bucketSort(ArrayList<Book> books) { Book[] booksToSort = new Book[books.size()]; for(int c = 0; c < books.size(); c++) { booksToSort[c] = books.get(c); } @SuppressWarnings("unchecked") Vector<Book>[] buckets = new Vector[booksToSort.length]; for(int c = 0; c < booksToSort.length; c++) { buckets[c] = new Vector<Book>(); } // Put array elements in different buckets for(int c = 0; c < booksToSort.length; c++) { int idx = (booksToSort[c].getBookCount() - 1); if(idx >= booksToSort.length) { buckets[buckets.length - 1].add(booksToSort[c]); } else { buckets[idx].add(booksToSort[c]); } } //Sort individual buckets for(int c = 0; c < booksToSort.length; c++) { Collections.sort(buckets[c]); } // Concatenate information int index = 0; for (int c = 0; c < booksToSort.length; c++) { for (int j = 0; j < buckets[c].size(); j++) { booksToSort[index++] = buckets[c].get(j); } } return booksToSort; }</pre>	Temporal
Book[] booksToSort = new Book[books.size()];	1
for(int c = 0; c < books.size(); c++) {	n + 1
booksToSort[c] = books.get(c);	n
}	
@SuppressWarnings("unchecked")	
Vector<Book>[] buckets = new Vector[booksToSort.length];	1
for(int c = 0; c < booksToSort.length; c++) {	n + 1
buckets[c] = new Vector<Book>();	n
}	
// Put array elements in different buckets	
for(int c = 0; c < booksToSort.length; c++) {	n + 1
int idx = (booksToSort[c].getBookCount() - 1);	n
if(idx >= booksToSort.length) {	n
buckets[buckets.length - 1].add(booksToSort[c]);	n
} else {	n
buckets[idx].add(booksToSort[c]);	n
}	
}	
//Sort individual buckets	
for(int c = 0; c < booksToSort.length; c++) {	n + 1
Collections.sort(buckets[c]);	n
}	
// Concatenate information	
int index = 0;	1
for (int c = 0; c < booksToSort.length; c++) {	n + 1
for (int j = 0; j < buckets[c].size(); j++) {	[n(n + 1)]/2 + n
booksToSort[index++] = buckets[c].get(j);	[n(n + 1)]/2
}	
}	
return booksToSort;	1
}	

Espacial

Complejidad espacial

Entrada array-> n
auxiliares buckets>n
output->n
salida -> n

O(n)

Complejidad temporal total: 1 + (n + 1) + n + 1 + (n + 1) + n + (n + 1) + n+ n+ n+ n+ n + (n + 1) + n + 1 + (n + 1) + ([n(n + 1)]/2 + n) + ([n(n + 1)]/2) + 1
Complejidad temporal total: 4 + 8n + 5(n + 1) + ([n(n + 1)]/2 + n) + [n(n + 1)]/2
Complejidad temporal total: 4+ 14n+ 5+ 2([n(n+1)]/2)
Complejidad temporal total: 9+ 14n + n^2+n
Complejidad temporal total: n^2+15n+9

O(N^2) en el peor de los casos