Sorting matched arrays in Java

Asked 16 years, 3 months ago Modified 7 years, 4 months ago

Viewed 12k times

26

Let's say that I have two arrays (in Java),

int[] numbers; and int[] colors;

Each ith element of numbers corresponds to its ith element in colors. Ex, numbers = {4,2,1} colors = {0x11, 0x24, 0x01}; Means that number 4 is color 0x11, number 2 is 0x24, etc.

I want to sort the numbers array, but then still have it so each element matches up with its pair in colors.

Ex. numbers = {1,2,4}; colors = {0x01,0x24,0x11};

What's the cleanest, simplest way to do this? The arrays have a few thousand items, so being in place would be best, but not required. Would it make sense to do an Arrays.sort() and a custom comparator? Using library functions as much as possible is preferable.

Note: I know the "best" solution is to make a class for the two elements and use a custom comparator. This question is meant to ask people for the quickest way to code this. Imagine being at a programming competition, you wouldn't want to be making all these extra classes, anonymous classes for the

comparator, etc. Better yet, forget Java; how would you code it in C?

java sorting list

Share

Improve this question

Follow

edited Sep 21, 2008 at 21:42

asked Sep 21, 2008 at 21:28

user16773

343 ● 1 ● 3 ● 9

13 Answers

Sorted by: Highest score (default)

20

You could use sort() with a custom comparator if you kept a third array with the index, and sorted on that, leaving the data intact.

Java code example:

```
Integer[] idx = new Integer[numbers.length];
for( int i = 0 ; i < idx.length; i++ ) idx[i] = i;
Arrays.sort(idx, new Comparator<Integer>() {
    public int compare(Integer i1, Integer i2) {
        return Double.compare(numbers[i1], numbers[i2]
    }
});

// numbers[idx[i]] is the sorted number at index i
// colors[idx[i]] is the sorted color at index i
```

Note that you have to use `Integer` instead of `int` or you can't use a custom comparator.

Share Improve this answer

Follow

edited Feb 25, 2013 at 17:36

Andrew Mao
36.8k ● 24 ● 147 ● 228

answered Sep 21, 2008 at 21:43

tovare
4,087 ● 5 ● 32 ● 30

This is the fastest way to do this... it's not clean, but it's the quickest to code. – billjamesdev Sep 21, 2008 at 22:48

9

It seems like the cleanest thing to do would be to create a custom property class that implements Comparable. For example:

```
class Color implements Comparable {
  private int number;
  private int color;

  // (snip ctor, setters, etc.)

  public int getNumber() {
    return number;
  }
  public int getColor() {
    return color;
  }

  public int compareTo(Color other) {
    if (this.getNumber() == other.getNumber) {
      return 0;
    } else if (this.getNumber() > other.getNumber) {
      return 1;
      } else {
        return -1;
      }
    }
  }
```

Then you can separate your sorting algorithm from the ordering logic (you could use Collections.sort if you use a List instead of an array), and most importantly, you won't have to worry about somehow getting two arrays out of sync.

Share Improve this answer

Follow

answered Sep 21, 2008 at 21:40

Frank Pape

692 ● 1 ● 7 ● 8

Yeah, this is what you would do in a normal situation or applications. But at something like a programming competition where you need to code this quickly, I bet you wouldn't want to write this extra fluff. – user16773 Sep 21, 2008 at 21:41

On the contrary, this took me about a minute to write, which would be a lot less time than I would take trying to keep two arrays in sync, and convincing myself that I'd done it correctly. – Frank Pape Sep 21, 2008 at 21:44

you should absolutely be making a custom class. That is by far the fastest way to do it. If it takes too long, invest in a learning how to use a good IDE. – ykaganovich Sep 22, 2008 at 3:58

If you'd be willing to allocate some extra space, you could generate another array, call it extra, with elements like

4

this:

```
extra = [0,1,...,numbers.length-1]
```

Then you could sort this extra array using Arrays.sort() with custom comparator (that, while comparing elements i and j really compares numbers[extra[i]] and numbers[extra[j]]). This way after sorting the extra array, extra[0] would contain the index of the smallest number and, as numbers and colours didn't move, the corresponding colour.
This isn't very nice, but it gets the job done, and I can't really think of an easier way to do it.

As a side note, in the competition I usually find the C++ templated pairs and nice maps indispensable ;)

Share Improve this answer Follow

answered Sep 21, 2008 at 21:42

finrod
1,408 ● 12 ● 8

BTW, you need an array of Integer and not of int to be able to use a custom comparator. Fundamental types can only be sorted according to their natural order when you're limited to Arrays.sort. – Torsten Marek Sep 21, 2008 at 22:04

Oh, sorry about it. I got a bit rusty with Java these days. – finrod Sep 21, 2008 at 22:28

Why not introduce an object to represent a number and a color and implement a comparator function for that?

3

Also, do you really need an array, why not use something derived from Collection?

Share Improve this answer Follow

answered Sep 21, 2008 at 21:34

JeffFoster
400 ● 1 ● 3

This situation comes up pretty often. I want to be able to code it quickly, without extra crud, for example in a programming competition. – user16773 Sep 21, 2008 at 21:37

3

I like @tovare's solution. Make a pointer array:

```
int ptr[] = { 1, 2, 3 };
```

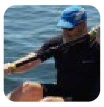
and then when you sort on numbers, swap the values in ptr instead of in numbers. Then access through the ptr array, like

```
for (int i = 0; i < ptr.length; i++)
{
   printf("%d %d\n", numbers[ptr[i]], colors[ptr[i]]);
}
```

Update: ok, it appears others have beaten me to this. No XP for me.

Share Improve this answer Follow

answered Sep 21, 2008 at 21:58



Paul Tomblin
183k ● 59 ● 323 ● 410

3

An example illustrating using a third index array. Not sure if this is the best implementation.

```
    import java.util.*;
```

```
public class Sort {

    private static void printTable(String caption, In
                Integer[] colors, Integer[] sortOrde

        System.out.println(caption+
                "\nNo   Num   Color"+
                "\n----------------");

        for(int i=0;i<sortOrder.length;i++){
            System.out.printf("%x    %d     %d\n",
                    i,numbers[sortOrder[i]],colors[so

        }
    }


    public static void main(String[] args) {

        final Integer[] numbers = {1,4,3,4,2,6};
        final Integer[] colors  = {0x50,0x34,0x00,0x
        Integer[] sortOrder = new Integer[numbers.le

        // Create index array.
        for(int i=0; i<sortOrder.length; i++){
            sortOrder[i] = i;
        }
        printTable("\nNot sorted",numbers, colors, so

        Arrays.sort(sortOrder,new Comparator<Integer
            public int compare(Integer a, Integer b)
                return numbers[b]-numbers[a];
            }});
        printTable("\nSorted by numbers",numbers, co

        Arrays.sort(sortOrder,new Comparator<Integer
            public int compare(Integer a, Integer b)
                return colors[b]-colors[a];
            }});
        printTable("\nSorted by colors",numbers, col
    }
}
```

The output should look like this:

```
Not sorted
No   Num   Color
----------------
0    1     80
1    4     52
2    3     0
3    4     254
4    2     255
5    6     255

Sorted by numbers
No   Num   Color
----------------
0    6     255
1    4     52
2    4     254
3    3     0
4    2     255
5    1     80

Sorted by colors
No   Num   Color
----------------
0    6     255
1    2     255
2    4     254
3    1     80
4    4     52
5    3     0
```

Share Improve this answer Follow   edited Sep 22, 2008 at 23:57

answered Sep 22, 2008 at 23:44

tovare
4,087 ● 5 ● 32 ● 30

2

One quick hack would be to combine the two arrays with bit shifts. Make an array of longs such that the most significant 32 bits is the number and the least significant 32 is the color. Use a sorting method and then unpack.

Share Improve this answer
Follow

answered Sep 21, 2008 at 22:24

theycallhimtom
345 ● 1 ● 5

1

Would it suffice to code your own sort method? A simple bubblesort would probably be quick to code (and get right). No need for extra classes or comparators.

Share Improve this answer
Follow

answered Sep 21, 2008 at 22:03

Zach Scrivena
29.5k ● 12 ● 65 ● 73

Memorizing Donald. D. Knuth/ The Art of Computer Programming volume 3 - sorting and searching and implementing the optimal algorithm on the fly would be ... impressive! :-) – tovare Sep 21, 2008 at 22:22

Credit to `@tovare` for the original best answer.

1

My answer below removes the (now) unnecessary autoboxing via Maven dependency {net.mintern : primitive : 1.2.2} from this answer: https://stackoverflow.com/a/27095994/257299

```
int[] idx = new int[numbers.length];
for( int i = 0 ; i < idx.length; i++ ) idx[i] = i;
final boolean isStableSort = false;
Primitive.sort(idx,
               (i1, i2) -> Double.compare(numbers[i1],
               isStableSort);

// numbers[idx[i]] is the sorted number at index i
// colors[idx[i]] is the sorted color at index i
```

Share Improve this answer

Follow

edited May 23, 2017 at 12:08

Community Bot
1 ●1

answered Apr 18, 2016 at 11:24

kevinarpe
21.3k ●28 ●132 ●164

1

I guess you want performance optimization while trying to avoid using array of objects (which can cause a painful GC event). Unfortunately there's no general solution, thought. But, for your specific case, in which numbers are different from each others, there might be two arrays to be created only.

```
/**
 * work only for array of different numbers
 */
private void sortPairArray(int[] numbers, int[] colors
    int[] tmpNumbers = Arrays.copyOf(numbers, numbers.
    int[] tmpColors = Arrays.copyOf(colors, colors.len
    Arrays.sort(numbers);
    for (int i = 0; i < tmpNumbers.length; i++) {
        int number = tmpNumbers[i];
        int index = Arrays.binarySearch(numbers, numbe
be found
        colors[index] = tmpColors[i];
    }
}
```

Two sorted arrays can be replace by [Int2IntOpenHashMap](), which performs faster run, but memory usage could be double.

Share Improve this answer Follow

edited Aug 4, 2017 at 9:47

answered Aug 4, 2017 at 9:41

Tung Ha
11 ● 2

0

You need to sort the colors array by its relative item in the numbers array. Specify a comparator that compares numbers and use that as the comparison for the colors array.

Share Improve this answer Follow

answered Sep 21, 2008 at 21:32

1800 INFORMATION
135k ● 30 ● 163 ● 242

0

The simplest way to do this in C, would be bubblesort + dual pointers. Ofcourse the fastest would be quicksort + two pointers. Ofcourse the 2nd pointer maintains the correlation between the two arrays.

I would rather define values that are stored in two arrays as a struct, and use the struct in a single array. Then use quicksort on it. you can write a generic version of sort, by calling a compare function, which can then be written for each struct, but then you already know that :)

Share Improve this answer Follow

answered Sep 22, 2008 at 0:40

shiva
741 ● 6 ● 14

0

Use a [TreeMap](#)

Share Improve this answer Follow

answered Sep 13, 2010 at 9:44

st0le
33.5k ● 8 ● 92 ● 89