My Java Solution to share

0

CodingGeek
Reputation: ★ 2

Same code by user Daring with comments : https://leetcode.com/discuss/21488/java-code-by-providing-comparator-with-explaination-nlogn

0

VectorChange
Reputation: ★ 1

Compare between the string needs O(len) and sort needs O(len*log(len)), the time complexity is O(len^2*log(len))

0

codecola
Reputation: ★ 5

may be Integer.toString(num[i]) is better than num[i] + ""

2

fun4LeetCode
Reputation: ★ 1247

I would recommend using the following comparator if you wish to reduce memory footprint, since the comparator above generates a lot of concatenated strings (of the order of O(n^2), with n the total number of elements). Also for large-size input arrays, the following comparator will have a smaller chance to trigger GC, which is detrimental to the time performance.

```
Comparator<String> cmp = new Comparator<String>() {
        @Override
        public int compare(String str1, String str2) {
                sb1.delete(0, sb1.length()).append(str1).append(str2);
                sb2.delete(0, sb2.length()).append(str2).append(str1);

                for (int i = 0; i < sb1.length(); i++) {
                        if (sb1.charAt(i) == sb2.charAt(i)) continue;
                        return (sb1.charAt(i) > sb2.charAt(i) ? -1 : 1);
                }

                return 0;
        }
};
```

I assume you have initialized two final StringBuilder objects sb1 and sb2 somewhere before the comparator.

0

freezaku
Reputation: ★ 0

This post is deleted!

0

saharH @ran3
Reputation: 0

@ran3 could you explain how you get nklognk ?

0

touchdown
Reputation: 5

Great Solution! You have a deep understanding of Arrays.sort()

1

whglamrock
Reputation: 19

Maybe the time complexity is O(knlogn)? The typical sort takes O(nlogn) when the comparison between two elements takes O(1). Then in this case each comparison takes O(k), I guess the overall time complexity should be O(knlogn)?

0

tankztc @whglamrock
Reputation: 52

@whglamrock I think you're right, I've the same opinion. Typical sort like merge sort have O(logn) level, and each level have O(n) times compare, so the overall time complexity is O(knlogn)

19
POSTS

10587
VIEWS

Log in to reply