Interview Cake

← course home (/table-of-contents#section_sorting-searching-logarithms_question_top-scores)

You created a game that is more popular than Angry Birds.

Each round, players receive a score between 0 and 100, which you use to rank them from highest to lowest. So far you're using an algorithm that sorts in $O(n \lg n)$ time, but players are complaining that their rankings aren't updated fast enough. You need a faster sorting algorithm.

Write a function that takes:

1. a list of `unsorted_scores`
2. the `highest_possible_score` in the game

and returns a sorted list of scores in less than $O(n \lg n)$ time.

For example:

```
unsorted_scores = [37, 89, 41, 65, 91, 53]
HIGHEST_POSSIBLE_SCORE = 100


# Returns [91, 89, 65, 53, 41, 37]
sort_scores(unsorted_scores, HIGHEST_POSSIBLE_SCORE)
```

Python 2.7

We're defining n as the number of `unsorted_scores` because we're expecting the number of players to keep climbing.

And, we'll treat `highest_possible_score` as a constant instead of factoring it into our big O time and space costs because the highest possible score isn't going to change. Even if we *do* redesign the game a little, the scores will stay around the same order of

magnitude.
