2418. SORT THE PEOPLE

29 June 2024 Saturday
Kaushal Pratap

In this Leetcode problem, we'll be given two arrays called *names* and *heights* of the same length *n*. These two arrays will contain the name of the person and the respective height. I.e. *names[i] = heights[i]. We have to return an array containing the names of all persons mentioned in names* array in a descending order of height. I.e. from tallest to shortest.

For example -

```
Input: names = ["Mary","John","Emma"], heights = [180,165,170]
Output: ["Mary","Emma","John"]
```

As you can see the output is Mary, Emma and then John because Mary is tallest having height of 180 and Emma is of 170 and then John of 165.

Logic-

The approach to this problem is very simple. We will first create an empty array and then we will store the height with its respective name in the form of a list.

Like this-

```
new_array = [[180, Mary], [165, John], [170, Emma]]
```

It is important to note that we will first store the height then the name in each list. Because, we'll sort this list in descending order. And for sorting, the first element of each list will be checked. So, we will add the height to the name not the name to the height.

And after sorting, we'll append each name to our final output array using a loop. Because the list is already sorted and we just need to iterate over it.

Now, let's look at the code for a better understanding.

Code-

```
6   class Solution:
7       def sortPeople(self, names: list[str], heights: list[int]) -> list[str]:
8           new_array = []
9           result = []
```

- We will begin by creating our class and function in line 6 and 7.
- Function will take *names* array and *heights* array.
- Then, we will initialize our empty list called *new_array* in which we will store heights to the names in line 8.
- Also, we'll initialize our final output array named as *result* in line 9.

```
10          for i in range(len(names)):
11              new_array.append([heights[i],names[i]])
```

- Now in line 10, we will use a *for loop* to access the elements of *names* and *heights*.
- In line 11, we will add a list of *height element* with its respective *names element* in *new_array*.
- Now, the *new_array* looks like this - `[[180, Mary], [165, John], [170, Emma]]`

```
12          new_array.sort(reverse = True)
```

- Now we will simply sort the *new_array* in reverse order i.e. from tallest to the shortest.

```
13          for i in range(len(names)):
14              result.append(new_array[i][1])
```

- Now, the *new_array* is sorted. So, we will simply add the names to our *result* array in line 13 and 14. We will first iterate over each pair and then select the element located on 1th index. I.e. the name.

```
16          return result
```

- And finally, we'll return our final *result* array in line 16.

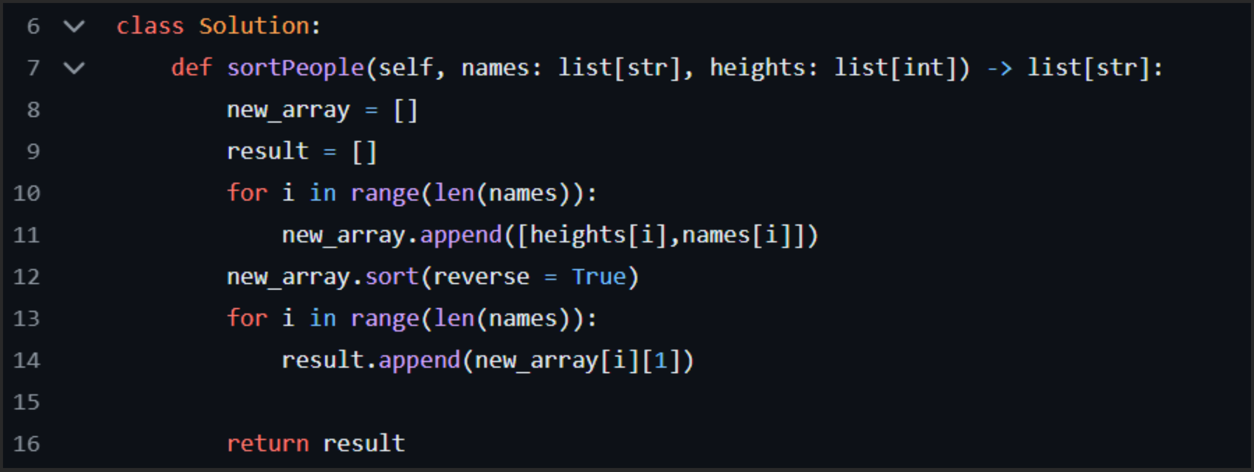
Now, let's understand the time complexity.

Time complexity analysis-

Line 6 to 9: Creating class, function and arrays run in O(1) time.
Line 10 and 11: A *for loop* iterating over every element, takes O(n) time.
Line 12: Python's built-in sort function uses Tim Sort. So, its time complexity is O(n logn).
Line 13 and 14: This *for loop* will also run in O(1) time.
Line 15: Returning a function is a O(1) runtime complexity.

The dominating term here is O(n logn). So, the time complexity of this algorithm is O(n logn)

Leetcode: https://leetcode.com/problems/sort-the-people/
GitHub: https://github.com/kaushal-pratap/code-portifolio/blob/main/Leetcode%20Problems/2418.%20Sort%20the%20People.py



```
6   class Solution:
7       def sortPeople(self, names: list[str], heights: list[int]) -> list[str]:
8           new_array = []
9           result = []
10          for i in range(len(names)):
11              new_array.append([heights[i],names[i]])
12          new_array.sort(reverse = True)
13          for i in range(len(names)):
14              result.append(new_array[i][1])
15
16          return result
```