

LeetCode 59: Merge Intervals

Arlie

Problem

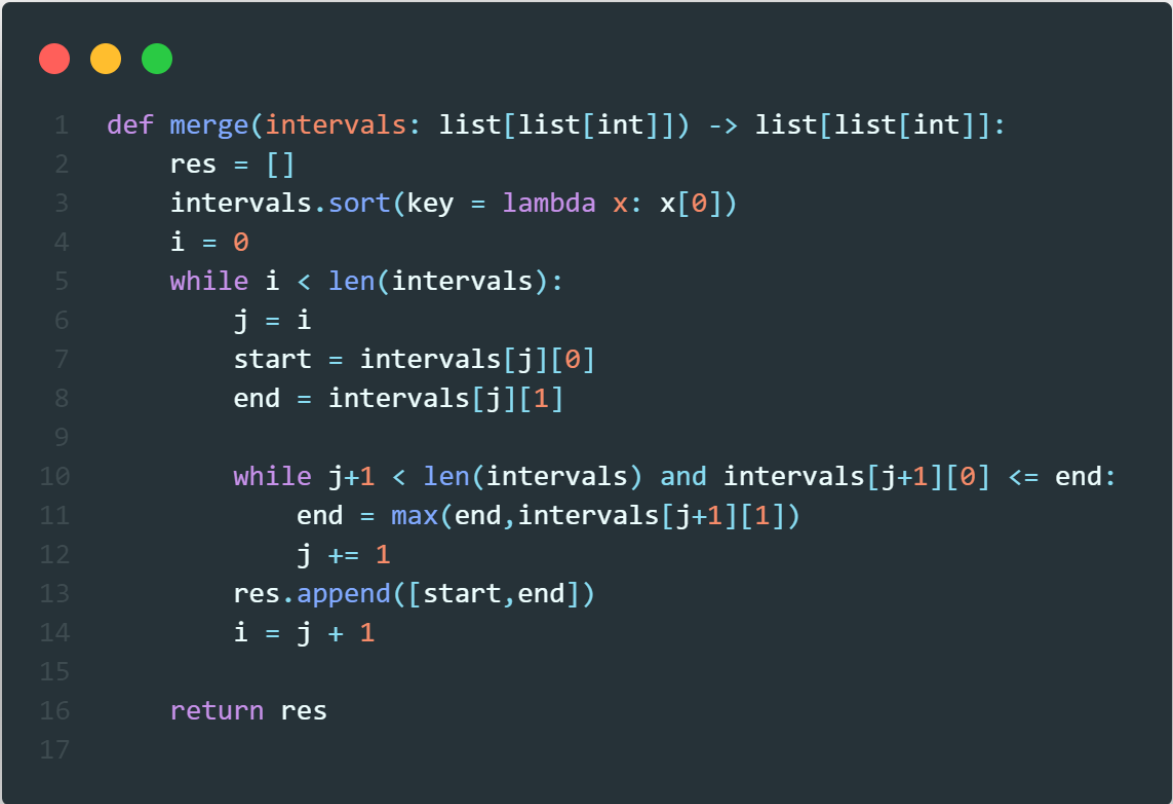
Given an array of `intervals` where `intervals[i] = [starti, endi]`, merge all overlapping intervals, and return an array of the non-overlapping intervals that cover all the intervals in the input

Difficulty: *Medium*

Solution

First, we need to understand what an *overlapping interval* is. An interval I_i will overlap with interval I_j if and only if $I_i[end] \leq I_j[start]$ i.e., interval I_i will end when or after I_j starts. Now for each interval I_i we need to merge all intervals that overlap with it, if we simply search for each interval where $end_i \leq start_j$ we would have quadratic complexity, we want to merge as many intervals possible using the least amount of time. We can do this by sorting the array of intervals by their starting time. Then for each interval we simply look at the next interval and check if it overlaps current end time we continue to look to the next interval until we are either reached the end of the array, or the next interval does not overlap in which case we append the start and end time to a resultant list.

Code



```
1  def merge(intervals: list[list[int]]) -> list[list[int]]:
2      res = []
3      intervals.sort(key = lambda x: x[0])
4      i = 0
5      while i < len(intervals):
6          j = i
7          start = intervals[j][0]
8          end = intervals[j][1]
9
10         while j+1 < len(intervals) and intervals[j+1][0] <= end:
11             end = max(end, intervals[j+1][1])
12             j += 1
13         res.append([start, end])
14         i = j + 1
15
16     return res
17
```

Figure 1: image

Analysis

Time Complexity

Sorting the list will take $n \log n$ operations, and looping through the intervals will take n operations, hence we have a time complexity of

$$\mathcal{O}(n \log(n))$$

Space Complexity

We create a resultant array which will contain at most n intervals hence we have a space complexity of

$$\mathcal{O}(n)$$