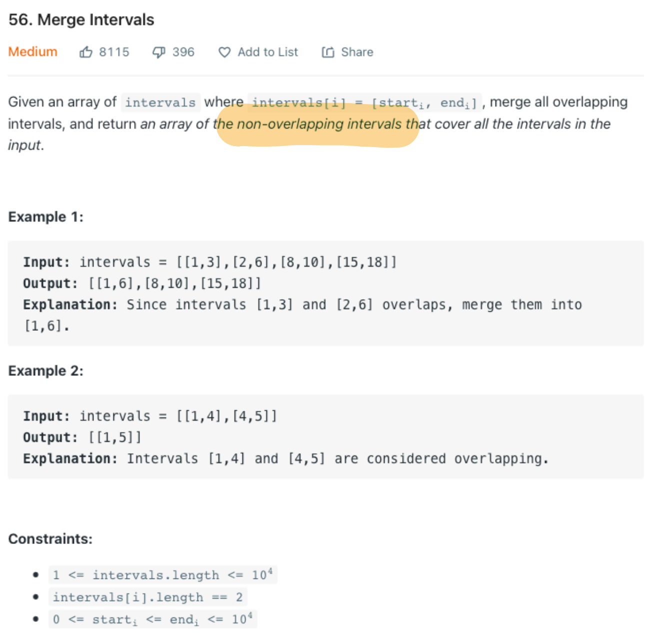


56. Merge Intervals

Medium 8115 396 Add to List Share

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

Example 1:

```
Input: intervals = [[1,3],[2,6],[8,10],[15,18]]
Output: [[1,6],[8,10],[15,18]]
Explanation: Since intervals [1,3] and [2,6] overlaps, merge them into
[1,6].
```

Example 2:

```
Input: intervals = [[1,4],[4,5]]
Output: [[1,5]]
Explanation: Intervals [1,4] and [4,5] are considered overlapping.
```

Constraints:

- `1 <= intervals.length <= 10^4`
- `intervals[i].length == 2`
- `0 <= start_i <= end_i <= 10^4`

Medium

A. ① Overlapping

inter1	inter2	
[1 , 8]	[8, 12]	✓
[1, 8]	[7, 12]	✓
[x , 8]	[9 , 2]	✗
[1 , 8]	[1, 9]	✓

∴ if inter1[0] <= inter2[0] <= inter1[1]

② Sort all intervals in advance

[[1,3] , [8,10] , [2,6] , [15,18]]

⇓

$[\underbrace{[1,3]}, \underbrace{[2,6] , [8,10] , [15,18]}]$

③ ans = [[1,3]] ← 最小的先拿下來

for interal in ______ :

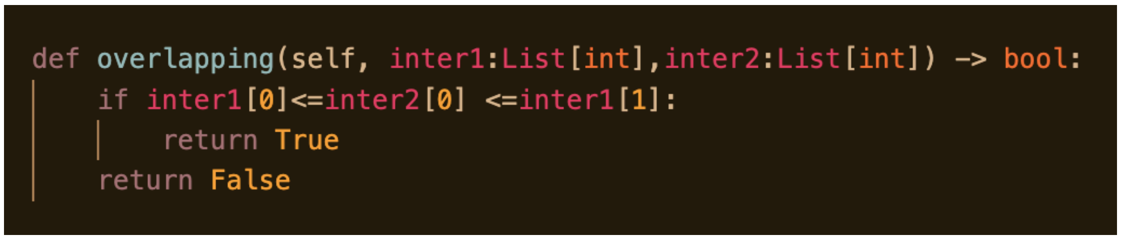
    if ans[-1] not overlapping with interval:

        ans.append (interval)

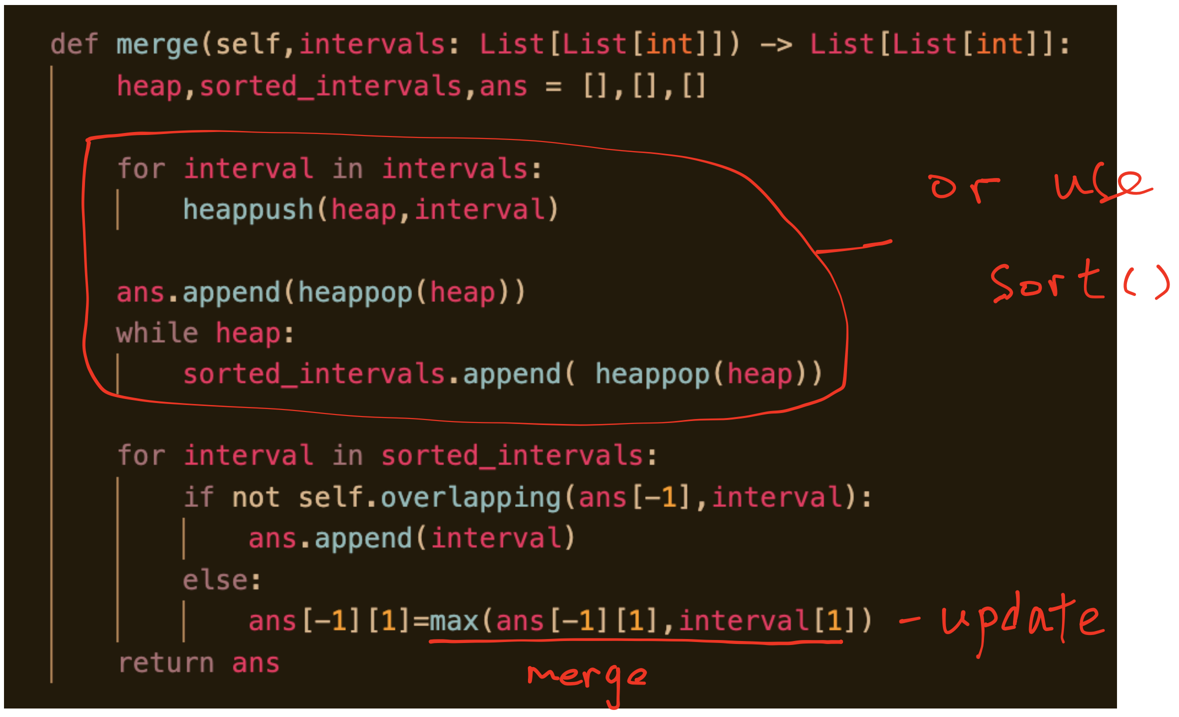
else :

1) merge

2) update the ans[-1] to be merged interval



```
def overlapping(self, inter1:List[int],inter2:List[int]) -> bool:
    if inter1[0]<=inter2[0] <=inter1[1]:
        return True
    return False
```



```
def merge(self,intervals: List[List[int]]) -> List[List[int]]:
    heap,sorted_intervals,ans = [],[],[]

    for interval in intervals:
        heappush(heap,interval)

    ans.append(heappop(heap))
    while heap:
        sorted_intervals.append( heappop(heap))

    for interval in sorted_intervals:
        if not self.overlapping(ans[-1],interval):
            ans.append(interval)
        else:
            ans[-1][1]=max(ans[-1][1],interval[1]) - update
    return ans
```