Problem # 57: Insert Interval

<https://leetcode.com/problems/insert-interval/>

My Solution:

<https://leetcode.com/problems/insert-interval/discuss/845502/Simple-Python-3-Solution-Runtime-beats-99.77>

Simple Python 3 Solution -- Runtime beats 99.77%

1. The intervals are either smaller\_intervals, overlap\_intervals or larger\_intervals, where smaller\_intervals are those with end values less than the start value of newInterval, larger\_intervals are those with start values greater than the end value of newInterval, and overlap\_intervals are the intervals that have overlap with newInterval.
2. Merge the overlap\_intervals with the newInterval -- for this starting point of the merged interval will be the minimum of the start of newInterval and the start of the first interval in overlap\_intervals, and the ending point of the merged interval will be the maximum of the end of the newInterval and the ending point of the last interval in the overlap\_intervals.
3. If there are no overlap intervals, then return the concatenation of the lists consisting of the smaller\_intervals, newInterval and larger\_intervals. Otherwise, return the concatenation of smaller\_intervals, merged\_interval and larger\_intervals

class Solution:

def insert(self, intervals: List[List[int]], newInterval: List[int]) -> List[List[int]]:

smaller\_intervals = [] # intervals whose end value < newInterval start value

overlap\_intervals = [] # intervals that need to be merged i.e. end value <= newInterval start value or whose start value is >= newInterval end value

larger\_intervals = [] # intervals whose start value > newInterval end value

for interval in intervals:

if interval[1] < newInterval[0]:

smaller\_intervals.append(interval)

elif interval[0] > newInterval[1]:

larger\_intervals.append(interval)

else:

overlap\_intervals.append(interval)

#print("smaller\_intervals = ", smaller\_intervals)

#print("overlap\_intervals = ", overlap\_intervals)

#print("larger\_intervals = ", larger\_intervals)

# merge intervals in the overlap\_intervals

if len(overlap\_intervals) == 0:

return(smaller\_intervals + [newInterval] + larger\_intervals)

merged\_interval = [[min(overlap\_intervals[0][0], newInterval[0]), max(overlap\_intervals[-1][1], newInterval[1])]]

return(smaller\_intervals + merged\_interval + larger\_intervals)