```
from typing import List
def binary_search(x, n):
    s = 0
    e = len(x) - 1
    while s<=e:
        mid = (s+e)//2
        if x[mid] == n:
            return mid
        elif x[mid] < n:
            s  = mid + 1
        else:
            e = mid - 1
    return e+1
def is_intersect(s1, t1, s2, t2):
        # return length of intersection if intersect else zero
        if min(t1, t2) <= max(s1, s2):
            return [min(s1, s2), max(t1, t2)]
        else:
            return []
class Solution:
    def insert(self, intervals: List[List[int]], newInterval: List[int]) ->
List[List[int]]:
        # sorted in ascending order by starti
        if not intervals:
            intervals.append(newInterval)
            return intervals
        x = []
        for i in range(len(intervals)):
            x.append(intervals[i][1]) # 为了k之前的interval都不用考虑

        k = binary_search(x,newInterval[0]) # 之前的interval都没有交集 不用考虑
        res = intervals[:k] #to store intervals till `k`th index
        while k < len(intervals) and intervals[k][0] <= newInterval[1]: #
until 当前interval的左端点 大于 想merge的interval的右端点
            newInterval[0] = min(intervals[k][0], newInterval[0])
            newInterval[1] = max(intervals[k][1], newInterval[1])
            k += 1
        res.append(newInterval)
        #adding remaining elements to the list
        res += intervals[k:]
        return res
```