Sample Solution Discussion: Forum on Backtracking and Greedy Algorithms

Given a collection of intervals, find the minimum number of intervals you need to

remove to make the rest of the intervals non-overlapping.

Example 1

Input: [[1,2],[2,3],[3,4],[1,3]]

Output: 1

Explanation: [1,3] can be removed and the rest of intervals are non-overlapping.

Example 2

Input: [[1,2],[1,2],[1,2]]

Output: 2

Explanation: You need to remove two [1,2] to make the rest of intervals non-overlapping.

Example 3

Input: [[1,2],[2,3]]

Output: 0

Explanation: You don't need to remove any of the intervals since they're already non-overlapping.

Sort the intervals with the finish time. We start with empty set. Then we iterate through the list, if current interval overlaps with any previous interval.

```
import sys

def nonOverlapIntervals(intervals):

    if not intervals:
        return 0
    min_rmv = 0
    intervals.sort(key=lambda x: x.end)
    last_end = - sys.maxsize
    for i in intervals:
        if i.start < last_end:  # overlap, delete this one, do not update the
end
            min_rmv += 1
        else:
            last_end = i.end
    return min_rmv

print(nonOverlapIntervals([[1,2],[1,2],[1,2]]))
```

Sorting time O(nlogn) will dominate.