Explanation

Why Sorting is Necessary

Basically you cannot draw a graph without sorting the intervals; hence, we will not know the overlapping intervals.

More Accurately

The key insight is about **making the problem systematic and predictable**. Here's why:

Without Sorting - The Problem

If intervals aren't sorted, you can't make local decisions about merging. Consider this example:

```
intervals = [[1,4], [0,2], [3,5]]
```

If you process them in this order:

- Start with [1,4]
- [0,2] overlaps with [1,4] → merge to [0,4]
- [3,5] overlaps with [0,4] → merge to [0,5]

But what if the order was different?

```
intervals = [[3,5], [1,4], [0,2]]
```

- Start with [3,5]

- [1,4] overlaps with [3,5] → merge to [1,5]
- [0,2] overlaps with [1,5] → merge to [0,5]

You get the same result, but the process is unpredictable and you might miss overlaps or process them inefficiently.

With Sorting - The Solution

When you sort by start time, you create a **guarantee**:

- Any interval that can overlap with the current interval must have already been processed
- You only need to check the **last interval** in your result (the most recently processed one)

How to Recognize You Need Sorting

Look for these patterns in interval problems:

1. **Local Decision Making**: You want to make decisions by only looking at adjacent or nearby elements
2. **Overlapping Relationships**: When elements can have relationships with multiple other elements
3. **Greedy Approach**: When you want to process elements in a specific order to make optimal local choices

The Mental Model

Think of it like organizing papers on your desk:

- **Unsorted**: Papers are scattered. To find overlapping topics, you'd have to compare every paper with every other paper
- **Sorted**: Papers are arranged chronologically. Now you only need to check if the current paper relates to the previous one

Code Insight

```
# This line is the key insight:
if cur_interval[0] <= prev_interval[1]:
```

This comparison only works reliably because:

1. All previous intervals with smaller start times have been processed
2. The current interval has the smallest start time among remaining intervals
3. Therefore, if there's going to be an overlap, it must be with the most recent interval in our result

General Rule

For interval problems, ask yourself: "Do I need to make decisions based on relationships between intervals?" If yes, sorting by start time (and sometimes end time) usually makes the problem much simpler by creating predictable patterns.

The sorting transforms a complex "compare everything with everything" problem into a simple "compare with your immediate neighbor" problem.
