Why Greedy Fails

The greedy approach fails for several fundamental reasons:

1. Wrong Optimization Target

Your greedy approach tries to minimize meetings per room by sorting by end time, but the actual goal is to **minimize the maximum number of simultaneous meetings**. These are different objectives!

2. Suboptimal Room Assignment

When you sort by end time, you process meetings in an order that doesn't reflect when they actually need to start. This leads to poor room assignment decisions.

3. Missing Overlaps

The key insight is that this problem is really about **interval overlap counting**. The minimum rooms needed equals the maximum number of meetings that overlap at any point in time. Greedy by end time doesn't directly address this.

4. Local vs Global Optimum

Consider this example: `[[0,30], [5,10], [15,20]]`

Greedy by end time:

- Sort: `[[5,10], [15,20], [0,30]]`
- Process `[5,10]`: Room 1
- Process `[15,20]`: Room 1 (since 10 ≤ 15) ✓
- Process `[0,30]`: Room 2 (since 20 > 0) ✓
- Result: 2 rooms ✓

Wait, this actually works! But here's a case where it fails:

Better failure example: `[[1,4], [5, 9], [3,8]]`

Greedy by end time:

- Sort: `[[1,4], [3,8], [5,9]]`
- Process `[1,4]`: Room 1
- Process `[3,8]`: Room 2 (since 3 ≤ 4) ✓
- Process `[5,9]`: Room 3 (since 8 > 5) x
- Result: 3 rooms

Optimal solution:

- At time 3: `[1,4]` and `[3,8]` overlap
- At time 5: `[3,8]` and `[5,9]` overlap
- But `[1,4]` ends before `[5,9]` starts
- So we actually need 2 rooms, and greedy got it wrong!