

12.4

N Meetings in One Room

MEDIUM

What You Will Learn

- ✓ Activity Selection Problem — a classic greedy algorithm
- ✓ Why sorting by end time is the key greedy insight
- ✓ How to select maximum non-overlapping intervals
- ✓ The pattern: finish early, leave room for more meetings

Intuition First

You have one meeting room and N meetings, each with a start and end time. You want to hold as many meetings as possible without any overlap.

The magic insight: always pick the meeting that ends the earliest. Why? Because finishing early leaves the maximum time for future meetings. It's like a relay race — hand off the baton as soon as possible.

The Greedy Rule

- Sort all meetings by END time (earliest ending first)
- Pick the first meeting (it ends earliest — always safe to pick)
- For each next meeting: if start > last_end → pick it (no overlap)
- If start ≤ last_end → skip it (overlaps with last picked meeting)

Visual Timeline

Meetings (before sort):

M1: start=1, end=2 ■ ■ ■ ■

M2: start=3, end=4 ■ ■ ■ ■

M3: start=0, end=6 ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■

M4: start=5, end=7 ■ ■ ■ ■ ■ ■ ■ ■

M5: start=8, end=9 ■ ■ ■ ■

After sorting by end time: M1(end=2), M2(end=4), M3(end=6), M4(end=7), M5(end=9)

Sorted by end time — green = selected, gray = skipped

Event	Start	End	Selected?
M1 [1-2]	1	2	✓ YES

M2 [3–4]	3	4	✓ YES
M3 [0–6]	0	6	✗ skip
M4 [5–7]	5	7	✗ skip
M5 [8–9]	8	9	✓ YES

Select M1 (end=2). Next: M2 start=3 > 2 → select. Next: M3 start=0 ≤ 4 → skip.
 Next: M4 start=5 ≤ 4? No, 5 > 4 → wait, but M4 end=7, selected M2 end=4... M4
 start=5 > 4 → select!

Actually: M3 sorted BEFORE M4. M3 start=0 ≤ last_end=4 → skip M3.

M4: start=5 > 4 → select M4! last_end=7. M5: start=8 > 7 → select M5!

Result: M1, M2, M4, M5 = 4 meetings ✓

■ Dry Run — Sorted: [(1,2),(3,4),(0,6),(5,7),(8,9)]

Step 1	Pick M1(1,2) — always pick first. count=1, lastEnd=2
Step 2	M2(3,4): start=3 > lastEnd=2 → PICK. count=2, lastEnd=4
Step 3	M3(0,6): start=0 ≤ lastEnd=4 → SKIP (overlaps)
Step 4	M4(5,7): start=5 > lastEnd=4 → PICK. count=3, lastEnd=7
Step 5	M5(8,9): start=8 > lastEnd=7 → PICK. count=4, lastEnd=9. Done!

Maximum meetings = 4 ✓

■ Java Code

```
public static int maxMeetings(int[] start, int[] end) {
    int n = start.length;
    // Create meeting objects and sort by end time
    int[][] meetings = new int[n][2];
    for (int i = 0; i < n; i++) {
        meetings[i][0] = start[i];
        meetings[i][1] = end[i];
    }
    // Sort by end time (greedy choice)
    Arrays.sort(meetings, (a, b) -> a[1] - b[1]);
    int count = 1; // Always pick first meeting
    int lastEnd = meetings[0][1];
    for (int i = 1; i < n; i++) {
        if (meetings[i][0] > lastEnd) {
            // No overlap – pick this meeting
            count++;
            lastEnd = meetings[i][1];
        }
    }
}
```

```
// else: overlaps, skip
}
return count;
}
```

■ Time & Space Complexity

Case	Time	Space	What it really means
All Cases	$O(n \log n)$	$O(n)$	Sort dominates; $O(n)$ for meetings array

■ Why Sort by End Time, Not Start?

- Sort by start time: might pick a long meeting that blocks many short ones
- Sort by duration: same issue — short duration doesn't mean early end
- Sort by end time: finishing early is the ONLY thing that maximizes future opportunities
- This is the mathematical proof of optimality — greedy choice is correct

■ Common Mistakes

- Sorting by start time instead of end time — different and wrong!
- Using `start >= lastEnd` instead of `start > lastEnd` (boundary meetings CAN coexist)
- Not initializing `lastEnd` from `meetings[0]` after sorting

■ Pro Tips

- This is the Activity Selection Problem — a textbook greedy
- Same pattern: Interval Scheduling, Non-overlapping Intervals (LeetCode #435)
- For minimum rooms needed: sort by start, use a min-heap → see Minimum Platforms

■ Visual Thinking

Picture a timeline drawn on a whiteboard. Each meeting is a colored bar.

You're selecting bars that don't overlap — maximize the count.

The greedy intuition: always pick the bar that ends first.

It's like planning your day — finish each task early to fit in more tasks.

Animation: bars on timeline, cursor sweeps left-to-right, selecting non-overlapping ones.

Selected bars glow green; overlapping ones fade gray when skipped.

■ Practice Problems

[Easy] Meetings: start=[1,3,0,5,8], end=[2,4,6,7,9] — max meetings?

[Med] Find the meetings that are selected (not just count)

[Med] Non-overlapping intervals: min removals to make all non-overlapping (LeetCode #435)

[Hard] Weighted job scheduling: each meeting has a profit — maximize total profit

■ Quick Revision Box

- Sort by END time — this is the key greedy choice
- Always pick first meeting
- Pick next if: `meeting.start > lastEnd`
- $O(n \log n)$ sort + $O(n)$ scan = $O(n \log n)$
- Activity Selection — classic greedy, provably optimal
- Boundary ok: `start == lastEnd` is a valid pick (`>` not `>=`)