

12
.7

Minimum Platforms

MEDIUM

■ What You Will Learn

- ✓ The event-based sweep line technique
- ✓ Sort arrivals and departures independently
- ✓ The 'simultaneous trains' problem — a classic interval problem
- ✓ How minimum platforms relates to maximum overlapping intervals

■ Intuition First

A railway station has trains arriving and departing. You need to find the minimum number of platforms required so no train has to wait.

Think of it this way: the minimum platforms needed = maximum number of trains present at the station at any single point in time. To find that maximum overlap, use two sorted pointers — one for arrivals, one for departures. When a train arrives, you need a new platform. When one departs, a platform is freed.

■ The Sweep Line Algorithm

- Sort arrival[] and departure[] arrays independently
- Two pointers i (arrivals), j (departures)
- If arrival[i] <= departure[j]: new train arrives → platforms++, i++
- Else: a train departs → platforms--, j++
- Track maxPlatforms = max platforms seen at any moment

■ Visual Timeline

Trains (arrival → departure):

```
Train 1: arr=900, dep=910
Train 2: arr=940, dep=1200
Train 3: arr=950, dep=1120
Train 4: arr=1100, dep=1130
Train 5: arr=1500, dep=1900
Train 6: arr=1800, dep=2000
```

Timeline (time →):

T1: ■■



Max overlap at ~1100: T2, T3, T4 all present → 3 platforms ✓

■ Dry Run

```
arr = [900, 940, 950, 1100, 1500, 1800] (sorted)
```

```
dep = [910, 1130, 1120, 1200, 1900, 2000] (sorted)
```

Step 1	arr[0]=900 ≤ dep[0]=910: ARRIVE. platforms=1, max=1, i=1
Step 2	arr[1]=940 ≤ dep[0]=910? No, 940>910: DEPART. platforms=0, j=1
	Wait: 940 > 910 → train departs first. platforms = 1-1 = 0
Step 3	arr[1]=940 ≤ dep[1]=1130: ARRIVE. platforms=1, max=1, i=2
Step 4	arr[2]=950 ≤ dep[1]=1130: ARRIVE. platforms=2, max=2, i=3
Step 5	arr[3]=1100 ≤ dep[1]=1130: ARRIVE. platforms=3, max=3, i=4
Step 6	arr[4]=1500 > dep[1]=1130: DEPART. platforms=2, j=2
Step 7	Continue... final max = 3 platforms ✓

■ Java Code

```
return maxPlatforms;  
}
```

Time & Space Complexity

Case	Time	Space	What it really means
All Cases	$O(n \log n)$	$O(1)$	Two sorts dominate; two-pointer scan is $O(n)$

The Core Insight — Independent Sorting

- The key trick: sort arr and dep SEPARATELY (not as pairs)
- We don't care which specific train departs — just that A departure frees a platform
- Think of it as two event streams: arrivals (+1) and departures (-1)
- Running sum of these events gives current platform count at any moment

Why $arr[i] \leq dep[j]$ (not $<$)?

- If a train arrives at the same time another departs: the departing train frees up BEFORE the new one needs a platform
- But we use $\leq$: we count the arrival first (conservative, safe count)
- Both $\leq$ and $<$ give same final answer in most implementations — depends on problem definition

Common Mistakes

- Sorting (arrival, departure) pairs together — DON'T do this, you need them independent
- Loop condition: while $i < n$ (not while $i < n \ \&\& \ j < n$) — process all arrivals
- Forgetting to update maxPlatforms inside the loop (not just at end)

Pro Tips

- Same algorithm solves: max meeting rooms needed, max CPU processes at any time
- Alternative: create events array $[(time, +1), (time, -1)]$, sort, scan — $O(n \log n)$ same
- If asked for which trains share a platform: use a min-heap on departure times instead

Visual Thinking

Picture a live train display board at a station. New rows appear (arrivals), rows disappear (departures).

The number of rows visible at peak is your answer.

Two sorted lists run like two film reels — one showing only arrivals, one only departures.

A cursor sweeps through both simultaneously, comparing the next event.

Animation: timeline with trains as horizontal colored bars.

A vertical red line sweeps right — at peak overlap, 3 bars are cut → 3 platforms.

■ Practice Problems

[Med] $arr=[200,210,300,320,350,500]$, $dep=[230,340,320,430,400,520]$ — min platforms?

[Med] If $arr[i] == dep[j]$ for some i, j , does order of processing matter? Prove it.

[Hard] What if trains can be reassigned (train can use any free platform)? Same answer?

[Hard] Meeting Rooms II (LeetCode #253) — same algorithm with heap variant

■ Quick Revision Box

- Sort $arr[]$ and $dep[]$ independently (not as pairs)
- Two pointers: i for arrivals, j for departures
- $arr[i] \leq dep[j] \rightarrow$ train arrives, $platforms++$
- Else $\rightarrow$ train departs, $platforms--$
- $maxPlatforms = \max$ seen throughout
- $O(n \log n)$ time, $O(1)$ space
- Equivalent to max overlapping intervals problem