THOMAS EAPEN · SELECTED WORK

Counting time without counting it twice

Merge overlapping and contained intervals before summing their duration.

Synthetic example · independently invented data

One-page print edition · A3 landscape

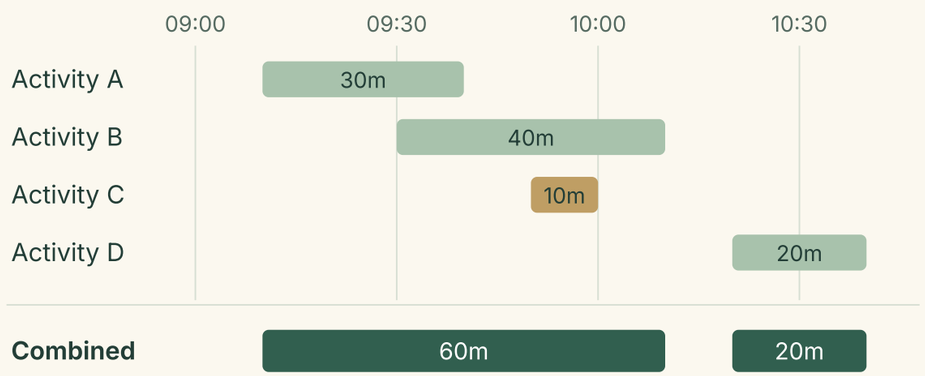
100 min	80 min	20 min	10 min
sum of raw durations	distinct covered time	duplicate coverage removed	gap preserved

SYNTHETIC INPUT / ONE DAY

Four records, two covered spans

Invented activity intervals

Activity	Start	End	Duration
Activity A	09:10	09:40	30 min
Activity B	09:30	10:10	40 min
Activity C	09:50	10:00	10 min
Activity D	10:20	10:40	20 min



Contained intervals add no new coverage; the ten-minute gap remains.

The merged result is 09:10–10:10 and 10:20–10:40: 60 + 20 = 80 minutes.

THE METHOD / PYTHON

Sort and extend the current span

```
def merge_intervals(intervals):
    if any(end < start for start, end in intervals):
        raise ValueError('An interval cannot end before it starts.')
    merged = []
    for start, end in sorted(intervals):
        if start == end:
            continue
        if merged and start <= merged[-1][1]:
            merged[-1][1] = max(merged[-1][1], end)
        else:
            merged.append([start, end])
    return merged
```

Selected code from the new demonstration that computes these results.

- Normalise timestamps to one unit before calculating.
- Reject reversed intervals; zero-length records add no time.
- Retain gaps instead of spanning the entire first-to-last window.

The important distinction Adding record durations gives 100 minutes; measuring the union gives 80. The nested 09:50–10:00 record contributes no additional covered time.

Thomas Eapen
nowlabs.net/tangerine-woa

New illustrative material explaining the approach. Values, configurations and results shown here are not historical workplace records.