

Analysis: Record Breaker

We can check whether the current element is the last element and, if not, if it is greater than the next element in constant time. To check whether i is a record breaking day, we also need to check whether the number of visitors of day i is greater than number of visitors from all the previous days.

Test Set 1

For each element j such that $(1 \leq j < i)$, check that the number of visitors on day j are less than number of visitors on day i . Hence, for each day we would compare it with all the previous days and it would take $O(N)$ time. Therefore, for N days, the time complexity of this solution would be $O(N^2)$.

Test Set 2

However that wouldn't be fast enough for Test Set 2, so we need a faster approach. Instead of comparing the number of visitors of day i against *all* the previous days one by one, we can compare the number of visitors of day i against the *greatest number of visitors* from all previous days. That reduces our processing time for each day from $O(N)$ to $O(1)$. Therefore, for N days, the time complexity of this solution would be $O(N)$, which is sufficiently fast for both Test Set 1 and Test Set 2.

Sample Code (C++)

```
int countRecordBreakingDays(vector<int> visitors) {
    int recordBreaksCount = 0;
    int previousRecord = 0;
    for(int i = 0; i < checkpoints.size(); i++) {
        bool greaterThanPreviousDays = i == 0 || visitors[i] > previousRecord;
        bool greaterThanFollowingDay = i == checkpoints.size()-1 || visitors[i] > visitors[i+1];
        if(greaterThanPreviousDays && greaterThanFollowingDay) {
            recordBreaksCount++;
        }
        previousRecord = max(previousRecord, visitors[i]);
    }
    return recordBreaksCount;
}
```