# Problem 1986: Minimum Number of Work Sessions to Finish the Tasks

## Problem Information

**Difficulty:** Medium
**Acceptance Rate:** 34.24%
**Paid Only:** No
**Tags:** Array, Dynamic Programming, Backtracking, Bit Manipulation, Bitmask

## Problem Description

There are `n` tasks assigned to you. The task times are represented as an integer array `tasks` of length `n`, where the `ith` task takes `tasks[i]` hours to finish. A **work session** is when you work for **at most** `sessionTime` consecutive hours and then take a break.

You should finish the given tasks in a way that satisfies the following conditions:

* If you start a task in a work session, you must complete it in the **same** work session. * You can start a new task **immediately** after finishing the previous one. * You may complete the tasks in **any order**.

Given `tasks` and `sessionTime`, return _the**minimum** number of **work sessions** needed to finish all the tasks following the conditions above._

The tests are generated such that `sessionTime` is **greater** than or **equal** to the **maximum** element in `tasks[i]`.

**Example 1:**

**Input:** tasks = [1,2,3], sessionTime = 3 **Output:** 2 **Explanation:** You can finish the tasks in two work sessions. - First work session: finish the first and the second tasks in 1 + 2 = 3 hours. - Second work session: finish the third task in 3 hours.

**Example 2:**

**Input:** tasks = [3,1,3,1,1], sessionTime = 8 **Output:** 2 **Explanation:** You can finish the tasks in two work sessions. - First work session: finish all the tasks except the last one in 3 + 1 + 3 + 1 = 8 hours. - Second work session: finish the last task in 1 hour.

**Example 3:**

**Input:** tasks = [1,2,3,4,5], sessionTime = 15 **Output:** 1 **Explanation:** You can finish all the tasks in one work session.

**Constraints:**

* `n == tasks.length` * `1 <= n <= 14` * `1 <= tasks[i] <= 10` * `max(tasks[i]) <= sessionTime <= 15`

## Code Snippets

**C++:**

```
class Solution {
public:
int minSessions(vector<int>& tasks, int sessionTime) {

}
};
```

**Java:**

```
class Solution {
public int minSessions(int[] tasks, int sessionTime) {

}
}
```

**Python3:**

```
class Solution:
def minSessions(self, tasks: List[int], sessionTime: int) -> int:
```