Idle Time Problem Solution

Problem

A parallel machine processes N jobs, each with a start and finish time. The goal is to find:

- The largest interval during which the machine is idle (no job is running).
- The largest interval during which the machine is continuously busy (at least one job is running).

Approach

1. Sorting the Jobs

We begin by sorting the jobs based on their start times. This ensures that we can process the jobs sequentially and check for idle gaps between jobs. Sorting the jobs takes $O(N \log N)$ time.

2. Iterating Through the Jobs

Once sorted, we iterate through the jobs and:

- Calculate idle intervals: The gap between the end time of the current job and the start time of the next job is the idle period.
- Calculate busy intervals: Track overlapping or contiguous job periods where the machine is continuously running.

3. Finding Largest Intervals

We track the largest idle interval and the largest busy interval during the iteration.

Python Code Implementation

```python
def find_intervals(jobs):
    # Sort jobs by their start times
    jobs.sort()

    largest_idle = 0
    largest_busy = 0

    # Initialize with the first job
    current_start, current_end = jobs[0]

    for i in range(1, len(jobs)):
        next_start, next_end = jobs[i]

        if next_start > current_end:
            # There's an idle period
            idle_time = next_start - current_end
            largest_idle = max(largest_idle, idle_time)

            # Update current interval to the next job's interval
            current_start, current_end = next_start, next_end
        else:
            # There's an overlap or contiguous busy time
            current_end = max(current_end, next_end)

        # Calculate the busy time for the current busy interval
        busy_time = current_end - current_start
        largest_busy = max(largest_busy, busy_time)

    return largest_idle, largest_busy

# Example usage:
jobs = [(1, 3), (5, 6), (2, 7), (9, 12)]
largest_idle, largest_busy = find_intervals(jobs)
print(f"Largest idle interval: {largest_idle}")
print(f"Largest busy interval: {largest_busy}")
```

Explanation

- **Input:**

  - `jobs`: A list of tuples, where each tuple represents a job's start and end time, e.g., `[(start1, end1), (start2, end2), ...]`.

- **Sorting:** The jobs are sorted by their start times to check for idle gaps and continuous busy intervals sequentially.
- **Iteration:** For each pair of consecutive jobs, we check if there is an idle period between the end of the current job and the start of the next job. We also track the longest continuous busy interval.
- **Return Values:** The function returns two values:
  - `largest_idle`: The largest idle interval.
  - `largest_busy`: The largest continuous busy interval.

Time Complexity

- **Sorting:** Sorting the jobs takes $O(N \log N)$, where N is the number of jobs.
- **Iteration:** Iterating through the jobs takes $O(N)$.
- **Overall:** The total time complexity is $O(N \log N)$.

Example

For the input jobs:

$$\text{jobs} = [(1, 3), (5, 6), (2, 7), (9, 12)]$$

- The largest idle interval is between time 7 and 9 (length = 2).
- The largest busy interval is from time 1 to 7 (length = 6).

The output will be:

```
Largest idle interval: 2
Largest busy interval: 6
```