Registrar Scheduling Conflict Detection

The registrar can prevent scheduling conflicts by using a **time slot conflict detection method**. Here's a detailed approach to ensure that no instructor is double-booked.

Conflict Detection Method

1. **Data Representation**:
   - Use a dictionary (or hash table) where the keys represent the available time slots and the values are sets of instructors scheduled for those time slots.
   - The time slots are: `9:00`, `10:00`, `11:00`, `1:00`, `2:00`, and `3:00`.
2. **Algorithm**:
   - For each class being scheduled:
     (a) Check if the instructor is already listed in the set for the corresponding time slot.
     (b) If the instructor is present, a conflict has been detected.
     (c) If the instructor is not present, add the instructor to the set for that time slot.
3. **Step-by-Step Implementation**:
   - Initialize a dictionary with empty sets for each time slot.
   - As you iterate over the class schedules, check and update the sets accordingly.

Example Code in Python

```
def check_schedule_conflicts(schedule):
    # Initialize dictionary with sets for each time slot
    time_slots = {
        "9:00": set(),
        "10:00": set(),
        "11:00": set(),
        "1:00": set(),
        "2:00": set(),
        "3:00": set()
    }

    # Iterate over the schedule
    for class_time, instructor in schedule:
        if instructor in time_slots[class_time]:
            print(f"Conflict␣detected␣for␣instructor␣{instructor}␣at␣{class_ti
            return True  # Conflict found
        else:
            time_slots[class_time].add(instructor)

    print("No␣conflicts␣found.")
    return False  # No conflict

# Example schedule (time, instructor)
schedule = [
    ("9:00", "Prof.␣Smith"),
    ("10:00", "Prof.␣Johnson"),
    ("10:00", "Prof.␣Smith"),
    ("11:00", "Prof.␣Smith"),
    ("1:00", "Prof.␣Johnson"),
    ("2:00", "Prof.␣Smith"),
    ("3:00", "Prof.␣Johnson"),
    ("10:00", "Prof.␣Smith")  # Conflict: Prof. Smith at 10:00
]

# Check for conflicts
check_schedule_conflicts(schedule)
```

Explanation

- **Data Structure**: The dictionary `time_slots` uses time slots as keys and sets of instructors to track who is teaching at each time.
- **Conflict Detection**: The method checks if an instructor is already in the set for the given time slot, indicating a conflict.

Complexity Analysis

- **Time Complexity**: $O(N)$, where N is the number of class schedules. Each insertion and lookup operation in a set is $O(1)$, making the overall complexity linear.
- **Space Complexity**: $O(K)$, where K is the number of unique time slots (in this case, 6 time slots). The space needed to store the sets is minimal and fixed.

Handling Edge Cases

- **No Classes Scheduled**: The algorithm should handle an empty schedule gracefully.
- **Multiple Classes for Different Instructors**: If two different instructors are scheduled for the same time slot, it is not a conflict.
- **Large Number of Instructors**: The set data structure ensures efficient handling of multiple instructors.

Conclusion

This method efficiently checks for scheduling conflicts by using a dictionary of sets to track which instructors are assigned to each time slot. If an instructor is found to be double-booked, the conflict is detected and reported. This approach is simple, efficient, and suitable for the given constraints.