DATE:01/08/2025

EXP:04

CONSTRAINT SATISFACTION PROBLEM

AIM:

Problem statement
Design a timetable for a university department that schedules classes for a given set of courses, ensuring no conflicts for students, teachers, or rooms.

Objective:

Assign time slots and classrooms to a set of university courses such that:

1. **No teacher** is scheduled to teach more than one course at the same time.
2. **No student** is assigned to attend more than one class at the same time.
3. **No classroom** is used for more than one course at the same time.
4. Each course is assigned to a classroom with enough capacity to hold all enrolled students.
5. Courses must be scheduled only within the department's working hours (e.g., 9:00 AM to 5:00 PM).
6. Certain professors may have availability constraints (e.g., Prof. A cannot teach on Friday afternoons).

ALGORITHM:

Algorithm: CSP function

```
BACKTRACKING_SEARCH(CSP):
    return BACKTRACK({}, CSP)        // start with empty assignment
```

```
function BACKTRACK(assignment, CSP):
   if all variables are assigned in assignment:
return assignment             // solution found
   X ← SELECT_UNASSIGNED_VARIABLE(CSP, assignment)     for
each value v in ORDER_DOMAIN_VALUES(X, CSP, assignment):
      if v is consistent with assignment and constraints:
         add {X = v} to assignment             inferences ← INFERENCE(CSP, X,
v)   // e.g. forward checking, AC-3
         if inferences ≠ failure:
            add inferences to assignment
result ← BACKTRACK(assignment, CSP)
            if result ≠ failure:
return result
         remove {X = v} and inferences from assignment
return failure
```

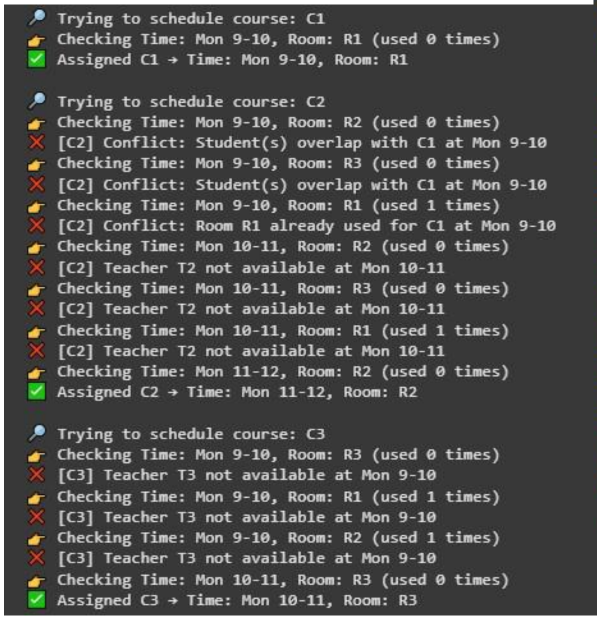
CODE:

```
from collections import defaultdict
# Time slots available time_slots = ['Mon 9-10', 'Mon 10-11', 'Mon 11-12',
'Tue 9-10', 'Tue 10-11'] # Room availability (same for all) rooms = {
   'R1': set(time_slots),
    'R2': set(time_slots),
    'R3': set(time_slots)
}
# Courses and their assigned teacher + students courses
= {
    'C1': {'teacher': 'T1', 'students': ['S1', 'S2', 'S3']},
    'C2': {'teacher': 'T2', 'students': ['S2', 'S4']},
    'C3': {'teacher': 'T3', 'students': ['S1', 'S4']},
    'C4': {'teacher': 'T1', 'students': ['S3', 'S5']},
    'C5': {'teacher': 'T2', 'students': ['S5', 'S1']}
}
# Teacher availability teachers
= {
    'T1': ['Mon 9-10', 'Mon 10-11', 'Tue 9-10'],
    'T2': ['Mon 9-10', 'Mon 11-12', 'Tue 10-11'],
    'T3': ['Mon 10-11', 'Tue 9-10', 'Tue 10-11']
}
# Assignments + room usage tracking assignments
= {}  # course -> (time, room) room_usage =
defaultdict(int)

# Check if this time-room assignment is valid for the course def
is_valid(course, time, room):
  teacher = courses[course]['teacher']

students = courses[course]['students']

# Check teacher availability     if time

not in teachers[teacher]:

      print(f"   [{course}] Teacher {teacher} not available at {time}")

return False

  # Check room availability

if time not in rooms[room]:

      print(f"   [{course}] Room {room} not available at {time}")

return False

  # Check for conflicts with already scheduled courses     for other_course,

(assigned_time, assigned_room) in assignments.items():

      other_teacher = courses[other_course]['teacher']

other_students = courses[other_course]['students']

      # Teacher conflict        if time == assigned_time and

teacher == other_teacher:

         print(f"   [{course}] Conflict: Teacher {teacher} already teaching {other_course} at
{time}")

         return False        # Room conflict        if time ==

assigned_time and room == assigned_room:           print(f"

[{course}] Conflict: Room {room} already used for

{other_course} at

{time}")
```

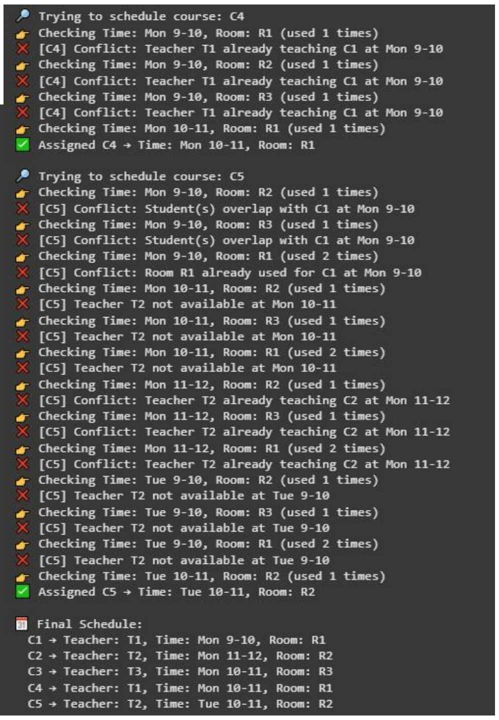
```
        return False         # Student conflict         if time == assigned_time
and any(s in students for s in other_students):
        print(f"   [{course}] Conflict: Student(s) overlap with {other_course} at {time}")
return False     return True
# Backtracking search def backtrack(index,
course_list):     if index == len(course_list):
return True  # all courses assigned     course =
course_list[index]     print(f"\n   Trying to schedule
course: {course}")     for time in time_slots:
      # Choose rooms with least usage first         sorted_rooms =
sorted(rooms.keys(), key=lambda r: room_usage[r])         for room in
sorted_rooms:
        print(f"   Checking Time: {time}, Room: {room} (used {room_usage[room]} times)")
if is_valid(course, time, room):
          # Assign course
assignments[course] = (time, room)
room_usage[room] += 1

          print(f"   Assigned {course} → Time: {time}, Room: {room}")


          if backtrack(index + 1, course_list):
             return True
          # Backtrack                print(f"   Backtracking {course} from Time:
{time}, Room: {room}")                del assignments[course]
```

```
room_usage[room] -= 1     print(f"   No valid assignment found for {course}")

return False # Driver course_list = list(courses.keys()) if backtrack(0,

course_list):     print("\n   Final Schedule:")     for c in sorted(assignments.keys()):

        t = courses[c]['teacher']         tm, rm = assignments[c]

print(f"  {c} → Teacher: {t}, Time: {tm}, Room: {rm}") else:

    print("   No valid schedule could be generated.")
```

OUTPUT:



```
🔎 Trying to schedule course: C1
👉 Checking Time: Mon 9-10, Room: R1 (used 0 times)
✅ Assigned C1 → Time: Mon 9-10, Room: R1

🔎 Trying to schedule course: C2
👉 Checking Time: Mon 9-10, Room: R2 (used 0 times)
❌ [C2] Conflict: Student(s) overlap with C1 at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R3 (used 0 times)
❌ [C2] Conflict: Student(s) overlap with C1 at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R1 (used 1 times)
❌ [C2] Conflict: Room R1 already used for C1 at Mon 9-10
👉 Checking Time: Mon 10-11, Room: R2 (used 0 times)
❌ [C2] Teacher T2 not available at Mon 10-11
👉 Checking Time: Mon 10-11, Room: R3 (used 0 times)
❌ [C2] Teacher T2 not available at Mon 10-11
👉 Checking Time: Mon 10-11, Room: R1 (used 1 times)
❌ [C2] Teacher T2 not available at Mon 10-11
👉 Checking Time: Mon 11-12, Room: R2 (used 0 times)
✅ Assigned C2 → Time: Mon 11-12, Room: R2

🔎 Trying to schedule course: C3
👉 Checking Time: Mon 9-10, Room: R3 (used 0 times)
❌ [C3] Teacher T3 not available at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R1 (used 1 times)
❌ [C3] Teacher T3 not available at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R2 (used 1 times)
❌ [C3] Teacher T3 not available at Mon 9-10
👉 Checking Time: Mon 10-11, Room: R3 (used 0 times)
✅ Assigned C3 → Time: Mon 10-11, Room: R3
```



```
🔎 Trying to schedule course: C4
👉 Checking Time: Mon 9-10, Room: R1 (used 1 times)
❌ [C4] Conflict: Teacher T1 already teaching C1 at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R2 (used 1 times)
❌ [C4] Conflict: Teacher T1 already teaching C1 at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R3 (used 1 times)
❌ [C4] Conflict: Teacher T1 already teaching C1 at Mon 9-10
👉 Checking Time: Mon 10-11, Room: R1 (used 1 times)
✅ Assigned C4 → Time: Mon 10-11, Room: R1

🔎 Trying to schedule course: C5
👉 Checking Time: Mon 9-10, Room: R2 (used 1 times)
❌ [C5] Conflict: Student(s) overlap with C1 at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R3 (used 1 times)
❌ [C5] Conflict: Student(s) overlap with C1 at Mon 9-10
👉 Checking Time: Mon 9-10, Room: R1 (used 2 times)
❌ [C5] Conflict: Room R1 already used for C1 at Mon 9-10
👉 Checking Time: Mon 10-11, Room: R2 (used 1 times)
❌ [C5] Teacher T2 not available at Mon 10-11
👉 Checking Time: Mon 10-11, Room: R3 (used 1 times)
❌ [C5] Teacher T2 not available at Mon 10-11
👉 Checking Time: Mon 10-11, Room: R1 (used 2 times)
❌ [C5] Teacher T2 not available at Mon 10-11
👉 Checking Time: Mon 11-12, Room: R2 (used 1 times)
❌ [C5] Conflict: Teacher T2 already teaching C2 at Mon 11-12
👉 Checking Time: Mon 11-12, Room: R3 (used 1 times)
❌ [C5] Conflict: Teacher T2 already teaching C2 at Mon 11-12
👉 Checking Time: Mon 11-12, Room: R1 (used 2 times)
❌ [C5] Conflict: Teacher T2 already teaching C2 at Mon 11-12
👉 Checking Time: Tue 9-10, Room: R2 (used 1 times)
❌ [C5] Teacher T2 not available at Tue 9-10
👉 Checking Time: Tue 9-10, Room: R3 (used 1 times)
❌ [C5] Teacher T2 not available at Tue 9-10
👉 Checking Time: Tue 9-10, Room: R1 (used 2 times)
❌ [C5] Teacher T2 not available at Tue 9-10
👉 Checking Time: Tue 10-11, Room: R2 (used 1 times)
✅ Assigned C5 → Time: Tue 10-11, Room: R2

📅 Final Schedule:
  C1 → Teacher: T1, Time: Mon 9-10, Room: R1
  C2 → Teacher: T2, Time: Mon 11-12, Room: R2
  C3 → Teacher: T3, Time: Mon 10-11, Room: R3
  C4 → Teacher: T1, Time: Mon 10-11, Room: R1
  C5 → Teacher: T2, Time: Tue 10-11, Room: R2
```

RESULT:

The programs have been completed and the outputs have been verified.