# Problem: Matching Group Schedules

The group schedule matching takes two or more arrays as input. The arrays represent slots that are already booked and login/logout time of group members. It outputs an array containing intervals of time when all members are available for a meeting for a minimum duration expected.

The group schedule matching takes following inputs:

1. Busy\_Schedule: A list of list that represent the persons existing schedule (they can’t plan any other engagement during these hours)
2. Working\_period: Daily working periods of group members. (login,logout)
3. Duration of the meeting

It outputs a list of list containing intervals of time when all members are available for a meeting for the minimum duration of the meeting required.

Assume there are two persons in your class project group. You want to schedule a meeting with another group member. The members decide to provide you with (a) a schedule of their daily activities, containing times of planned engagements. They are not available during these periods. (b) the earliest and latest times at which they are available for meetings daily. Your schedule and availability are provided too.

Write an algorithm that takes in your schedule, your daily availability (earliest time, latest time) and that of your group member (or members), and the duration of the meeting you want to schedule. Time is given and should be returned in military format. For example: 9:30, 22:21. The given times (output) should be sorted in ascending order.

Inputs are also in sorted order.

An algorithm for solving this problem involves combing the two sub-arrays into an array containing of a set unavailability, with consideration of the daily active periods.

**Sample input**

person1\_busy\_Schedule =[ [’12:00’, ’13:00’], [’16:00’, ’18:00’]]

person1\_work\_hours = [‘9:00’, ’19:00’]

person2\_busy\_Schedule = [[ ‘9:00’, ’10:30’], [’12:20’, ’14:30’], [’14:30’, ’15:00’], [’16:00’, ’17:00’ ]]

person2\_work\_hours = [‘9:00’, ’18: 30’]

duration\_of\_meeting =30

**Sample output**

[[’10:30’, ’12:00’], [’15:00’, ’16:00’], [’18:00’, ’18:30’]]

# Implementation

Have following files

1. Project1\_starter.py or project1\_starter.cpp that defines functions for the algorithm described above. You will need to develop and write the functions. Describe how to run your program
2. Input.txt containing the sample input files. Use these sample files to run your program to see whether your algorithm implementations are working correctly. Have a new line character separating the sample test cases (10)
3. Output.txt – load the sample test case result to output.txt

# To Do

1. Study the sample input and output above. Write your own complete and clear code for an algorithm to solve this problem.
2. Implement your algorithm using either Python or C++.
3. Run your code using different data inputs
4. A screenshot showing the output of your code for a minimum of 10 test cases defined by yourself.

# Mathematical Analysis

Analyze your algorithm mathematically. You should prove a specific big-O efficiency class for the algorithm. The analysis should be routine, similar to the ones we have done in class and in the textbook. The algorithm’s efficiency class will be one of or

# Grading Rubric

The suggested grading rubric is below.

1. Algorithm design and implementation = 50 points, divided as follows:
   1. Clear and complete code = 20 points
   2. Complete and clear README.md file = 3 points
   3. Successful compilation = 15 points
   4. Produces accurate result = 12 points
2. Analysis = 50 points, divided as follows
   1. Mathematical analysis and proof, including step count =22
   2. Report document presentation = 20 points
   3. Screenshot = 5 points
   4. Comments on possible improvements = 3

# Submitting your code

Submit your project as a zip folder with following format <Team\_member\_name1\_member\_2\_member\_3>.zip to the Project 1 link on Canvas. It allows for multiple submissions.

Include following files in the zip folder:

Readme

Input.txt

Project1\_starter.py or project1\_starter.cpp

Output.txt