**Project 2 CPSC 335**

**Matching Group Schedules**

Authors: *Joseph Bobby, Conrad Chu, Cyril Youssef, Jonathan Bird*

Emails: jbobby0@csu.fullerton; cbchu@csu.fullerton.edu; cyrily@csu.fullerton.edu; birdrjonathan@csu.fullerton.edu

Github: https://github.com/cyrilyoussef/Proj2_335

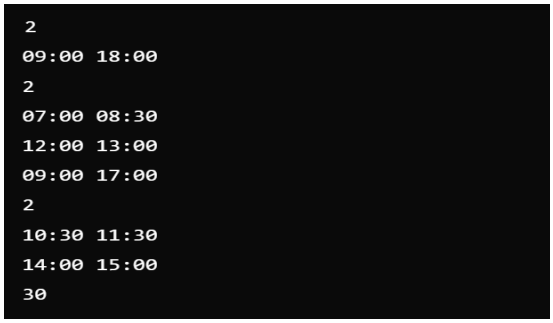
11/10/2024

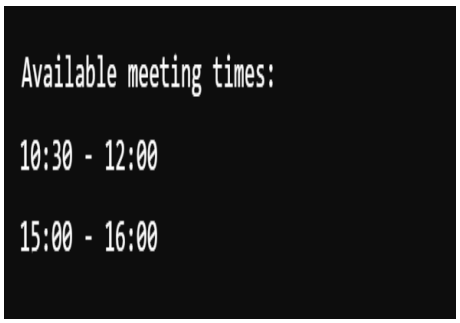
**Abstract:**

This is a python program that implements an algorithm to determine common available time slots among a group of individuals based on their daily schedules and availability windows. The inputs given are decided by the user, with multiple schedules and login/logout times, the program outputs intervals when all group members are free for a specified minimum meeting duration. The solution integrates efficient data structures, including 2D arrays and hash maps, to handle input schedules and compute overlapping free periods. By parsing schedules and aligning availability times, the algorithm identifies and returns intervals that are the minimum meeting duration requirements. The program contains; structured functions for schedule processing and meeting slot extraction, and is tested with various input cases, including edge cases.

**Test Cases:**

**1. Simple Overlap**




**2. Complex Overlap**

```
3
09:00 19:00
3
08:00 09:30
12:00 13:30
16:00 17:30
09:00 18:30
2
10:00 11:30
14:00 15:30
09:30 17:30
3
09:00 09:45
11:00 12:15
16:00 16:30
45
```

```
Available meeting times:

13:30 - 14:00
```

### 3. No Overlap

```
2
09:00 17:00
1
09:00 17:00
09:00 17:00
1
09:00 17:00
30
```

```
No common available times meet the required duration.
```

### 4. Minimal Free Time (Edge Case)

```
2
09:00 18:00
2
10:00 12:00
13:30 15:00
09:00 18:00
2
08:00 10:30
15:00 17:30
10
```

```
Available meeting times:

10:30 - 12:00

13:00 - 13:30

15:00 - 15:30
```

**5. Large Meeting Duration (Edge Case)**

```
2
08:00 20:00
3
08:30 10:00
11:00 12:30
15:30 18:00
08:00 19:00
2
09:00 11:00
14:30 16:00
90
```

```
Available meeting times:

11:00 - 12:30
```

**6.**

```
3
00:00 23:59
2
02:00 04:00
06:00 08:00
00:00 23:59
2
01:00 03:00
07:00 09:00
00:00 23:59
1
10:00 12:00
30
```

```
Available meeting times:

12:00 - 23:59
```

**7.**

```
2
09:00 17:00
1
10:00 12:00
09:00 17:00
1
10:30 12:30
5
```

```
Available meeting times:

09:00 - 10:00

12:30 - 17:00
```

**8.**

```
2
09:00 17:00
2
11:00 11:30
13:00 13:30
09:00 17:00
2
11:15 11:45
13:15 13:45
15
```

```
Available meeting times:

11:45 - 13:00
```

**9.**

```
3
08:00 16:00
2
09:00 09:30
14:00 14:30
08:30 16:00
2
08:30 10:00
15:00 16:00
08:00 15:30
1
12:00 12:30
20
```

```
Available meeting times:

10:00 - 12:00
```

**10.**

```
2
06:00 20:00
1
07:00 08:00
06:00 20:00
1
09:00 10:00
60
```

```
Available meeting times:

10:00 - 20:00
```

**Time Complexity:**

The time complexity of the entire algorithm, assuming n is the maximum number of busy intervals per participant and m is the maximum number of free intervals per participant after processing, is: O(p · n + p^2 · m). In most cases, the O(p^2 · m) term dominates, especially as the number of participants increases.

Step Count Summary - For each participant:

- Convert times in busy intervals to minutes (constant-time operations per interval).
- Calculate free intervals by iterating through busy intervals (O(n)).
- Compute common intervals with other participants (O(p^2 · m)).

This complexity is efficient for small to moderately-sized inputs but may scale with noticeable increases in runtime if there are many participants and intervals.