Available Meeting Slot Finder

By: Ethan Paransky(<u>eparansky@csu.fullerton.edu</u>), Jing Nie(<u>jingnie@csu.fullerton.edu</u>), Leonel Noriega-Rojas(<u>leonelnr8@csu.fullerton.edu</u>)

This Python program finds mutually available time slots for a meeting between two individuals based on their respective busy schedules and working hours.

Features

- Convert Time Formats: Converts time strings (e.g., "09:30") to minutes and vice versa for easier calculations.
- **Identify Unbusy Slots**: Determines free time slots between meetings for each individual within their specified working hours.
- **Find Common Slots**: Finds overlapping free time between two people that can accommodate a given meeting duration.
- Filter by Working Periods: Ensures available slots fall within each person's working hours.

How It Works

- 1. **Convert Schedules**: All time data (busy schedules and working periods) are converted from "HH" format to minutes.
- 2. **Calculate Free Slots**: For each person, free time slots are calculated by comparing busy times against their working hours.
- 3. **Find Common Availability**: Using a two-pointer approach, the code identifies overlapping free time slots between two people.
- 4. **Filter by Duration**: Ensures the common slots meet the required meeting duration.

Functions

- time to minutes (time str): Converts time from "HH" format to total minutes.
- minutes to time (minutes): Converts minutes back to "HH" format.
- find_unbusy_slots (busy_schedule, daily_act): Calculates free time slots for a person based on their busy schedule and daily working hours.
- find_common_unbusy_slots(unbusy1, unbusy2, duration): Finds mutually free slots between two people that meet the required duration.
- filter by working period(common slots, working periods): Ensures the

free slots fall within specified working periods.

• find_available_slots(busy_schedules, working_periods, duration):
Combines all functions to find and return available meeting slots in "HH\" format.

Usage

Run the find available slots function with the following parameters:

- busy_schedules: A list of busy schedules for each person (list of lists in "HH " format).
- working_periods: Each person's daily working period as a start and end time in "HH"
 "format.
- duration: Desired meeting duration in minutes.

Analyze

Efficiency Class:

The primary efficiency gain comes from the two-pointer technique, which avoids nest loops for this case.

find_unbusy_slots:

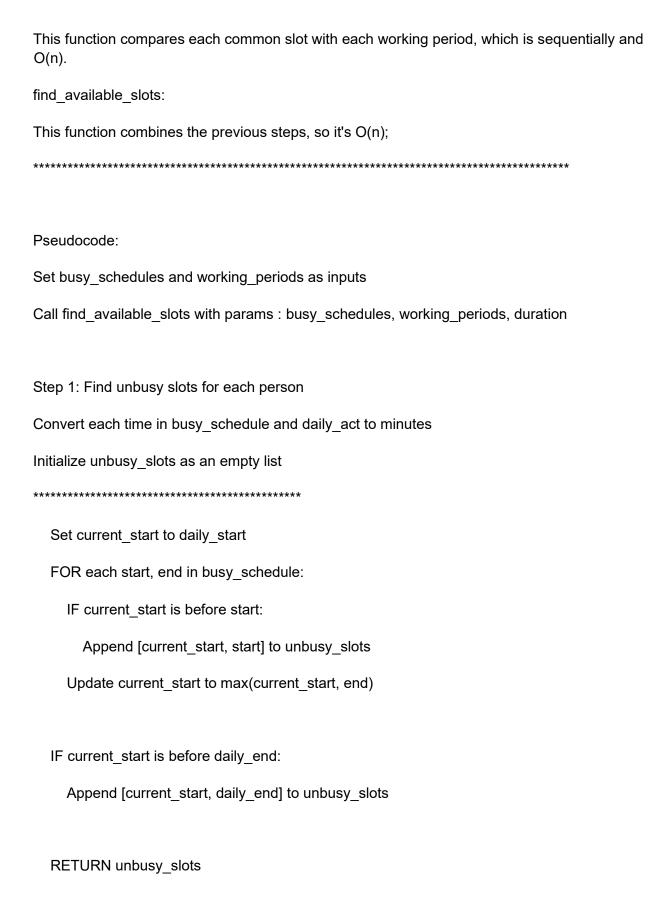
The function loop through the busy schedule, which has a complexity of O(n).

It calculates unbusy slots by processing each interval sequentially.

find common unbusy slots:

The function uses a two-pointer technique, which means each element from unbusy1 and unbusy2 is processed once, which is O(n).

filter by working period:



Using it find the unbusy slots for person1 and person2 Step 2: Find common unbusy slots between the first two schedules using find common unbusy slots ****************** Initialize common_slots as an empty list Set i, j to 0 WHILE i < length of unbusy1 AND j < length of unbusy2: Set start1, end1 to unbusy1[i] Set start2, end2 to unbusy2[j] Calculate common start as max(start1, start2) Calculate common end as min(end1, end2) IF common start is before common end AND duration fits within this slot: Append [common start, common end] to common slots Move pointer with earlier ending slot RETURN common slots ****************** Step 3: Filter common slots based on working periods ****************** Initialize filtered slots as an empty list FOR each start, end in common slots: FOR each work start, work end in working periods:

Calculate common_start as max(start, work_start)

Calculate common_end as min(end, work_end)

IF common_start is before common_end:

Append [common_start, common_end] to filtered_slots

Remove duplicates from filtered_slots

RETURN unique_filtered_slots

Step 4: Convert available slots back to time format

Print the available slots