Generate a List of Available Time for Meetings

By Boxiang Lin

Input:

Person1: `Given a nested list of current schedule person1 currently hold; a list of person's daily bound`

Person2: `Given a nested list of current schedule person1 currently hold; a list of person's daily bound`

Meeting_time frame: `30 minutes at least`

Output:

Return a nested list of all available time to schedule meetings for Person1 and Person2

example:

```
Input:
p1_schedule = [['9:00', '10:45'], ['10:45', '13:00'], ['16:00', '18:00'],
['18:30','19:00']]
p2_schedule = [['10:00', '10:30'], ['10:30', '14:30'], ['14:30', '15:00'],
['16:00', '17:00']]
p1_daily_bound = ['9:00','20:00']
p2_daily_bound = ['10:00','18:30']
meeting_time = 30

output: [['15:00', '16:00'], ['18:00', '18:30']]
```

Pre 1. Formal Definition of Input

Suppose P is a list of schedules for a particular day that contains $t^N_{(i,j)}$ where t is a schedule between i to j time at Nth index of P
We are also given a list of B that map to a P, where B contains two values B_0 *and* B_1
Now, each input comply with the conditions below:

$$p1.1)\ B_0 \leq [t^N_{(i,j)} \in P] \leq B_1$$
$$p1.2)\ t_i < t_j \leq t_{i+1} < t_{j+1} \leq t_{N=i} < t_{N=j}$$

Pre 2. Formal Definition of Output

Suppose O list contains time segments $T^M_{(i,j)}$, O should be comply with below conditions

Suppose, $\Delta t1_N = (t1_{j=0}, t1_{i=1}), \ldots\ldots, (t1_{j=N-1}, t1_{i=N})$, so is $\Delta t2_N$
$p2.1)\ \forall T^M_{(i,j)} \not\subset (\Delta t1_N + \Delta t2_N)$
$p2.2)\ Max(B(0)_0, B(1)_0) \leq \forall\ T^M_{(i,j)} \leq Min(B(0)_0, B(1)_1)$ where B(X), X represent a particular person
$p2.3)\ T_i < T_j \leq T_{i+1} < T_{j+1}$

Step 1. Determine The Final Bound Between Two Persons

Pick the largest B_0 and smallest B_1 between B(X) to be Final Bound, we have to use this to cut off the out-of-bound time segments.

$$s1.1) finalBound = [max(B(0)_0, B(1)_0, min(B(0)_1, B(1)_1)]$$

Step 2. Combine Both P lists

Since we know schedules of each P in sorted. We could merge both P lists by sorted order of t_i. Let the merge list called C, then C has this unique condition to be aware of:

$$s2.1)\ \forall C_{(i,j)}^{N+M=S}, C_i \leq C_{i+1} \text{ holds, but } C_j \nleq C_{i+1}\ and\ C_j \nleq C_{j+1} \text{ turns out to be not necessary}$$

```
Take the example above:
C = [['9:00', '10:45'], ['10:00', '10:30'], ['10:30', '14:30'], ['10:45',
'13:00'], ['14:30', '15:00'], ['16:00', '17:00'], ['16:00', '18:00'], ['18:30',
'19:00']]
```

Notice from above example:

$$C_j^0('10:45') > C_i^1\$('10:00')and\$C_j^0('10:45') > Cj^1('10:30')$$

Step 3. Trim The Combine List

We have to trim the Combine List, the purpose is to trim the C list to below condition:

$s3.1)$Place the upper bound to first index $C = [finalBound[0], finalBound[0]] + C_{i,j}^S$
$s3.2)\ \forall C_i^S, \forall C_j^S \leq finalBound[1]$, To acheive this we have to do the followings combine list modifications :

$$= \begin{cases} C_i^S \geq finalBound[1], & \text{Remove current } C_{i,j}^S \\ C_i^S \leq finalBound[1]\ and\ C_j^S > finalBound[1], & finalBound[1] = C_i^s \text{ (We pick the smallest possible i) and } C_j^S = C_i^S \end{cases}$$

$s3.3)$Append [finalBound[1],finalBound[1]] to C

finalBound for the example is = ['10:00','18:30']:

After s3.1, s3.2, s.3.3, C for the above example will be:

```
C = [['10:00', '10:00'], ['9:00', '10:45'], ['10:00', '10:30'], ['10:30',
'14:30'], ['10:45', '13:00'], ['14:30', '15:00'], ['16:00', '17:00'], ['16:00',
'18:00'], ['18:30', '18:30']]
```

Step 4. Main Logic: Compute the O (Output)

Lets first assuming our C has this inequality relation

$C_i \leq C_j \leq C_{i+1} \leq C_{j+1}$ then we are confident to say that each $[C_j, C_{i+1}] = T_{i=j,j=i+1}\ if \geq 30\ minutes$

is a solution. Why?

Proof
Given the above inequality C possess of, and indecies are defined and bounded $i, j \in \mathbb{Z}, 0 \leq i, j \leq S$;
We consider a set of time points between this range (C_i, C_j) are the only no solutions for any i, j.
We also know that, by inequality, the set of points between (C_{i+x}, C_{j+x}) will never collapse with any otherwise (C_i, C_j).
Then, we see a set of time points between $[C_j, C_{i+1}]$ for any i, j does not collapse with any $(C_i, C_j), [C_{j+x}, C_{i=j+x+1}]$ as well
Hence, $\forall [C_{j+1,}, C_i]$ are the solutions because by set differences $1 - (C_i^S, C_j^S)$.

However, this is not the case, recall (s2.1), we have these two uncertain condition

$$C_j \nleq C_{i+1} \; and \; C_j \nleq C_{j+1}$$

Which turns out if

$$C_j > C_{i+1} \; or \; C_j > C_{j+1}, the \; [C_j, C_{i+1}]$$

is not a solution, intuitively, that the first case resulting negative time range, and second case producing overlaps which shows

$$(C_i, C_j) \subseteq [C_j, C_{i+1}]$$

Mechanism is to loop through the C list from index 0 to S and do the followings each iteration:

$$= \begin{cases} C_j > C_{i+1} \; and \; C_j \leq C_{j+1}, & \text{Skip Current } C_{i,j} \\ C_j > C_{i+1} \; and \; C_j > C_{j+1}, & \text{Skip Current } C_{i,j} \text{ and marked } C_{j+1} = C_j \\ C_j \leq C_{i+1} \; then \; of \; course \; C_j \leq C_{j+1}, & [C_j, C_{i+1}] \text{ is solution if } \geq 30 \; min \end{cases}$$

Python Implementation

```
###################################---MAIN FUNCTION ---
###################################
def aSchedule(p1S, p2S, p1B, p2B, meet_time):
    res = []
    #Step 1
    final_bound = dailyBound(p1B, p2B)
    #Step 2 and 3
    combineS = combine(p1S, p2S,final_bound)
    temp_end = []

    #Step 4
    for i in range(1, len(combineS)):
        if temp_end:
            prev_end = temp_end[0]
            temp_end.pop()
        else:
            prev_end = combineS[i - 1][1]
        start = combineS[i][0]
        end = combineS[i][1]
        # Main Logic-----
        if compare(start, prev_end) <= 0:
            # if end < prev_end, prev_end should be the next (prev_end) to
compare with the start
            if compare(end,prev_end) < 0:
                temp_end.append(prev_end)

        elif compare(start, prev_end) > 0:  # start > prev_end
            if difference(prev_end, start) >= meet_time:  # start - prev_end >=
meet_time
```

```
                res.append([prev_end, start])

    return res

################################################################################
###
def dailyBound(p1B, p2B):
    finalBound = []
    # pick largest start
    if compare(p1B[0],p2B[0]) >= 0: finalBound.append(p1B[0])
    else: finalBound.append(p2B[0])

    # pick smallest start
    if compare(p1B[1],p2B[1]) <= 0: finalBound.append(p1B[1])
    else: finalBound.append(p2B[1])
    return finalBound

#
# <<Helper>> combine two sorted nested array
#
def combine(p1S, p2S,final_bound):
    combine = []
    combine.append([final_bound[0],final_bound[0]])
    i, j = 0, 0
    while i < len(p1S) and j < len(p2S):
        if compare(p1S[i][0], p2S[j][0]) < 0:
            combine.append(p1S[i])
            i += 1
        else:
            combine.append(p2S[j])
            j += 1

    while i < len(p1S):
        combine.append(p1S[i])
        i += 1

    while j < len(p2S):
        combine.append(p2S[j])
        j += 1

    # Remove the largest schedule beyong the final_bound and search for
potentially new end bound
    new_bound = []
    for i in range(len(combine)):
        # when start < finalbound but end >= finalbound, [i] = [start,start]
        if compare(combine[i][0],final_bound[1]) <= 0 and compare(combine[i][1],
final_bound[1]) > 0:
            combine[i] = [combine[i][0],combine[i][0]]
            if not new_bound: #we only take the first encounter one to be new
bound
                new_bound.append(combine[i][0])
            final_bound[1] = new_bound[0]
        # since the prev if condition valid, this will not execute, but next
loops it will remove start >= finalBound
        if compare(combine[i][0],final_bound[1]) >= 0:
            combine.remove(combine[i])

    combine.append([final_bound[1],final_bound[1]])
```

```
    return combine

#
# <<Helper>> compare
#
def compare(t1, t2):
    hr1, min1 = t1.split(':')
    hr2, min2 = t2.split(':')
    t1 = int(hr1) * 60 + int(min1)  # t1 total minutes
    t2 = int(hr2) * 60 + int(min2)  # t2 total minutes
    if t1 < t2:
        return -1
    elif t1 > t2:
        return 1
    else:
        return 0


def difference(t1, t2):
    hr1, min1 = t1.split(':')
    hr2, min2 = t2.split(':')
    t1 = int(hr1) * 60 + int(min1)  # t1 total minutes
    t2 = int(hr2) * 60 + int(min2)  # t2 total minutes
    return t2 - t1


p1S = [['9:00', '10:45'], ['10:45', '13:00'], ['16:00', '18:00'],
['18:30','19:00']]
p2S = [['10:00', '10:30'], ['10:30', '14:30'], ['14:30', '15:00'], ['16:00',
'17:00']]
p1B = ['9:00','20:00']
p2B = ['10:00','18:30']
print(aSchedule(p1S, p2S, p1B, p2B, 30))
```