Esteban Zapata

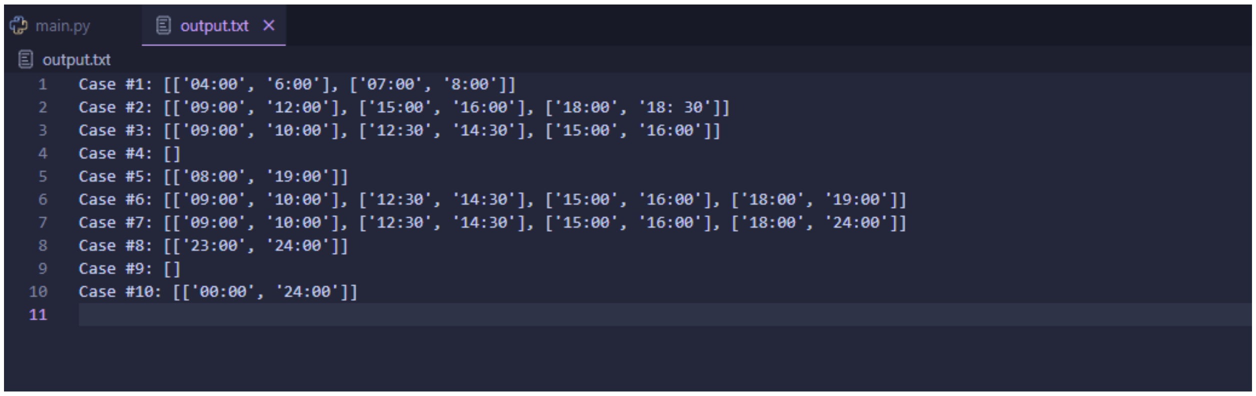
Raxel Ortiz

Jacob Corletto

https://github.com/jacob-corletto/MeetingAlgorithm

```
import ast

def to_hours(time):
  hours = time // 60               #+1
  minutes = time % 60              #+1
  if hours < 10:                   #+1
    hours = '0' + str(hours)       #+1
  else:                            #+1
    hours = str(hours)             #+1
  if minutes < 10:                 #+1
    minutes = '0' + str(minutes)   #+1
  else:                            #+1
    minutes = str(minutes)         #+1
  return hours + ':' + minutes     #+1

def to_mins(time):
  """Converts a time string in the format "HH:MM" to minutes."""
  hours, minutes = time.split(":") #+1
  return int(hours) * 60 + int(minutes) #+1

def combine_daily_active(person_one, person_two, agenda):
  """Combines the daily active times of two people into a single list."""
  if (to_mins(person_one[0]) > to_mins(person_two[0])):        +1
      agenda.append(person_one[0])                             +1
  elif (to_mins(person_one[0]) < to_mins(person_two[0])):      +1
      agenda.append(person_two[0])                             +1
  elif (to_mins(person_one[0]) == to_mins(person_two[0])):     +1
      agenda.append(person_one[0])                             +1

  if (to_mins(person_one[1]) < to_mins(person_two[1])):        +1
      agenda.append(person_one[1])                             +1
  elif (to_mins(person_one[1]) > to_mins(person_two[1])):      +1
```

```
      agenda.append(person_two[1])                             +1
  elif (to_mins(person_one[1]) == to_mins(person_two[1])):     +1
      agenda.append(person_one[1])                             +1

def schedule_meeting(person_one_schedule, person_two_schedule, agenda,
                      duration, times_available):
  earliest_start = to_mins(agenda[0])  +1
  latest_end = to_mins(agenda[1])     +1

  merged_schedule = sorted(person_one_schedule + person_two_schedule,
key=lambda x: x[0]) +O(n) + O(n log n) - bc of sorted

  end = earliest_start +1
  for i in range(len(merged_schedule)): +O(n)
      start_time = to_mins(merged_schedule[i][0]) +1
      diff = start_time - end +1
      if diff >= duration: +1
          times_available.append([to_hours(end), merged_schedule[i][0]])
+1
      end = max(end, to_mins(merged_schedule[i][1]))  +1

  if latest_end - end >= duration: +1
      times_available.append([to_hours(end), agenda[1]]) +1

  return times_available          +1


var1 = [] +1
var2 = [] +1
var3 = [] +1
var4 = [] +1
var5 = 0  +1
agenda = [] +1
times_available = [] +1

# Open the file in read mode ('r')
with open("input.txt") as file:
  # Read all lines in the file one by one
  x = 1
  counter = 1
  with open("output.txt", "w") as f:
    f.write("")

  for line in file: O(n)
    if line == "\n":  +1
      x = 1            +1
      Continue         +1
    exec(f"var{x} = line")    +1

    if x % 5 == 0:     +1

      person_one_schedule = ast.literal_eval(str(var1))        +1
      person_one = ast.literal_eval(str(var2))                 +1
      person_two_schedule = ast.literal_eval(str(var3))        +1
      person_two = ast.literal_eval(str(var4))                 +1
      duration = ast.literal_eval(str(var5))                   +1
      combine_daily_active(person_one, person_two, agenda)     +1
      schedule_meeting(person_one_schedule, person_two_schedule, agenda,
duration, times_available)                                     +1

      with open("output.txt", "a") as f: +1
        f.write("Case #" + str(counter) + ": " + str(times_available) +
"\n") +1

      var1 = []               +1
      var2 = []               +1
      var3 = []               +1
      var4 = []               +1
      var5 = 0                +1
      agenda = []             +1
      times_available = []    +1
      person_one_schedule = ast.literal_eval(str(var1))  +1
      person_one = ast.literal_eval(str(var2))           +1
      person_two_schedule = ast.literal_eval(str(var3))  +1
      person_two = ast.literal_eval(str(var4))           +1
      duration = ast.literal_eval(str(var5))             +1
      counter += 1                                       +1
      x = 1                                              +1
      Continue                                           +1
    x += 1                                                      +1
```
Output:



```
Case #1: [['04:00', '6:00'], ['07:00', '8:00']]
Case #2: [['09:00', '12:00'], ['15:00', '16:00'], ['18:00', '18: 30']]
Case #3: [['09:00', '10:00'], ['12:30', '14:30'], ['15:00', '16:00']]
Case #4: []
Case #5: [['08:00', '19:00']]
Case #6: [['09:00', '10:00'], ['12:30', '14:30'], ['15:00', '16:00'], ['18:00', '19:00']]
Case #7: [['09:00', '10:00'], ['12:30', '14:30'], ['15:00', '16:00'], ['18:00', '24:00']]
Case #8: [['23:00', '24:00']]
Case #9: []
Case #10: [['00:00', '24:00']]
```

Time Complexity: O(n log n)

```
  merged_schedule = sorted(person_one_schedule + person_two_schedule,
key=lambda x: x[0]) +O(n) + O(n log n) - bc of sorted
```
Using sorted method → O(n log n)

73 total steps
O(3n)
O(n log n)

T(n) = 3n + n log n + 73
O(f(n)) = n log n

We don’t believe we can do better because O(n log n) is better than n^2 and we don’t believe this can be exponential. An increase in n would change the complexity because of an increase in schedules would make for more iteration.

$$T(n) = 3n + n\log n + 73$$

$$T(n) = 3n + n\log n + 73 \in \Theta(n\log n)$$

$$\lim_{n\to} \frac{T(n)}{F(n)} = \lim_{n\to\infty} \frac{3n + n\log n + 73}{n\log n}$$

$$\lim_{n\to\infty} \frac{3n}{n\log n} + \lim_{n\to\infty} \frac{n\log n}{n\log n} + \lim_{n\to\infty} \frac{73}{n\log n}$$

$$\downarrow \qquad\qquad \downarrow \qquad\qquad \downarrow$$

$$0 \qquad\qquad 1 \qquad\qquad 0$$

$$0 + 1 + 0 = 1$$

which is non-negative and constant with respect to $n\log n$

therefore $3n + n\log n + 73 \in \Theta(n\log n)$