Project 1 Submission

Names: Isaac Perez, Jason Lee
Emails: isaacdan@csu.fullerton.edu , jason8452@csu.fullerton.edu
Github link: Isaacdan333/CPSC335-Project-1 (github.com)

Overall the code will run in O(n) time and it will be linear time.

Algorithm Analysis:
When taking in the person's array list, consisting of their schedule including their working period it converts the time of each set from string to an int for later calculations. Then when we have to output the time we have to convert it back to string time. **These two functions, timeToMinutes() and minutesToTime() take O(1) as it only deals with if conditions. The order of growth therefore is constant time.**

In the function findAvailableTimes(), we declare a vector that takes an interval of available times in between schedules as a pair. An int variable, called lastEnd, is declared to keep track of the current time we are comparing as we pass through the person's schedule starting at the beginning of their DailyAct. A for loop is made to pass through the current person's schedule. Within the current element in the array, it grabs the start and end time of that specific period in the schedule. It compares the difference in time between the start time and lastEnd to the duration of meeting required. If start time minus lastEnd is greater/equal to the duration time then a pair between the lastEnd and start time is made and is pushed back to availableTime vector. lastEnd is then reassigned to the end time of that period within the schedule. After traversing through the list, comparing the time difference of endtime and startime to the duration time required, one last comparison is made with the end time of the dailyAct and the finishing period of the schedule to the duration time to check if there is enough time to meet up for the duration. The function returns the availableTimes vector. **The efficiency of this function is O(n) because it uses a for loop but no nested loops within it. The step count of it is 3n+6 therefore the degree is O(n). Space complexity is also O(n) since it runs in linear time.**
Mathematical analysis:
To show that 3n+6 = O(n):
Let f(n) = 3n + 6 and g(n) = n
3n + 6 <= c * n, n > n0
Let n0 = 1 and c = 8
3n + 6 <= 8 * n, n > 1
3n + 6 <= 3n + 6n, n > 1
Therefore O(n) is the result

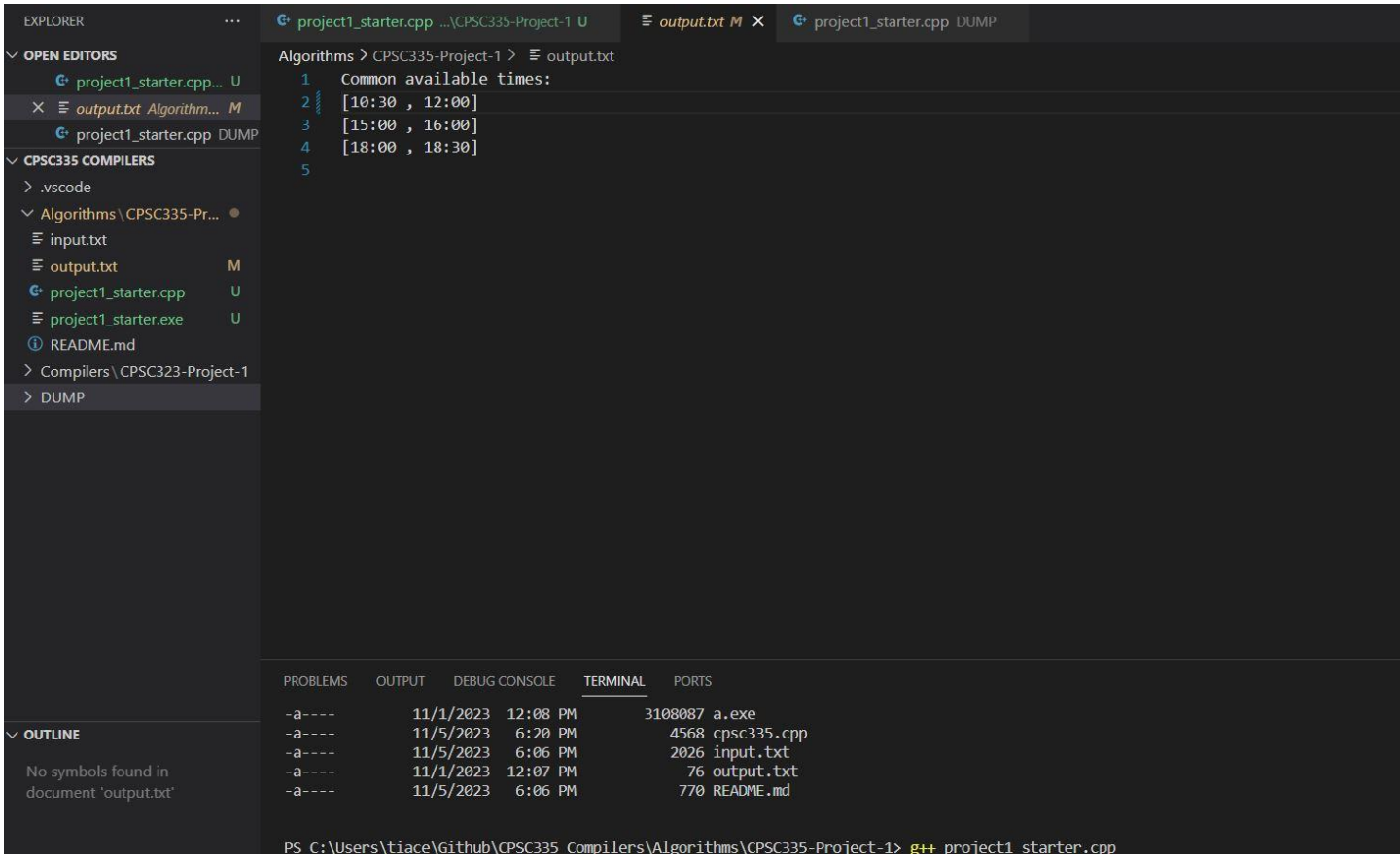
In the function findCommonTimes(), both schedules, dailyActivities, and duration time are taken as parameters. Both persons schedules are called into the findAvailableTimes function to find when they are free in between the busy schedules. After that a while loop is made to make sure we have not reached the end of both times available until we've checked for any common times that both parties are free. Starting at the beginning of both schedules we have variables for the

start and end time of their available times. If any of the schedules ending available time happens sooner than when the other schedules free time starts then we iterate through that person's available time until we find a time where their end time will happen later than the free time of the other. When we do find a time where both available times overlap, we find the start time that will happen later of the two using the max function and the end time that will happen sooner using the min function. The difference in time between the ending and starting time to the duration time required is made and if it is greater than the duration, it is pushed back to the made commonTimes vector. Once we are done traversing through available time periods, it returns the given times found where the duration of the meetings can happen.
Time complexity is also O(n) because the for loop runs separately and there are no nested loops. The step count is 18n + 21. Reducing constant time, brings it to O(n). Order of growth is linear therefore space complexity is also O(n).
Mathematical analysis:
To show that 18n+21 = O(n):
Let f(n) = 18n + 21 and g(n) = n
18n + 21 <= c * n, n > n0
Let n0 = 1 and c = 40
18n + 21 <= 40 * n, n > 1
18n + 21 <= 18n + 21n, n > 1
Therefore O(n) is the result

The main function is where we put the schedule, dailyActs and duration of meeting to find the common times for a meeting to happen. Once it does all of that, it converts the integers of common times back to a string to output it on an output.txt file using a for loop. **This will also run in O(n) time.**

I am unsure of how to make this code any better because it already runs in O(n) time. If we had an increase of n, number of persons in the group, our algorithm currently would be a lot slower because we would use a for loop to go through each person's schedules and call the availableTime and commonTime functions. Since each already runs in O(n) time and we are using a for loop to iterate through each person or time complexity would increase to O(n^2).

Screenshot:


EXPLORER
OPEN EDITORS
project1_starter.cpp... U
output.txt Algorithm... M
project1_starter.cpp DUMP
CPSC335 COMPILERS
.vscode
Algorithms \ CPSC335-Pr...
input.txt
output.txt M
project1_starter.cpp U
project1_starter.exe U
README.md
Compilers \ CPSC323-Project-1
DUMP
OUTLINE
No symbols found in document 'output.txt'
project1_starter.cpp ...\CPSC335-Project-1 U
output.txt M
project1_starter.cpp DUMP
Algorithms > CPSC335-Project-1 > output.txt
1 Common available times:
2 [10:30 , 12:00]
3 [15:00 , 16:00]
4 [18:00 , 18:30]
5
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
-a---- 11/1/2023 12:08 PM 3108087 a.exe
-a---- 11/5/2023 6:20 PM 4568 cpsc335.cpp
-a---- 11/5/2023 6:06 PM 2026 input.txt
-a---- 11/1/2023 12:07 PM 76 output.txt
-a---- 11/5/2023 6:06 PM 770 README.md
PS C:\Users\tiace\Github\CPSC335_Compilers\Algorithms\CPSC335-Project-1> g++ project1_starter.cpp