Project 1

Matching Group Schedules

CPSC 335-02: Algorithms Engineering
Professor Ornella Irene Dsouza
Fall 2023

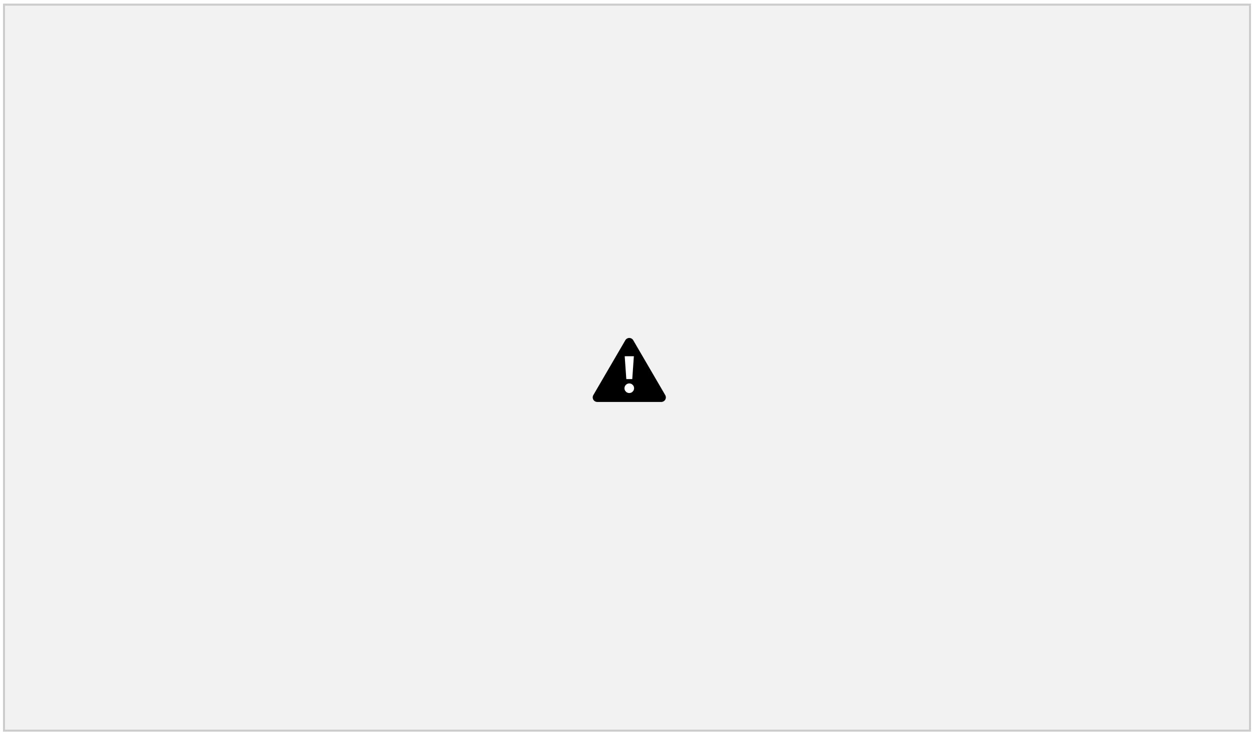
Tiffany Buu
tiffanybuu@csu.fullerton.edu
https://github.com/BuuTiffany/CSPS-335-Project-1
2nd October, 2023

Inputs

Used many “bad inputs” to test the durability of the algorithm

Group 1: Testing no times unavailable [] ['9:00','19:00'] [] ['9:00','19:30'] 30	Group 2: Testing no working hours [['12:00':'13:00']] [] [['14:00':'15:00']] [] 30
Group 3: Testing no meeting duration [['10:00':'13:00'],['14:00':'15:00']] ['9:00','19:00'] [['10:15':'11:00'],['11:30':'12:45'],['14:00':'16:00']] ['9:00','18:30']	Group 4: Testing No Potential Meeting Times [['14:00':'16:00']] ['12:00','23:59'] [['10:15':'11:00'],['11:30':'12:45']] ['0:00','12:00'] 180
Group 5: Testing only 1 group member [['00:00':'12:30'],['13:00':'18:00']] ['10:00','22:00'] 45	Group 6: Testing 3 Group Members [['10:00':'13:00']] ['9:00','19:00'] [['10:15':'11:00'],['11:30':'12:45'],['14:00':'16:00']] ['9:00','18:30'] [['10:30':'11:45'],['14:00':'16:00']] ['8:00','20:30'] 30
Group 7: Testing missing information for different group members [['00:00':'6:00'],['12:00':'13:00'],['23:00':'24:00']] [] [] ['12:00','18:30'] 30	Group 8: Testing if working hours was missing for second group member [['0:00':'13:00'],['20:00','23:59']] ['8:00':'22:00'] [['10:15':'11:00'],['11:30':'12:45'],['14:00':'16:00']] 15
Group 9: Test if extra time frame is given for working hours [['10:00':'13:00'],['14:00':'15:00']] ['9:00','19:00','21:00','22:00'] [['10:15':'11:00'],['11:30':'12:45'],['14:00':'16:00']] ['9:00','18:30'] 15	Group 10: Test to pass empty group [] [] [] []

Output



Mathematical Analysis

Note: My code included some `cout`s for debugging purposes and were not included in the mathematical analysis

After analyzing the step count, I propose that my time function would be:

$$121n^4 + 86n^3 + 172n^2 + 81n + 8$$

Meaning that my algorithm runs in the order of

$$O(n^4)$$

Given:

$$T(n) = 121n^4 + 86n^3 + 172n^2 + 81n + 8$$

$$f(n) = n^4$$

Finding the limit:

$$\lim_{n \to \infty} \frac{T(n)}{f(n)} = \lim_{n \to \infty} \frac{121n^4 + 86n^3 + 172n^2 + 81n + 8}{n^4}$$

$$= \lim_{n \to \infty} \frac{121n^4}{n^4} + \lim_{n \to \infty} \frac{86n^3}{n^4} + \lim_{n \to \infty} \frac{172n^2}{n^4} + \lim_{n \to \infty} \frac{81n}{n^4} + \lim_{n \to \infty} \frac{8}{n^4}$$

$$= 121 + 0 + 0 + 0 + 0$$

$$\lim_{n \to \infty} \frac{T(n)}{f(n)} = 121$$

The limit of $\frac{T(n)}{f(n)}$ is 121 which is non-negative and constant with respect to n

Therefore: $121n^4 + 86n^3 + 172n^2 + 81n + 8 \in O(n^4)$

Can we do better?

What changes do you think can be made to your algorithm to increase its time complexity/efficiency?

When creating my initial algorithm I had added extra “protective” measures to my algorithm to ensure a bad input wouldn’t break the system which bogged down the algorithm a bit.

Safeguards included:

- Ability to detect missing times unavailable
- Ability to detect missing working hours
    - Default working hours [00:00, 23;59]
- Ability to detect missing meeting duration
    - Default meeting duration is 1 (minute)
- Ability to detect non-valid time frames & remove
    - Non-valid: start time is after end time
- Ability to handle one or more group members
- Ability to handle “bad” inputs

While I would still add safeguards if I were to redo this project, I would opt to only have safeguards for the bad initial inputs into the algorithm. This way I can trust that because the initial input is good, later steps wouldn’t need to safeguard against bad inputs; Especially for this assignment where I had complete control over what the inputs into the project would be.

Will an increase in “n” change the complexity class?

“n” being the number of people in a group

No, while there would be an increase in time, the class complexity would stay the same. Due to the definition of Big O, the complexity class would be the upper bound of the algorithm, meaning that even as n values change, there is still some value n_0 and scale factor c such that the complexity class function is greater than the n.