Christian Lussier, Ben Watto, Mikey Spurr

CS202 Final Project Report

Every semester at Allegheny College, students go through the registration process to enter courses for the upcoming semester. Each student is placed into a letter group for their class year and these eight letter groups are then randomized. Students than register by class year and by their specified letter group within each class. However, with this randomized letter group system, students are often forced to have to register at the end of the letter group order for their class. With student anger surrounding this system, we decided to create a new letter group organizer algorithm and a program to interact with it.

We had a variety of motivations when we decided to begin work on this project. Our first motivation was based off of our groups interest in creating a new algorithm, as described in Track 2 of the project's specification sheet. Our groups main motivation was to improve upon the current letter group based registration system in place at Allegheny College. This is because this student angers us and many of our colleagues on campus. The current system used for determining letter groups is completely randomized and doesn't take anything into account when determining this order. If a student letter group resides near the end of the order, there is still a chance that the student letter group will remain in a similar position the next time registration rolls around. This could happen time and time again, meaning that some students could never register near the front of the letter group order. As a result, our group was motivated to create a letter group ordering algorithm that would make the registration more efficient and more fair for all students.

Every school has their own registration system so it is hard to say what exactly has been done to improve these systems as these details are often private information. However, there is no question that poor registration systems are a problem for students and others have looked into the impacts registrations can have on students. One study describes that assigned registration times greatly impact course availability for college students (Gurantz). With fewer course offerings in today's colleges, students forced to register later than their peers because of a set registration system are less likely to get into classes they need which can impact their graduation deadline (Gurantz). The current registration systems in today's schools may be one of the factors causing many students to remain in school for an additional year, delaying their entrance into the workforce and increasing their school debts. Additionally, Bahr, Peter Riley, et al. mentions that with increasing enrollment and less classes in colleges, schools have resorted to having a registration system that gives some students preferential course enrollment based on specified criteria. This disallows some students from being able to register for classes and dissuades others from entering college at all (Bahr, Peter Riley, et al.). There is no question that unfair and flawed registration system have negative effects on the students using them.

The algorithm takes in the list containing the student information and also the letter groups. It begins by splitting the list of 8 letter groups from the previous semester into halves. Once the letter groups are split into halves, it swaps those two halves to create the new order for the upcoming semester registration. It then splits these halves in half again, creating quarters or "sub-halves". After this, it will decide to randomly swap these sub-halves around, with there being a 50-50 chance of a swap occurring. This randomization adds some variation in the letter group orders every semester within each half, so that even when the halves are swapped, the order inside each half can still change. Afterwards, the program will go inside each of the sub-halves and sort the 2 letters groups in there by calculating the average GPA of each group by going through the list of inputted students' GPAs. With this, the letter group with the higher average GPA is moved to the first position in the sub-halve. This gives a small reward to the letter groups whose students are performing better. It could also give students an incentive to take more interest in their schoolwork because if they have a good GPA it gives their letter group a better chance of registering earlier. After this, the students are then sorted based on their class year along with the new letter group order. The working algorithm then returns the sorted list of students.

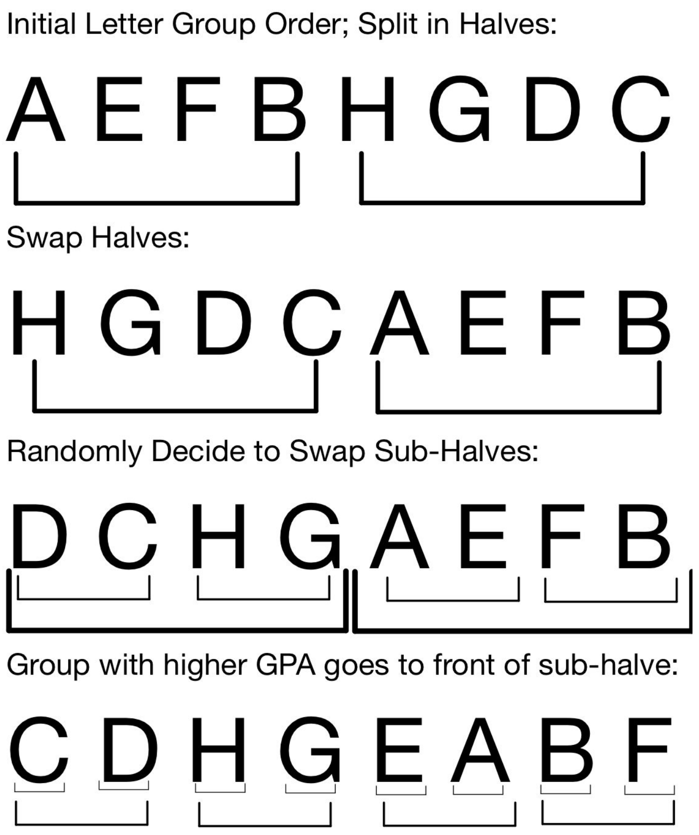
Initial Letter Group Order; Split in Halves:
A E F B H G D C
Swap Halves:
H G D C A E F B
Randomly Decide to Swap Sub-Halves:
D C H G A E F B
Group with higher GPA goes to front of sub-halve:
C D H G E A B F

Figure 1: Diagram of an example run of the algorithm.

Overall, our project's code functions as intended. We implemented a working algorithm that successfully organizes the letter groups. Additionally this algorithm also successfully sorts a list of students based on their class year and the new letter group order. We also created sample input files of students and a file reader to read these input files in. With this, we also created a working interface runs the algorithm and allows users to interact with it, giving them the ability to do things like enter the previous semester's letter group order and choose an input file of students. This information is then fed into the algorithm. This gives users more choices and customization options when running the program, thus enhancing the user experience.

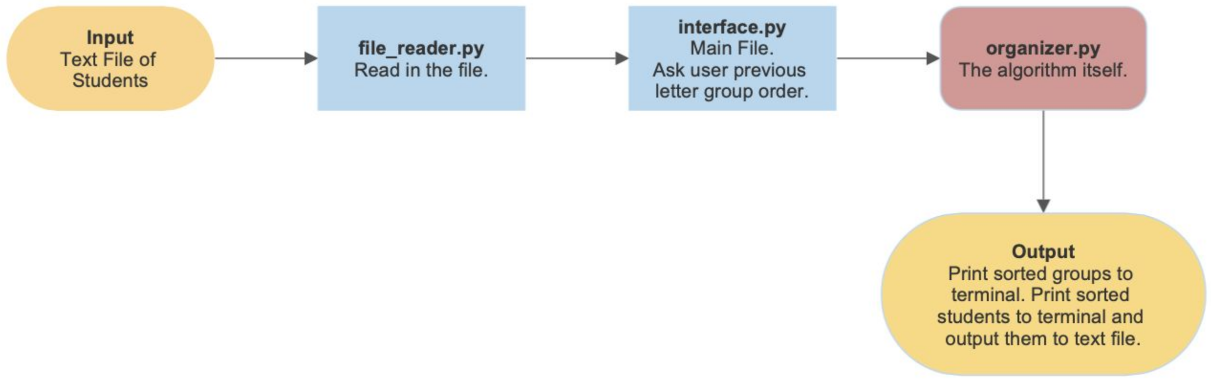
Input
Text File of Students
file_reader.py
Read in the file.
interface.py
Main File.
Ask user previous letter group order.
organizer.py
The algorithm itself.
Output
Print sorted groups to terminal. Print sorted students to terminal and output them to text file.

Figure 3: Program interaction flowchart.

In terms of our algorithm, it seems to function correctly. It correctly follows the steps that appear in the diagram above and in the algorithm description, following our original plan from when we wrote our project proposal. Our algorithm successfully sorts letter groups into halves, randomly swaps the sub-halves, and organizes the letter groups inside each sub-half by GPA. Additionally, the algorithm then takes an inputted list of students and sorts it based on the new letter group order.

Our algorithm seems to run with a worst-case time complexity of O(n^3). We calculated this by analyzing the pseudocode for our algorithm, finding that at one point in the algorithm there are three nested for loops that have a runtime of “n”. This means that our worst-case time complexity should be around O(n^3). We also took the time to try and verify the correctness of this estimated worst-case time complexity by consulting with TAs. While analyzing the algorithm, we also did a quick experimental analysis on the algorithm, running a simple doubling experiment. We found that even with an input size of 120 students, the algorithm took less than a second to run. Despite this, our future works would take steps to improve our algorithms worst-case time complexity by refactoring code to remove nested for-loops.

Input Size	Running Time	Ratio
10	0.0001628398895263672	0
20	0.0003559589385986328	2.19
40	0.0005738735198974609	1.62
80	0.0008790493011474609	1.52
160	0.0016260147094726562	1.83

Figure 4: Results of a doubling experiment on the Letter Group Organizer Algorithm.

Our Registration Letter Group Project successfully sorts the letter groups based on our specified goals. This solves the original problem presented in which students repeatedly schedule at the end of each period. Using our algorithm, every letter group can potential to register first instead of the previously random system. Incorporating this algorithm into our schools current system could be beneficial for all students. Overall, our program functions as intended with a worst case time complexity of O(n^3). This project exposed us to many learning experiences through the challenges we faced. The initial challenge we faced involved scheduling our weekly meetings. This was challenging because we are all full-time students with many other responsibilities. Another major challenge was creating our algorithm. To make progress we continually used trial and error. Our group also struggled to implement a priority feature to our algorithm. We were able to successfully implement it, but it undermined the fairness within our algorithm. The final challenge we faced involved analyzing our algorithm, which involved us examining the code. Our greatest reward for this project was creating a working algorithm to sort letter groups.

Works Cited

Bahr, Peter Riley, et al. "First in line: Student registration priority in community colleges."

Educational Policy 29.2 (2015): 342-374.

Gurantz, Oded. "Who loses out? Registration order, course Availability, and Student Behaviors in Community College." *The Journal of Higher Education* 86.4 (2015): 524-563.