Christian Lussier, Ben Watto, Mikey Spurr
Spring 2019

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. The randomization often does not work out favorably for some students, creating anger surround this process. With this in mind, we set out to create a new letter group organizing algorithm. The algorithm keeps some randomization elements in play, while making the system more fair. With this, we also took steps to create sample input files of students and a program to interact with the algorithm which creates a interface which gives users more customization options. We completed a variety of tasks while completing this project, which included successfully implementing a working algorithm and 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 group's 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.

Our group found it hard to find what has been done to improve registration systems, not just at Allegheny College, but at schools across the country. 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.

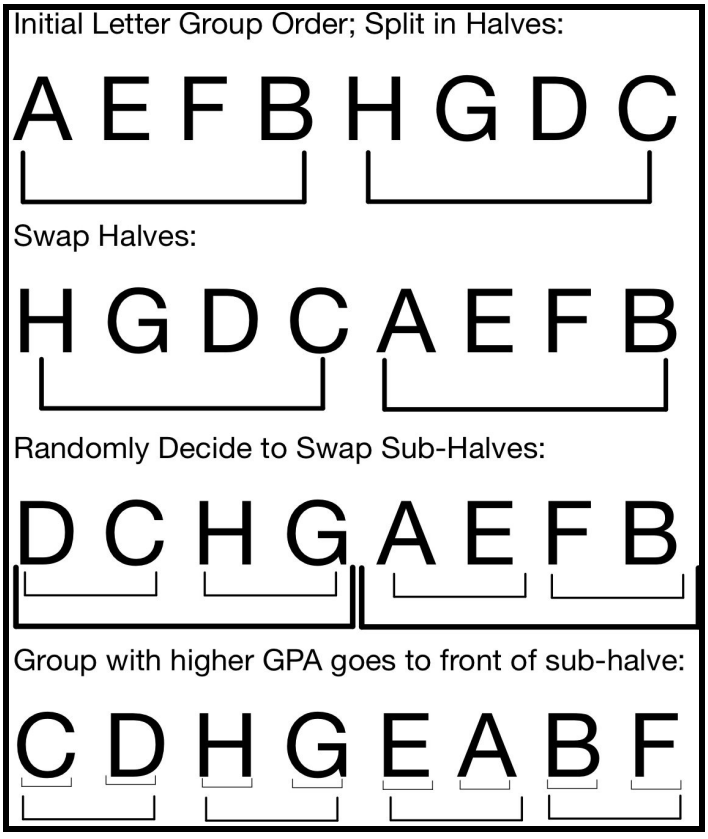

Figure 1: Diagram of an example run of the algorithm.

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. The file reader functions as intended and when called by the interface, it inputs a list of students flawlessly. 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, which is then parsed and read in by the file reader. This information is then fed into the algorithm. Additionally, once the algorithm is done running, the new letter group order and list of sorted students are outputted to the terminal, while the interface asks users if they would like to output the sorted list of students to a text file of their choosing. This gives users more choices and customization options when running the program, thus resulting in a simplistic yet functional user experience.

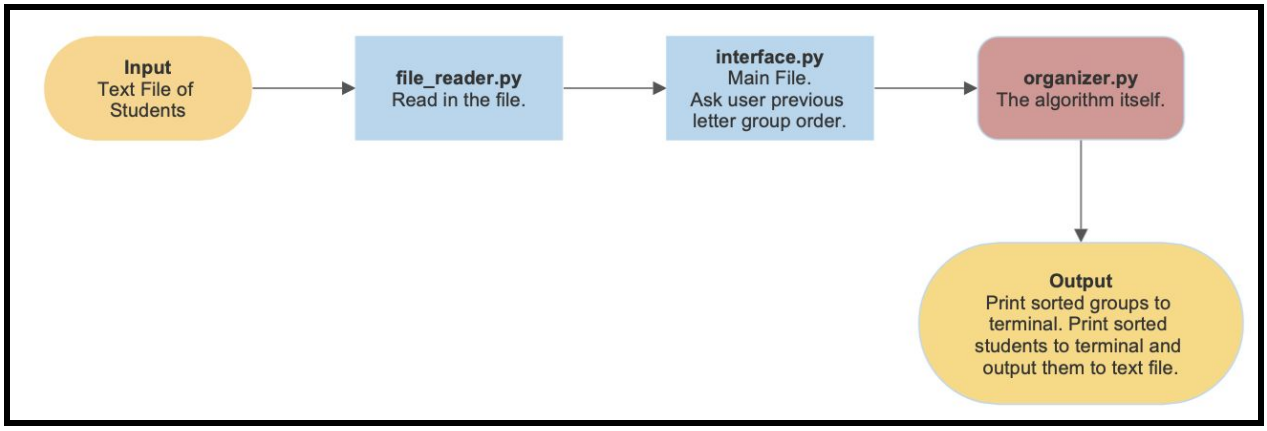

Figure 2: Program interaction flowchart.

In terms of the algorithm itself, we have found that it functions correctly. It correctly follows the steps that appear in Figure 1 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. In order to help verify the correctness of the algorithm, the team implemented code within the algorithm to show how the list of letter groups was changing. It does this by printing out what is occurring to the terminal and the result of what occurred. This allows the user of the algorithm to see what is occurring and for us, as the developers of the algorithm, to verify that it is working as intended.

```
def letter_group_organizer(letter_groups, students):
        Split letter group list in half saving each half in an object
        Swap halfs inside of list
        Split halves into sub-halves; each sub-half remains in half
        for half in all_groups:
                Randomly swap the sub-halves in each half
                for sub-halve in half:
                        for i in range(len(students)):
                                Calculate average GPA for letter groups
                                Move higher average GPA to front of each sub-halve
        Add subhalves to halves and halves to all_groups list
        Sort students based on class year and letter group
```

Figure 3: Pseudocode for where the algorithm performs letter group organization.

Our algorithm, which functions in an iterative fashion, seems to run with a worst-case time complexity of O(n^3). We calculated this by analyzing the implemented code and 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. With this, the calculated ratio indicates that our algorithm is running faster than our estimated worst-case time complexity, indicating that we are not providing a large enough or diverse enough dataset to reach the worst-case. Despite this, if we had more time before the project deadline or if we worked on this algorithm more in the future, we would take steps to improve our algorithms worst-case time complexity by refactoring code to remove the three 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.

We created an algorithm that would help prevent student anger surrounding the current letter group based registration system in place at Allegheny college. By separating the letter group order into halves and keeping some sense of randomization, our letter group organizer algorithm makes the process of registering more fair, giving groups a chance to register first every other semester. We learned a variety of things while completing this project, such as learning more about analyzing an algorithm to find the big-O time complexity, learning how to do research and build an algorithm from the ground-up, and learning more about Python. Our

challenges also enhanced our learning experiences as creating our algorithm required a lot of time, effort, and critical thinking. When creating this algorithm we also had to figure out problems through trial and error to find bugs and improve our algorithm. Another challenge came when we tried to implement a proposed priority index feature into the algorithm. This was challenging because it took a while to implement but it seems to undermine the fairness our original algorithm strived for. Despite these challenges, the rewards were worth it as we created a working algorithm and program, found that our student peers liked the idea of our letter group organizer and the fairness it would bring, and we provided a solution to a real-life problem. Overall, this project presented a challenging problem in building a working algorithm from scratch while providing a variety of learning experiences and rewarding moments.

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.