UNIVERSITY OF CENTRAL FLORIDA
DEPARTMENT OF COMPUTER SCIENCE

Computer Science 1

Group Meeting Report

Group Members:
Derek Lopes Oliveira
Kiara Yost
Dhyan Suresh
Gregory Aguila

December 4, 2025

1 MEETING DATES AND TIMES

1.1 In Recitation

- 11/06/2025 @ 8:30:00

1.2 Outside of Recitation

- 11/08/2025 @ 12:00:00 - 13:00:00
- 12/01/2025 @ 12:00:00 - 13:00:00
- 12/03/2025 @ 10:00:00 - 11:00:00

2 STRATEGIES TO PREPARE FOR FINAL EXAM

THIS DOCUMENT WILL OUTLINE THE STRATEGIES OUR GROUP USED TO PREPARE FOR THE FINAL EXAM IN CS1. WE WILL DISCUSS HOW WE ORGANIZED OUR MEETINGS, WHAT TOOLS WE USED TO COLLABORATE, AND HOW WE DIVIDED THE WORK AMONG GROUP MEMBERS. ADDITIONALLY, WE WILL REFLECT ON OUR EXPERIENCES AND THE EFFECTIVENESS OF OUR STUDY METHODS AND HIGHLIGHT KEY DISCOVERIES MADE DURING OUR GROUP STUDY SESSIONS.

3 Understanding our Group Activity

3.1 In Recitation

On 11/06/2025 our group was restructured to combine several people from other groups that had more than 80% of their members drop. This lead to a new group dynamic and we had to quickly adapt to working with new members. We discussed our strengths and weaknesses in the subject matter and decided to focus on areas where we felt less confident. We also established a schedule for our meetings and set goals and explectaions for each session to ensure we stayed on track. We agreed that the best way to meet up was via Discord. This woudl allow us to be flexible with our meeting times and accomodate everyone's schedules. Discord also allowred us to shared our screens and work through problems together in real-time.

3.2 Outside of Recitation

- Meeting 1: 11/08/2025:
  During our first meeting as a newly formed group, we focused on reviewing the midterm to identify where each member's weaknesses lay. From that discussion, we reached the consensus that concentrating our efforts on algorithm analysis would benefit the group the most.
  We began with a problem we considered relatively simple. We selected the first question from Section C of the Spring 2023 FE, which presented a segment of code and asked us to determine its runtime. We solved this fairly quickly, as the solution primarily involved examining the nested loops and identifying the resulting time complexity.
  During the second part of our session, we shifted to the Spring 2016 FE Part B time-complexity question. The problem stated that a function with time complexity $O(n^2)$ takes 338 ms for an input of size 13,000, and asked how long it would take for an input of size 8,000. We initially assumed the change would scale proportionally, but quickly remembered that quadratic time does not behave linearly. To correctly approach the problem, we used the standard form:

  $$T(n) = cn^2$$

  Given $T(13000) = 338$ ms, we solved for the constant:

  $$c = \frac{338}{13000^2}$$

  After calculating c, we substituted $n = 8000$:

  $$T(8000) = c(8000^2)$$

  This resulted in a runtime of approximately 128 ms. Working through the full expression clarified why a simple proportional scale-down would have been incorrect. From this problem we are understood the best method to approch our meetings. We saw the most brnfit when one of us shared our screen and walked through each step so the group could verify the calculations together. Whenever arithmetic uncertainty came up, we checked our work using our notes and online tools. This collaborative process helped solidify our understanding of hard topics and problems.
- Meeting 3: 12/04/2025:
  During our last meeting, we reviewed material taught after the midterm. After some discussion on what we were most interested in reviewing, we agreed to cover sorting algorithms, as they cover a relatively large percentage of the final exam.

We began with a medium difficulty problem on FEPrep.net, "Identify Sorting Algorithm from First Pass".The problem was structured into three parts, with parts A and B featuring diagrams that showcased an initial array followed by the transformed array after one pass of an unspecified sorting algorithm. The problem asked us to indicate which sorting algorithm is being applied, and give that algorithm's worst-case runtime using big-oh notation.
After refreshing ourselves on all the major sorting algorithms, we easily identified the algorithm in part A as selection sort, a method that finds the smallest element and swaps it with the first element. In part B, we recognized bubble sort, which operates by repeatedly pushing the largest element to the end of the array with each pass. However, many of us couldn't immediately recall the worst-case runtime for these algorithms. After some discussion, we collectively remembered that both selection sort and bubble sort exhibit a worst-case runtime of $O(n^2)$. Moreover, we took the opportunity to compare the worst-case runtimes of all the major sorting algorithms. We noted that while bubble sort, selection sort, and insertion sort each display $O(n^2)$ complexity, more efficient algorithms such as merge sort and quick sort have a better performance with $O(n \log n)$ runtimes.
Part C of the problem asked to give a recurrence relation that represents the runtime for a Merge Sort of n items. This stumped us, as none of us could readily recall the appropriate relation. As a result, we reviewed previous class notes to determine how to find the proper recurrence relation for merge sort. Eventually, we discovered the proper relation to be $T(n) = 2T(n/2) + O(n)$.
Because of this problem, we were able to review the behavior and worst-case runtimes for all the major sorting algorithms. Additionally, we deepened our understanding of merge sort's recurrence relation.
Next, we completed another medium difficulty problem on FEPrep.net, "Median of Three for Quick Sort Partitioning". In this problem, we were tasked to write a function that takes in an array, a low index to the array, and a high index to the array, designating a subarray, generates three random indexes in between low and high inclusive (indexA, indexB and indexC), and returns the corresponding index (either indexA, indexB, or indexC) to the middle value of the three designated values array[indexA], array[indexB], or array[indexC]. While we initially struggled to devise a suitable approach, we collaboratively brainstormed potential solutions. We decided to leverage the provided randInt function (which returns a random integer in between a and b, inclusive) to create the random indexA, indexB, and indexC within the subarray. Independently, we wrote our own solutions for determining the middle value of these three indices. Many of us struggled with how to approach this problem, but after sharing our code snippets and discussing our approaches, we arrived at a functional solution involving a series of if-else comparisons to pinpoint the median index.
Because of this problem, we better understood the purpose of using a median of three for quick sort partitioning and how to determine this median.

4 Tools Used

The following are tools we used to solve some problems and brainstorm ideas.

- Midterm Exam to review and identify weak points.
- Exams from previous semesters from Professor Guha's website.
- Class notes from lectures.
- "FEPrep.net" - Website created by fellow UCF student: Dmytro Chygarov. The site contains practice problems and solutions for the questions that appreaed on the foundation

exam from previous years.

5 Summary

- Meeting 1: 11/08/25: Overall, this meeting established our foundation and expectations for how we will operate as a group moving forward. By beginning with a quick review problem and then working collaboratively through the more challenging questions, we set a clear norm for future meetings: start with brief confidence-building review, then transition into deeper analysis with shared screens, open discussion, and collective verification of each step. This session helped us better understand algorithm analysis and demonstrated the value of walking through solutions together rather than working in isolation. The team helped one another through arithmetic, referencing notes, and checking calculations—gave us a structure that will guide our next sessions and ensured that each member feels supported.

  Meeting 2: 12/01/25: In this meeting we talked about summations and we worked through a couple of different problemes regarding summations. We first began by going through some question through the website FEPrep.net while checking class notes and the professor's notes on the COP website. Through the website the team collaborated to work through several summation problems, focusing on recognizing patterns, applying summation rules, and simplifying expressions. For example, we reviewed basic formulas such as

  $$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \quad \text{and} \quad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}.$$

  We also practiced breaking down more complex expressions. For instance, in a problem like

  $$\sum_{k=1}^{n} (3k+2),$$

  we rewrote it using linearity of summation:

  $$\sum_{k=1}^{n} 3k + \sum_{k=1}^{n} 2 = 3\left(\frac{n(n+1)}{2}\right) + 2n.$$

  This approach helped us understand how to breakdown these problems and helped us better understand the steps needed to get to the final result.
- Meeting 3: 12/04/2025: During our last meeting, we reviewed material taught after the midterm. After some discussion on what we were most interested in reviewing, we agreed to cover sorting algorithms, as they cover a relatively large percentage of the final exam.
  We began with a medium difficulty problem on FEPrep.net, "Identify Sorting Algorithm from First Pass".The problem was structured into three parts, with parts A and B featuring diagrams that showcased an initial array followed by the transformed array after one pass of an unspecified sorting algorithm. The problem asked us to indicate which sorting algorithm is being applied, and give that algorithm's worst-case runtime using big-oh notation.
  After refreshing ourselves on all the major sorting algorithms, we easily identified the algorithm in part A as selection sort, a method that finds the smallest element and swaps it with the first element. In part B, we recognized bubble sort, which operates by repeatedly pushing the largest element to the end of the array with each pass. However, many of us couldn't immediately recall the worst-case runtime for these algorithms. After some

discussion, we collectively remembered that both selection sort and bubble sort exhibit a worst-case runtime of $O(n^2)$. Moreover, we took the opportunity to compare the worst-case runtimes of all the major sorting algorithms. We noted that while bubble sort, selection sort, and insertion sort each display $O(n^2)$ complexity, more efficient algorithms such as merge sort and quick sort have a better performance with $O(n \log n)$ runtimes.
Part C of the problem asked to give a recurrence relation that represents the runtime for a Merge Sort of n items. This stumped us, as none of us could readily recall the appropriate relation. As a result, we reviewed previous class notes to determine how to find the proper recurrence relation for merge sort. Eventually, we discovered the proper relation to be $T(n) = 2T(n/2) + O(n)$.
Because of this problem, we were able to review the behavior and worst-case runtimes for all the major sorting algorithms. Additionally, we deepened our understanding of merge sort's recurrence relation.
Next, we completed another medium difficulty problem on FEPrep.net, "Median of Three for Quick Sort Partitioning". In this problem, we were tasked to write a function that takes in an array, a low index to the array, and a high index to the array, designating a subarray, generates three random indexes in between low and high inclusive (indexA, indexB and indexC), and returns the corresponding index (either indexA, indexB, or indexC) to the middle value of the three designated values array[indexA], array[indexB], or array[indexC]. While we initially struggled to devise a suitable approach, we collaboratively brainstormed potential solutions. We decided to leverage the provided randInt function (which returns a random integer in between a and b, inclusive) to create the random indexA, indexB, and indexC within the subarray. Independently, we wrote our own solutions for determining the middle value of these three indices. Many of us struggled with how to approach this problem, but after sharing our code snippets and discussing our approaches, we arrived at a functional solution involving a series of if-else comparisons to pinpoint the median index.
Because of this problem, we better understood the purpose of using a median of three for quick sort partitioning and how to determine this median.

6 Key Discoveries

- Resources and Tools: We discovered the value of various tools and resources, such as FEPrep.net, previous exams, and class notes, which provided valuable practice and insights, helping us reinforce learned concepts and identify weak areas. FePrep.net is the most notable resource, as it provided us with a plethora of example problems to exercise our comprehension of the material.
- :

7 Member's Reflection

7.1 Derek Lopes Oliveira

AT FIRST I WAS SENT TO A GROUP WHO DIDN'T REALLY TALK APART FROM THREE DIFFERENT OCASSIONS. I WAS THINKING THAT THIS WAS A FAILED EXPERIMENT AND ALL MY HOPE WAS LOST ON

THIS PROJECT. LUCKILY, JUST BEFORE DECEMEBER STARTED, I WAS TRANSFERED TO THE HARD-WORKING GROUP THAT I AM NOW AND I TRULLY UNDERSTAND THE PURPOSE OF THIS EXCERCISE. TOGETHER WE WORKED THROUGH PROBLEMAS AND OVERCAME CHALLENGES THAT WE WOULD OTHERWISE NOT HAVE SOLVED IT ON OUR OWN. THE COLLABORATION WITH THIS TEAM HAS SHOWED ME THE POSSIBILITIES OF TEAM WORK GOING INTO MY FIELD OF COMPUTER SCIENCE AND HOPE TO ONE DAY HAVE A GREAT TEAM LIKE THIS ONE AGAIN.

7.2 Kiara Yost

I FOUND COLLABORATING WITH MY FELLOW GROUP MEMBERS TO BE ENRICHING FOR MY LEARNING EXPERIENCE. FROM THE BEGINNING, I TOOK THE INITIATIVE TO HELP MY PEERS GRASP CONCEPTS IN WHICH I FELT CONFIDENT. IN RETURN, MY CLASSMATES SHARED THEIR OWN AREAS OF EXPERTISE, WHICH HELPED TO STRENGTHEN MY UNDERSTANDING OF TOPICS I FOUND MORE CHALLENGING. THROUGH OUR DISCUSSIONS, WE NOT ONLY FILLED IN EACH OTHER'S KNOWLEDGE GAPS BUT ALSO CULTIVATED A SENSE OF CAMARADERIE THAT MADE THE LEARNING PROCESS ENJOYABLE. COLLECTIVELY, WE BUILT A SOLID FOUNDATION BY LEVERAGING OUR DIVERSE STRENGTHS, ENSURING THAT WE FELT WELL-PREPARED FOR OUR FINAL EXAM.

7.3 Dhyan Suresh

THE GROUP STUDY SESSIONS WERE INCREDIBLY BENEFICIAL FOR MY UNDERSTANDING OF THE COURSE MATERIAL. IT WAS VERY RELEVANT EARLY ON THAT WE IDENTIFIED OUR WEAK POINTS AND FOCUSED OUR EFFORTS THERE. THE COLLABORATIVE ENVIRONMENT ALLOWED US TO HELP ONE ANOTHER BY ALLOWING EACH MEMBER TO TAKE THE LEAD WHEN TOPICS THEY WERE STRONG IN CAME UP. HAVING MORE TIME WITH TOPICS BEYOND THE CLASSROOM PROVIDED A DEEPER UNDERSTANDING AND HELPED SOLIDIFY CONCEPTS THAT WERE PREVIOUSLY UNCLEAR. OVERALL, THE GROUP STUDY SESSIONS WERE A VALUABLE EXPERIENCE THAT ENHANCED MY LEARNING AND PREPARED ME WELL FOR THE FINAL EXAM.

7.4 Gregory Aguila

THIS GROUP EXPERIMENT HAS HELPED ME UNDERSTAND MY WEAKNESSES WITH TOPICS SUCH AS AVL TREES AND SUMMATIONS. WORKING IN THE GROUP ALSO HELPED ME UNDERSTAND HOW TO TACKLE SOME OF THE WORD PROBLEMS THAT MAY BE ON THE FINAL EXAM. WE MOSTLY UTILIZED FOUNDATION EXAM QUESTIONS SO WE ARE BETTER PREPARED WITH SOME OF THE HARDER CONCEPTS.