Integrating Generative AI with Data Structures and Algorithm Analysis Course Homework

Cong Pu
Department of Computer Science, Oklahoma State University, United States. Email: congpu@acm.org

***Abstract*—This innovative practice full paper describes how to integrate generative Artificial Intelligence (AI) with *Data Structures and Algorithm Analysis* (CS2) homework at Oklahoma State University. *Data Structures and Algorithm Analysis* (CS2) course covers extremely important knowledge and skills of becoming a computer scientist. However, students might fail to meet the learning outcomes of CS2 course, somewhat due to the abstract nature of concepts but also because of a misunderstanding of concepts, the selection of inappropriate data structure and algorithm, a lack of effective debugging skills, and writing inefficient code. Currently we are in an Artificial Intelligence (AI) revolution, and generative AI (also widely known as AI chatbots) are already popular across college and university campuses. Generative AI that are designed to learn and mimic human conversation is capable of generating, translating, or paraphrasing text and answering questions in a way that is often indistinguishable from human-generated content. We investigate the above-mentioned potential challenges faced by students while learning CS2 course at Oklahoma State University (OSU) and redesign the course homework in Fall 2023 semester. The objectives of the redesigned course homework are to provide students with opportunities to use generative AI to support their learning in the CS2 course as well as measure the effectiveness of utilizing generative AI to improve student learning outcomes in the CS2 course. At the end of Fall 2023 semester, we conducted a student perception survey in the CS2 course and collected valuable feedback from 47 out of 61 students (77% response rate). In summary, 85.1%, 76.6%, 74.5%, 63.8%, and 70.2% respondents indicated that generative AI help to understand testing and debugging better, improve coding skills and code quality, design and implement efficient data structures and algorithms, select appropriate algorithms and data structures with the assistance of generative AI, and understand the importance of designing and implementing efficient data structures and algorithms, respectively. In this paper, we summarize the experience of redesigning CS2 course homework at OSU, share lessons learned, and provide candid suggestions for utilizing course homework in CS2 courses at other institutions.**

***Index Terms*—Computer Science, Data Structures, Algorithm Analysis, CS2, Artificial intelligence (AI), Generative AI, AI Chatbots**

I. Introduction

Data Structures and Algorithm Analysis (CS2) are widely regarded as fundamental and important knowledge and skills to learn for students who major in computing-related degrees (e.g., computer science, computing engineering, electrical engineering, management information systems, etc.) [1]. In general, students enroll in the CS2 course during the period of their degrees because the CS2 concepts will become the building block for improving problem solving skills, realizing the importance of resource and performance optimization, elevating code quality and efficiency, facilitating program scalability and flexibility, as well as developing robust and reliable software. According to the ‘‘Computer Science Curricula 2023’’ [2] prepared by the joint task force of IEEE Computer Society and ACM, the CS2 concepts are essential to computer scientists and software developers because every theoretical and practical computational program consists of algorithm(s) that operate on data items possessing an underlying structure. Typically, the CS2 courses cover static and dynamic data structures as well as various algorithms such as complexity analysis of algorithms, linked list, recursion, stacks, queues, trees, priority queues, heaps, hash tables, graphs, searching, and sorting [3]. In general, the CS2 courses place the stress on learning-by-doing for developing students' algorithmic reasoning and programming skills. However, the abstract nature of CS2 concepts and other factors such as conceptual misunderstanding, the selection of inappropriate data structures and algorithms, the lack of effective debugging skills, and writing inefficient code post great challenges to novice computer programmers [4]. As a result, there is general agreement that learning CS2 concepts is particularly challenging, and there is little understanding of how novice computer programmers develop proficiencies in DS2 [5].

In the third decade of the 21st century, generative Artificial Intelligence (AI) and derivatives are no longer a fad/hype, and have expeditiously become a subject of concern and interest for different sectors of society [6]. Industry and government are enthusiastically embracing the AI revolution, taking advantage of AI to help improve the lives of citizens and customers they serve. For example, the municipal government of Yokosuka (a city in central Japan) announced on April 20, 2023 that ChatGPT will be adopted to assist with certain administrative tasks (e.g., summarizing sentences, generating ideas, etc.) [7]. Higher education institutions are also faced with the ‘‘disruption’’ of generative AI and have responded quickly. For example, many universities (including Oklahoma State University) have provided sample statements regarding the use of generative AI (e.g., do not allow; allow in some circumstances or with explicit permission; and allow full use in the learning environment) in the classroom [8]. Unlike electronic devices or Wikipedia that can be easily banned or

avoided in the classroom, generative AI are totally different; they are either not the devices that can be prohibited, not public resources that students can be asked not to access, or not tools that their outputs can be easily detected by plagiarism detection software. On the contrary, college students and faculty have already been using generative AI which are embedded in other application programs (e.g., Grammarly, Google Docs, and Microsoft Word) without noticing [9]. In addition, ACM permits authors to use generative AI to produce manuscript content such as text, images, tables, code, etc. as long as they acknowledge its usage in the manuscript prominently [10]. Everything has positive and negative side. If there is no proper way to ban generative AI in the classroom, we might need to think about how to use them in the careers students are preparing to enter.

To help address students' challenges (e.g., conceptual misunderstanding, the selection of inappropriate data structures and algorithms, the lack of effective debugging skills, and writing inefficient code) in learning CS2 concepts, we redesigned the CS2 course homework at Oklahoma State University (OSU) in Fall 2023 semester. The objectives of the redesigned course homework are to provide students with opportunities to use generative AI to support their learning in the CS2 course as well as measure the effectiveness of utilizing generative AI to improve student learning outcomes in the CS2 course. In order to evaluate the effectiveness of redesigned course homework in addressing students' learning challenges and improving student learning outcomes in the CS2 course, we conducted an end-of-course perception survey and collected valuable feedback from 47 out of 61 students (77% response rate). In summary, our contribution can be summarized in the following:

- We provide a detailed description of CS2 course homework before and after redesign, along with sample course homework and grading policy.
- We demonstrate a summary of student responses to the end-of-course perception survey about the use of generative AI in the redesigned course homework.
- We share lessons learned about the use of generative AI in the CS2 course and provide candid suggestions for adopting sample course homework at other institutions.

The rest of the paper is organized as follows. The existent literature are reviewed in Section II. We present an overview of the CS2 course at OSU in Section III. In Section IV and V, we describe CS2 course homework before and after redesign, respectively. We present the results of perception survey in Section VI. Lastly, we conclude the paper in Section VIII.

II. Related Work

The authors in [11] focus on the problem of student-teacher ratio, and argue that it is challenging for students to obtain prompt assistance from teachers if the ratio is high. After conducting an interview with over 200 undergraduate students, the authors conclude that the AI chatbots can be adopted as pedagogical tools to provide student assistance through explaining basic business concepts and sharing relevant educational resources. Nevertheless, their conclusion is drawn from subjective interview and the participants might not have a similar level of experience about the usage of generative AI. Before we conduct a perception survey at the end of CS2 course, each student has engaged with generative AI extensively through various tasks and obtained roughly the same experience through completing a sequence of course homework. In [12], the authors from University of Central Florida use AI to generate a majority of course materials. The generated course materials rely on the content of textbooks and contain a variety of practice questions, which provide students with opportunities for learning-by-doing. However, AI cannot automatically align course materials with teaching practices, thus, additional time and effort are still needed from the instructor.

Hall et al. investigated the impact of open access resources for learning CS2 course [13] at Virginia Tech. Since the CS2 course concentrates on the dynamic of structures and algorithms, visualizations and interactive assessment could enhance student learning. After analyzing student survey results, Hall et al. found that students hold an optimistic attitude towards open access resources. In order to help students understand the importance of data structure selection and algorithm analysis, King integrated the CS2 concepts into the software development life cycle at North Carolina State University [14]. King redesigned the CS2 course project so that students were offered with opportunities to link theory to applications. Su et al. took advantage of game-based instructional strategy along with various visualization techniques to teach the CS2 course, where the data structures and algorithms are demonstrated as in-game objects and animations within a 2.5 dimension game world [1]. Students' feedback has shown that the visualizations of data structures and algorithms assist them with understanding the CS2 concepts. In [15], the author quantified student learning outcomes of CS2 course at University of North Carolina at Charlotte. The authors discovered that several factors such as course modality, pedagogical approach, and student engagement would determine student learning outcomes. In addition, better student learning outcomes can be achieved when students are encouraged to ask questions and then receive prompt feedback, as well as engage in the classroom discussions and group activities.

In [16], the authors adopted a new teaching methodology to address the issues of high dropout rates in computer science degrees. The new teaching methodology is realized based on two key ideas: (i) consider using a continuous evaluation approach with a variety of activities for summative evaluation, instead of the conventional final exam; and (ii) making those activities more appealing to the students. The majority of the activities involve online programming competitions, which

are conducted using a web-based automatic evaluation system known as the online judge. According to the evaluation results, the dropout rate decreased to 45% while the pass rate doubled on average. Nathasya et al integrate the algorithm visualization tool with program visualization tool to help students understand the implementation of data structures [17]. First, the integrated tool benefits both moderate-paced and slow-paced students by improving their assessment scores. Second, for moderate-paced students, it allows faster completion of assessments, while accommodating additional time for slow-paced students. Third, students perceive that the integrated tool enhances their understanding of data structure materials. The work [18] provides a concise overview of how the authors seamlessly integrated algorithm visualizations into a CS2 course focused on algorithms and data structures. To integrate algorithm visualizations into the CS2 course on algorithms and data structures, the authors followed the engagement taxonomy proposed by the working group on Improving the Educational Impact of Algorithm Visualization. Specifically, they incorporated five forms of engagement: no-viewing, viewing, changing, constructing, and presenting.

In [19], the authors provide an overview of recent research studies presented at the Technical Symposium on Computer Science Education (SIGCSE) related to teaching programming and data structures to university students in computer science courses. The authors' significant contribution lies in categorizing and subcategorizing three aspects related to teaching programming: (i) content characterization, (ii) pedagogical strategy identification, and (iii) support tool grouping. In [20], the authors outlined the 100 problems curriculum, provided specifics about its format, and shared their firsthand experiences implementing it for CS2 and data structures courses. Based on the study, the authors discovered that the 100 problems can lead students to explore the essential knowledge and skills expected of a degree program graduate. In addition, the curriculum encourages students to personalize their learning experience, fostering deep understanding and stimulating intellectual growth. The author in [21] creates an adaptive learning module designed to enhance data structures and algorithms courses (CS2/DS) by incorporating introductory parallel computing. Specifically, the author's focus lies on harnessing shared-memory parallelism, aiming to teach students how to break down problems or their underlying large data structures into components that can efficiently execute across many- or multi-core processes. The [22] discussed ways to weave responsible-computing content into data structures and introductory algorithms courses, inspiring computer science faculty to work similar content into their corresponding courses.

III. DS2 Course at OSU

In Fall 2023 semester, the Department of Computer Science at OSU offered three sections of CS2 course: one traditional face-to-face section and two online sections. The actual enrollment of traditional face-to-face section was 62 (maximum 65), while the number of students enrolled in online sections were 42 and 7, respectively. The first author was the instructor of face-to-face section. The class length of face-to-face section was 75 minutes and the class meeting was held on Monday and Wednesday for 17 weeks. In addition, academic support was available to students in the form of teaching assistants (TAs) in the class, where the TA-to-student ratio was 1:20.

Students who take the CS2 course at OSU are either sophomore/junior computer science students, or students who are majoring in other computing-related degrees such as computing engineering, electrical engineering, and management information systems. The story behind this is that those non-computer science majors or concentrations requires their students to take the CS2 course, however, their departments do not offer the course. Before enrolling in the CS2 course, students need to complete two prerequisite courses which are Computer Science II and Discrete Mathematics for Computer Science. The Computer Science II introduces Java programming language to students and covers the use of object-oriented programming to design and implement software solutions. The Discrete Mathematics for Computer Science covers elementary discrete mathematics for computer science and engineering disciplines. It emphasizes mathematical definitions and proofs as well as applicable methods.

The CS2 course covers complexity analysis of algorithms, linked list, recursion, stacks, queues, trees, priority queues, heaps, hash tables, graphs, as well as searching and sorting algorithms. The course description from University Catalog is provided below:

> Storage, structures, data and information structures, list processing, trees and tree processing, graphs and graph processing, searching, and sorting.

The following two textbooks were used to prepare course materials: Data Structures and Algorithms in Java, M. T. Goodrich, R. Tamassia and M. H. Goldwasser, 6th Edition, Wiley [23] and Data Structures and Algorithms in C++, A. Drozdek, 4th Edition, Cengage Learning [24]. The textbooks were recommended but not required. Students who have completed this course should have:

1) Understand basic data structures and abstract data types including stacks, queues, lists, sets, maps and graphs.
2) Use recursion as a powerful problem solving technique in design and development of data structures and understand when it is not appropriate to use.
3) Gain an appreciation of the variety, theoretical nature, and practical uses of data structures.
4) Analyze the efficiency of data structures and select the most appropriate data structure for applications.
5) Build data structures and use them as building blocks to form more complex and advanced data structures in a hierarchical manner.

Students practiced each learning outcome through lecture examples and in-class exercise/discussion. The instructor used

review quiz, homework, and exam to assess student achievement of learning outcome.

IV. Course Homework Prior Redesign

Before Fall 2023 semester, there were five course homework throughout the entire semester. Homework #1 focused on singly and doubly linked lists and their operations such as insertion, deletion, search, and print. Homework #2 let students get familiar with stacks and queues through implementing number conversion systems. Homework #3 requested students to use recursion to find all routes from a source to a destination in the maze. Homework #4 involved binary tree and various tree traversal algorithms. Homework #5 was devoted to graphs and a number of searching and sorting algorithms. Students were required to work on each course homework individually with the provided pseudo-code, however, they were encouraged to seek assistance from TAs if necessary.

The instructions and requirements of each course homework were straightforward and prescriptive to students; they knew explicitly what data structures and algorithms were required to complete the course homework. Students' course homework were graded exclusively based on the correctness of programs. Since the course homework did not include any other coding-related activities, students only got implementation practice on the pre-determined data structures and algorithms, and other essential programming skills could not be improved.

V. Redesigned Course Homework

The redesigned course homework was composed of three components:

- 80% Data structures and algorithms implementation
- 10% Program testing
- 10% Exploiting generative AI

For each course homework, students were given appropriate two weeks to complete. The topics that were covered by the course homework were algorithm analysis, linked lists, stacks and queues, recursion, binary trees, graph, and sorting and searching. Before releasing each course homework, students had been introduced to the required knowledge, skills, and abilities. The simplified course schedule is provided in the following. The complexity analysis of algorithms is introduced in week 1. Week 2-4 are devoted to discuss linked lists and their operations. In week 5, stacks and queues are introduced in the class. Recursion is discussed in week 7 after having the 1st exam in week 6. From week 8 to 10, we focus on trees and various traversal algorithms. The 2nd exam is scheduled in week 11. After that, it takes two weeks to discuss graphs, traversal algorithms, and topological ordering. From week 15 to 16, we discuss sorting, searching, and hashing. The 3rd exam is scheduled in week 17.

A. Homework #1

The *objective* of this homework is to let students practice and refine the techniques used to assess the efficiency of algorithms and select the most appropriate data structure and algorithm for applications.

Homework Description:

Part One: Given the following program, calculate its asymptotic complexity (Big-O) in terms of n. Show all your work step-by-step as the examples provided in the class. Answer only without process receives half points. (This is the last question in Homework #1, where other complexity analysis questions have similar description with different program.)

```
public static int function(int[ ] first, int[ ] second) {
    int n = first.length, count = 0;
    for (int i=0; i < n; i++) {
        int total = 0;
        for (int j=0; j < n; j++)
            for (int k=0; k <= j; k++)
                total += first[k];
        if (second[i] == total)
            count++;
    }
    return count;
}
```

Fig. 1. Complexity analysis program.

Part Two: Use a generative AI, such as ChatGPT, Bing AI, or Bard, to generate a ''solution'' for the program in the above question. (a) Attach the screenshot of ''solution'' generated by AI tool. and (b) Compare your solution with the ''solution'' generated by generative AI, and summarize your findings (at least 50 words).

Grading Policy:

- No answer and no process: receive 0 pt.
- Correct answer without process: deduct 0.5 pt.
- Correct answer with wrong process: deduct 1 pt or 1.5 pts depending on the degrees of wrongness.
- Correct answer with correct process: receive all pts.
- Wrong answer without process: receive 0.5 pt.
- Wrong answer (e.g., math calculation error, typo. etc) with correct process: deduct 0.5 pt.
- Wrong answer (e.g., math calculation error, typo. etc) with wrong process: receive 1 pt.
- The instructor will decide the grade policy of any scenario which is not covered by the above list. Meanwhile, please kindly contact the instructor if you have any questions regarding the grading policy.

B. Homework #2

The *objective* of this homework is to help students understand and master the concepts and implementation of linked lists and their operations

Homework Description:

Part One: Write a program that can insert, delete, search, and print nodes using singly linked lists and doubly linked lists. Your program should be menu-driven and execute the chosen operation as shown in

Fig .2. If you type 12, then exit the program. Here, IH (Insert Head), IT (Insert Tail), DH (Delete Head), DT (Delete Tail), SD (Search & Delete), PS (Print Single Linked List), and PD (Print Double Linked List). Display an ''error'' message when a node to be searched or deleted does not exist. Or try to delete a node in the empty list.

MENU
SLL: IH(0), IT(1), DH(2), DT(3), SD(4), PS(5)
DLL: IH(6), IT(7), DH(8), DT(9), SD(10), PD(11)
Exit Program (12)

Fig. 2. Homework #2.

Part Two: Use a generative AI, such as ChatGPT, Bing AI, or Bard, to debug one programming problem (e.g., error message from compiler, unexpected behavior, etc.) you encounter while programming, obtain a valid response from generative AI, and fix the problem accordingly.

Grading Policy:

- Compilation error/run-time error: Deduct 5 pts first, and then apply the following grading policy
- Singly linked list: 9 pts
  - Each operation has 1.5 pts
    - Correct output: receive 1.5 pts
    - Incorrect output:
      - The idea/logic of algorithm is wrong: deduct 1 pt
      - Partial points (e.g., 0.5 pts) are deducted depending on the degrees of wrongness.
- Doubly linked list:9 pts
  - Each operation has 1.5 pts.
    - Correct output: receive 1.5 pts
    - Incorrect output:
      - The idea/logic of algorithm is wrong: deduct 1 pt
      - Partial points (e.g., 0.5 pts) are deducted depending on the degrees of wrongness.
- No testing conducted/No WORD document: deduct 1 pt.
- No generative AI debugging/No WORD document: deduct 1 pt.
- Using Linked List library/package in Java and C++ is Not Allowed to use. Deduct 10 pts first. and then apply the above grading policy.

C. Homework #3

The *objective* of this homework is to help students understand and master the concepts and implementation of stacks and queues and their operations.

Homework Description:

Part One: Write a program to convert a number from a decimal notation to a number expressed by a number system whose base is 2 (binary), 8 (octal), or 16 (hexadecimal). The conversion is performed by repetitious division by the base to which a number is being converted and then taking the remainders of division in the reverse order. For example, in converting to binary, number 6 requires three such divisions: 6/2 = 3 remainder 0, 3/2 = 1 remainder 1, and finally, 1/2 = 0 remainder 1. The remainders 0, 1, and 1 are put in a reverse order so that the binary equivalent of 6 is equal to 110. Your program should be menu-driven and execute the chosen command. If you type 3, then exit the program.

MENU
Binary (0), Octal (1), Hexadecimal (2)
Exit Program (3)

Fig. 3. Homework #3.

Part Two: Use a generative AI, such as ChatGPT, Bing AI, or Bard, to generate a stack implemented by a linked list. (only the stack implementation which is independent of the number conversion problem) (a) Attach the screenshot of ''solution'' generated by generative AI. and (b) Compare your implementation with the ''program'' generated by generative AI and summarize your findings (at least 50 words).

Grading Policy:

- Compilation error/run-time error: Deduct 5 pts first, and then apply the following grading policy
- Stack implementation in linked list: 4 pts
  - Stack implementation correctly: receive 4 pts
  - Stack implemented incorrectly: Partial points (e.g., 0.5 pt) are deducted depending on the degrees of wrongness
    - Stack class: 1 pt
    - Pop: 1 pt
    - Push: 1 pt
  - Stack implementation in non-linked list: deduct 2.5 pts
  - No stack implementation: deduct 3.5 pts
- Binary/Octal/Hexadecimal number conversion: 4 pts
  - Correct output: 4 pts
    - Using Stack: receive 4 pts
    - Not using Stack: deduct 3 pts
  - Incorrect output:
    - Simple mathematical calculation/formula error: deduct 1 pt
    - The idea/logic of algorithm is wrong: deduct 3 pts (Partial points (e.g., 2 pts) are deducted depending on the degrees of wrongness.)
- No testing conducted/No WORD document: deduct 2 pts
- AI tool Stack implementation: 2 pts
  - ''Program'' from generative AI: 1 pt
  - Comparison between student's program and ''program'' from generative AI: 1 pt

D. Homework #4

The *objective* of this homework is to help students understand and master the concepts and implementation of recursion, and apply recursion to solve real-world application problems.

Homework Description:

Part One: Given an M × N rectangular grid, write a program to discover all routes in the grid starting at the source (0, 0) and ending at the destination (M-1, N-1). During the discovery, you can move down or right or diagonally (down-right), but not up or left. Your program should be menu-driven and execute the chosen command. If you type 3, then exit the program.

MENU

Horizontal Axis (0), Vertical Axis (1), Start Discovery (2), Exit Program (3)

Fig. 4. Homework #4.

Part Two: Use a generative AI, such as ChatGPT, Bing AI, or Bard, to optimize program, function, or code segment. (a) Attach the screenshot of ''solution'' generated by generative AI. and (b) Summarize your findings (at least 50 words). Here are several ways that you can use generative AI to optimize program, function, or code segment for performance. (i) Best practices: Having been trained on a wide range of code patterns, ChatGPT can help you follow best practices for coding, leading to more efficient and optimized code. (ii) Refactoring: ChatGPT can help to reorganize existing code to improve its efficiency and maintainability without affecting its functionality. and (iii) Code Suggestions: ChatGPT can suggest code snippets or alternative solutions to improve the performance of your existing code. (You need to have your own implementation first before asking AI tool for alternative solution.)

Grading Policy:

- Compilation error/run-time error: Deduct 5 pts first, and then apply the following grading policy
- Use recursion to discover all routes: 8 pts
  - Recursion implementation correct: receive 8 pts
  - Recursion implementation wrong: Partial points (e.g., 1 pt or 1.5 pts) are deducted depending on the degrees of wrongness.
    - Base case: 3 pts
    - Inductive clause: 5 pts
  - No recursion implementation: deduct 8 pts
- Program menu as required: 1 pt
- Horizontal Axis input: 1 pt
- Vertical Axis input: 1 pt
- Flexibility of changing Horizontal Axis and Vertical Axis before ''Start Discovery'': 1 pt
- Program output: 4 pts
  - Correct output: receive 4 pts
  - Wrong output: Partial points (e.g., 1 pt or 2 pts) are deducted depending on the degrees of wrongness.
- No testing conducted/No WORD document: deduct 2 pts
- Generative AI: deduct 2 pts
  - Screenshots of generative AI responses: 1 pt
  - Student's summary: 1 pt

E. Homework #5

The *objective* of this homework is to help students understand and master the concepts and implementation of trees, and various tree traversal algorithms.

Homework Description:

Part One: Write a program to build a binary tree from a sequence of data. Once the tree is constructed, conduct search and tree traversal functions including breadth-first traversal and depth-first traversal (preorder, inorder, and postorder). Search and tree traversal functions are applied to the most recently constructed tree. Your program should be menu-driven and execute the chosen command. If you type 6, then exit the program.

MENU

Create (0), Search (1), Breadth-First Traversal (2) Depth-First Traversal: preorder (3), inorder (4), postorder (5) Exit Program (6)

Fig. 5. Homework #5.

Part Two: Use a generative AI, such as ChatGPT, Bing AI, or Bard, to generate the idea of creating binary tree, Breadth-First Traversal, and Depth-First Traversal (preorder, inorder, and postorder), verify/validate the correctness of ideas, and summarize your findings (at least 50 words).

Grading Policy:

- Compilation error/run-time error: Deduct 5 pts first, and then apply the following grading policy
- Creating binary tree: 3 pts
- Breadth-First Traversal: 3 pts
  - Correct output: 3 pts
  - Wrong output: Partial points (e.g., 1 pt or 2 pts) are deducted depending on the degrees of wrongness
- Depth-First Traversal – Preorder/Inorder/Postorder: 3 pts
  - Correct output: receive 3 pts
  - Wrong output: receive 1 pt
- Program menu as required: 1 pt
- No testing conducted/No WORD document: deduct 2 pts
- Generative AI: 2 pts
  - Screenshots of generative AI responses: 1 pt
  - Student's summary: 1 pt

F. Homework #6

The *objective* of this homework is to help students understand and master the concepts and implementation of graphs, and various graph search algorithms.

Homework Description:

Part One: Write a program to conduct a depth-first search using a stack and the minimum path search (e.g., breadth-first search) using a queue based on the following directed graph and its adjacency lists. In the depth-first search, any starting node can be selected (e.g., user input) in the graph. In the minimum path search, both starting and ending nodes should be selected (e.g., user input). Your program should be menu-driven and execute the chosen command. If you type 3, then exit the program.

MENU
Depth-First Search (0), Minimum Path Search (1)
Exit Program (3)

Fig. 6. Homework #6.

Part Two: Use a generative AI, such as ChatGPT, Bing AI, or Bard, to generate the idea of topological ordering algorithm, verify/validate the correctness of ideas, and summarize your findings (at least 50 words).

Grading Policy:

- Compilation error/run-time error: Deduct 5 pts first, and then apply the following grading policy.
- Program menu as required: 2 pts
- Depth-First Search: 7 pts
  - Correct output: 7 pts
  - Incorrect output:
    - The idea/logic of algorithm is wrong: deduct 5 pts
    - Partial points (e.g., 2 pts) are deducted depending on the degrees of wrongness.
- Minimum Path Search: 7 pts
  - Correct output: 7 pts
  - Incorrect output:
    - The idea/logic of algorithm is wrong: deduct 5 pts
    - Partial points (e.g., 2 pts) are deducted depending on the degrees of wrongness.
- No testing conducted/No WORD document: deduct 2 pts
- Generative AI: 2 pts
  - Screenshots of generative AI responses: 1 pt
  - Student's summary: 1 pt

VI. Evaluation Survey

To collect students' valuable opinions about the usage of generative AI to improve student learning outcomes in the CS2 course, we conducted a student perception survey at the end of Fall 2023 semester. The perception survey was organized and coordinated by Dr. Kristi Dickey and Dr. Gina Morris from OSU Institute for Teaching and Learning Excellence (ITLE). Students who attended the class were provided 30 minutes of class time to complete the survey without feeling rushed while the instructor was not present in the classroom.

The perception survey is composed of six (6) questions: five (5) likert scale questions and one (1) open-ended question. The likert scale questions and their corresponding responses are shown in Table I, where 47 out of 61 students submitted the perception survey (77% response rate). In addition, the open-ended question is as follows: If you were the instructor, what changes would you like to make for the usage of generative AI to improve programming skills as well as design and implement efficient algorithms and data structures in this class?

Overall, 85.1% of respondents agree or strongly agree that generative AI can help them understand testing and debugging better. 76.6% of respondents agree or strongly agree that generative AI have helped to improve their coding skills and code quality. 74.5% of students agree or strongly agree that generative AI is able to help them design and implement efficient algorithms and data structures. 63.8% of students agree or strongly agree that they can select appropriate algorithms and data structures with the assistance of generative AI. Finally, 70.2% of students agree or strongly agree that generative AI can help them understand the importance of designing and implementing efficient algorithms and data structures.

The selected student feedback from the open-ended question is provided below.

- ''Harder course homework.''
- ''I would make students explain more about their programs to prevent cheating using generative AI.''
- ''I have no improvements to suggest. Generative AI is used correctly in this class.''
- ''Do more cases of 'improve students' code' to see how students can improve their programs.''
- ''I would only use the responses of generative AI to compare with my own code to see which part can be more efficient.''
- ''I don't think any changes need to be made. The use of generative AI in this class works for my learning.''
- ''Allow generative AI more often for programming assignment. Generative AI provides assistance to those who are struggling with code and syntax. If you're having trouble, you can look it up on Google and never find a solution. Generative AI makes it much easier and quicker to find a solution.''
- ''I actually think the way the instructor has implemented them in each of our assignments is very useful and discourages cheating.''
- ''Visually show us how we should be using generative AI to our advantage.''
- ''I like what the instructor does with generative AI already. Helps with debugging as well as comparing your code to what generative AI produces. It allows you to see

TABLE I
STUDENT PERCEPTION SURVEY QUESTIONS AND RESPONSES.

Question: The generative AI tools helped me	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree	Total
understand testing and debugging better.	2	0	5	23	17	47
improve coding skills and code quality.	2	3	6	19	17	47
design and implement efficient algorithms and data structures.	2	1	9	23	12	47
select appropriate algorithms and data structures.	1	3	13	21	9	47
understand the importance of designing and implementing efficient algorithms and data structures.	3	2	9	22	11	47

what you could work on.''

- ''I wouldn't change how the instructor has us use it, I personally use it for debugging purposes or to help with any roadblocks that I have.''
- ''Maybe teach us how to use generative AI effectively to get them to help us with our code instead of just having generative AI fix something. I'm not sure how feasible that is though.''
- ''I would make sure that errors in generative AI judgment were emphasized.''
- ''As the instructor, I would introduce several different ways to use generative AI while also restricting the usage. Overreliance on AI could hamper the learning experience of students.''
- ''I would have them use generative AI to debug each time they program, as this will quickly find faults, but when even generative AI can't find the bug, it improves debugging skills.''
- ''Compare the output of generative AI to a working implementation.''
- ''Not sure that's difficult to say but I appreciate the instructor's openness to generative AI and informing us how to use them as a study tool. Maybe demonstrate generative AI in class as in comparing the instructor's algorithm to the response of generative AI.''
- ''I would have more descriptive or directed questions that need to be asked of generative AI when debugging the code so that students have more to compare and contrast for better understanding. I like the ones we have had so far, but more directed questions would be good.''
- ''I would do the same thing but ask students to add comments in their programs; they have to provide comments in the programs to show that they understand what is going on.''

VII. LESSONS LEARNED

In this section, we describe the lessons learned after adopting the redesigned course homework with the integration of generative AI.

Communicate expectation of the use of generative AI to students: As generative AI becomes more pervasive, many colleges and universities have developed sample statements about their usage in the classroom. For example, OSU provides the following sample statement for three scenarios of using generate AI in the learning environment.

- For Faculty Who Do NOT Allow Generative AI Tools in the Learning Environment: ''*Students are prohibited from using any generative AI tools such as ChatGPT, Bing AI, or Bard when completing course assignments. Use of these tools, or other similar generative AI tools, will not be tolerated and will be considered plagiarism and could result in the student failing the course. Any incident detected will be addressed through the university's academic integrity procedures.*''
- For Faculty Who Allow Generative AI Tools to be Used in Some Circumstances or With Explicit Permission: ''*Students may access and use generative AI tools, such as ChatGPT, Bing AI, or Bard, to assist them in their learning of the course content. Appropriate uses may include generating ideas for writing assignments, checking facts of a phenomenon, and assessing a paper for grammatical errors that are written by the student. Such uses of the tool assist students in learning the content and will therefore be permitted. However, students are prohibited from using generative AI tools to completely produce, reproduce, and/or manufacture paper and/or other assignments without using any personal effort devoted to the learning process. Before using generative AI tools, students should check to ensure they do not conflict with copyright laws or other's proprietary information.*''
- For Faculty Who Allow Full Use of Generative AI Tools in the Learning Environment: ''*Students may use generative AI tools such as ChatGPT, Bing AI, or Bard, to help them learn course content, complete course assignments, or do other course-related tasks. Students are expected to provide attribution for any text created using generative AI tools as appropriate.*''

Since many universities still do not have universal policy about generative AI, it is the instructors' responsibilities to provide students with clear guidance about what is and is not allowed on each course work. Communicating the expectation of generative AI should be done orally in class when discussing each and every homework or exam. In the course syllabus, specific scenarios in which it is allowed to use generative AI and when it is not should be clearly indicated. Last but not least, TAs should also be informed about the expectations of

the use of generative AI in the classroom.

Help students understand the rules round generative AI in the classroom: If instructors allow generative AI in the learning environment with or without explicit permission, they should explain and justify the rationale for their rules around generative AI as crystal as possible. In many cases, students do not fully understand how the course work instructors create for them achieve learning objectives. Thus, it is extremely important for instructors to help students identify in what ways using generative AI can assist them achieving learning outcomes. The more students are aware of why generative AI is not allowed, the more likely they will follow the rules.

Use generative AI to apply CS2 concepts rather than simply recall definitions: One of attractive features of generative AI is to provide fast and easy information access, however, this might not be very appropriate in the CS2 courses. Instead of asking generative AI to provide the definition of data structures and the basic ideas of algorithms, they should be used to help students think critically about data structures and algorithms they're already familiar with. For example, students can ask generative AI why one algorithm is better than another algorithm. Then, they can ask again why the second algorithm is better. In this way, students can learn to appreciate other arguments and see both sides objectively before establishing an opinion about the strengths and weaknesses of both algorithms.

VIII. Conclusion

In Fall 2023 semester, we redesigned all course homework in the CS2 course at OSU. The redesigned course homework were integrated with generative AI so that students had opportunities to use this emerging advanced technology to support and improve learning outcomes in the CS2 course. In order to collect students' valuable opinions about the usage of generative AI in the classroom, we conducted a student perception survey at the end of course. Based on the survey responses, a majority of respondents perceived that generative AI helped them understand testing and debugging better, improve their coding skills and code quality, design and implement efficient algorithms and data structures, select appropriate algorithms and data structures with the assistance of generative AI, and understand the importance of designing and implementing efficient algorithms and data structures. In addition, we shared lessons learned and provided candid suggestions for utilizing similar course homework in CS2 courses at other institutions.

Acknowledgment

The author is extremely grateful to Dr. Kristi Dickey and Dr. Gina Morris from the Institute for Teaching and Learning Excellence at Oklahoma State University for discussing the goals of the course observation, conducting the course observation noting items related to the predetermined goals, as well as providing constructive feedback and student survey results based on the course observation.

References

[1] S. Su, E. Zhang, P. Denny, and N. Giacaman, "A Game-Based Approach for Teaching Algorithms and Data Structures using Visualizations," in *Proc. ACM SIGCSE*, 2021, pp. 1128–1134.

[2] *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, 2023.

[3] L. Wittie, A. Kurdia, J. Peng, J. Kelly, and M. Huggard, "Developing a Concept Inventory for Computer Science 2: What should it focus on and what makes it challenging?" in *Proc. IEEE FIE*, 2020, pp. 1–8.

[4] L. Porter, D. Zingaro, C. Lee, C. Taylor, K. Webb, and M. Clancy, "Developing Course-Level Learning Goals for Basic Data Structures in CS2," in *Proc. ACM SIGCSE*, 2018, pp. 858–863.

[5] A. Barczak, A. Mathrani, B. Han, and N. Reyes, "Automated assessment system for programming courses: a case study for teaching data structures and algorithms," *Educational Technology Research and Development*, pp. 1–24, 2023.

[6] C. Cath, S. Wachter, B. Mittelstadt, M. Taddeo, and L. Floridi, "Artificial intelligence and the 'good society': the US, EU, and UK approach," *Science and Engineering Ethics*, vol. 24, pp. 505–528, 2018.

[7] *As Japan's population drops, one city is turning to ChatGPT to help run the government*, https://www.cnn.com/2023/04/21/asia/japan-yokosuka-government-chatgpt-intl-hnk/index.html.

[8] *Chat GPT & Other Generative AI Software*, https://itle.okstate.edu/chatgpt.html.

[9] S. Murugesan and A. K. Cherukuri, "The Rise of Generative Artificial Intelligence and Its Impact on Education: The Promises and Perils," *IEEE Computer*, vol. 56, no. 5, pp. 116–121, 2023.

[10] A. Varga, *ACM Policy on Authorship*, https://www.acm.org/publications/policies/frequently-asked-questions.

[11] Y. Chen, S. Jensen, L. Albert, S. Gupta, and T. Lee, "Artificial Intelligence (AI) Student Assistants in the Classroom: Designing Chatbots to Support Student Success," *Information Systems Frontiers*, vol. 25, no. 1, pp. 161–182, 2023.

[12] K. Schroeder, M. Hubertz, R. V. Campenhout, and B. Johnson, "Teaching and Learning with AI-Generated Courseware: Lessons from the Classroom," *Online Learning*, vol. 26, no. 3, pp. 73–87, 2022.

[13] S. Hall, C. Shaffer, E. Fouh, M. H. ElShehaly, and D. Breakiron, "Evaluating Online Tutorials for Data Structures and Algorithms Courses," in *Proc. ASEE Annual Conference & Exposition*, 2013, p. 14.

[14] J. King, "Evaluating Online Tutorials for Data Structures and Algorithms Courses," in *Proc. ACM SIGCSE*, 2021, pp. 959–965.

[15] D. Ahmed, "Analysis of Student Learning Outcomes in Data Structures and Algorithms," in *Proc. IEEE CSCI*, 2022, pp. 2003–2007.

[16] G. Gárcia-Mateos and J. Fernández-Alemán, "A course on algorithms and data structures using on-line judging," in *Proc. ACM SIGCSE*, 2009, pp. 45–49.

[17] R. Nathasya, O. Karnalim, and M. Ayub, "Integrating program and algorithm visualisation for learning data structure implementation," *Egyptian Informatics Journal*, vol. 20, no. 3, pp. 193–204, 2019.

[18] P. Crescenzi and C. Nocentini, "Fully integrating algorithm visualization into a CS2 course. a two-year experience," *ACM SIGCSE Bulletin*, vol. 39, no. 3, pp. 296–300, 2007.

[19] D. Silva, R. de Lima Aguiar, D. Dvconlo, and C. Silla, "Recent studies about teaching algorithms (cs1) and data structures (cs2) for computer science students," in *Proc. IEEE FIE*, 2019, pp. 1–8.

[20] N. Kraft, X. Hong, J. Lusth, and D. ßMcCallum, "Experiences with CS2 and data structures in the 100 problems format," in *Proc. IEEE FIE*, 2011, pp. 1–7.

[21] Q. Cheng, "Enrich a data structures course with parallelism," in *Proc. IEEE FIE*, 2020, pp. 1–5.

[22] K. Fisler, S. Friedler, K. Lin, and S. Venkatasubramanian, "Approaches for weaving responsible computing into data structures and algorithms courses," in *Proc. ACM SIGCSE*, 2022, pp. 1049–1050.

[23] M. Goodrich, R. Tamassia, and M. Goldwasser, *Data structures and algorithms in Java*. Wiley, 2014.

[24] A. Drozdek, *Data Structures and Algorithms in C++*. Cengage Learning, 2012.