TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
Quezon City

College of Computer Studies
Computer Science Department

STUDENT PORTFOLIO

(CS 006 – Algorithm and Complexity)

2nd Semester
SY 2024-2025

Jaspher F. Cadelina

<SECTION CS32S3>

STUDENT PORTFOLIO

CS 006 -ALGORITHM AND COMPLEXITY

TABLE OF CONTENTS

SUMMARY RESULTS OF STUDENT ASSESSMENT

TABLE OF CONTENTS

Summary Results of Student Assessment

CS 404 – EXPERT SYSTEMS
<INSERT YOUR FULL NAME HERE>

Prelim Period

Prelim Grade = 50% Class Standing + 50% Prelim Exam (Major Exam)

Class Standing:

Assignments//Discussions/Recitation	= 10 %
Laboratory Activities	= 30 %
Quizzes/Project	= 60 %
TOTAL	= **100 %**

Midterm Period

Midterm Grade = 33% Prelim Grade + 67% Tentative Midterm Grade

Tentative Midterm Grade = 50% Class Standing + 50% Midterm Exam (Major Exam)

Class Standing:

Assignments//Discussions/Recitation	= 10 %
Laboratory Activities	= 30%
Quizzes/Project	= 60 %
TOTAL	**= 100 %**

Final Period

Final Grade = 33% Midterm Grade + 67% Tentative Final Grade

Tentative Final Grade = 50% Class Standing + 50% Final Exam (Final Project)

Class Standing:

Recitation	= 10 %
Laboratory Activities	= 30 %
Quizzes/Project	= 60 %
TOTAL	**= 100 %**

Preliminary Period

Class Standing	Average
Assignments//Discussions/Recitation	
Laboratory Activities	
Quizzes/Project	
Class Standing Grade	
Major Examination (Written and Laboratory Exam)	
Prelim Grade:	

Midterm Period

Class Standing	Average
Assignments//Discussions/Recitation	
Laboratory Activities	
Quizzes/Project	
Class Standing Grade	
Major Examination (Written and Laboratory Exam)	
Midterm Grade:	

Final Period

Class Standing	Average
Recitation	
Laboratory Activities	
Quizzes/Project	
Class Standing Grade	
Major Examination (Project)	
Final Grade:	

The student overall performance in the course – 95.00 is above the acceptable level of attainment of 75%, thus it is a clear indication that the student attained the specified SOs for the course.

Details of Student's Assessment Tasks:

PRELIMINARY			
Major Examinations	**Description**	**Score**	**Rating**
Preliminary Examination	Preliminary Examination (Laboratory Examination) Note:50% of Final Exam	28	93.33
		AVERAGE:	93.33
MIDTERM			
Midterm Examination	Midterm Examination(Laboratory Examination) Note:50% of Final Exam	44	88.00
		AVERAGE:	88.00
FINAL			
Final Examination	Final Examination (Final Project Requirements)		
		AVERAGE:	
		Major Examination Grade:	

PRELIMINARY			
Quizzes	**Description**	**Score**	**Rating**
Quiz No. 1.1	Quiz 1: Prelim Quiz	28	93.33
		AVERAGE:	**93.33**
MIDTERM			
Quiz No. 2.1	Quiz 1 | Midterm Handwritten Quiz	8	80.00
Quiz 2 - Midterms	Quiz 2 - Midterms Quiz 2 – Midterms	24	96
		AVERAGE:	**88.00**
FINAL			
		AVERAGE:	
		Quiz Grade:	

PRELIMINARY			
Assignments / Discussion	**Description**	**Score**	**Rating**
Discussion No. 1.0	Assignment #1 | Flowchart, Pseudocode and Algorithm Assignment #1 | Flowchart, Pseudocode and Algorithm	18	90.00
Discussion No. 2.0	Assignment #1 | Flowchart, Pseudocode and Algorithm Assignment #1 | Flowchart, Pseudocode and Algorithm	9	90.00
Discussion No. 3.0	Assignment # 2 | Time and Space Complexity [Coding Implementation] Assignment # 2 | Time and Space Complexity [Coding Implementation]	20	100.00

AVERAGE:	**93.33**
Assignment / Discussion Grade:	**93.33**

MIDTERM			
Assignments / Discussion	**Description**	**Score**	**Rating**
Discussion No. 3.1	Discussion #2 | Graph Theory Discussion #2 | Graph Theory /	20	100.00
Discussion No. 4.1	Discussion #3 | Jump and Interpolation Search Algorithm	15	100.00
Discussion No. 5.1	Discussion 4.1 | Brute Force Algorithm Discussion 4.1 | Brute Force	19	95.00
Discussion No. 6.1	Assignment 4.1 || Midterm Reflection Paper Assignment 4.1 || Midterm Reflection Paper	10	100
		AVERAGE:	**98.75**
		Assignment / Discussion Grade:	**98.75**

Finals			
Assignments / Discussion	**Description**	**Score**	**Rating**
		AVERAGE:	
		Assignment / Discussion Grade:	

Course Reflection			
Reflection	**Description**	**Score**	**Rating**
Reflection	Course Reflection	10	100.00
Reflection Paper	Reflection Paper	10	100.00
		AVERAGE:	100.00
		Course Reflection Grade:	**100.00**

Project Requirements			
Project	**Description**	**Score**	**Rating**
Final Project	Quiz #1 | Final Project Documentation and PPT	28	93.33
Final Project	Quiz #1 | Final Project Application	28	93.33
		AVERAGE:	**93.33**

ASSESSMENT TASKS

TABLE OF CONTENTS

LABORATORY ACTIVITIES

TABLE OF CONTENTS

PRELIMINARY PERIOD

TABLE OF CONTENTS

<Screenshot the all Laboratories Task that found in canvas under prelim>

<after the screenshot put Figure No.and the title of the Laboratory>
<figure no. start at 1.1>

MIDTERM PERIOD

TABLE OF CONTENTS

<Screenshot the all Laboratories Task that found in canvas under midterm>
<after the screenshot put Figure No.and the title of the Laboratory>

<figure no. will continue counting until the end of laboratory in finals>

FINAL PERIOD

TABLE OF CONTENTS

<Screenshot the all Laboratories Task that found in canvas under final>
<after the screenshot put Figure No.and the title of the Laboratory>

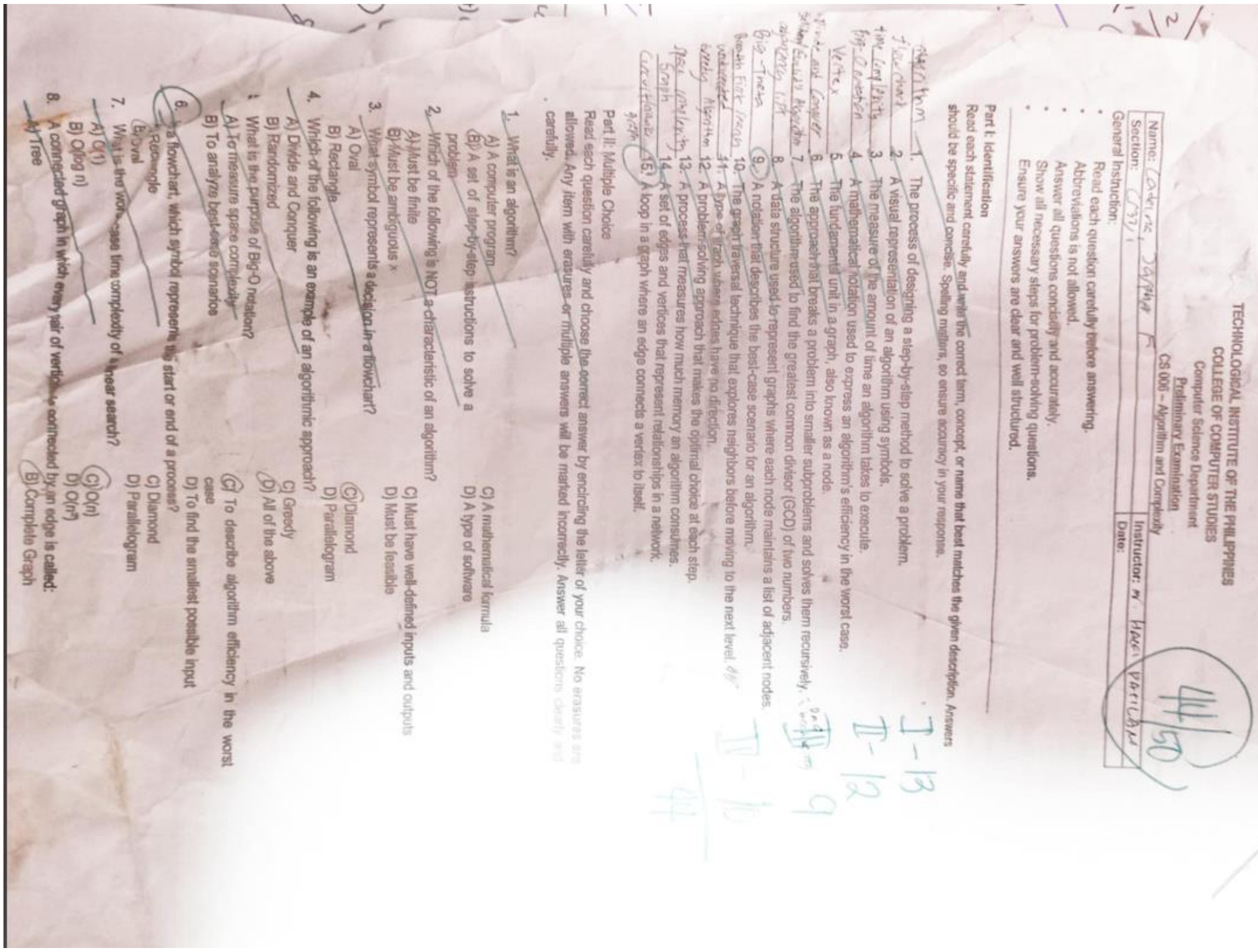
<figure no. will continue counting until the end of laboratory in finals>

MAJOR EXAMINATIONS

TABLE OF CONTENTS

PRELIMINARY PERIOD

TABLE OF CONTENTS



TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
COLLEGE OF COMPUTER STUDIES
Computer Science Department
Preliminary Examination
CS 000 – Algorithm and Complexity

44/50

Name: Cadeins, Jarpha F	Instructor: M. Haec Patilan
Section: C33/1	Date:

General Instruction:

- Read each question carefully before answering.
- Abbreviations is not allowed.
- Answer all questions concisely and accurately.
- Show all necessary steps for problem-solving questions.
- Ensure your answers are clear and well structured.

Part I: Identification

Read each statement carefully and write the correct term, concept, or name that best matches the given description. Answers should be specific and concise. Spelling matters, so ensure accuracy in your response.

I – 13
II – 12
III – 9
IV – 10
44

algorithm 1. The process of designing a step-by-step method to solve a problem.
flowchart 2. A visual representation of an algorithm using symbols.
time complexity 3. The measure of the amount of time an algorithm takes to execute.
Big-O notation 4. A mathematical notation used to express an algorithm's efficiency in the worst case.
Vertex 5. The fundamental unit in a graph, also known as a node.
Divide and Conquer 6. The approach that breaks a problem into smaller subproblems and solves them recursively.
Euclid's Algorithm 7. The algorithm used to find the greatest common divisor (GCD) of two numbers.
adjacency list 8. A data structure used to represent graphs where each node maintains a list of adjacent nodes.
Big-Theta 9. A notation that describes the best-case scenario for an algorithm.
Breadth First Search 10. The graph traversal technique that explores neighbors before moving to the next level.
undirected 11. A type of graph where edges have no direction.
Greedy Algorithm 12. A problem-solving approach that makes the optimal choice at each step.
space complexity 13. A process that measures how much memory an algorithm consumes.
Graph 14. A set of edges and vertices that represent relationships in a network.
Circuit/loop graph 15. A loop in a graph where an edge connects a vertex to itself.

Part II: Multiple Choice

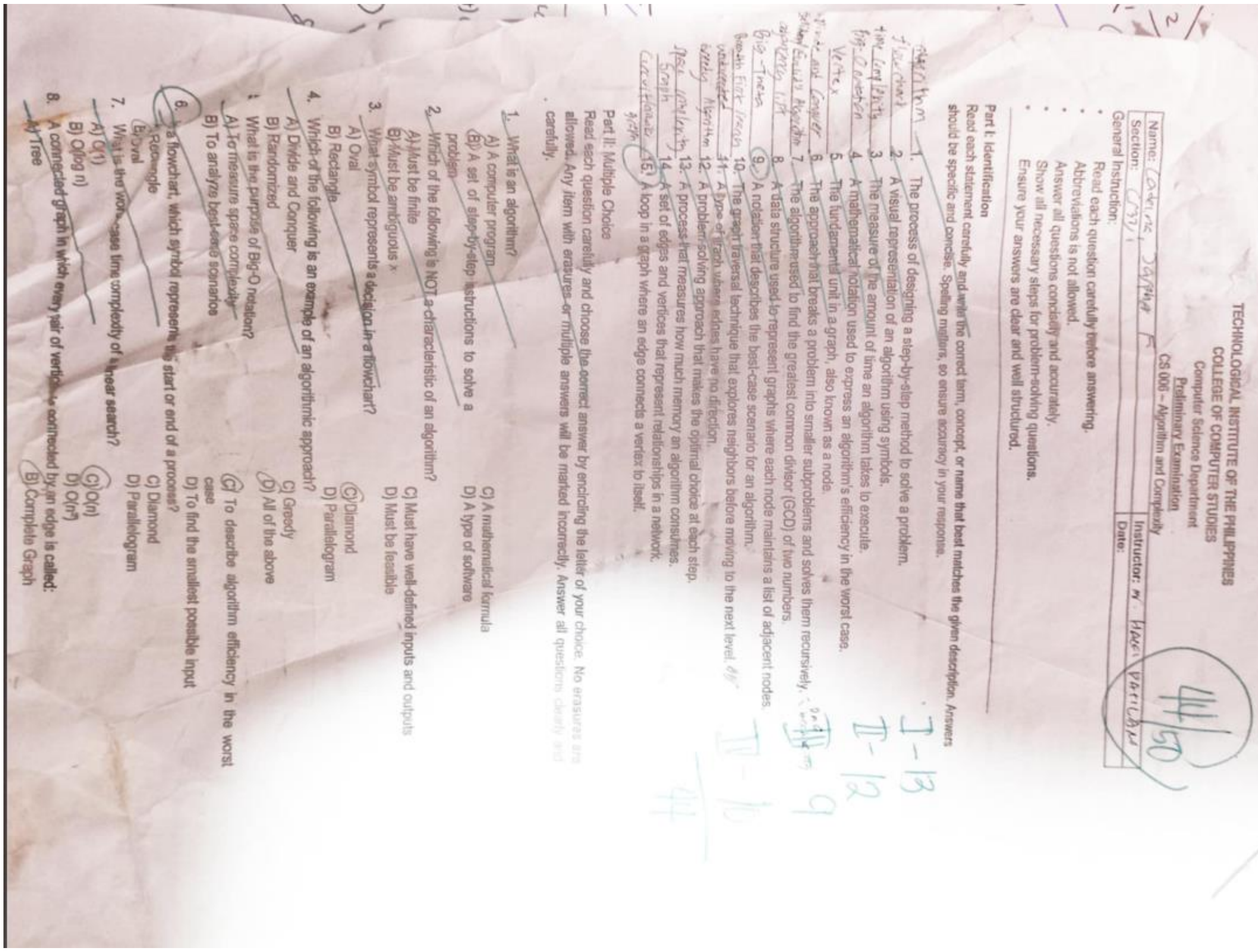
Read each question carefully and choose the correct answer by encircling the letter of your choice. No erasures are allowed. Any item with erasures or multiple answers will be marked incorrectly. Answer all questions clearly and carefully.

1. What is an algorithm?
   A) A computer program
   B) A set of step-by-step instructions to solve a problem
   C) A mathematical formula
   D) A type of software
2. Which of the following is NOT a characteristic of an algorithm?
   A) Must be finite
   B) Must be ambiguous
   C) Must have well-defined inputs and outputs
   D) Must be feasible
3. What symbol represents a decision in a flowchart?
   A) Oval
   B) Rectangle
   C) Diamond
   D) Parallelogram
4. Which of the following is an example of an algorithmic approach?
   A) Divide and Conquer
   B) Randomized
   C) Greedy
   D) All of the above
5. What is the purpose of Big-O notation?
   A) To measure space complexity
   B) To analyze best-case scenarios
   C) To describe algorithm efficiency in the worst case
   D) To find the smallest possible input
6. In a flowchart, which symbol represents the start or end of a process?
   A) Rectangle
   B) Oval
   C) Diamond
   D) Parallelogram
7. What is the worst-case time complexity of a linear search?
   A) O(1)
   B) O(log n)
   C) O(n)
   D) $O(n^2)$
8. A connected graph in which every pair of vertices is connected by an edge is called:
   A) Tree
   B) Complete Graph

Preliminary Examination
Figure 2.1

MIDTERM PERIOD

[TABLE OF CONTENTS](#)



TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
COLLEGE OF COMPUTER STUDIES
Computer Science Department
Preliminary Examination
CS 000 – Algorithm and Complexity

44/50

Name:	Instructor:
Section:	Date:

General Instruction:

- Read each question carefully before answering.
- Abbreviations is not allowed.
- Answer all questions concisely and accurately.
- Show all necessary steps for problem-solving questions.
- Ensure your answers are clear and well structured.

Part I: Identification

Read each statement carefully and write the correct term, concept, or name that best matches the given description. Answers should be specific and concise. Spelling matters, so ensure accuracy in your response.

1. The process of designing a step-by-step method to solve a problem.
2. A visual representation of an algorithm using symbols.
3. The measure of the amount of time an algorithm takes to execute.
4. A mathematical notation used to express an algorithm's efficiency in the worst case.
5. The fundamental unit in a graph, also known as a node.
6. The approach that breaks a problem into smaller subproblems and solves them recursively.
7. The algorithm used to find the greatest common divisor (GCD) of two numbers.
8. A data structure used to represent graphs where each node maintains a list of adjacent nodes.
9. A notation that describes the best-case scenario for an algorithm.
10. The graph traversal technique that explores neighbors before moving to the next level.
11. A type of graph where edges have no direction.
12. A problem-solving approach that makes the optimal choice at each step.
13. A process that measures how much memory an algorithm consumes.
14. A set of edges and vertices that represent relationships in a network.
15. A loop in a graph where an edge connects a vertex to itself.

I – 13
II – 12
III – 9
IV – 10
44

Part II: Multiple Choice

Read each question carefully and choose the correct answer by encircling the letter of your choice. No erasures are allowed. Any item with erasures or multiple answers will be marked incorrectly. Answer all questions clearly and carefully.

1. What is an algorithm?
   A) A computer program
   B) A set of step-by-step instructions to solve a problem
   C) A mathematical formula
   D) A type of software
2. Which of the following is NOT a characteristic of an algorithm?
   A) Must be finite
   B) Must be ambiguous
   C) Must have well-defined inputs and outputs
   D) Must be feasible
3. What symbol represents a decision in a flowchart?
   A) Oval
   B) Rectangle
   C) Diamond
   D) Parallelogram
4. Which of the following is an example of an algorithmic approach?
   A) Divide and Conquer
   B) Randomized
   C) Greedy
   D) All of the above
5. What is the purpose of Big-O notation?
   A) To measure space complexity
   B) To analyze best-case scenarios
   C) To describe algorithm efficiency in the worst case
   D) To find the smallest possible input
6. In a flowchart, which symbol represents the start or end of a process?
   A) Rectangle
   B) Oval
   C) Diamond
   D) Parallelogram
7. What is the worst-case time complexity of a linear search?
   A) O(1)
   B) O(log n)
   C) O(n)
   D) O(n²)
8. A connected graph in which every pair of vertices is connected by an edge is called:
   A) Tree
   B) Complete Graph

Midterms Examination
Figure 2.2

FINAL PERIOD

TABLE OF CONTENTS

<screenshot the examination and also put figure no and name o here>

QUIZZES

TABLE OF CONTENTS

PRELIMINARY PERIOD

TABLE OF CONTENTS

<screenshot the quiz and also put figure no and name of the screenshot here>

Quiz 1: Prelim

CADELIÑA, JASPHER submitted Feb 24 at 10:18pm

Quiz 1: Prelim

Due Feb 24 at 11:59pm **Points** 30 **Questions** 1 **Available** Feb 17 at 12am - Mar 17 at 11:59pm **Time Limit** None

Instructions

General Instruction:

- The quiz will be conducted onsite, and you will receive a printed copy of the questionnaire.
- Read all the instructions carefully before answering the questions.
- Use the provided answer sheet or write your answers directly on the questionnaire (as instructed).
- Ensure your handwriting is clear and legible.
- Double-check your answers before submission.
- Once the checked quiz is returned to you, that's when you can upload it.
- Upload your scanned or photographed answer sheet. Ensure the file is clear and readable.

This quiz was locked Mar 17 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	5 minutes	28 out of 30

Quiz results are protected for this quiz and are not visible to students.

Correct answers are hidden.

Score for this quiz: **28** out of 30

Quiz 1: Prelim

Figure 3.1

MIDTERM PERIOD

TABLE OF CONTENTS

<screenshot the quiz and also put figure no and name of the screenshot here>

Submission Details

Grade: **8 / 10**

Quiz 1 | Midterm Handwritten

No Preview Available

Add a Comment:

Quiz 2 | Midterm Handwritten
Figure 3.2

Quiz 2 - Midterms

Due Apr 22 at 11:59pm **Points** 25 **Questions** 25 **Available** Apr 22 at 10:30am - Apr 22 at 11:59pm 13 hours and 29 minutes
Time Limit 120 Minutes

This quiz was locked Apr 22 at 11:59pm.

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	13 minutes	24 out of 25

Quiz results are protected for this quiz and are not visible to students.

Correct answers are hidden.

Score for this quiz: **24** out of 25

◂ Previous Next ▸

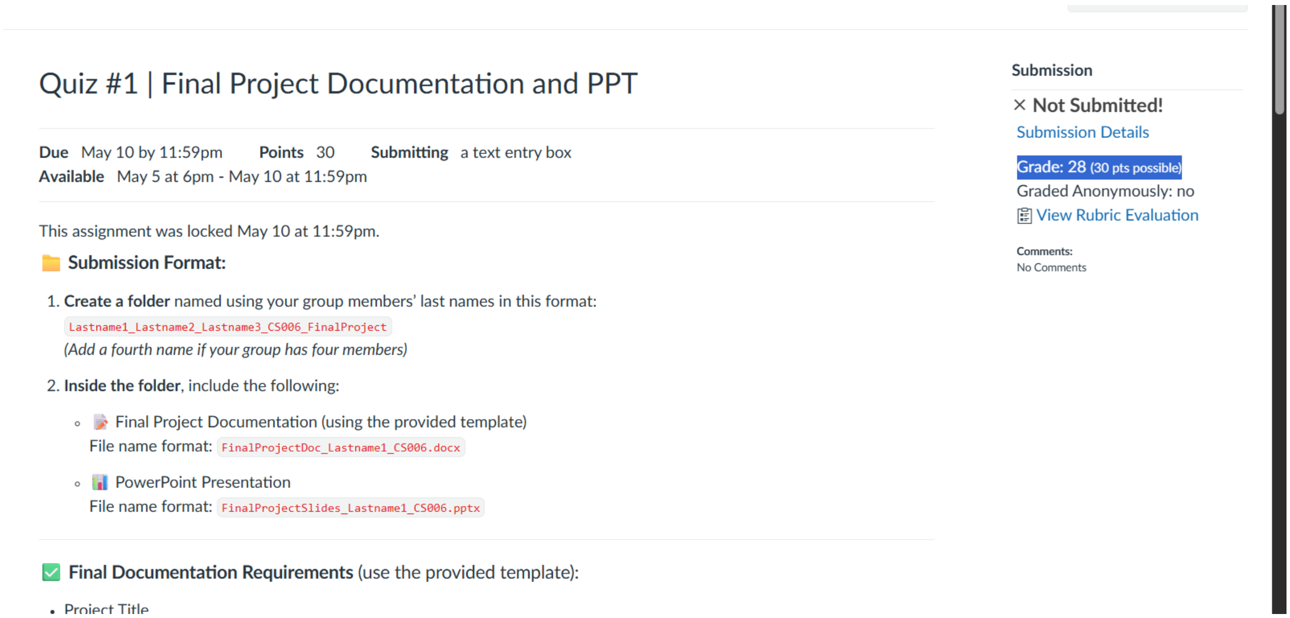
Submission Details:

Time:	13 minutes
Current Score:	24 out of 25
Kept Score:	24 out of 25

Quiz 3 - Midterms
Figure 3.3

FINAL PERIOD

TABLE OF CONTENTS



Quiz #1 | Final Project Documentation and PPT

Due May 10 by 11:59pm **Points** 30 **Submitting** a text entry box
Available May 5 at 6pm - May 10 at 11:59pm

This assignment was locked May 10 at 11:59pm.

Submission Format:

1. **Create a folder** named using your group members' last names in this format:
   `Lastname1_Lastname2_Lastname3_CS006_FinalProject`
   (Add a fourth name if your group has four members)
2. **Inside the folder**, include the following:
   - Final Project Documentation (using the provided template)
     File name format: `FinalProjectDoc_Lastname1_CS006.docx`
   - PowerPoint Presentation
     File name format: `FinalProjectSlides_Lastname1_CS006.pptx`

Final Documentation Requirements (use the provided template):

- Project Title

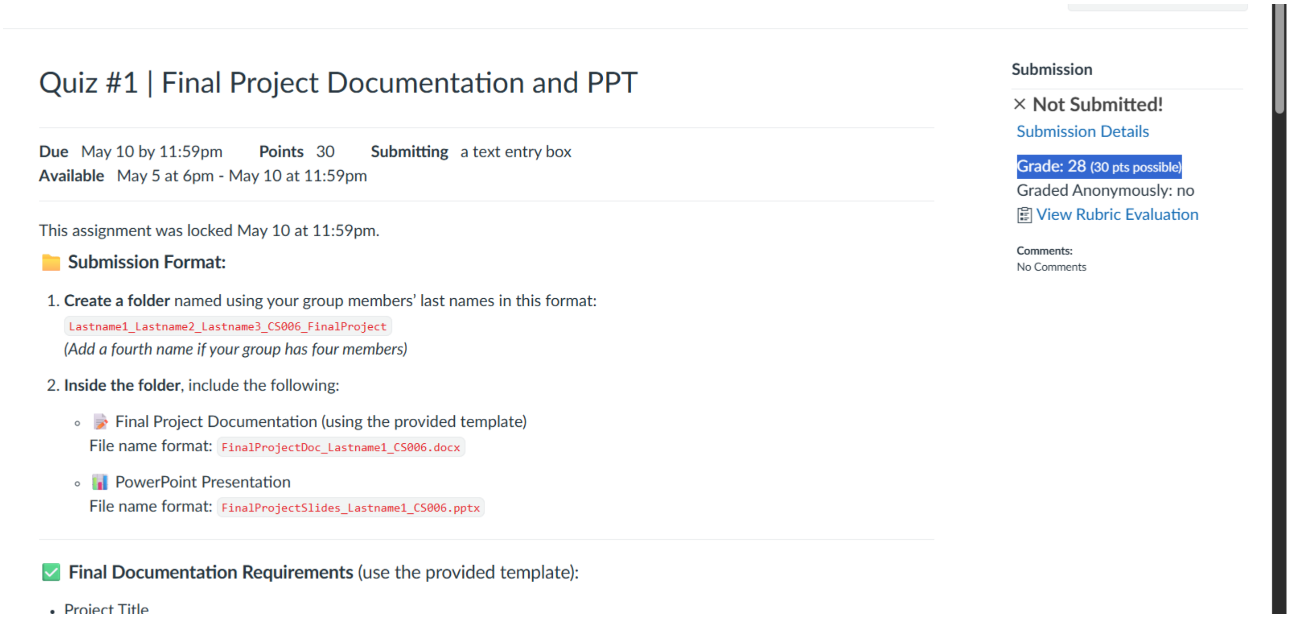
Submission

× Not Submitted!
Submission Details

Grade: 28 (30 pts possible)
Graded Anonymously: no
View Rubric Evaluation

Comments:
No Comments

Quiz 4 - Final Project Documentation and PPT
Figure 3.4



Quiz #1 | Final Project Documentation and PPT

Due May 10 by 11:59pm **Points** 30 **Submitting** a text entry box
Available May 5 at 6pm - May 10 at 11:59pm

This assignment was locked May 10 at 11:59pm.

Submission Format:

1. **Create a folder** named using your group members' last names in this format:
   `Lastname1_Lastname2_Lastname3_CS006_FinalProject`
   (Add a fourth name if your group has four members)
2. **Inside the folder**, include the following:
   - Final Project Documentation (using the provided template)
     File name format: `FinalProjectDoc_Lastname1_CS006.docx`
   - PowerPoint Presentation
     File name format: `FinalProjectSlides_Lastname1_CS006.pptx`

Final Documentation Requirements (use the provided template):

- Project Title

Submission

× Not Submitted!
Submission Details
Grade: 28 (30 pts possible)
Graded Anonymously: no
View Rubric Evaluation

Comments:
No Comments

Quiz 5 - Final Project Presentation and Application
Figure 3.

ASSIGNMENTS / DISCUSSION

TABLE OF CONTENTS

PRELIMINARY PERIOD

[TABLE OF CONTENTS](#)

<screenshot all assignment / discussion and also put figure no and name of the screenshot her

CADELIÑA, JASPHER

Feb 21 11:03pm

1. What is your takeaways from the lecture?

- A key takeaway I learned from this lecture is that the Euclid's Algorithm, where technically we just demo it for algorithm's discussion, is a powerful example about the word of "how we should implement an algorithm.". I learned that if we are creating an algorithm, we have to do it in step-by-step manner, not our intuition first. We have to understand the root solution, then break it down that the computer program can follow up the logic that we want to tell to computer program.

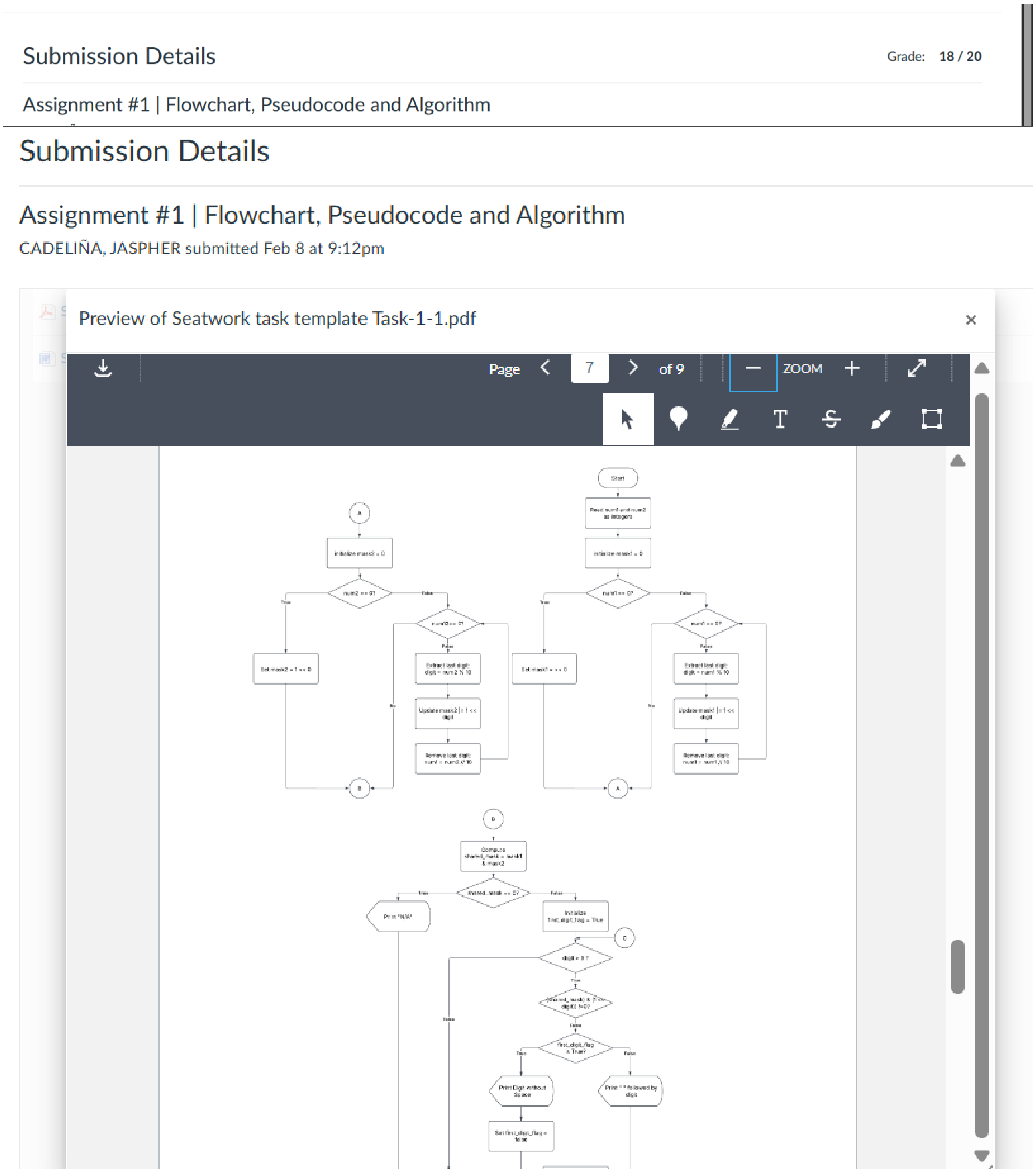
2. How will you relate concepts of Euclid's Algorithm /GCD in computer science?

In modular arithmetic, optimization, and algorithm design, GCD calculations are essential. Euclid's Algorithm is a prime example of effective problem-solving and has numerous uses in computer science, number theory, and numerous other areas of technology, all of which depend on the efficient problem-solving, scheduling, and resource allocation that GCD offers.

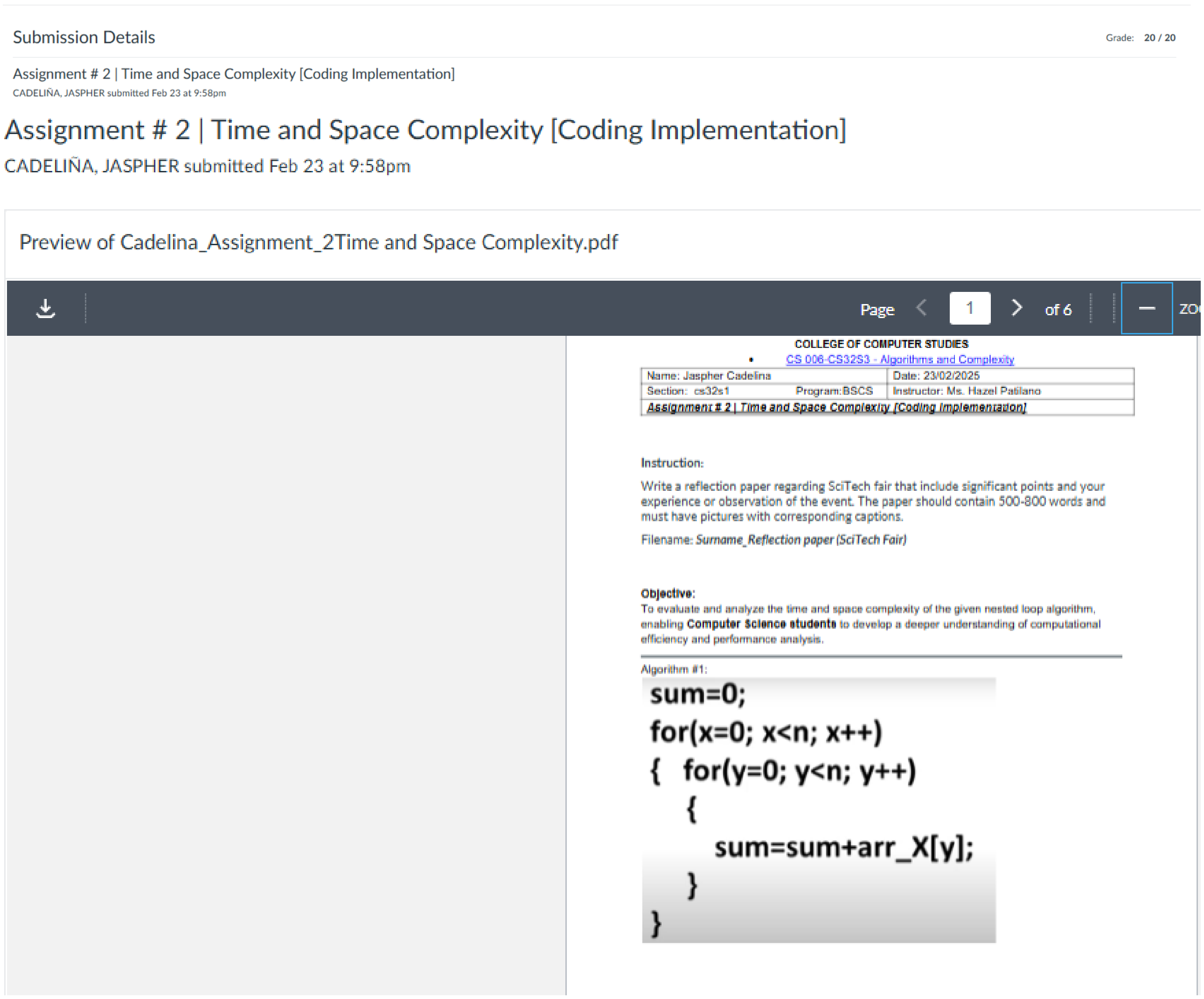
Mark as Unread

Discussion #1: GCD

Figure 4.1



Assignment #1 | Flowchart, Pseudocode and Algorithm
Figure 4.2



Assignment # 2 | Time and Space Complexity [Coding Implementation]

Figure 4.3

MIDTERM PERIOD

TABLE OF CONTENTS

<screenshot all assignment / discussion and also put figure no and name of the screenshot her

Submission Details

Grade: **20 / 20**

Assignment # 2 | Time and Space Complexity [Coding Implementation]

CADELIÑA, JASPHER submitted Feb 23 at 9:58pm

Discussion #2 | Graph Theory

CADELIÑA, JASPHER submitted Mar 3 at 6:54pm

1. What is a graph in the context of graph theory?

In graph theory, a **graph** is a mathematical structure used to model pairwise relationships between objects. It consists of a set of objects, called **vertices** (or nodes), and a set of connections betw
transportation systems, or communication systems.

WHERE G = (V,E)

- V is a set of vertices.
- E is a set of edges, which are pairs of vertices indicating a connection between them.

Graphs can be **directed** (edges have a specific direction) or **undirected** (edges have no direction). They can also be **weighted** (edges have associated numerical values) or **unweighted** .

2. What are the basic elements of a graph?
The two basic elements of a graph are:

Vertices (Nodes): These are the individual points or objects within the graph. Vertices are used to represent entities like people, cities, web pages, or any other discrete items being modeled.
Edges (Links): These are the links or relationships between vertices. Edges specify how the vertices interact or are related to one another.
Other attributes that can be present in a graph are:

Weights: Numerical weights given to edges to specify cost, distance, or relationship strength.
Directionality: In directed graphs, edges have direction, indicated by arrows.
Loops: Edges going from a vertex to itself.
Multiple Edges: In a multigraph, there can be several edges between the same vertices.

3. **Vertices and edges representation:**

- **Vertices:** Represented as points, circles, or labeled nodes.
- **Edges:** In undirected graphs, edges are lines connecting vertices. In directed graphs, edges are arrows indicating direction.
- **Graph representation:** Can be visual (dots/circles and lines/arrows) or mathematical (adjacency matrix or adjacency list).
- **Representation:**
  Visual Representation: Graphs are usually sketched with vertices as circles or dots and edges as arrows or lines.
  Mathematical Representation: A graph is represented by an adjacency matrix where rows and columns are vertices, and entries in the matrix point to the absence or presence of edges.
  Or else, a graph can be shown with an adjacency list, wherein every vertex maintains a list of its neighboring vertices.

Discussion #2 | Graph Theory

FIGURE 4.4

Submission Details

Grade: **15 / 15**

Discussion #3 | Jump and Interpolation Search Algorithm

CADELIÑA, JASPHER submitted Mar 24 at 6:50pm

1. Jump Search

- Advantages:
  - Technically, I see this to be faster than linear search for large sorted arrays because it skips elements in fixed steps SUCH AS USING SQUARE ROOT OF (n)
  - Personally, this is simpler to implement compared to interpolation but NOT in binary search
- Disadvantages:
  - Requires the array to be sorted. So not for dynamic datasets
  - If the step size is too large or too small, efficiency drops. After jumping past the target, a linear search is needed in the last "block," which causing overhead

https://www.geeksforgeeks.org/jump-search/

2. Interpolation Search vs. Binary Search

- Binary Search:
  - Splits the array into half repeatedly, guaranteeing O(log n) time complexity
    - HOWEVER, the data must be sorted initially for this to perform better, that's my personal take.
  - Works well for all sorted datasets, regardless of distribution.
- Interpolation Search:
  - Estimates the target's position using a formula
    - **pos = low + (target - arr[low]) * (high - low) / (arr[high] - arr[low])**.
    - https://www.geeksforgeeks.org/g-fact-84/
  - Faster than binary search for uniformly distributed O(log log n) only in average cases.
  - NOT ideal when using for skewed or non-uniform data (worst-case O(n))

https://www.geeksforgeeks.org/g-fact-84/
https://www.geeksforgeeks.org/interpolation-search/

3. Scenario Where Interpolation Search Outperforms Binary Search

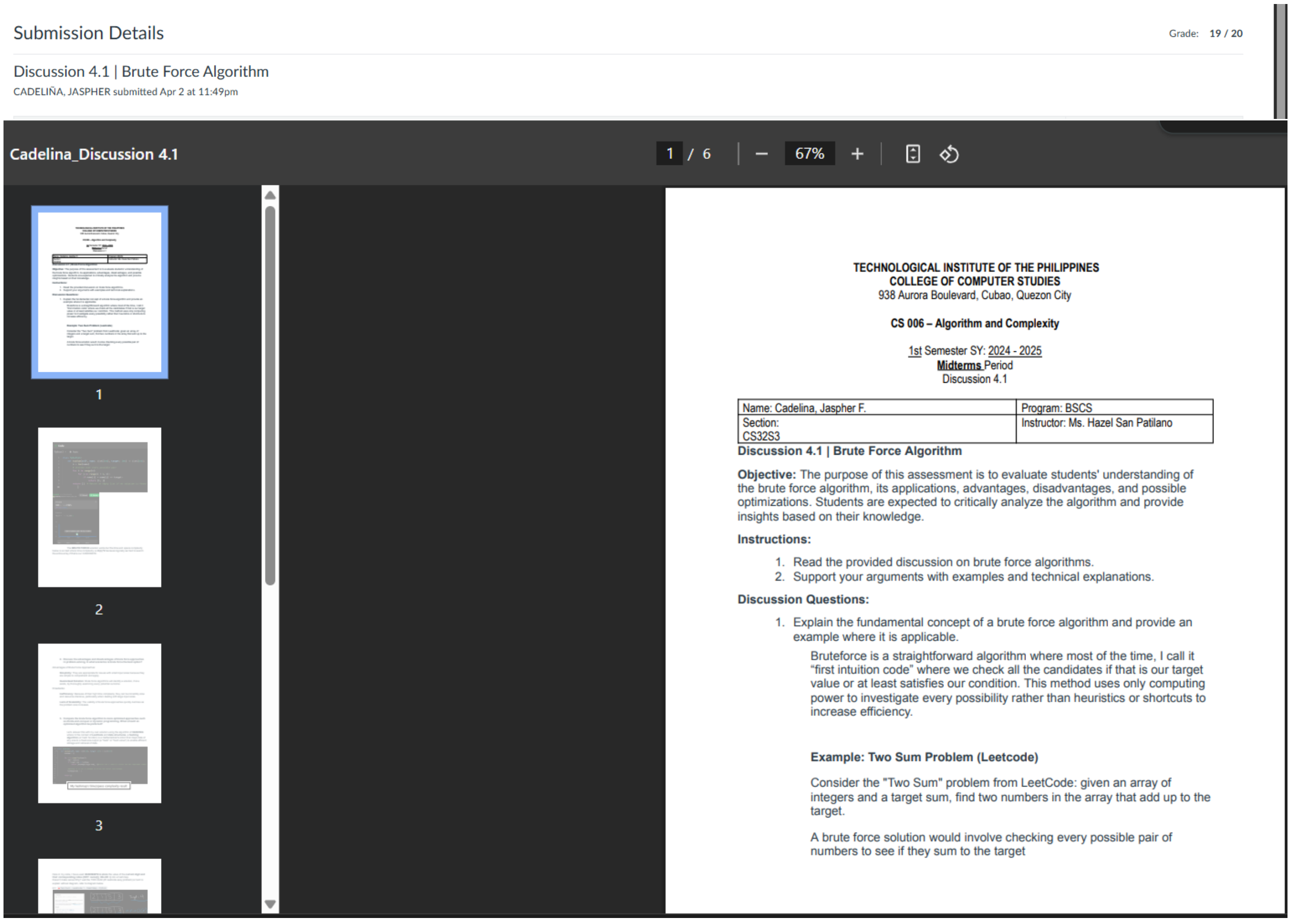
Imagine a sorted array of evenly spaced integers like **[1, 2, 3, ..., 1000]**. If searching for **500**, interpolation search would directly calculate the middle index (500th position) in one step , whereas binary search would take $\log_2 1000 \approx 10$ steps. This works because the data is perfectly uniform, allowing interpolation to predict the exact position

https://www.geeksforgeeks.org/g-fact-84/
https://stackoverflow.com/questions/74752770/choice-between-interpolation-search-and-binary-search

My personal take here after the discussion was:

- Jump search is a middle-ground between linear and binary search.
- Interpolation search shines with uniform data but risks inefficiency otherwise.
- Binary search is reliable for all sorted data but lacks interpolation's "best-case" speed.

Discussion #3 | Jump and Interpolation Search Algorithm
FIGURE 4.3



Submission Details

Grade: 19 / 20

Discussion 4.1 | Brute Force Algorithm

CADELIÑA, JASPHER submitted Apr 2 at 11:49pm

Cadelina_Discussion 4.1

1 / 6 67%

TECHNOLOGICAL INSTITUTE OF THE PHILIPPINES
COLLEGE OF COMPUTER STUDIES
938 Aurora Boulevard, Cubao, Quezon City

CS 006 – Algorithm and Complexity

1st Semester SY: 2024 - 2025
Midterms Period
Discussion 4.1

Name: Cadelina, Jaspher F.	Program: BSCS
Section: CS32S3	Instructor: Ms. Hazel San Patilano

Discussion 4.1 | Brute Force Algorithm

Objective: The purpose of this assessment is to evaluate students' understanding of the brute force algorithm, its applications, advantages, disadvantages, and possible optimizations. Students are expected to critically analyze the algorithm and provide insights based on their knowledge.

Instructions:

1. Read the provided discussion on brute force algorithms.
2. Support your arguments with examples and technical explanations.

Discussion Questions:

1. Explain the fundamental concept of a brute force algorithm and provide an example where it is applicable.

   Bruteforce is a straightforward algorithm where most of the time, I call it "first intuition code" where we check all the candidates if that is our target value or at least satisfies our condition. This method uses only computing power to investigate every possibility rather than heuristics or shortcuts to increase efficiency.

   Example: Two Sum Problem (Leetcode)

   Consider the "Two Sum" problem from LeetCode: given an array of integers and a target sum, find two numbers in the array that add up to the target.

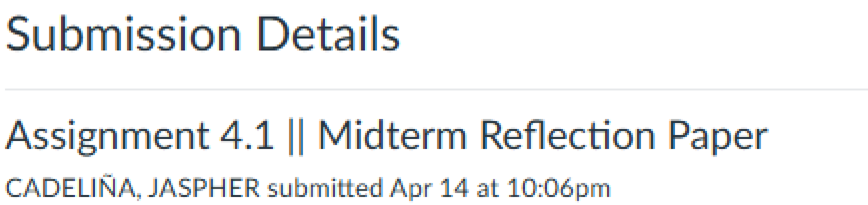
   A brute force solution would involve checking every possible pair of numbers to see if they sum to the target

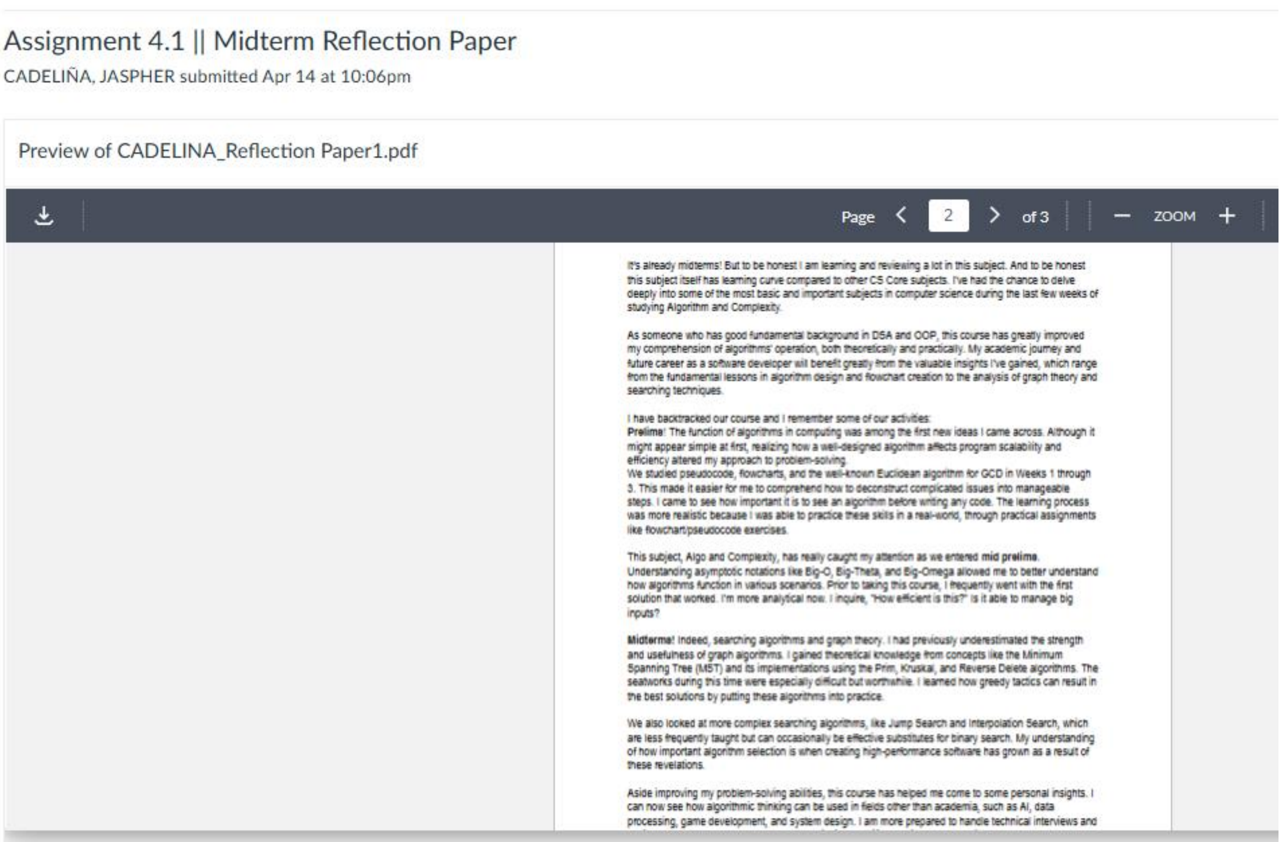
Discussion 4.1 | Brute Force Algorithm
Figure 4.4

FINAL PERIOD

TABLE OF CONTENTS

<screenshot all assignment / discussion and also put figure no and name of the screenshot here>





It's already midterms! But to be honest I am learning and reviewing a lot in this subject. And to be honest this subject itself has learning curve compared to other CS Core subjects. I've had the chance to delve deeply into some of the most basic and important subjects in computer science during the last few weeks of studying Algorithm and Complexity.

As someone who has good fundamental background in DSA and OOP, this course has greatly improved my comprehension of algorithms' operation, both theoretically and practically. My academic journey and future career as a software developer will benefit greatly from the valuable insights I've gained, which range from the fundamental lessons in algorithm design and flowchart creation to the analysis of graph theory and searching techniques.

I have backtracked our course and I remember some of our activities:
Prelims! The function of algorithms in computing was among the first new ideas I came across. Although it might appear simple at first, realizing how a well-designed algorithm affects program scalability and efficiency altered my approach to problem-solving.
We studied pseudocode, flowcharts, and the well-known Euclidean algorithm for GCD in Weeks 1 through 3. This made it easier for me to comprehend how to deconstruct complicated issues into manageable steps. I came to see how important it is to see an algorithm before writing any code. The learning process was more realistic because I was able to practice these skills in a real-world, through practical assignments like flowchart/pseudocode exercises.

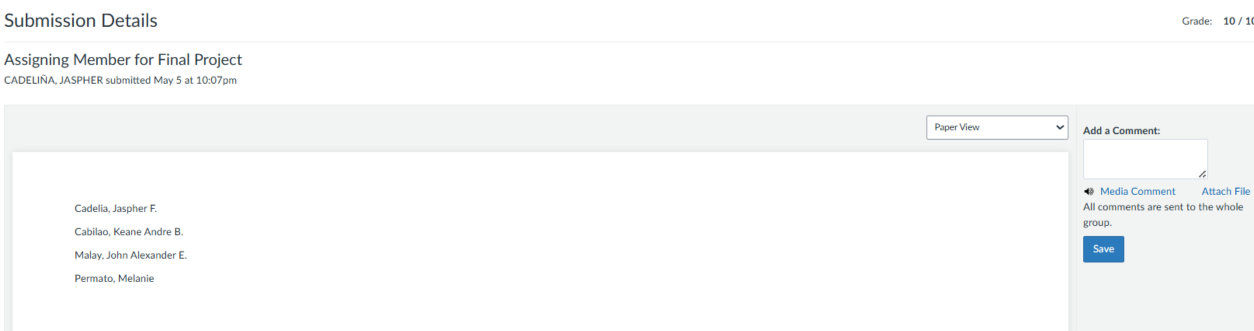
This subject, Algo and Complexity, has really caught my attention as we entered **mid prelims**. Understanding asymptotic notations like Big-O, Big-Theta, and Big-Omega allowed me to better understand how algorithms function in various scenarios. Prior to taking this course, I frequently went with the first solution that worked. I'm more analytical now. I inquire, "How efficient is this?" Is it able to manage big inputs?

Midterms! Indeed, searching algorithms and graph theory. I had previously underestimated the strength and usefulness of graph algorithms. I gained theoretical knowledge from concepts like the Minimum Spanning Tree (MST) and its implementations using the Prim, Kruskal, and Reverse Delete algorithms. The seatworks during this time were especially difficult but worthwhile. I learned how greedy tactics can result in the best solutions by putting these algorithms into practice.

We also looked at more complex searching algorithms, like Jump Search and Interpolation Search, which are less frequently taught but can occasionally be effective substitutes for binary search. My understanding of how important algorithm selection is when creating high-performance software has grown as a result of these revelations.

Aside improving my problem-solving abilities, this course has helped me come to some personal insights. I can now see how algorithmic thinking can be used in fields other than academia, such as AI, data processing, game development, and system design. I am more prepared to handle technical interviews and

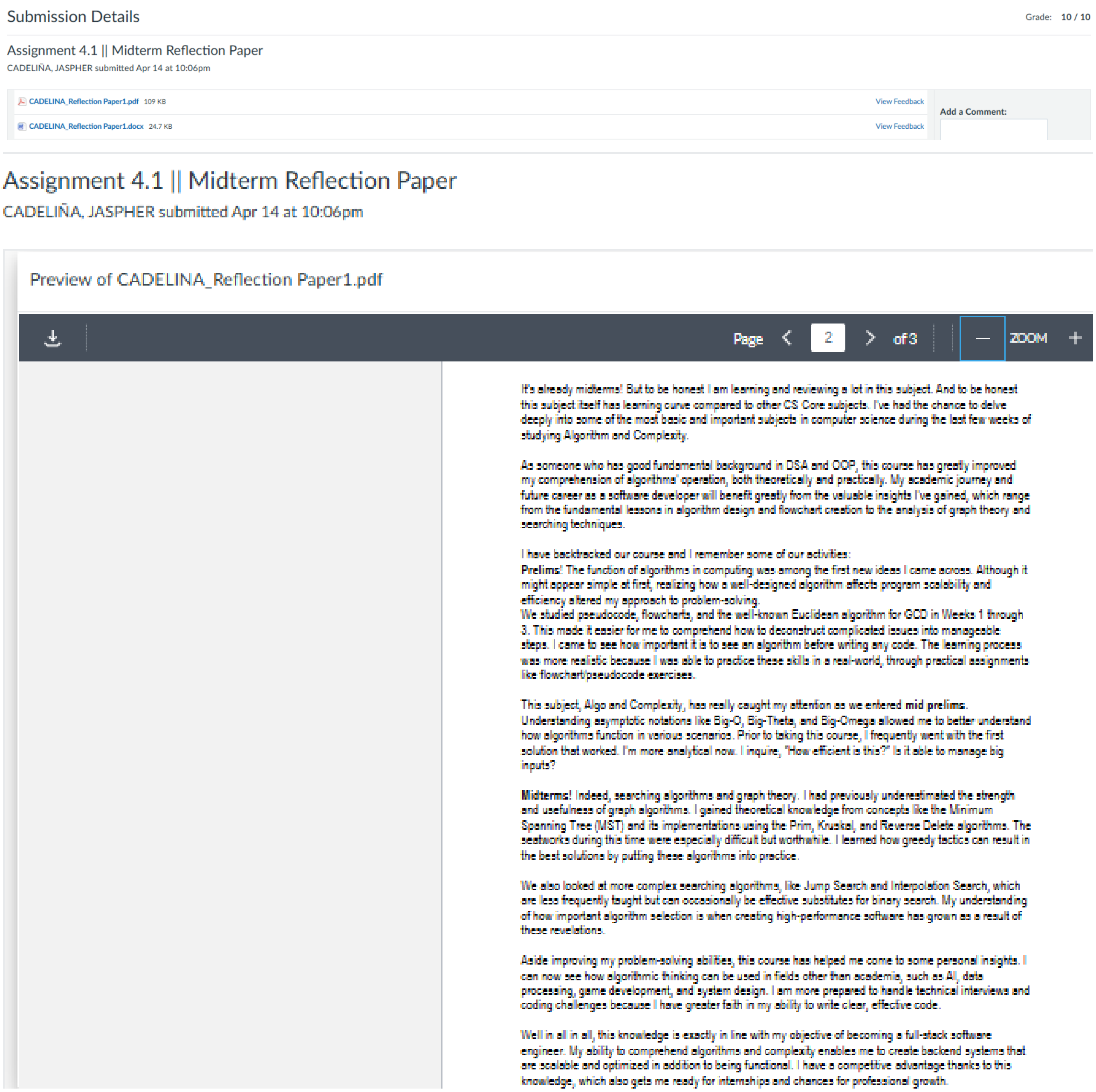
Assignment 4.1 || Midterm Reflection Paper
Figure 4.5



Assigning Member for Final Project
Figure 4.6

COURSE REFLECTION

TABLE OF CONTENTS



Submission Details

Grade: 10 / 10

Assignment 4.1 || Midterm Reflection Paper

CADELIÑA, JASPHER submitted Apr 14 at 10:06pm

CADELINA_Reflection Paper1.pdf 109 KB View Feedback

CADELINA_Reflection Paper1.docx 24.7 KB View Feedback

Add a Comment:

Assignment 4.1 || Midterm Reflection Paper

CADELIÑA, JASPHER submitted Apr 14 at 10:06pm

Preview of CADELINA_Reflection Paper1.pdf

Page 2 of 3 ZOOM

It's already midterms! But to be honest I am learning and reviewing a lot in this subject. And to be honest this subject itself has learning curve compared to other CS Core subjects. I've had the chance to delve deeply into some of the most basic and important subjects in computer science during the last few weeks of studying Algorithm and Complexity.

As someone who has good fundamental background in DSA and OOP, this course has greatly improved my comprehension of algorithms' operation, both theoretically and practically. My academic journey and future career as a software developer will benefit greatly from the valuable insights I've gained, which range from the fundamental lessons in algorithm design and flowchart creation to the analysis of graph theory and searching techniques.

I have backtracked our course and I remember some of our activities:
Prelims! The function of algorithms in computing was among the first new ideas I came across. Although it might appear simple at first, realizing how a well-designed algorithm affects program scalability and efficiency altered my approach to problem-solving.
We studied pseudocode, flowcharts, and the well-known Euclidean algorithm for GCD in Weeks 1 through 3. This made it easier for me to comprehend how to deconstruct complicated issues into manageable steps. I came to see how important it is to see an algorithm before writing any code. The learning process was more realistic because I was able to practice these skills in a real-world, through practical assignments like flowchart/pseudocode exercises.

This subject, Algo and Complexity, has really caught my attention as we entered **mid prelims.** Understanding asymptotic notations like Big-O, Big-Theta, and Big-Omega allowed me to better understand how algorithms function in various scenarios. Prior to taking this course, I frequently went with the first solution that worked. I'm more analytical now. I inquire, "How efficient is this?" Is it able to manage big inputs?

Midterms! Indeed, searching algorithms and graph theory. I had previously underestimated the strength and usefulness of graph algorithms. I gained theoretical knowledge from concepts like the Minimum Spanning Tree (MST) and its implementations using the Prim, Kruskal, and Reverse Delete algorithms. The seatworks during this time were especially difficult but worthwhile. I learned how greedy tactics can result in the best solutions by putting these algorithms into practice.

We also looked at more complex searching algorithms, like Jump Search and Interpolation Search, which are less frequently taught but can occasionally be effective substitutes for binary search. My understanding of how important algorithm selection is when creating high-performance software has grown as a result of these revelations.

Aside improving my problem-solving abilities, this course has helped me come to some personal insights. I can now see how algorithmic thinking can be used in fields other than academia, such as AI, data processing, game development, and system design. I am more prepared to handle technical interviews and coding challenges because I have greater faith in my ability to write clear, effective code.

Well in all in all, this knowledge is exactly in line with my objective of becoming a full-stack software engineer. My ability to comprehend algorithms and complexity enables me to create backend systems that are scalable and optimized in addition to being functional. I have a competitive advantage thanks to this knowledge, which also gets me ready for internships and chances for professional growth.

Assignment 4.1 || Midterm Reflection Paper
Figure 5.1

PROJECT

[TABLE OF CONTENTS](#)

PROJECT
Final Project Requirements

TABLE OF CONTENTS

Name	Owner	Last modified	File size
Group 4 - CS 006_Final Project Presentation.pp...	cadelinajasp...	May 14, 2025	4.8 MB
Group 4 - CS 006_Final Project Presentation.pdf	cadelinajasp...	May 14, 2025	1.1 MB
FinalProjectDoc_Cabilao_Cadelina_Malay_Perm...	cadelinajasp...	May 14, 2025	637 KB
FinalProjectDoc_Cabilao_Cadelina_Malay_Perm...	cadelinajasp...	May 14, 2025	938 KB
Cadelina_Cabilao_Permato_malay_CS006_Final...	cadelinajasp...	May 14, 2025	497.1 MB

CS 006 - CS32S3

Final Project

IN CS 006

"VISUALIZATION GRID GRAPHS FOR ENHANCING ALGORITHM UNDERSTANDING AND ANALYSIS"

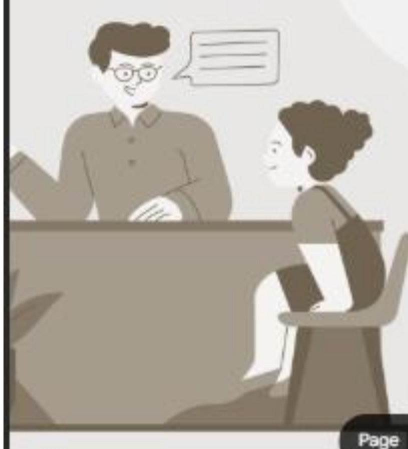
ALGORITHMS & COMPLEXITY

GROUP 4

Introduction

This project aims to address the need for a clear and intuitive understanding of fundamental pathfinding algorithms (Brute Force, Greedy, Backtracking) by visualizing their step-by-step execution on a grid.

- Brute Force: Tries every possible path.
- Greedy: Chooses the seemingly closest next step.
- Backtracking: Explores one path at a time, retracing if stuck.



Objectives

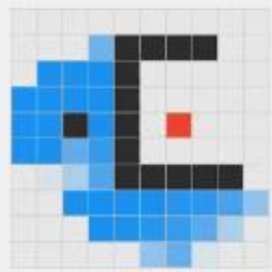
- To develop an interactive grid-based visualization tool for pathfinding algorithms.
- To demonstrate how different algorithms such as: Backtracking, Greedy, and Brute Force, work in finding a path from a start point to a goal.
- To help users, especially students and beginners, understand how these algorithms explore paths, make decisions, and handle obstacles on the grid.
- To provide a visual and easy-to-follow experience that supports learning.
- To compare the performance of each algorithm.
- To highlight their strengths, weaknesses, and best use cases in solving different pathfinding problems.

Analysis

Greedy Algorithm

KEY METRICS

- Effectiveness of the heuristic
- Path cost or length
- Speed



Greedy BFS
Time: O(V log V)
Space: O(V)
Time: 214.20 ms

HOW IT WORKS

It always picks the path that looks closest to the goal, using a rule called heuristic.

STRENGTH

Quick path finding

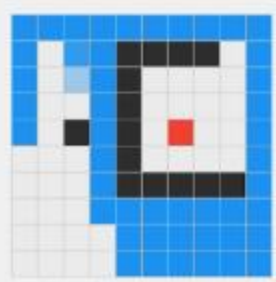
WEAKNESSES

Could fail to find the optimal route and might get stuck in tricky situations.

Backtracking Algorithm dfs

KEY METRICS

- Time complexity
- Space Complexity
- Correctness
- Scalability



DFS (Backtracking)
Time: O(V + E)
Space: O(N)
Time: 3651.80 ms

HOW IT WORKS

Explores one path at a time, backtracks

Systematically checks every possibility,

WEAKNESSES

Searching all options makes it slow for

Algorithm

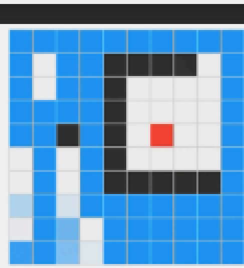
Algorithm

TWO ALGORITHM HAS THE SAME IMPLEMENTATION

But, the Backtracking has optimization to SKIP unncessesary works, and Brute Force algorith is just dfs WITHOUT shortest path tracker

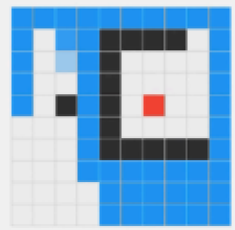
```
//BASE CASE IF meet the end
//"Finds the shortest path during traversal, avoiding unnecessary work."
if (shortestPath !== null && path.length >= shortestPath.length) {
  return;
}
```

Because you're not waiting until the end to compare paths (like brute force). You're constantly checking while exploring — and backing out early when it's clear the current path won't help.



Brute Force

Time: 6174.90 ms



DFS (Backtracking)

Time: 3651.80 ms

Concept	With Pruning (dfsGrid)	Without Pruning (bruteForceGrid)
Efficiency	Smart — avoids bad paths early	Dumb — explores everything blindly
Memory usage	Low — stores only 1 best path	High — stores every valid path
Speed	Faster in large grids	Slower, especially in large grids
That line does?	**Avoids unnecessary work**	No such optimization

Discussion

- **Brute Force: Slow but thorough, checks everything. Bad for big grids.**
- **Greedy: Fast, but might miss the best way.**
- **Backtracking: Finds all paths, but slow, no optimization.**

Conclusion: Algorithm choice depends on needing speed, accuracy, or guaranteed solutions. No single algorithm is best for all grid problem

Pathfinding Algorithm Visualizer
Color Legend
Pathfinding Algorithm Visualizer
Page 16 / 17

Figure 6.1

Figure 6.2

Quiz #1 | Final Project Documentation and PPT

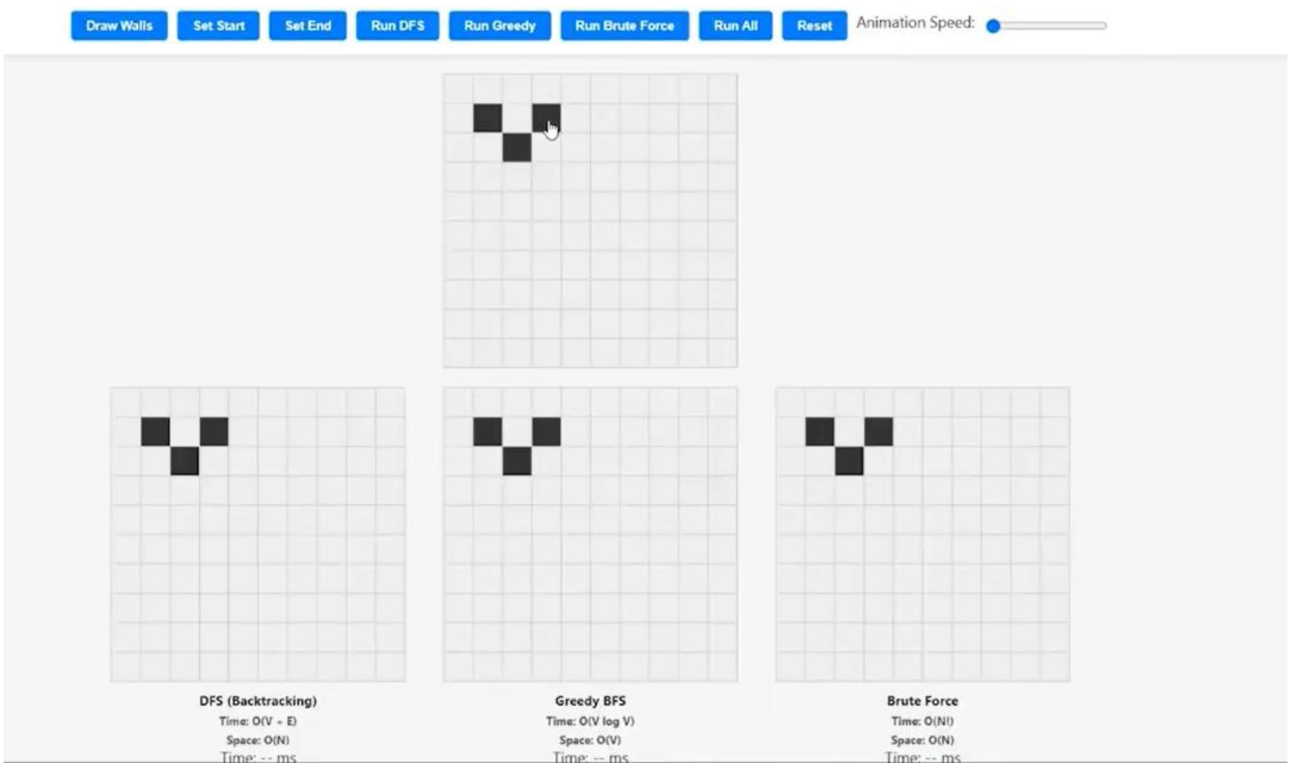


Figure 6.3 App demo #1

Figure 6.4 App Demo #2

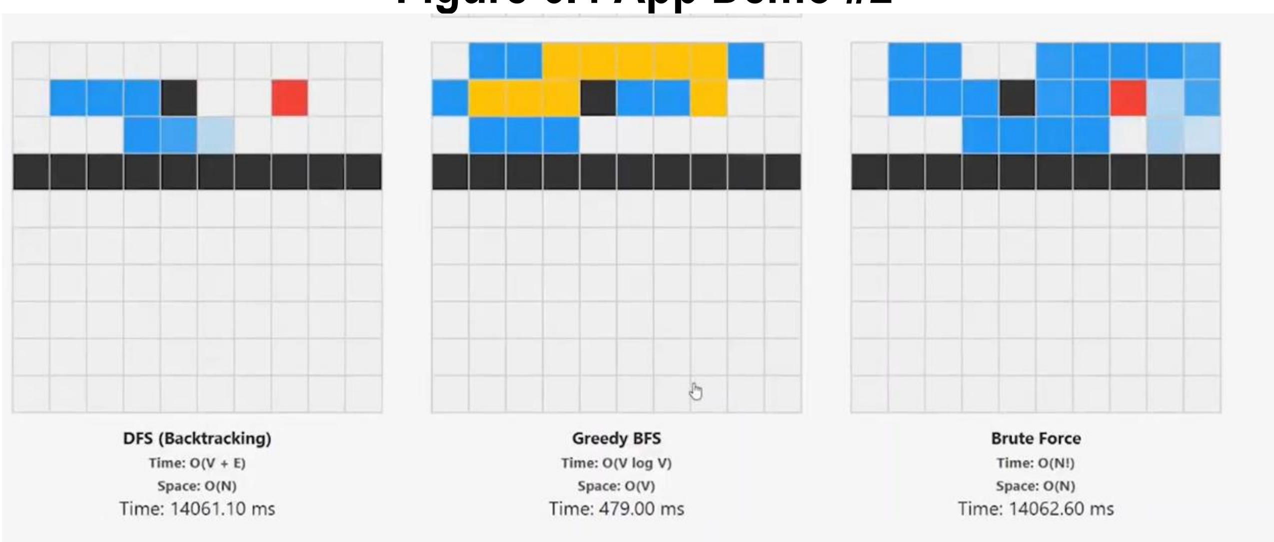


Figure 6.5 App Demo #3

Quiz #2 | Final Project Presentation and Application