

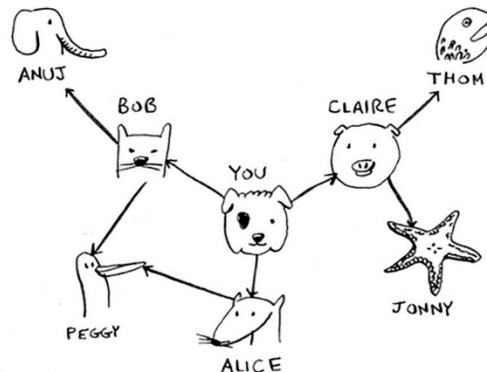
ADVANCED ALGORITHM

Course Syllabus

Course Title	Advanced Algorithm
Department	IDT Computer Sciences
Credit	3
In-class hours	45 hours (30 sessions)
Outside-class hours	6 hours per week (<i>recommended for assignments, projects, and self-study</i>)
Period	Year 2 –Term 1
Revision	Generation 10 (SEP 2024)
Author	R. OGOR

Instructors

GROUP	THEORY	PRACTICE
G1 - DS	VA HONGLY	VA HONGLY
G2 - DS		VA HONGLY
G1 - SE	BOU CHANNA	BOU CHANNA
G2 - SE		BOU CHANNA
G3 - SE	LAY VATHNA	KORAT NATT
G4 - SE		KORAT NATT



1. COURSE DESCRIPTION

Welcome to Advanced Algorithms course!

In computer science, an Abstract Data Type (ADT) formally defines a data type and the operations on that type of data, independently of any implementation. The objective of this course is to familiarize with a **rigorous programming approach** to **design abstract data types** and use them wisely to **solve real-world problems**.

Through a combination of **theoretical** knowledge and hands-on **practice**, this course will enable you to **specify**, **implement**, and **analyze** various abstract data types, while fostering a **collaborative** and **rigorous approach** to programming.

2. COURSE LEARNING OUTCOMES

By the end of the course, you should gain the following outcomes:

Knowledge	<i>Theoretical or factual information you need to understand.</i>
<ul style="list-style-type: none"> ✓ Understand C++ syntax and object-oriented approach. ✓ Understand the concepts of complexity through various real-world problems. ✓ Understand the functioning of different ADTs and associated problem types <ul style="list-style-type: none"> <i>Aggregated structures: Array, Classes</i> <i>Dynamic structures: Array, Linked Lists, Stack, Trees, Graphs</i> 	
Skills	<i>Practical abilities you need to acquire</i>
<ul style="list-style-type: none"> ✓ Be able to specify an abstract data type <ul style="list-style-type: none"> <i>Data structure and operations involved, pre/post conditions, complexity</i> ✓ Write test cases to validate the correctness of an algorithm ✓ Be able to conduct performance analysis of an algorithm <ul style="list-style-type: none"> <i>In terms of complexity, memory usage</i> ✓ Be able to justify the factors that influence the choice of an ADT for a given problem <ul style="list-style-type: none"> <i>Operations to be performed, data size, available space...</i> ✓ Be able to solve a real-world problem using well-chosen data structures 	
Attitudes	<i>Values, motivations, and dispositions you need to develop.</i>
<ul style="list-style-type: none"> ✓ Give and receive feedback <ul style="list-style-type: none"> - Provide constructive comments respectfully. - Receptive to suggestions for improvement from peers or instructors. ✓ Cultivate collaborative learning <ul style="list-style-type: none"> - Teamwork spirit to solve complex problems. - Engage in technical discussions to deepen understanding of concepts. ✓ Develop rigor and discipline in programming <ul style="list-style-type: none"> - Attention to detail when specifying and implementing ADTs and algorithms. - Perseverance in the face of technical challenges and programming errors. ✓ Critical and analytical approach <ul style="list-style-type: none"> - Ability to justify technical choices based on rigorous analysis. - Openness to reconsider and improve proposed solutions. 	

3. COURSE SESSIONS

This course is composed of 30 sessions, described as follows:

	PRE-ASSESSMENT	<p><i>(Before course start)</i></p> <p><i>Let's assess your skills on C language!</i></p> <ul style="list-style-type: none"> - Data Types Understand and manipulate integers, floats, strings, and Booleans - Loops Implement and control loops ('for', 'while') for repetitive tasks - Conditions Use conditional statements to make decisions in code - Arrays Manipulate array of data - Structs Manipulate structs to model complex entities - Variables Identify and use variables effectively to solve problems - Decomposition Break down complex problems into smaller tasks - Functions Modularize code using functions for reuse and organization
C1 – C++ BASICS		
S1	KICK OFF	<p>Course Kick off</p> <ul style="list-style-type: none"> • Overview of programs you will be able to code at the end of this course • Overview of course learning objectives & evaluation strategy • Rules about the usage of AI within this course + Plagiarism • What we expect in terms of work and attitudes • Overview of the tools to work efficiently <p>A quick start</p> <ul style="list-style-type: none"> • A short introduction to C++ • Challenge yourself with a few problems! <p>Feedbacks</p> <ul style="list-style-type: none"> • How this course can serve your future career? • What kind of involvement do you like to commit during this course?
	HOME	<p>Research</p> <ul style="list-style-type: none"> • How to use the IDE properly to work on C++
S2	PRACTICE	<p>Manipulate C++ syntax</p> <ul style="list-style-type: none"> • Execute and debug C++ programs • Transition from C-style printf and scanf to cout and cint • Manipulate if-else and switch-case conditions • Manipulate std::string and std::vector • Iterate through array elements or vector with a for loop • Iterate with a do-while loop to validate user inputs
S3	PRACTICE	<p>Allocate Memory Dynamically</p> <ul style="list-style-type: none"> • Visualize dynamic memory allocation in stack and heap • Manipulate pointers • Evaluate the memory state in stack and heap along the execution • Compute the required memory space for a data or a program
S4	LEARNING	<p>Classes and Objects</p> <ul style="list-style-type: none"> • Benefits of class vs structures • Class members (attributes & methods) • Class instantiation and constructors • Class members visibility





		<ul style="list-style-type: none"> Passing objects by value vs reference
	HOME	Research <ul style="list-style-type: none"> Dynamic instantiation & destructors
S5	PRACTICE	Classes and Objects <ul style="list-style-type: none"> Implement a class in C++ with specific attributes and methods Instantiate objects of the defined class and utilize methods to manipulate attributes. Examine the object instantiation in memory using debugger tools
C2 – ANALYSIS OF ALGORITHMS		
S1	LEARNING	Algorithm Complexity <ul style="list-style-type: none"> Calculation of the number of operations according to N Understand the Big-O notation <i>Ex : Constant / Linear / Logarithmic / Quasilinear / Polynomial</i>
	HOME	Research <ul style="list-style-type: none"> Research in team based on demonstrating a sorting algorithm principle
S2	LEARNING	Sorting Algorithms <ul style="list-style-type: none"> Understand 3 sorting approaches: <i>selection, insertion, bubble</i> Evaluate each algorithm complexity (time and space)
S3	PRACTICE	Algorithm Complexity <ul style="list-style-type: none"> Understand some algorithms based on numbers <i>Ex: Search problems, PGDD problems, Fibonacci problems</i> Evaluate each algorithm complexity (time and space) Identify possible improvements
S4	PRACTICE	Sorting Algorithms <ul style="list-style-type: none"> Use std vector data structure Implement algorithms to sort numbers <ul style="list-style-type: none"> Selection - Insertion - Bubble Analyse algorithm performances with large amount of data
S5	REVIEW	<i>Quiz - Research Presentation - Peer Review - Self Evaluation</i>
C3 – ADT		
S1	LEARNING	Abstract Data Type <ul style="list-style-type: none"> Make the difference between interface and implementation Understand invariant, pre and post conditions Design different ADT (<i>a rectangle, a coffee machine...</i>)
S2	LEARNING	Dynamic Table <ul style="list-style-type: none"> ADT internal data and operations behaviour ADT operations pre and post conditions ADT operations pseudo code ADT operations complexity
S3	LEARNING	Linked list <ul style="list-style-type: none"> ADT internal data and operations behaviour ADT operations pre and post conditions ADT operations pseudo code ADT operations complexity Single/Double linked list

		<ul style="list-style-type: none"> Circular linked list
S4	PRACTICE	Dynamic table <ul style="list-style-type: none"> Create the ADT classes (attributes, methods, constructor, destructor) Implementation the ADT methods Implement more advanced methods
S5	PRACTICE	Linked list <ul style="list-style-type: none"> Create the ADT classes (attributes, methods, constructor, destructor) Implementation the ADT methods Implement more advanced methods
S6	LEARNING	Stack / Queue <ul style="list-style-type: none"> ADT internal data and operations behaviour ADT operations pre and post conditions ADT operations pseudo code ADT operations complexity Handle recursive problems with a stack Evaluate an expression with a stack
S7	PRACTICE	Stack / Queue <ul style="list-style-type: none"> Create the ADT classes (attributes, methods, constructor, destructor) Implementation the ADT methods Implement more advanced methods Problem based on validity of parenthesis an expression Problem based on Polish notation
S8	REVIEW	<i>Quiz - Research Presentation - Peer Review - Self Evaluation</i>
S9	LEARNING	Tree <ul style="list-style-type: none"> ADT internal data and operations behaviour ADT operations pre and post conditions ADT operations pseudo code ADT operations complexity Different kind of tree (binary tree.)
S10	PRACTICE	Tree <ul style="list-style-type: none"> Create the ADT classes (attributes, methods, constructor, destructor) Implementation the ADT methods Implement more advanced methods <ul style="list-style-type: none"> insertElement searchElement getTreeHeight getRouteWidth
S11	PRACTICE	Tree, Stack, Queue <ul style="list-style-type: none"> Real word problems
S12	LEARNING	Graphs <ul style="list-style-type: none"> ADT internal data and operations behaviour ADT operations pre and post conditions ADT operations pseudo code ADT operations complexity

S13	PRACTICE	Graphs <ul style="list-style-type: none"> Create the ADT classes (attributes, methods, constructor, destructor) Implementation the ADT methods Implement more advanced methods
S14	PRACTICE	Graphs <ul style="list-style-type: none"> Real word problems
S15	REVIEW	<i>Quiz - Research Presentation - Peer Review - Self Evaluation</i>
S16	PRACTICE	All <ul style="list-style-type: none"> Real word problems
S17	PRACTICE	All <ul style="list-style-type: none"> Real word problems
C5 – LET S END		
S1	REVIEW	Course retrospective and revisions before exam
S2	EXAM	Final exam
S3	PROJECT	Project jury

4. COURSE BEHAVIOR

According to each type of session, here is what you need to do and how to behave:

 LEARNING	<p>Before class</p> <ul style="list-style-type: none"> ✓ Make sure to complete the assigned reading and exercises ✓ Take the individual quiz to prepare <p>During class</p> <ul style="list-style-type: none"> ✓ Participate actively in group quiz reviews ✓ Engage with the teacher's inquiry questions, ✓ collaborate with your peers on group exercises
 PRACTICE	<p>During class</p> <ul style="list-style-type: none"> ✓ Focus on working individually through a set of problem-based exercises to apply what you've learned. ✓ Use this time to practice and deepen your understanding
 REFLECT	<p>During class</p> <ul style="list-style-type: none"> ✓ Engage in peer reviews and debates, ✓ Contribute to class presentations and demonstrations. ✓ Be open to giving and receiving feedback <p>After class</p> <ul style="list-style-type: none"> ✓ Complete your individual evaluation ✓ Review the feedback provided by the peers and teacher to improve your understanding and performance
 PROJECT	<p>During class</p> <ul style="list-style-type: none"> ✓ Work individually or in group on your project. ✓ Stay focused and make use of the teacher's guidance and monitoring to enhance your work.

5. GRADING & ACADEMIC POLICY

TYPE OF EVALUATION	RATIO
Attendance and class participation	10 %
Quizzes	10 %
Peer review	10 %
Practice assessments	20 %
Project	20 %
Exam	30 %

Attendance & Class Participation

You must be present in all sessions of this course for the physical class and only absences authorized by the academic affair are allowed.

Participation is also scored: by participation we mean your involvement in your group activities, your action to support other students, your questions to the lecturer.

Peer reviews

In addition to each assignment, you must evaluate 3 of your peers, for a total of 15 evaluations at the end of the course.

You will learn in this course not only to give feedback, but also to receive them positively.

Project

You will build a personal project along the course to showcase your skills (see below the description) . We will evaluate not only the quality of your works, but also your out-of-the-box thinking and your creativity

Final Exam

Final exam will be performed without internet. Only 1 sheet of paper with your own note will be allowed

6. PROJECT DEFINITION

As part of this course, you will undertake a personal project aimed at applying the knowledge and skills you have acquired. Your project will involve the following steps:

1. Problem Selection

- ✓ You will choose a real-world problem to solve, such as finding *the shortest road*, *optimizing an agenda*, or *any other relevant issue*.

2. ADT Selection

- ✓ You will identify and justify the choice of one or more ADTs that are best suited to solving your chosen problem.

3. Program Development

- ✓ You will develop a C++ program that provides an optimized solution to the problem.
- ✓ You will explain your choices of implementations for the different functions and algorithms developed in relation to performance and robustness.

4. Performance Analysis

- ✓ You will analyze the performance of your solution, considering both time and space complexity.

5. Documentation and Presentation

- ✓ You will document your project, explaining your design choices, implementation, and performance analysis.
- ✓ You will present your findings to the jury.

7. REQUIRED SOFTWARE

You will need to install the following software/application:

CATEGORY	SOFTWARE
IDE	Visual Studio Code
C++ Compiler	MinGW
Version Controller	Git / GitHub
Communication Tool	Telegram
LMS	MS Teams

Important: Please read the **Course Tool Installation Guide** for details about installation process

8. RESOURCES

Course books

- ✓ Problem Solving in Data Structures & Algorithms Using C++ *Hemant_Jain_ 2016*
- ✓ Learning Algorithms Through Programming and Puzzle Solving
- ✓ [Grokking-Algorithms](#) (Aditya Bhargava - 2016)
- ✓ Cracking-the-Coding-Interview

Data Structure & Algorithms Resources

- ✓ https://www.tutorialspoint.com/data_structures_algorithms/index.htm

C++ Resources

- ✓ <https://cplusplus.com/doc/tutorial>
- ✓ <https://cplusplus.com/reference/>
- ✓ <https://google.github.io/styleguide/cppguide.html>

Tools

- ✓ Online C++ debugger and visualizer
<https://pythontutor.com/cpp.html#mode=edit>
- ✓ Online C++ IDE
<https://replit.com/>
<https://onecompiler.com/cpp>