**Data Structures and Algorithms (sp23)**

**Schedule**

* 2023-01-30: [Lecture 1](https://github.com/TT00FE39-3001/lecture1)
* 2023-01-30: [Lecture 2](https://github.com/TT00FE39-3001/lecture2)
* 2023-02-06: [Lecture 3](https://github.com/TT00FE39-3001/lecture3)
* 2023-02-13: [Lecture 4](https://github.com/TT00FE39-3001/lecture4)
* 2023-02-20: [Review -- optional Session](https://github.com/TT00FE39-3001/lecture-2023-02-20-review)
* 2023-02-27: [Lecture 5](https://github.com/TT00FE39-3001/lecture5)
* 2023-03-06: [Lecture 6](https://github.com/TT00FE39-3001/lecture6)
* 2023-03-13: [Lecture 7](https://github.com/TT00FE39-3001/lecture7)
* 2023-03-20: Review -- optional Session
* 2023-03-27: [Lecture 8](https://github.com/TT00FE39-3001/lecture8)
* 2023-04-03: Exam

**Content**

**Classification of Algorithm techniques**

* Brute Force
  + Linear search
  + Bubble sort
  + Selection sort
* Decrease and conquer
  + Binary Search
  + Insertion sort
* Divide-and-Conquer
  + pow()
  + Quick Sort
  + Merge Sort
* Dynamic Programming (DP)
  + Overlapping Sub-problems
  + Bottom Up (Tabulation) vs Top Down (Memoization)
  + 0/1 Knapsack & the Staircase Problems
* Greedy Technique
* Transform-and-Conquer (**If we have time**)

**Data Structures & ADT**

* Arrays, Linked Lists
* Queues, Stacks
* Hash Tables
* Binary Tree, Binary Search Tree
* Heaps
* Graphs

**Analysis of Algorithm Efficiency**

* Big O Complexity
* Average case vs worst case
* Space vs Time
* Little-O
* Theta
* Little Omega, Big Omega

**Standard Template Library (STL)**

* Containers
* Algorithms
* Iterators

**Misc**

* FIFO vs LIFO
* Recursion vs Iteration
  + The Top-Down Thought Process
* Logarithms vs Exponential

**Description**

In today's digital age, the field of computer science has become increasingly important in a wide range of industries. As such, it is becoming more and more essential for students from diverse fields to have a basic understanding of algorithms and data structures, even if they are not majoring in computer science.

This course is designed to introduce major/non-major students to the fundamental concepts of algorithms and data structures. Students will learn how to analyze problems and design algorithms. They will also learn about basic data structures such as arrays, linked lists, stacks, queues, and trees, as well as more advanced topics such as graph algorithms, sorting, and searching.

Throughout the course, students will be encouraged to think critically and apply their knowledge to real-world problems. They will also have hands-on experience on how algorithms and data structures concepts are implemented in the C++ programming language.

**Grading & Learning Outcomes**

**Grade 1 and 2**

By the end of this course, students will be able to:

* Understand different algorithms and data structure from high level.
* Understand the role of algorithms and data structures in a variety of fields.
* Work collaboratively with peers to solve problems.

**Grade H**

By the end of this course, students will be able to:

* Understand concepts in algorithms and data structures, including analyzing problems.
* Understand and explain C++ code for different algorithms and data structures.
* Understand the role of algorithms and data structures in a variety of fields and how they can be used to solve complex problems in different domains.
* Work collaboratively with peers to solve problems.

**Grade 4 and 5**

By the end of this course, students will be able to:

* Understand and apply fundamental concepts in algorithms and data structures, including analyzing problems, designing algorithms, and implementing data structures.
* Evaluate the efficiency of algorithms and data structures and select appropriate techniques for solving problems.
* Demonstrate proficiency in implementing algorithms and data structures using C++.
* Apply their knowledge of algorithms and data structures to real-world problems and effectively communicate solutions to others.
* Analyze and design basic data structures such as arrays, linked lists, stacks, queues, and trees, as well as more advanced topics such as graph algorithms, sorting, and searching.
* Understand the role of algorithms and data structures in a variety of fields and how they can be used to solve complex problems in different domains.
* Work collaboratively with peers to solve problems and complete programming assignments.

Overall, by achieving these learning outcomes, students will have gained a solid foundation in algorithms and data structures that will enable them to succeed in a variety of fields, even if they do not intend to major in computer science.

**# Data Structures and Algorithms (sp23)**

**## Schedule**

- 2023-01-30: [Lecture 1](https://github.com/TT00FE39-3001/lecture1)

- 2023-01-30: [Lecture 2](https://github.com/TT00FE39-3001/lecture2)

- 2023-02-06: [Lecture 3](https://github.com/TT00FE39-3001/lecture3)

- 2023-02-13: [Lecture 4](https://github.com/TT00FE39-3001/lecture4)

- 2023-02-20: [Review -- optional Session](https://github.com/TT00FE39-3001/lecture-2023-02-20-review)

- 2023-02-27: [Lecture 5](https://github.com/TT00FE39-3001/lecture5)

- 2023-03-06: [Lecture 6](https://github.com/TT00FE39-3001/lecture6)

- 2023-03-13: [Lecture 7](https://github.com/TT00FE39-3001/lecture7)

- 2023-03-20: Review -- optional Session

- 2023-03-27: [Lecture 8](https://github.com/TT00FE39-3001/lecture8)

- 2023-04-03: Exam

**## Content**

**### Classification of Algorithm techniques**

- Brute Force

  - Linear search

  - Bubble sort

  - Selection sort

- Decrease and conquer

  - Binary Search

  - Insertion sort

- Divide-and-Conquer

  - `pow()`

  - Quick Sort

  - Merge Sort

- Dynamic Programming (DP)

  - Overlapping Sub-problems

  - Bottom Up (Tabulation) vs Top Down (Memoization)

  - 0/1 Knapsack & the Staircase Problems

- Greedy Technique

- Transform-and-Conquer (**\*\*If we have time\*\***)

**### Data Structures & ADT**

- Arrays, Linked Lists

- Queues, Stacks

- Hash Tables

- Binary Tree, Binary Search Tree

- Heaps

- Graphs

**### Analysis of Algorithm Efficiency**

- Big O Complexity

- Average case vs worst case

- Space vs Time

- Little-O

- Theta

- Little Omega, Big Omega

**### Standard Template Library (STL)**

- Containers

- Algorithms

- Iterators

**### Misc**

- FIFO vs LIFO

- Recursion vs Iteration

  - The Top-Down Thought Process

- Logarithms vs Exponential

**## Description**

<!--

This course is designed for non-major students who are interested in learning the fundamentals of algorithms and data structures. You will gain an understanding of the basic concepts of algorithms and data structures, including sorting, searching, and data organization. You will learn the different types of algorithms and data structures, their uses, and how to implement them in a variety of contexts. Topics covered in this course include the analysis of algorithms, data structures and strategies for problem-solving. Through a combination of lectures and programming exercises, you will gain a solid understanding of algorithms and data structures and how to use them to solve real-world problems.

-->

In today's digital age, the field of computer science has become increasingly important in a wide range of industries. As such, it is becoming more and more essential for students from diverse fields to have a basic understanding of algorithms and data structures, even if they are not majoring in computer science.

This course is designed to introduce major/non-major students to the fundamental concepts of algorithms and data structures. Students will learn how to analyze problems and design algorithms. They will also learn about basic data structures such as arrays, linked lists, stacks, queues, and trees, as well as more advanced topics such as graph algorithms, sorting, and searching.

Throughout the course, students will be encouraged to think critically and apply their knowledge to real-world problems. They will also have hands-on experience on how algorithms and data structures concepts are implemented in the C++ programming language.

<!--

By the end of the course, students shall have a solid understanding of the key concepts in algorithms and data structures, and will be able to use this knowledge to tackle complex problems in a variety of fields.

-->

**## Grading & Learning Outcomes**

**### Grade 1 and 2**

By the end of this course, students will be able to:

- Understand different algorithms and data structure from high level.

- Understand the role of algorithms and data structures in a variety of fields.

- Work collaboratively with peers to solve problems.

**### Grade H**

By the end of this course, students will be able to:

- Understand concepts in algorithms and data structures, including analyzing problems.

- Understand and explain C++ code for different algorithms and data structures.

- Understand the role of algorithms and data structures in a variety of fields and how they can be used to solve complex problems in different domains.

- Work collaboratively with peers to solve problems.

**### Grade 4 and 5**

By the end of this course, students will be able to:

- Understand and apply fundamental concepts in algorithms and data structures, including analyzing problems, designing algorithms, and implementing data structures.

- Evaluate the efficiency of algorithms and data structures and select appropriate techniques for solving problems.

- Demonstrate proficiency in implementing algorithms and data structures using C++.

- Apply their knowledge of algorithms and data structures to real-world problems and effectively communicate solutions to others.

- Analyze and design basic data structures such as arrays, linked lists, stacks, queues, and trees, as well as more advanced topics such as graph algorithms, sorting, and searching.

- Understand the role of algorithms and data structures in a variety of fields and how they can be used to solve complex problems in different domains.

- Work collaboratively with peers to solve problems and complete programming assignments.

Overall, by achieving these learning outcomes, students will have gained a solid foundation in algorithms and data structures that will enable them to succeed in a variety of fields, even if they do not intend to major in computer science.

<!--

## Course Mechanics

-->