Recep Tayyip Erdogan University

Faculty of Engineering and Architecture

Computer Engineering

CE205- Data Structures

Syllabus

Fall Semester, 2022-2023

Instructor	**Asst. Prof. Dr. Uğur CORUH**
Contact Information	ugur.coruh@erdogan.edu.tr
Office No	**F-301**
Google Classroom Code	**d5yg4hi**
Lecture Hours and Days	**Tuesday 15:00/15:45 - 16:00-16:45 (2 hours)) (Theory) / Friday (09:00-09:45) (Theory) 10:00/10:45-11:00/11:45 (Lab)**
Lecture Classroom	**İBBF 402 Level-4**
Office hours	Meetings will be scheduled over Google Meet with your university account and email and performed via demand emails. Please send emails with the subject starts with [CE205] tag for the fast response and write formal, clear, and short emails.
Lecture and Communication Language	English
Theory/Laboratory Course Hour Per Week	3/2 hours
Credit	4
Prerequisite	CE103- Algorithms and Programming I CE100- Algorithms and Programming II
Corequisite	TBD
Requirement	TBD

**TBD: To Be Defined.*

A. Course Description

This course covers the fundamentals of data structure and file organization. The course scope explains how to use digital data mapping in programming to use data in application run-time memory or long-term file storage. The course discusses various implementations of these data objects, as well as programming styles and run-time representations. The course also looks at sorting, searching, and graph algorithms. The goal of this course is to provide digital data structures for real-world problems, as well as how data is shaped and mapped to memory or storage solutions. The class will be based on sharing expertise and guiding students in the discovery of learning methods and practice for data structure topics. By making programming applications and projects in the courses, the learning process will be strengthened by practicing rather than theory.

B. Course Learning Outcomes

After completing this course satisfactorily, a student will be able to:

- Describe how common linear and non-linear data structures such as arrays, matrices, linked structures, queues, stacks, trees and graphs are represented in run-time and storage memory and used by algorithms.
- Compare and contrast the benefits of dynamic and static data structures implementations.
- Understand basic industrial data structure definitions such as ASN.1 / BER TLV / PER TLV.
- Describe how run-time application data stored in a file and organized.
- Interpret a problem and define data structures for solution by using a C/C++, Java or C# application solve that problem in data structure manner.
- Compare alternative implementations of data structures with respect to performance and analysis space and time complexity.
- Understand data structure based sorting and searching algorithms.
- Describe hashing and indexing methods for file organization and processing.
- Discuss the computational efficiency of the principal algorithms for sorting, searching, and hashing in memory and file storage.
- Combine programming skills with data structures know-how and generate efficient solutions for real-life problems.

C. Course Topics

- Data-in-use, Data-in-transit and Data-at-rest concepts.
- Data Structures Space and Time Complexity Analysis
- Data and Variable Mappings
- ASN.1 / BER TLV / PER TLV
- Linked Lists (Single, Circular, Double, XOR)
- Skip List
- Strand Sort
- Arrays (Rotations, Arrangement, Rearrangement, Searching and Sorting)
- Matrices and Spare Matrices
- Stacks (Array and Linked List) and FILO (First in Last Out)
- Expressions (Infix, Postfix and Prefix) and Infix to Postfix Conversions and Postfix Evaluation

- Queues (Standard, Circular and Double Ended) (Array and Linked List) (FIFO-First-in-First-Out or FCFS-Fist Come First Serve)
- Multievel Queues (MLQ)
- Hanoi Tower
- Tree Structures and Binary Tree and Traversals (In-Order, Pre-Order, Post-Order)
- Heaps (Max, Min, Binary , Binomial, Fibonacci, Leftist, K-ary) and Priority Queue
- Heap Sort
- Huffman Coding
- Graph Representations (Adjency Matrix, Incidence Matrix, Adjency List) and Basics
- Graph Traversals (Depth-First Search (DFS), Iterative Deepening Search(IDS) or Iterative Deepening Depth First Search(IDDFS), Breadth-First Search (BFS), Depth-limited Search, Uniform Cost Search, Bidirectional Search)
- Water Jug Problem
- Graph Topological Sorting
- Graph Minimum Spanning Tree (MST)
- Graph Backtracking (Tug of War, n-Queen's, m Coloring, Euler& Hamiltonian Path)
- Graph Shortest Paths
- Graph Connectivity, Max Flow, Isomorphism, Canonization and Cuts (Max /Min)
- Alpha-Beta Prunning
- Hasse Diagrams
- Petri Nets
- Bipartite Graphs
- Graph Cycle Detection (Brent's, Hare and Tortoise Algorithms)
- Bayesian Network
- Linear, Binary, Interpolation and Fibonacci Search
- Hashing and Hash Tables (Direct-Adress Tables, Hash Tables, Hash Functions, Open Adressing, Perfect Hashing)
- Common Sorting Algorithms (Insertion, Selection, Radix, Quick, Heap, Permutation, Gnome, Comb, Flash, Stooge, Bees, Lucky, Indirect (Pointer), External (Segmented), Shaker/Bidirectional Bubble, Shell Sort)
- Comparison of Sorting Methods
- Common Tree Data Structures and Operations (Binary Search Tree, AVL Tree, B Tree and Derivations (2 3 4 Trees, 2 3 Trees, B+ Trees, B# Trees), R Tree, Red-Black Tree, Splay Tree, Van Emde Boas Tree, Binomial Tree, Minimax Tree)
- Comparison of Search Trees
- Augmenting Data Structures
- String LCS Problem (Hunt Macllory, Levenstein, Wagner-Fischer)
- String Alignment (Needleman Wunsch, Smith Waterman, Hunt Macllory), Tokenizer and Comparison
- String Search (Reverse Factor) Algorithms (Knuth-Morris-Pratt, Horspool, Boyer-Moore, Brute-Force, DFA Text Search)
- Tries and Patricia Tree (Radix Tree)
- Data Structure for Disjoint Sets
- Sequential File Organization (Binary Search, Interpolation Search, Self-Organizing Sequential Search)
- Direct File Organization Locating Information
- Direct File Organization Hashing Functions (MD5, HAVAL, SHA1, Key Mod N, Key Mod P, Truncation, Folding, Squaring, Radix Conversion, Polynomial Hashing, Alphabetic Keys, Collisions)
- Direct File Organization Collision Resolution
- Direct File Organization Coalesced Hashing (EISCH, LISCH, BEISCH, BLISCH, REISCH, RLISCH, EICH, LICH)
- Direct File Organization Progressive Overflow (Linear Probing, Quadratic Probing)

- Direct File Organization Double Hashing, Use of Buckets, Linear Quotient, Brent's Method, Binary Tree and Computed Chaining Insertion (CCI)
- Perfect Hashing and SimHash for Direct File Organization
- Comparison of Collision Resolution Methods
- Indexed Sequential File Organization
- Secondary Key Retrivals and Bits and Hashing for Classification and Checking
- Binary Tree Structures for Files (Binary Search, AVL Trees, Internal Path Reduction Trees)
- B-Trees and Derivates for Files (B Tree, B+Tree, B# Tree)
- Hashing Techniques for Expandable Files (Extendible, Dynamic and Linear Hashing)
- Tries, Approximate String Matching, Trie Hashing, Patricia Tree and Digital Search Tree for File Organization
- Secondary Key Retrivial (K-d Trees and Grid Files)
- File Sorting (Insertion, Quick, Heap Sorts, External Sorting, Sorting By Merging and Disk Sort)

D. Textbooks and Required Hardware

This course does not require a coursebook. If necessary, you can use the following books and open-source online resources.

- *C How to Program, 7/E. Deitel & Deitel. 2013, Prentice-Hall.*
- *Intro to Java Programming, Comprehensive Version (10th Edition) 10th Edition by Y. Daniel Liang*
- *Introduction to Algorithms, Third Edition By Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein*
- *Problem Solving and Program Design in C, J.R. Hanly, and E.B. Koffman, 6th Edition.*
- *Alan L. Tharp. 1988. File organization and processing. John Wiley & Sons, Inc., USA.*
- *Richard Jankowski. 2010. Advanced data structures by Peter Brass Cambridge University Press 2008. SIGACT News 41, 1 (March 2010), 19–20. DOI:https://doi.org/10.1145/1753171.1753176*
- *Robert Sedgewick and Kevin Wayne. 2011. Algorithms (4th. ed.). Addison-Wesley Professional.*
- *Additional Books TBD*

During this course, you should have a laptop for programming practices. You will have your development environment, and you will use this for examination and assignments also classroom practices.

E. Grading System

Midterm and Final grades will be calculated with the weighted average of the project or homework-based examinations. Midterm grades will be calculated between term beginning to the midterm week, and Final grades will be calculated between Midterm and Final week homeworks or projects as follow.

$$a_n = Homework\ or\ Project\ Weight$$

$$HW_n = Homework\ or\ Project\ Points$$

$$n = Number\ of\ Homework\ or\ Project$$

$$Grade = \frac{(a_1 HW_1 + a_2 HW_2 + \cdots + a_n HW_n)}{n}$$

Homework/Exam	Weight
$Midterm$	$\%40$
$Final$	$\%60$

$$Passing\ Grade = \frac{40 * Midterm_{Grade} + 60 * Final_{Grade}}{100}$$

F. Instructional Strategies and Methods

The basic teaching method of this course will be planned to be face-to-face in the classroom, and support resources, homeworks, and announcements will be shared over google classroom. Students are expected to be in the university. This responsibility is very important to complete this course with success. If pandemic situation changes and distance education is required during this course, this course will be done using synchronous and asynchronous distance education methods. In this scenario, students are expected to be in the online platform, zoom, or meet at the time specified in the course schedule. Attendance will be taken.

G. Late Homework

Throughout the semester, assignments must be submitted as specified by the announced deadline. Overdue assignments will not be accepted. Unexpected situations must be reported to the instructor for late homeworks by students.

H. Course Platform and Communication

Google Classroom will be used as a course learning management system. All electronic resources and announcements about the course will be shared on this platform. It is very important to check the course page daily, access the necessary resources and announcements, and communicate with the instructor as you needed to complete the course with success.

I. Academic Integrity, Plagiarism & Cheating

Academic Integrity is one of the most important principles of RTEÜ University. Anyone who breaches the principles of academic honesty is severely punished.

It is natural to interact with classmates and others to "study together". It may also be the case where a student asks to help from someone else, paid or unpaid, better understand a difficult topic or a whole course. However, what is the borderline between "studying together" or "taking private lessons" and "academic dishonesty"? When is it plagiarism, when is it cheating?

It is obvious that looking at another student's paper or any source other than what is allowed during the exam is cheating and will be punished. However, it is known that many students come

to university with very little experience concerning what is acceptable and what counts as "copying", especially for assignments.

The following are attempted as guidelines for the Faculty of Engineering and Architecture students to highlight the philosophy of academic honesty for assignments for which the student will be graded. Should a situation arise which is not described below, the student is advised to ask the instructor or assistant of the course whether what they intend to do would remain within the framework of academic honesty or not.

a. What is acceptable when preparing an assignment?

- Communicating with classmates about the assignment to understand it better
- Putting ideas, quotes, paragraphs, small pieces of code (snippets) that you find online or elsewhere into your assignment, provided that
    - these are not themselves the whole solution to the assignment,
    - you cite the origins of these
- Asking sources for help in guiding you for the English language content of your assignment.
- Sharing small pieces of your assignment in the classroom to create a class discussion on some controversial topics.
- Turning to the web or elsewhere for instructions, for references, and solutions to technical difficulties, but not for direct answers to the assignment
- Discussing solutions to assignments with others using diagrams or summarized statements but not actual text or code.
- Working with (and even paying) a tutor to help you with the course, provided the tutor does not do your assignment for you.

b. What is not acceptable?

- Asking a classmate to see their solution to a problem before submitting your own.
- Failing to cite the origins of any text (or code for programming courses) that you discover outside of the course's lessons and integrate into your work
    - Giving or showing a classmate your solution to a problem when the classmate is struggling to solve it.

J. Expectations

You are expected to attend classes on time by completing weekly course requirements (readings and assignments) during the semester. The main communication channel between the instructor and the students will be emailed. Please send your questions to the instructor's email address about the course via the email address provided to you by the university. ***Ensure that you include the course name in the subject field of your message and your name in the text field***. In addition, the instructor will contact you via email if necessary. For this reason, it is very important to check your email address every day for healthy communication.

K. Lecture Content and Syllabus Updates

If deemed necessary, changes in the lecture content or course schedule can be made. If any changes are made in the scope of this document, the instructor will inform you about this.

Course Schedule Overview

Weeks	Dates	Subjects	Other Tasks
Week 1	20.09.2022 23.09.2022	Course Plan and Communication, Introduction to Linear & Non-Linear Data Structure and Performance Analysis Implementing Pointer and Objects for Data and Variables Basic of ASN.1 / BER TLV / PER TLV	TBD
Week 2	27.09.2022 30.09.2022	Linked Lists and Related Algorithms Arrays and Matrices	TBD
Week 3	04.10.2022 07.10.2022	Stacks, Queue Structures and Related Algorithms and Problems.	TBD
Week 4	11.10.2022 14.10.2022	Tree Data Structure Types and Applications (Binary Tree, Tree Traversals, Heaps)	TBD
Week 5	18.10.2022 21.10.2022	Graph Data Structure and Traversals	TBD
Week 6	25.10.2022 28.10.2022	Graph MST, Backtracking, Topological Sorting, Shortest Paths, Connectivity,Max Flow and Cycle Detection Algorithms. Graph Isomorphism and canonization Graph Cuts	TBD
Week 7	01.11.2022 04.11.2022	Linear, Binary and Fibonacci Search Hashing and Hash Tables with Perpect Hashing	TBD
Week 8	**08.11.2022 11.11.2022**	**Midterm**	TBD
Week 9	15.11.2022 18.11.2022	Sorting Algorithms, Taxonomy and Comparisons	TBD
Week 10	22.11.2022 25.11.2022	Advaced Tree Data Structures (Binary Search Tree, AVL Tree, B Trees and derivations,Red-Black trees, Splay Trees and Augmented Data Structures, van Emde Boas Trees, Binomial and Minimax Trees) and Comparisons.	TBD
Week 11	29.11.2022 02.12.2022	String Data Structure, Subsequence Search, Alignment and Comparison Algorithms.	TBD
Week 12	06.12.2022 09.12.2022	String Search Algorithms, Tries, Data Structures for Disjoint Sets.	TBD
Week 13	13.12.2022 16.12.2022	Introduction to File Organization and Processing Sequential File Organization,Direct File Organization Hash Methods	TBD

Week 14	20.12.2022 23.12.2022	Direct File Organization Indexes Binary and B Tree Structures for File.	TBD
Week 15	27.12.2022 30.12.2022	Hashing Techniques for Expandable Files,Tries Approximate String Matching Trie Hashing Seconday Key Retrieval (2) File Sorting	TBD
Week 16	**03.01.2023** **06.01.2023**	**Final**	TBD

Course Schedule Details

Week-1

1. Introduction to Data Structure
    a. Data-in-use
    b. Data-in-transit
    c. Data-at-rest
2. Performance Analysis
3. Space Complexity
4. Time Complexity
5. Data and Variables
6. Implementing Pointer and Objects
7. Linear & Non-Linear Data Structures
8. ASN.1 / BER TLV / PER TLV

Week-2

1. Single Linked List
2. Circular Linked List
3. Double Linked List
4. XOR Linked List
5. Skip List
6. Strand Sort
7. Arrays
    a. Array Rotations
    b. Arrangement Rearrangement
    c. Searching and Sorting
    d. Optimization Problems
8. Matrix
9. Sparse Matrix

Week-3

1. Stack ADT
2. Stack Using Array
3. Stack Using Linked List
4. Expressions
    a. Infix
    b. Postfix
    c. Prefix
5. Infix to Postfix Conversion
6. Postfix Expression Evaluation
7. Queue ADT
    a. First Come First Serve, FCFS, FIFO
8. Queue Datastructure Using Array
9. Queue Using Linked List
10. Circular Queue Datastructure
11. Double Ended Queue Datastructure
12. Hanoi Tower

13. Multilevel Queue (MLQ)

Week-4

1. Tree – Terminology
2. Tree Representations
3. Binary Tree Datastructure
    a. Construction and Conversion
    b. Checking and Printing
    c. Summation
    d. Longest Common Ancestor
4. Binary Tree Representations
5. Binary Tree Traversals
    a. In-Order
    b. Pre-Order
    c. Post-Order
6. Threaded Binary Trees
7. Max Priority Queue
8. Heap Data Structure
    a. Max-Heap
    b. Min-Heap
    c. Binary Heap
    d. Binomial Heap
    e. Fibonacci Heap
        i. Structure of Fibonacci Heaps
        ii. Mergeable-heap operations
        iii. Decreasing a key and deleting a node
        iv. Bounding the maximum degree
    f. Leftist Heap
    g. K-ary Heap
    h. Heap Sort
    i. Huffman Coding

Week-5

1. Introduction to Graphs
    a. Vertex
    b. Edge
    c. Undirected Graph
    d. Directed Graph
    e. Mixed Graph
    f. End Vertices or Endpoints
    g. Origin
    h. Destination
    i. Adjacent
    j. Incident
    k. Outgoing Edge
    l. Incoming Edge
    m. Degree
    n. Indegree
    o. Outdegree

    p. Parallel edges or Multiple edges
    q. Self-loop
    r. Simple Graph
    s. Path
2. Graph Representations
    a. Adjacency Matrix
    b. Incidence Matrix
    c. Adjacency List
3. Graph Traversal
    a. Depth-First Search (DFS)
        i. Iterative Deepening Search(IDS) or Iterative Deepening Depth First Search(IDDFS)
    b. Breadth-First Search (BFS)
    c. Depth-limited Search
    d. Uniform Cost Search
    e. Bidirectional Search
    f. Water Jug Problem

Week-6

1. Graph Topological Sorting
2. Graph MST
3. Graph Backtracking
    a. Tug of War
    b. n-Queen's Problem
    c. m Coloring Problem
    d. Euler & Hamiltonian Path
4. Graph Sortest Paths
5. Graph Connectivity
6. Graph Max Flow
7. Graph Isomorphism
    a. https://github.com/Mith13/Graphs-isomorphism
8. Graph canonization
9. Graph Cuts
    a. Min Cut
    b. Max Cut
10. Alpha-Beta Pruning
11. Hasse Diagrams
12. Petri Nets
13. Bipartite Graphs
14. Cycle Detection
    a. Brent's Algorithm
    b. Hare and Tortoise Algorithm
15. Bayesian Network

Week-7

1. Linear Search
2. Binary Search
    a. Interpolation Search
3. Fibonacci Search

4. Hashing and Hash Tables
    a. Direct-Address Tables
    b. Hash Tables
    c. Hash Functions
    d. Open Adressing
    e. Perfect Hashing

Week-8 (Midterm)

Week-9

5. Sortings
    a. Insertion Sort
    b. Selection Sort
    c. Radix Sort
    d. Quick Sort
    e. Heap Sort
    f. Permutation Sort
    g. Gnome Sort
    h. Comb Sort
    i. Flash Sort
    j. Stooge Sort
    k. Bees Algorithm
    l. Lucky Sort
    m. Indirect Sort (Pointer Sort)
    n. External Sort (Segmented Sort)
    o. Shaker Sort / Bidirectional Bubble Sort
    p. Shell Sort
    q. Comparison of Sorting Methods

Week-10

1. Trees
    a. Binary Search Tree
        i. Search and Insertion
        ii. Delete
        iii. BST over Hash Table
        iv. Construction and Conversions
        v. Check Smallest/Largest Element
        vi. Red Black Tree and Threaded Binary Tree
    b. AVL Trees
    c. B Trees
        i. Defitinion of B Trees
        ii. Basic operations on B tree
        iii. Deleting a key from a B tree
    d. 2 3 4 Trees
    e. 2 3 Trees
    f. B+ Trees
    g. R Trees
    h. Red - Black Tree Datastructure
    i. Splay Tree Datastructure

j. Augmenting Data Structures
    i. Dynamic order statistics
    ii. How to augment a data structure
    iii. Interval trees
k. van Emde Boas Trees
    i. Preliminary approaches
    ii. A recursive structure
    iii. The van Emde Boas tree
l. Binomial Trees
m. Comparison of Search Trees
n. Minimax Tree

Week-11

1. Strings
    a. Longest common subsequence problem
        i. Longest increasing subsequence
        ii. Hunt–Szymanski algorithm (Hunt Macllory)
        iii. Levenshtein distance
        iv. Wagner–Fischer algorithm
    b. String Alignment
        i. Needleman Wunsch
        ii. Smith Waterman
        iii. Hunt Macllory
    c. String Tokenizer
    d. String Comparison

Week-12

2. Strings
    a. Reverse Factor Algorithm (String Search)
        i. Knuth-Morris-Pratt Algorithm
        ii. Horspool Algorithm
        iii. Boyer-Moore Algorithm
        iv. Brute-Force / Linear Text Search
        v. DFA Text Search

1. Tries
    a. Patricia Tree (Radix Tree)
2. Data Structures for Disjoint Sets
    a. Disjoint-set operations
    b. Linked-list representation of disjoint sets
    c. Disjoint-set forests
    d. Analysis of union by rank with path compression

Week-13

1. File Organization
    a. Sequential File Organization
        i. Binary Search
        ii. Interpolation Search
        iii. Self-Organizing Sequential Search
    b. Direct File Organization

i. Locating Information
ii. Hashing Functions (MD5, HAVAL, SHA1 etc.)
    1. Key mod N
    2. Key mod P
    3. Truncation
    4. Folding
    5. Squaring
    6. Radix Conversion
    7. Polynomial Hashing
    8. Alphabetic Keys
    9. Collisions
iii. Collision Resolution
    1. Collision resolution with links
    2. Collision resolution without links
        a. Static positioning of records
        b. Dynamic positioning of records
    3. Collision resolution with pseudolinks
iv. Coalesced Hashing
    1. EISCH
    2. LISCH
    3. BEISCH
    4. BLISCH
    5. REISCH
    6. RLISCH
    7. EICH
    8. LICH
v. Progressive Overflow
    1. Linear Probing
    2. Quadratic Probing
vi. Double Hashing
vii. Use of Buckets
viii. Linear Quotient
ix. Brent's Method
x. Binary Tree
xi. Computed Chaining Insertion(CCI)
xii. Comparison of Collision Resolution Methods
xiii. Perfect Hashing
xiv. SimHash

Week-14

a. Indexed Sequential File Organization
b. Bits of Information
c. Secondary Key Retrieval
    i. Multilist File Organization
    ii. Inverted Files
    iii. Partial Match Retrieval with Signature Trees
    iv. Partial Match Retrieval with Page Signatures
d. Bits and Hashing
    i. Signature Hashing

ii. Bloom Filters
iii. Classification Hashing
iv. Check Hashing

e. Binary Tree Structures
    i. Binary Search Trees
    ii. AVL Trees
    iii. Internal Path Reduction Trees
f. B-Trees and Derivatives
    i. B-Trees
    ii. B#-Trees
    iii. B+ -Trees

Week-15

g. Hashing Techniques for Expandable Files
    i. Extendible Hashing
    ii. Dynamic Hashing
    iii. Linear Hashing
h. Other Tree Structures
    i. Tries
    ii. Approximate String Matching
    iii. Trie Hashing
    iv. PATRICIA Trees
    v. Digital Search Trees
i. Seconday Key Retrieval (2)
    i. K-d trees
    ii. Grid Files
j. File Sorting
    i. Insertion Sort
    ii. Quicksort
    iii. Heapsort
    iv. External Sorting
    v. Sorting by Merging
    vi. Disk Sort

Week-16 (Final)