CADT IDT

ALGORITHM & DATA STRUCTURES

Final Project

Introduction

Welcome to the final project!

The final project aims at applying the concepts you have acquired during course to a real-world problem.

You will identify a **real-world problem,** determine and justify the **choice of one or more data structures**, **implement** a solution, **analyze your solution** in term of complexity, and finally **document the process** thoroughly.

Projects will be performed in teams. A team consists of 3 students.
Good luck everyone!

Agenda

- ✓ **Proposal Submission**
  *Teams will submit a one-page proposal outlining the problem, ADT(s) to be used, and the planned implementation strategy by **11th December.***

- ✓ **Project Defense and Final Submission**
  *Teams will deliver a final report, present their project, and defend their solution. Deliverables include the report, code, and presentation. Presentation date: **26th December***

Deliverables

Date	DELIVERABLE
By **11th December**	Proposal Document - *see template*
By **26th December**	Final Report - *see template* Presentation slides Source code

Proposal Document *Template*

The proposal should briefly outline **your problem, plan**, and **approach**.
Below is the template with examples to guide you:

Team Members:

Example: Alice Doe, Bob Smith

Project Title:

Example: Efficient Routing for a Delivery System

Problem Statement:

Describe the problem you aim to solve and its significance.

Example: Delivery companies face challenges in optimizing their routing to minimize fuel consumption and time.

Expected Outcomes:

Summarize what you aim to achieve.

Example: Develop an application that recommends optimal delivery routes and compare its efficiency with existing solutions.

Proposed ADT(s):

Identify and justify the choice of ADTs.

Example:
-We will use a Graph based ADT as its best suited for modeling delivery routes
-Nodes will represent locations and edges represent paths with weights indicating distance or cost.

Implementation Plan:

Outline how you will implement the solution.

Example:
- We will represent the city as a graph using an adjacency list
-We will implement Dijkstra's algorithm for shortest path computation
-We will finally evaluate performance with real-world data sets.

Final Report *Template*

The final report should be structured as follows:

1. Introduction

Brief description of the problem and its relevance.

Example: Delivery routing optimization reduces fuel costs and delivery times, directly impacting business efficiency.

2. Problem Definition and Requirements

Define the problem in detail and list any requirements or constraints.

Example: Routes must avoid traffic hotspots and minimize travel time.

3. Abstract Data Type (ADT) Selection

Justify the choice of ADT(s) and how they fit the problem.

Example: Graphs enable efficient representation of routes, and priority queues enhance shortest-path computations.

4. Implementation Details

Provide a thorough explanation of your implementation, covering the following aspects:

4.1 Data Structure(s)

Specify the **data structure(s)** and justify their use in solving the problem.

- Provide a clear API specification for the data structure(s), detailing methods, their parameters, and expected behavior.

Method	Description	Parameter	Returns
shortestPath	Computes the shortest path from a source node to destination node.	Node from Node to	List of nodes

- If applicable, include **constraints or limitations** of your data structure(s).
- If needed, include a UML class diagram to **represent relationships** between your classes.

For example, for a graph representation: -Graph class with attributes for nodes and edges. -PriorityQueue class used in conjunction with the Graph for Dijkstra's algorithm.

4.1 Key algorithms

Describe **1 or 2 key algorithms used** and how they integrate with the data structure(s).

- Input/Output: What the algorithm takes as input and produces as output.
- Step-by-Step Logic: Summarize the main steps and their purpose.

5. Performance Analysis

Analyze the time and space complexity of your solution.

Example: Time Complexity: Dijkstra's algorithm runs in O((V + E) log V) for a graph with V vertices and E edges. Space Complexity: Adjacency list requires O(V + E) space.

6. Results and Discussion

Summarize your findings and compare them with expectations. Include graphs or tables if applicable.

Example: Shortest route computation reduced delivery time by 20%.

7. Challenges and Future Improvements

Discuss challenges faced during the project and potential improvements.

Example: Incorporating real-time traffic data is a challenge we aim to address in future work.

9. References

List any references used during the project.

Suggestions of problems to solve

Here's a list of suggested problems
You are more than welcome to be creative an come up with your own problem to solve 😃

Optimal Delivery Routing

Design a system to optimize delivery routes in a city

Task Scheduler:

Develop a system to allocate tasks to workers efficiently, using priority queues or interval scheduling.

Undo-Redo application

Develop an undo-redo system for a text editor or drawing application.
Use two stacks: one for undo operations and another for redo.

Browser History Navigation

Simulate browser navigation using two stacks to manage forward and backward movements.

Expression Evaluation

Implement a program to evaluate postfix or prefix expressions using a stack.
Extend it to convert infix expressions to postfix or prefix.

Conference Room Allocation

Solve the problem of scheduling meetings in the minimum number of rooms using interval graphs.

External Sorting:

Implement a sorting algorithm to handle massive datasets that cannot fit in memory.

Coffee Shop System

-Console application
-Show menu to user
-CRUD coffee
-User buy coffee
-Read data from file
-Write data to file
-Store user's history
-Store user data
-Generate report (user, income, …)
-etc.

Student management System

-Console application
-Show menu to user
-CRUD student
-Read data from file
-Write data to file
-Store history of operations
-…etc.

Attendance System

-Console application
-Show menu to user
-CRUD attendance
-User register attendance
-Read data from file
-Write data to file
-Store user absent/present
-Store user data, …etc.

Quiz System

-Console application
-Show menu to user
-CRUD quiz by admin
-User take quiz (should random questions, …)
-Read data from file
-Write data to file
-Generate report (user score, quiz, …) etc.

Library management System

-Console application
-Show menu to user
-CRUD book, info in library
-Read data from file
-Write data to file
-Store history of operations
-Student requests borrow books
-etc….

Employee management system

-Console application
-Show menu to user

-CRUD employee
-Display upcoming retired employees in 5 years
-Read data from file
-Write data to file
-Generate report (user score, quiz, …)

Hotel management system

-Console application
-Show menu to user
-CRUD hotel room
-User books rooms, payment
-Read data from file
-Write data to file
-Store user test's history
-Store user data, …etc.

E-commerce application

-Console application
-Show menu to user
-CRUD product
-User buy product, add to card
-Read data from file
-Write data to file
-Create invoice/receipt
-Store user data & history
-Generate report (user, sales, …)

Remark about your project:

- CRUD operation should be able to perform by your program.
- The program should implement: Linked list, stack, queue, or hash table. And the program must use File IO to store data. Use header (.h) also store your data structure.
- The program read data from file to store in data structure. Program working with variable to perform CRUD. Finally, store data back to file when we stop the program.