Data Structures and Algorithms

Final Project - Problem Analysis

Week 11

Time:
5 min

Peer Project - 2 people

- You will be assessed on:
  - **1) Problem Analysis** - how well have you analysed and broken down the problem explaining why is it important to solve and what particular constraints the problem has.
  - **2) Problem Interpretation** - how well have you interpreted the problem in terms of data structures and algorithms. What are the requirements and resource constraints from a resource usage point of view? And what factors should we keep in mind when making decisions around the chosen data structures and algorithms.
  - **3) Problem Solution** - based on the interpretation of the problem, which specific data structures have you chosen to implement your solution and why? What are the advantages of your chosen solution and what are the limitations when another solution could have been better? How complex is the solution?
  - **4) Solution Implementation** - how have you implemented your solution? Does it solve the requirements of the problem? Does the code run smoothly and is it easy to read? Would a client or user be happy with your implementation? Is it tested thoroughly? Is it flexible to be changed if the problem scope changes?

Time:
5 min

Peer Project

- You will **not** be assessed on but you **may want to think about**:
    - The front-end user interface / experience
    - The data that you use - dummy data is fine
    - How well it integrates with other applications/interfaces (APIs)
    - What language you chose to use - any is fine
    - How secure is it?
    - How well does it run on different devices? (e.g. OS, Android etc)
- The focus is on the **functionality** of what you're building
- There should be some **data storage** and **manipulation** aspect

Time:
5 min

Peer Projects

- You **may** use other online repositories to help you with other aspects (e.g. integrations, user interface, security etc)
- You **must** write code for the core data structures and algorithms functionality.
- You **need** to present the problem and solution nicely in a PDF. You should discuss what you considered and why you made certain decisions.
- You **may** want to find a mentor (user) to help you think about the specific functions that would be useful for the use-case that you are looking for.

Time:
5 min

Project Example - Abstract

- I, as a facilitator, want to build an interactive question/answer tool that helps students improve their learning and helps me understand how students are learning.
- The facilitator interface should allow facilitators to input questions & answers, view performance of students and view questions/topics that were difficult
- The student interface should allow students to respond to questions, view the correct answer and track their own performance.

Time:
15 min

Problem Example - Analysis

There are multiple important problems here. In priority, these are

1. Students need more practice
2. Students need more feedback
3. Facilitators need to know which concepts to spend more time teaching
4. Students need to know which concepts to spend more time learning
5. Facilitators need to know which students are struggling

This means that the priority here is the students and their learning. The more questions that are on this platform, the better. The wider the variety of questions, the better. It cannot be laggy for students as they will lose attention. It should repeat questions to check that the student has understood but not too often. It should mix concepts focusing on concepts that students are struggling with.

Time:
15 min

Problem Example - Analysis

Individual student report - Facilitators should be able to see a report showing which topics/questions each student is struggling with. Students should also be get access to their own report.

Overall student report - Facilitators should be able to see a report showing a list of all students ordered by which students are struggling the most (in all topics)

Overall topic report - Facilitators should be able to see a report showing a list of all topics ordered by which topics are most difficult and need attention.

Individual topic report - Facilitators should be able to see a report for each topic showing which questions they are struggling with.

Individual question report - Facilitators should be able to see performance for a particular question

Generating questions - Students should be able to access a random question either in a topic of their choice or one generated for them in a topic of difficulty.

Time:
15 min

Problem Example - Interpretation

What data objects are needed? - student, question, facilitator

What individual attributes are needed for each data object?

- Students - Name, Email Address
- Question - Question Type (Multichoice, Free Answer), Question, Answer, Possible Answers
- Topic - Topic Name
- Facilitator - Name, Email Address

What relationships attributes are needed?

- Each student should store which questions they have answered marking them as having answered it correctly/incorrectly.
- Each question should store which students have answered it correctly and which topic it is part of

Time:
15 min

Problem Example - Interpretation 2

What methods are needed (other than initialise)?

- Students - Answer Question, View Student Report
- Question - View Question Report
- Topic - View Topic Report

What other functions/algorithms are needed?

- View overall topic report
- View overall student report
- Generate question based on chosen topic for student
- Generate question for student (preferring difficult topics)

What constraints should we keep in mind?

- Report generation should be fast - O(n log n) if possible where n is number of topics/students/questions
- Generate question should be fast - O(n log n) if possible where n is number of questions
- Storage should be limited - O(q+s+t+f) where q,s,t,f are number of questions, students, topics and

Time:
15 min

Problem Example - Interpretation 3

What collective data structures are needed? What operations should be easy on them?

- A DS to store all students - insert, access, sort (by performance)
- A DS to store all questions - insert, access, sort (by difficulty)
- A DS to store all topics - insert, access, sort (by difficulty)

Other things to consider:

- Stability: data (especially performance/difficulty) should be changing often
- Duplicates: duplicates should not be allowed
- No non-linear hierarchies (except that many questions make up a topic)
- Order (performance/difficulty) only changes when a student answers a question correctly. Only a small change is required to the overall order.

Time:
5 min

Data Structures Considerations

- Is space a constraint - do I have limited space or limited contiguous space?
- What kind of operations am I doing often? Accessing, Deleting, Inserting?
- How stable is the data? Will it be changed often?
- What primitive data types will I be storing? Can I limit some options?
- Does the order matter? Are duplicates a problem? How do I deal with them?
- Are there other hierarchies / relationships that matter (graphs/trees)
- What is easy to implement/use?

Time:
1 min

Questions