Data Structures and Algorithms in Python

Project 3: Data Structure Video

A data structure is a collection of data values, the relationships among them, and the functions or operations that can be applied to the data. Data structures are a central concept in computer science and are key to the development of efficient algorithms. You already know about a couple basic data structures - arrays and linked lists for instance. For this project, you will work with a team to learn about another data structure and produce a video and computer program using your chosen data structure. As the final deliverable we will play your video in class, and your group will present the code you wrote with an explanation of how your chosen data structure was used.

Your video should meet the following requirements:

- 5-10 minutes in length
- Specify the Abstract Data Type (ADT)
- Explain the implementation of at least one method using either:
    - pseudocode, or
    - a physical representation using real-world objects.
- Use of stock video footage
- Humans appearing in the video for a skit or explanation.

Your program should meet the following requirements:

- Uses your chosen data structure for a clear purpose. That purpose could be related to the video you create, or not – it's up to you!
- Displays output
- A non-trivial amount of code. You do not have to choose one of the projects from the textbook, but your program should be at least as complex as one of the suggested projects.

There are three options for topics that you can select. These topics include two or three related data structures. You should work with your team to learn the basics of all of them, and then select a particular focus for your video and computer program.

Option 1: Stacks, Queues, & Deques

Dive into the world of linear data structures with Stacks, Queues, and Deques, each offering unique ways to store and manage data. Stacks operate on a last-in, first-out (LIFO) principle, making them ideal for tasks like undo mechanisms in text editors or for parsing expressions. Queues follow a first-in, first-out (FIFO) approach, crucial for scheduling tasks in computing environments or managing any process that requires a sequence. Deques, or double-ended queues, combine the features of both stacks and queues, allowing insertion and deletion at both ends, making them versatile for scenarios requiring flexible access to data. This topic will challenge you to explore these structures' implementation and practical applications, enhancing your understanding of data management and algorithm efficiency.

Reference materials:

- HackerRank Stacks & Queues video: https://www.youtube.com/watch?v=wjI1WNcIntg
- *Data Structures and Algorithms in Python* Chapter 6 Reading, sections 6.1-6.3
- *Data Structures and Algorithms in Python* Chapter 6 Projects

Option 2: Tress & Binary Trees

Embark on a journey through hierarchical data structures with Trees and Binary Trees, fundamental for representing structured hierarchy. Trees consist of nodes connected by edges, without any cycles, creating a branching structure that mimics real-world systems like family trees or organizational charts. Binary Trees, a special category, restrict each node to have at most two children, leading to widespread use in searching algorithms, implementing databases, and more. This option invites you to explore the concepts of tree traversal, binary search trees, and balanced vs. unbalanced trees, offering a solid foundation in managing hierarchical data efficiently and understanding its impact on algorithm performance.

Reference materials:

- HackerRank Trees video: https://www.youtube.com/watch?v=oSWTXtMglKE
- *Data Structures and Algorithms in Python* Chapter 8 Reading, sections 8.1-8.3
- *Data Structures and Algorithms in Python* Chapter 8 Projects

Option 3: Heaps & Priority Queues

Explore the dynamic world of Heaps and Priority Queues, specialized data structures that prioritize elements based on their value. Heaps are primarily used for implementing priority queues; they ensure that the highest (or lowest) priority element is always at the front. This feature is crucial in algorithms like Heap Sort or in managing tasks with dynamic priorities, such as scheduling jobs in an operating system. Priority Queues, built on the principles of heaps, offer an efficient way to manage a set of data where the element with the highest priority is served before those with lower priority. This topic will allow you to explore the mechanics of heap operations, the implementation of priority queues, and their applications in real-world scenarios, such as traffic management or event simulation.

Reference materials:

- HackerRank Heaps Video: https://www.youtube.com/watch?v=t0Cq6tVNRBA
- Priority Queue Introduction Video: https://www.youtube.com/watch?v=wptevk0bshY
- *Data Structures and Algorithms in Python* Chapter 9 reading, sections 9.1-9.3
- *Data Structures and Algorithms in Python* Chapter 9 Projects