When designing a program that will be working with data structures, it is important to utilize big O notation to decide on the best algorithm to use. In this project, I focused on designing algorithms for hash tables, binary trees, and vectors. For this courses project, I designed pseudocode to store and display course information using those 3 data structures. Before deciding on which algorithm to use, it is important to review the advantages and disadvantages of both.

In a vector, data is displayed linear, which provides great sequential access to elements. It is helpful for iterating through lists and are simple to use since it possesses a straight forward interface. Despite making traversing through lists easier, using a vector can be time consuming for other functionalities. For example, if a course needs to be inserted or deleted at the head or tail of a vector, the time cost is very short, however, if an item needs to be inserted in the middle of a list, it can be costly due to elements needing to shift. Not only can insertion/deletion be costly, but searching can also be costly. A vector searches for a specific element by iterating through the entire vector.

In a hash table, it is best used for searching through data. Hash table keys are not restricted to a certain data type. A hash key can be various types, resulting in more flexibility when working with data. Although the time-average complexity is constant for insertion, search, and deletion, it does not mean there is no potential of worst average complexity. The main issue when inserting data in a hash table are collisions. Collisions can happen when two keys hash to the same value. Ensuring collisions from happening can be costly and requires incorporating mechanisms like chaining or open addressing. Another drawback of hash tables is the amount of memory used for a hash function and collision solutions.

A binary tree is best used to maintain order, which allows for efficient searches, insertions, and deletions. Binary trees are known for their simplicity, which provides users with a simple understanding of data structure management. Despite being one of the easier structures to understand, deletion of nodes can be complex. When a node is deleted, programmers need to make sure to understand how nodes shift.

I recommend using a vector due to its performance benefits. Due to how little the course list is, it would be beneficial to use a vector since it can be simpler to access elements. Users will want to access random courses and vectors support that functionality by using indices. It is important to choose a data structure based on specific requirements and usage patterns of the project. Since I anticipate this project not requiring a lot of insertions, I feel confident in recommending vectors.

References

1. baeldung, W. by: (2023, March 11). Binary trees vs. linked lists vs. hash tables. Baeldung on Computer Science. https://www.baeldung.com/cs/binary-trees-vs-linked-lists-vs-hash-tables#:~:text=The%20main%20advantage%20of%20using,to%20reflect%20relationships%20between%20data.