Vector will be the first one that we talk about here. Some advantages of vectors are the simplicity that it offers. It is easy to implement and provide straight forward access to elements by index. It’s efficient for sorting small amounts of data. It also provides random access. The elements in the vector can be accessed by index in constant time O(1) which is useful if you know the position of the course. So, disadvantages include sorting overhead, because vectors do not maintain a sorted order by default. Any time that I need to list courses in alphanumeric order I must sort it explicitly. Sorting requires O(n log\_n) which can be inefficient if I need to print courses in order often. The insertion and deletion are inefficient. If I must insert or delete an element from a vector in the middle it requires me to shift elements which takes O(n). Also searching is inefficient. To search for a specific course in a vector it requires O(n) as I must perform linier search unless I sort it first which adds more complexity.

Next, I will discuss Hash Tables. Som advantages of Hash Tables include fast look up and insertion. Hash Tables provide average case O91) time complexity for searching, inserting and deleting elements. This makes finding specific courses and printing efficient. There is also direct access. This is due to using course numbers as keys. Because of this you can access them quickly without iterating through the entire collection. A couple disadvantages include no natural order. Hash Tables don’t maintain any order among their elements. Thus, printing the courses in alphanumeric order requires extracting the keys, sorting, and then accessing the course data. Sorting requires O (n log\_n). Another disadvantage is memory overhead. They require additional memory to manage the hash function, buckets and potential collisions. The final would be collisions. Worst case scenario with poor hash functions or too many collisions the time complexity for operations can degrade to O(n).

Finally, is Binary Search Tree or BST. Some advantages include maintaining sorted order. This makes printing the courses in alphanumeric order straightforward and efficient via an in-order traversal which takes O(n). Next would be efficient search. With a balanced binary search tree, searching for a specific course takes O(log\_n) time. This is faster than a vector linear search and slower than a hash tables average case constant time search, but it’s consistent even in the worst case. Finally, an advantage is Dynamic Insertion and Deletion. Which means it takes O(log\_n) time making it more efficient than a vector. Some disadvantages are balancing issues. The worst case for unbalanced trees is the time complexity for search, insertion and deletion can degrade to O(n) which negates the advantages of the tree structure. It is also more complex to implement. They are more difficult to implement and maintain than vectors or hash tables. Self-balancing trees like AVL or Red-black trees require additional logic to ensure that the tree stays balanced. Finally, is the memory overhead. Trees require more memory than vectors because each node stores not only data but also points to its child node.