# Vector Program Analysis:

## Advantages:

Two of the largest advantages of a vector data structure in this particular case is they are simple, effective and easier to understand what is happening within the code. Basic iterators can be used to sort and print the information that is contained in the vector of courses and in the prerequisites vector. Depending on how many courses need to be added to the course vector, the best case scenario will be an O(1), or O(n) run time needed to execute the program.

## Disadvantages:

Nested loops need to be used to sort data and run the quality control portion of the program, leading to an O(n) worst case scenario runtime needed. While the time needed to execute the program is not as much as a sorted hash table, the time it will take to sort and execute will be a multiple as the size of the list grows.

# Hash Table Analysis:

Advantages:

The primary advantage of a hash table in this case is the method used to store data in the vector is seemingly random if you don’t know what the modulus for producing the key is based off. However, in our case information security is likely not important.

Disadvantages:

Hash tables are optimized for lists where order doesn’t matter. While they add a layer of security for their retrieval because of their hash key, which is a modulus of some number, whether its an arbitrary number or based on the size of the hash table. However, In this case in particular, the strength of being more secure is actually its weakness. To get the list of courses to print out in alphabetical order, one of two things need to be done, and in both cases the data of the hash table needs to be stored in a separate vector or array:

1. Both a vector *and* hash table can be populated at the same time when the courses are loaded from the txt file, which will effectively double the space requirement needed to store courses and will also negate security of the hash as the same information will be stored elsewhere. But in this case, you would be more storing a data in a vector than you would be a hash table.
2. A temporary vector can be populated from the hash table using additional functions within the class. This will almost certainly lead to a slow running speed as not only will the user need to wait for the hash table to populate, they will then need to wait for the hash table to transcribe to the temporary vector, wait for the temporary vector to be sorted *and* wait for the information to be printed. In my Pseudocode, this is what I demonstrated.

In both cases, we will be looking at an O(n) time requirement under best case conditions and O(n^2) worst case scenario since you would be accomplishing the Vector data storage technique twice: extracting and populating the hash table, then extracting and populating a vector from the hash table..

Another downfall of a hash table in this particular case is the space needed to store the information *will be doubled*. Not only do you need a vector to store the hashed nodes and its information, but you will also need some type of vector to store the information that has either been populated during the text file parsing or populated into a temporary vector to be printed out.

# Binary Search Tree:

## Advantages:

This data storage method is the quickest way to accomplish the task at hand specifically because there is no need implement an exclusive sorting function which sorts the list and a function that exclusively prints the list. The function that inputs information into the binary search tree mostly sorts it as the information is inputted and as it prints, it depends on a recursive call versus an iterator to look at the next item to print based on its alphanumeric value. This speeds up the processing time it takes with an O(n) worst case run time and a best case, O(log N) runtime: All that’s happening is a comparison between one node and another, at different levels.

## Disadvantages:

Can become complicated if client wants to remove courses from the course list. While this algorithm is fast, it also fast to determine the algorithm used for storing data making it less secure. But the things that ABCU seems to require don’t appear to need tight security to accomplish the job.

# Recommendation Moving Forward:

Given the different requirements from ABCU and the estimated run time analysis of each program, I would recommend pursuing a binary search tree data structure, purely because of the speed that the data structure moves at to accomplish the task at hand and how seamless it is to sort the courses. While a vector based data structure is more simplistic and straight forward, the requirement of several loops and functions to accomplish sorting make it a second choice because of the time analysis. However, with this being said, I do not think that the time difference between a Binary Search Tree and a Vector based data structure would be negligible considering the likelihood that there will not be a significant amount of information needing to be sorted.