Analyze vector, hash table, and tree

**For the vector**

for creating a course object

(n+1)+(1)+(n+1)+(n-1)+(1)= 3n+3

for reading a file

n+1+n+1= 2n+2

Total =(3n+3)+(2n+2)

=5n+5

Runtime = O(n)

**Hashtable**

for creating course object and file reading

n+n+n+1+n+1+n+1+n+n+1+n+1+1+5n+n+1 =14n+7

Runtime = O(n)

**Tree**

n+1+n+1+1+n+1+n+1+n+1+1+n+1+1+1=6n+10

Runtime = O(n)

**Explain the advantages and disadvantages of each structure in your evaluation.**

1. **Hashtables**

***Advantages***

The search and insert operations takes the order of O(1). This implies that it takes only one operation. Thus being very fast. One of its advantage is its speed in its operations. When the number of objects to be searched are trillions and trillions, it will only round them into O(logn) and O(1) this will imply the overheads adding up for different data. The independence of the size of the data structure becoming the main reason. Additionally, the growth of this data structure is linear and does not take a lot of time in execution.

Another advantage is its speed of searching keys even when the data is huge. Unlike other structures, the time complexity for the hashtable remains very high. The speed is as an increased usage of collision techniques. In addition to the above advantages, synchronization becomes one of the main advantage of hashtables.

***Disadvantages***

Considering that in hashtables that the use of hashing functions is of paramount importance, practically avoiding their collisions is impossible. This usually happens when hashing data or subsets with multiple possible keys. At this point, collisions appear making it inefficient that is, too many collisions will lead to it not delivering the expected results, whenever a search operation is executed, the initialized key collides with other keys, this results in its inefficiency. Additionally, the hastables do not null values.

1. **Vectors**

Vectors acts as dynamic arrays, this implies that, they have the ability to resize themselves with the increase of data objects. With this ability, it means that no specialized techniques needs to be performed on a vector whenever the size of the data increases and had not been accounted for. Also upon deletion of the elements from the vector, it resizes itself to fit the elements it holds. Another advantage is that, vectors stores its data elements in contiguous storage to allow iterators to access these elements. Time complexity for insertion and deletion operations at the beginning of the vector is linear, that is O(1). This ensures that it takes shortest time when it comes to retrieval of data elements that are at the beginning of the vector.

**Disadvantage**

Mostly in vectors data is inserted at the end of the vector, the time complexity for this operation is not linear, meaning it takes more time. The reason for this is as it takes more time to extend the the array.

1. **Tree**

The time complexity for traversing a tree is O(n). This practically means that it takes linear time for the growth of this data structure. This is a short time as compared to other data structure operations which would take exponential times. A tree takes the benefits of both linked lists and ordered arrays. This equates to the search and deletion operations being very fast, this implies that the time complexity for this data structure is low.

**Disadvantage**

Despite the search operations taking the shortest time, the number of operations can be very many if the tree has many children. When searching for the elements, it means that all the children must be traversed. If misses occurs, it passes to the next child until the hit is acquired.

For the implementation of my code, I will choose hashtable. Despite the disadvantage of collisions in keys, the code I intended to implement does not have multiple keys that will collide, this therefore means that it qualifies to be the best in terms of its time complexity.

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **for all courses** | 1 | n | n |
| **if the course is the same as courseNumber** | 1 | n | n |
| **print out the course information** | 1 | 1 | 1 |
| **for each prerequisite of the course** | 1 | n | n |
| **print the prerequisite course information** | 1 | n | n |
| **Total Cost** | | | 4n + 1 |
| **Runtime** | | | O(n) |