Hanna Parham

Professor Lebel

CS-300 DSA: Analysis and Design

18 August 2024

**Project One**

**Runtime Analysis**

The advantages of using a **VECTOR** would be that they are rather simple to implement and make it easy to retrieve data, they are very efficient when it comes to accessing elements, and they also have dynamic sizing. The disadvantages of using a Vector would include that it is often time consuming to add or delete elements within the middle of a vector versus in the beginning or end of it. Additionally, it does take time to search for any specific elements in an unsorted vector as it has to first scan through all of its elements.

The advantages of using a **HASH TABLE** would include how fast it is to not only look up something within it but to also insert and delete elements and just how efficient it is for checking for any duplicates within the code element wise. The disadvantages would include that they are not the best when it comes to maintaining order and its performance can degrade if there are too many collisions, ie. when two different pieces of data are assigned to the same key, which may be something I could potentially do by accident without realizing.

The advantages of using a **BINARY TREE** would include how easy it is to maintain things that need to be sorted while also making sorting and range queries easy and efficient. Balanced trees offer O(log n) time for adding, removing and finding items. The disadvantages would be that it is more complex compared to vectors or hash tables and accessing the index with a binary tree is slower than it is with vectors.

Based on these findings along with the chart below, I would suggest using Binary Trees for this project instead of vectors or hash tables because of how crucial it is for the information to be ordered alphanumerically. Binary trees are more advantageous in situations like this since they inherently maintain sorted order which would allow for efficient adding, removing and search queries, especially if the dataset needs to grow out of the courses that are only included on it now. They also provide a balanced approach to managing data which reduces the chance of performance degradation that can happen with hash tree collins and or the need for extensive sorting in vectors. I think that this would be the ideal choice for this scenario despite its given complexity as the emphasis for ordering elements in this project is large, and it would not entirely make sense to use hash tables or vectors instead.

**CHART FOR RUNTIME ANALYSIS**

|  | **VECTOR** | **HASH TABLE** | **BINARY TREE** |
| --- | --- | --- | --- |
| **Reading the File** | O(n) | O(n) | O(n) |
| **Loading Data into X** | O(n) | O(n) | O(n log n) |
| **Creating Course Objects** | O(n) | O(n) | O(n) |
| **Cost Per Line** | 1 per line | 1 per line | 1 per line |
| **Number of Executions** | n | n | n |
| **Total Cost** | O(n) | O(n) | O(n log n) |