Harley Reimels

04/11/2024

CS – 300 DSA

Module 6 – Runtime Analysis

Vector Data Structure

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Read in the file** | 1 | 1 | 1 |
| **For all items in file** | 1 | n | n |
| **Append item to vector** | 1 | 1 | 1 |
| **Total Cost** | | | n + 2 |
| **Runtime** | | | O(n) |

The main advantage of the Vector structure is that it is very simple to understand, and for the purposes of this project it would be absolutely fine to use. Whenever we create a new object, we just add it to the end. A vector is resizable, meaning that there is no issue if we need more space. This data structure in the worst case is run at a linear time complexity, which is good.

Hash Table Data Structure

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Read in the file** | 1 | 1 | 1 |
| **For item in file** | 1 | n | n |
| **Find the hash key** | 1 | 1 | 1 |
| **If key is taken, loop until availability is found (linear probing)** | 1 | n | n |
| **Assign item to key location** | 1 | n | n |
| **Total Cost** | | | 3n + 2 |
| **Runtime** | | | O(n) |

The hash table is a data structure that is very simple to use. The benefits of using a hash table would be to try and reduce collisions, but for a size of information this small, I do not think the need for a hash table would be necessary. The hash table worse case runtime is Linear.

Binary Tree Data Structure

| **Code** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| **Read in the file** | 1 | 1 | 1 |
| **For item in file** | 1 | n | n |
| **If root is empty assign item to root** | 1 | 1 | 1 |
| **If item is less than parent, assign as left child**  **If item is more than parent, assign as right child** | 1 | n | n |
| **Total Cost** | | | 2n + 2 |
| **Runtime** | | | O(n) |

The binary search tree is the most complex of the three to understand but it does have very efficient methods for searching. Each time an option is chosen, half of the remaining options are no longer possible, cutting down the search time incredibly compared to the other two. One disadvantage would be trying to store them using the class names, since we have different class titles, it would be difficult to store them without giving each object another unique identifiable variable.

My Suggestion:

I would recommend using a Vector Data Structure, as this program for ABCU Computer Science department is not very large, and all three structures run at linear time complexity. Since this is the case, we could analyze the average use time complexity, in which Hash Table is the only one that will run at constant time. Binary Search tree runs at Logarithmic time complexity as an average time complexity, which is also very good. Vector is the only one running at Linear time for average as well. Although, vector may be the absolute worse one to choose for this, the speed difference for a project so small is negligible. An absolute beginner could understand the concepts behind a vector, how it stores items, accesses items, meaning if any changes need to be made, it would not take an expert.