Joshua Pugh

CS-300

2/18/2024

6-2 Project One: Evaluation

**Worst-Case Running Time:**

**Reading the file:** Since two loops are being used to read from the file with additional operations within each loop, the Worst-Case running time would be O(n + m). The while loop will execute up to n times giving a time complexity of O(n) and the for loop will execute up to m times giving a time complexity of O(m). Combining these will give the worst-case running time of O(n + m). This assumes there is n number of lines in the file, a cost of 1 for each line, and all lines are executed one time.

**Create Course Objects:** O(n).

The code below is within a for loop which will execute n times.

|  |  |  |  |
| --- | --- | --- | --- |
| Code | Line cost | # Times executed | Total cost |
| If the line format is valid then: | 1 | n | n |
| Create a new Course object course | 1 | 1 | 1 |
| Course Set course number (line at the index of 0) | 1 | 1 | 1 |
| Course Set course name (line at the index of 1) | 1 | 1 | 1 |
| Course, add prerequisites (line at remaining indexes) | 1 | 1 | 1 |

Total Cost: n + 4

Runtime: O(n)

**Vector:**

**Time Complexity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Best time** | **Worst time** | **Average Time** |
| **Access** | O(1) | O(1) | O(1) |
| **Insert** | O(1) | O(n) | O(n) |
| **Search** | O(1) | O(n) | O(n) |
| **Delete** | O(1) | O(n) | O(n) |

**Advantages:**

* **Dynamic Size:** Vectors can dynamically grow or shrink as needed.
* **Fast Random Access:** Vectors provide constant-time access to elements by index.
* **Contiguous Memory:** Elements in a vector are stored in contiguous memory locations.
* **Predictable Iteration Order:** Vectors maintain the order of insertion, ensuring that elements are iterated in the same sequence they were added.

**Disadvantages:**

* **Fixed Capacity:** Unlike dynamic arrays, vectors have a fixed capacity.
* **Insertion and Deletion Overhead:** Adding or removing elements in the middle of a vector requires shifting subsequent elements.
* **Memory Reallocation:** When a vector needs to grow beyond its current capacity, it reallocates memory and copies existing elements.
* **Inefficient for Frequent Insertions/Deletions:** If frequent insertions or deletions at arbitrary positions are needed, other data structures may be more suitable to use.

**Hash Table:**

**Time Complexity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Best time** | **Worst time** | **Average Time** |
| **Access** | O(1) | O(n) | O(1) |
| **Insert** | O(1) | O(n) | O(1) |
| **Search** | O(1) | O(n) | O(1) |
| **Delete** | O(1) | O(n) | O(1) |

**Advantages:**

* **Fast Data Retrieval:** Hash tables provide rapid access times for elements and typically take constant time O(1) due to the use of a hash function that maps keys to array indices.
* **Efficient Insertions and Deletions:** Hash tables allow efficient insertion and deletion of elements.
* **Applications in Databases and Caches:** Hash tables play a vital role in computer science, including implementing symbol tables, caches, and databases.

**Disadvantages:**

* **Collision Handling:** Hash tables can encounter collisions when multiple keys map to the same index.
* **Memory Usage:** Hash tables require memory to store both the array and the hash function.
* **Hash Function Quality:** The effectiveness of a hash table heavily relies on the quality of the hash function.

**BST:**

**Time Complexity:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Operation** | **Best time** | **Worst time** | **Average Time** |
| **Access** | O(log n) | O(n) | O(log n) |
| **Insert** | O(log n) | O(n) | O(log n) |
| **Search** | O(log n) | O(n) | O(log n) |
| **Delete** | O(log n) | O(n) | O(log n) |

**Advantages:**

* **Efficient Searching:** BSTs allow for fast searching due to their ordered structure.
* **Fast Insertion and Deletion:** When balanced, BSTs are quick at insertion and deletion.
* **Memory Efficiency:** BSTs have lower overhead in terms of memory compared to hash tables.
* **Indexing and Range Queries:** BSTs are used for indexing data efficiently.

**Disadvantages:**

* **Balancing Issues:** If a BST becomes unbalanced, its performance degrades significantly.
* **Complexity in Handling Duplicates:** BSTs do not naturally handle duplicate keys.
* **Sensitive to Input Order:** The performance of a BST depends on the order of insertion.
* **Not Suitable for Small Data Sets:** BSTs perform well on large data sets but may not be highly suitable for small data sets with only a few elements.

**Recommendation:**

I recommend using a hash table (dictionary/map) for the given program. The average time complexity for the listed operations on a hash table is O(1) which is more than efficient for the program that we are designing. Hash tables provide fast access and insertion, which is crucial for efficient lookups and updates in the future. While they consume extra memory, their performance benefits often outweigh this drawback. In addition, most languages offer built-in hash tables. If coding in C++ for example, I would use a map to implement the hash table. Maps can also be sorted which is another plus. A BST could also be used in this situation but one major drawback is if the BST becomes unbalanced this can lead to the worst-case linear time complexity of O(n). From an implementation standpoint, a hash table is far easier to implement than a BST since a BST must be fully coded for implementation. A vector could also be used but if the changes are frequently made and items are added in the middle this would require shifting subsequent elements, resulting in linear time complexity of O(n). Vectors are also not as efficient as hash tables when it comes to insertion and deletion.

**References**

GeeksforGeeks. Time complexities of different data structures. (2020, December 29).

<https://www.geeksforgeeks.org/time-complexities-of-different-data-structures/>

GeeksforGeeks. Applications, Advantages, and Disadvantages of Hash Data Structure. (2022, June 7).

<https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-hash-data-structure>

GeeksforGeeks. Applications, Advantages and Disadvantages of Binary Search Tree. (2022, June 1).

<https://www.geeksforgeeks.org/applications-advantages-and-disadvantages-of-binary-search-tree/>

OpenGenus IQ: Computing Expertise & Legacy. Time and Space complexity of Binary Search Tree (BST). (2021, December 29).

<https://iq.opengenus.org/time-and-space-complexity-of-binary-search-tree/>

Stack Overflow. What is Vector data structure. (n.d.). (2015)

<https://stackoverflow.com/questions/32541194/what-is-vector-data-structure>