Jeffrey Flores

6-1 Project One

August 11, 2024

Pseudocode for Printing Sorted Course List

The pseudocode below will print out the list of courses in the Computer Science program in alphanumeric order using three different data structures: Vector, Hash Table, and Binary Search Tree.

Using Vector:

function printSortedCourses(vector<Course> courses): sort(courses by courseNumber) for each course in courses: print course.courseNumber, course.courseTitle

Using Hash Table:

function printSortedCourses(hashTable<Course> coursesHashTable): create an empty vector<Course> called courseVector for each course in coursesHashTable: insert course into courseVector sort(courseVector by courseNumber) for each course in courseVector: print course.courseNumber, course.courseTitle

Using Binary Search Tree (BST):

function printSortedCourses(binarySearchTree<Course> coursesBST): inOrderTraverse(coursesBST.root) function inOrderTraverse(node): if node is null: return inOrderTraverse(node.left) print node.course.courseNumber, node.course.courseTitle inOrderTraverse(node.right)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Data Structure | Operation | Line Cost | # Times Executes | Total Cost | Big O Runtime |
| Vector | Sorting | 1 | N log n | N log n | O(n log n) |
|  | Traversal | 1 | n | n |  |
| Hash Table | Transfer to Vector | 1 | n | n | O(n log n) |
|  | Sorting | 1 | N log n | N log n |  |
|  | Traversal | 1 | n | n |  |
| BST | Insertion | 1 | Log n | Log n | O(n) – O(n^2) |
|  | In-Order Traversal | 1 | n | n |  |

**Vector Advantages:**

* **Simple to implement** and manage.
* Sorting is straightforward with built-in functions.

**Disadvantages:**

* Sorting is O(n log n), which is less efficient than the tree structure's in-order traversal.
* Not efficient for frequent insertions and deletions since it requires shifting elements.

**Hash Table** **Advantages:**

* Average O(1) for insertion, deletion, and search.
* Handles large datasets efficiently when key lookups are necessary.

**Disadvantages:**

* Requires conversion to a vector for sorting, which adds overhead.
* Memory overhead due to hashing and potential collisions.

**Binary Search Tree (BST) Advantages:**

* Naturally keeps elements in order if balanced.
* Efficient in-order traversal is O(n) for printing sorted elements.

**Disadvantages:**

* In the worst case, an unbalanced tree could degrade to a linked list with O(n) operations.
* Requires careful management to maintain balance (or using a self-balancing BST)