Amelia Sivick

CS-300-15095-M01 DSA

Professor Lebel

December 22, 2024

Journal: Final Project Part One Run Time and Memory Analysis

Run Time Analysis Tables

Vector

Code	Line Cost	# Time Executes	Total Cost
For all courses	1	n	n
Insert course into vector	1	n	n
For each course in the file	1	n	n
For each prerequisite of the course	1	1	1
For each prerequisite of the course (vector)	1	n	n
Print prerequisite course info	1	n	n
Total Cost	4n + 1		

Runtime: O(n)

Hash Table

Code	Line Cost	# Time Executes	Total Cost
For all courses	1	n	n
Insert course into hash table	1	n	n
For each prerequisite	1	1	1
For each prerequisite lookup	1	n	n
Print prerequisite info	1	n	n
Total Cost	5n + 2		

Runtime: O(n)

Binary Search Tree

Code	Line Cost	# Time Executes	Total Cost
For all courses	1	n	n
Insert course into BST	2	n	2n
For each prerequisite search	1	1	1
For each prerequisite lookup (BST)	1	n	n
Print prerequisite course info (BST)	1	n	n
Total Cost	5n + 3		

Runtime: O(n)

Comparison

Data Structure	Time Complexity (Loading)	Time Complexity (Memory)	Advantages	Disadvantages
Vector	O(n)	O(n)	Ease of access when resizing. Access is efficient O(1). *Simple to implement. *Fast, random access.	Dependent on where insertion occurs for efficiency. Ex) inserting at the end runs quickly and easily O(1), elsewhere requires all elements to shift O(n). Search time is linear, and memory space is taken up easily when resizing. *Slow insertions.
Hash Table	O(n)	O(n)	Efficient with O(1) complexity for searching, inserting, and deleting. Direct hash lookup when finding and storing prerequisites. *Fast insert, search, delete	Extra memory for pointers. Unordered data. Risk of collisions. *Memory intensive *Complexity with collision handling
BST	O(n log n)	O(n)	Sorts list of courses using O(1). Average-case O(n log n) for searching, deleting, and inserting. *Sorted structure, alphanumeric.	The structure must maintain balance and requires memory for pointers. *Worst-case performance if unbalanced.

Recommendation

A Binary Search Tree would overall be the best utilization of a data structure when maintaining an ordered course list. A BST allows for a naturally sorted outcome while maintaining efficient timing for search, insertion, and traversal operations. However, a Hash Table would allow for the fastest search times. This structure can be utilized instead of the BST due to its abilities to quickly lookup and insert. This allows for fast access to course information. A Hash Table also stores data with O(n) requirements, making the storage better for memory capacity, making it more efficient in this aspect compared to utilizing a BST. It should be noted that a Hash Table allows for an increased chance in collisions, especially with large data sets, and may not be as productive as a BST. A Vector would allow for ease of access like dynamic resizing and elements can be removed and added efficiently with few insertions. This structure would not be recommended for this specific instance.