Ryan Reames

CS-300-T5598

6/21/2022

**RUNTIME AND MEMORY EVALUATION**

Runtime and Memory Evaluation for Vector, Hash Table, and Binary Tree (Using <https://www.bigocheatsheet.com/> ) – All worst-case scenarios

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Vector** | **Hash Table** | **Binary Tree** |
| **Access** | O(1) | N/A (no order) | O(n) |
| **Search** | O(n) | O(n) | O(n) |
| **Insertion** | O(n) | O(n) | O(n) |
| **Deletion** | O(n) | O(n) | O(n) |
| **Memory Usage** | O(n) | O(n) | O(n) |

Analysis of getFileData for each structure:

| **Vector getFileData** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| DECLARE string courseNumber, courseName, coursePrereq | 1 | 1 | 1 |
| Declare Vector<string> temp | 1 | 1 | 2 |
| DECLARE Course tempCourse | 1 | 1 | 3 |
| DECLARE Filereader read | 1 | 1 | 4 |
| SET int paramCount = 0 | 1 | 1 | 5 |
| **OPEN csv file with reader** | 1 | 1 | 6 |
| **IF (failure to open csv file THEN)** | 1 | 1 | 7 |
| **THROW error message** | 1 | 1 | 8 |
| **WHILE (reading line and not reaching new line character)** | 1 | N + 1 | N + 9 |
| **WHILE (reading line and reaching ‘,’ delimiter)** | 1 | N + 1 | 2N + 10 |
| **APPEND value prior to ‘,’ to temp** | 1 | N | 3N + 10 |
| **INCREMENT paramCount** | 1 | N | 4N + 10 |
| **APPEND last value prior to new line char to temp** | 1 | N | 5N + 10 |
| **INCREMENT paramCount** | 1 | N | 6N + 10 |
| **IF (paramCount < 2) THEN** | 1 | N | 7N + 10 |
| **THROW error message** | 1 | 1 | 7N + 11 |
| **tempCourse setCourseNumber (PASS IN temp value at index 0)** | 1 | N | 8N + 11 |
| **tempCourse setCourseName (PASS IN temp value at index 1)** | 1 | N | 9N + 11 |
| **IF (paramCount > 2)** | 1 | N | 10N + 11 |
| **FOR all values in temp from index 2 to end** | 1 | N | 11N + 11 |
| **tempCourse setCoursePrereq(string cPrereq)** | 1 | N | 12N + 11 |
| **tempCourse setPrereqCount** | 1 | N | 13N + 11 |
| **SET paramCount = 0** | 1 | N | 14N + 11 |
| **APPEND tempCourse to allCourses** | 1 | N | 15N + 11 |
| **RETURN allCourses** | 1 | 1 | 15N + 12 |
| **Total Cost** | | | 15N + 12 |
| **Runtime** | | | O(n) |
| **Memory** | | | O(n) |

| **HashTable getFileData** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| DECLARE string courseNumber, courseName, coursePrereq | 1 | 1 | 1 |
| Declare Vector<string> temp | 1 | 1 | 2 |
| DECLARE Course tempCourse | 1 | 1 | 3 |
| DECLARE Filereader read | 1 | 1 | 4 |
| SET int paramCount = 0 | 1 | 1 | 5 |
| **OPEN csv file with reader** | 1 | 1 | 6 |
| **IF (failure to open csv file THEN)** | 1 | 1 | 7 |
| **THROW error message** | 1 | 1 | 8 |
| **WHILE (reading line and not reaching new line character)** | 1 | N + 1 | N + 9 |
| **WHILE (reading line and reaching ‘,’ delimiter)** | 1 | N + 1 | 2N + 10 |
| **APPEND value prior to ‘,’ to temp** | 1 | N | 3N + 10 |
| **INCREMENT paramCount** | 1 | N | 4N + 10 |
| **APPEND last value prior to new line char to temp** | 1 | N | 5N + 10 |
| **INCREMENT paramCount** | 1 | N | 6N + 10 |
| **IF (paramCount < 2) THEN** | 1 | N | 7N + 10 |
| **THROW error message** | 1 | 1 | 7N + 11 |
| **tempCourse setCourseNumber (PASS IN temp value at index 0)** | 1 | N | 8N + 11 |
| **tempCourse setCourseName (PASS IN temp value at index 1)** | 1 | N | 9N + 11 |
| **IF (paramCount > 2)** | 1 | N | 10N + 11 |
| **FOR all values in temp from index 2 to end** | 1 | N | 11N + 11 |
| **FOR int i = 0, i < courseNames’ length, INCREMENT i** | 1 | N | 12N + 11 |
| **if temp EQUAL TO courseNames at index i** | 1 | N | 13N + 11 |
| **tempCourse setCoursePrereq(string cPrereq)** | 1 | N | 14N + 11 |
| **break from loop** | 1 | 1 | 14N + 12 |
| **RETURN error message “Prereq course not found”** | 1 | 1 | 14N + 13 |
| **tempCourse setPrereqCount** | 1 | N | 15N + 13 |
| **SET paramCount = 0** | 1 | N | 16N + 13 |
| **nodes Insert(PASS IN tempCourse)** | N | N | 17N + 13 |
| **RETURN allCourses** | 1 | 1 | 17N + 14 |
| **Total Cost** | | | 17N + 14 |
| **Runtime** | | | O(n) |
| **Memory** | | | O(n) |

| **Tree getFileData** | **Line Cost** | **# Times Executes** | **Total Cost** |
| --- | --- | --- | --- |
| DECLARE string courseNumber, courseName, coursePrereq | 1 | 1 | 1 |
| Declare Vector<string> temp | 1 | 1 | 2 |
| DECLARE Course tempCourse | 1 | 1 | 3 |
| DECLARE Filereader read | 1 | 1 | 4 |
| SET int paramCount = 0 | 1 | 1 | 5 |
| **OPEN csv file with reader** | 1 | 1 | 6 |
| **IF (failure to open csv file THEN)** | 1 | 1 | 7 |
| **THROW error message** | 1 | 1 | 8 |
| **WHILE (reading line and not reaching new line character)** | 1 | N + 1 | N + 9 |
| **WHILE (reading line and reaching ‘,’ delimiter)** | 1 | N + 1 | 2N + 10 |
| **APPEND value prior to ‘,’ to temp** | 1 | N | 3N + 10 |
| **INCREMENT paramCount** | 1 | N | 4N + 10 |
| **APPEND last value prior to new line char to temp** | 1 | N | 5N + 10 |
| **INCREMENT paramCount** | 1 | N | 6N + 10 |
| **IF (paramCount < 2) THEN** | 1 | N | 7N + 10 |
| **THROW error message** | 1 | 1 | 7N + 11 |
| **tempCourse setCourseNumber (PASS IN temp value at index 0)** | 1 | N | 8N + 11 |
| **tempCourse setCourseName (PASS IN temp value at index 1)** | 1 | N | 9N + 11 |
| **IF (paramCount > 2)** | 1 | N | 10N + 11 |
| **FOR all values in temp from index 2 to end** | 1 | N | 11N + 11 |
| **FOR int i = 0, i < courseNames’ length, INCREMENT i** | 1 | N | 12N + 11 |
| **if temp EQUAL TO courseNames at index i** | 1 | N | 13N + 11 |
| **tempCourse setCoursePrereq(string cPrereq)** | 1 | N | 14N + 11 |
| **break from loop** | 1 | 1 | 14N + 12 |
| **RETURN error message “Prereq course not found”** | 1 | 1 | 14N + 13 |
| **tempCourse setPrereqCount** | 1 | N | 15N + 13 |
| **SET paramCount = 0** | 1 | N | 16N + 13 |
| **courses Insert(PASS IN tempCourse)** | N | N | 17N + 13 |
| **RETURN allCourses** | 1 | 1 | 17N + 14 |
| **Total Cost** | | | 17N + 14 |
| **Runtime** | | | O(n) |
| **Memory** | | | O(n) |

**Advantages and Disadvantages of Each Structure**

|  |  |  |
| --- | --- | --- |
| **Vector** | **Hash Table** | **Binary Tree** |
| Advantages:  - Able to access data by index, meaning O(1) access  - Simple set up  - Can hold multiple data types | Advantages:  - Good choice of hash key and table size can potentially lead to O(1) search, insertion, and deletion | Advantages:  - Sorted upon creation / insertion  - Quicker average access, search, insertion, and deletion runtime of O(log(n)), assuming fairly balanced tree |
| Disadvantages:  - Insertion / deletion may require moving up to all the remaining values in the vector (O(n) runtime)  - Unsorted vector may take O(n) runtime to find specific data  - Resizing once vector is full may waste memory (ex. only one more element is added, but vector size has been doubled) | Disadvantages:  - No way to run through sorted data in hash table – must use different data structure meaning additional memory is used  - Poorly chosen hash value can lead to O(n) runtime for search, insertion, and deletion due to excessive collisions  - If data is constantly being added, array used for hash table will need to be resized | Disadvantages:  - Poorly chosen root node can mean that tree is just a linked list, losing advantageous average runtime  - Insertion / deletion can require moving significant amount of data |