**GIT Department of Computer Engineering**

**CSE 222/505 – Spring 2020**

**Homework #6 Report**

**Türker Tercan**

**171044032**

**Question 1:**

Shell Sort:

Firstly, initialize the value of the gap to length of the array / 2

Divide the list into smaller sub-list of equal intervals of gap

Sort these sub-lists with insertion sort

Repeat until complete list is sorted

* A -> Comparison: 22 Displacement: 0
* B -> Comparison: 35 Displacement: 24
* C -> Comparison: 43 Displacement: 28
* D -> Comparison: 38 Displacement: 23

Merge Sort:

Starts the dividing arrays from middle

Creates left and right array

While left and right is not ended compares left’s current element to right’s current element

If it is right element is lesser than left element, swap it

Adds remaining elements to main array

Divide array until it is not divisible.

* A -> Comparison: 19 Displacement: 0
* B -> Comparison: 15 Displacement: 15
* C -> Comparison: 31 Displacement: 15
* D -> Comparison: 33 Displacement: 16

Heap Sort:

Starts from the last parent index

Compares it’s children to current root if child is larger than root, swap it to current root and recursively call this.

Recursive runs it until it comes the root

* A -> Comparison: 28 Displacement: 31
* B -> Comparison: 15 Displacement: 22
* C -> Comparison: 30 Displacement: 36
* D -> Comparison: 31 Displacement: 37

Quick Sort:

Start the pivot value to middle of the array.

Start index is 0 and end index is last index

StartIndex of the sub-array and increment forward until we find a value that is > pivotValue

LastIndex of the sub-array and increment backward until we find a value that is < pivotValue

Swap values at the startIndex and endIndex

* A -> Comparison: 43 Displacement: 0
* B -> Comparison: 44 Displacement: 5
* C -> Comparison: 59 Displacement: 8
* D -> Comparison: 66 Displacement: 12

**Question 2:**

MyMergeSort

MyQuickSort

BubbleSort

BookHeapSort

BookInsertionSort

BookMergeSort

BookShellSort

BookSelectionSort

BookQuickSort

**Question 3:**

Problem Solution:

* We need to implement our nested map.
* In outmost map, the author name is used as a key and value is another map.
* Inside of it, Key must be book’s title and value is another map
* Inside of it, Key must be book’s location in the library and value is it’s current status
* So we need 3 nested maps. It’s look like this Map<String, Map<String, Map<String, String>>>
* An administrator can add, delete and change book’s informations
* User can just search library with author’s name or book’s title

Test Cases:

|  |  |  |
| --- | --- | --- |
| Test Subject | Test Number | Pass/Fail |
| Creates admin | T1 | Pass |
| Adds book to system | T2 | Pass |
| Search by author | T3 | Pass |
| Update information | T4 | Pass |
| Delete Book | T5 | Pass |
| Search by title | T6 | Pass |

Running and Results:

Test T1:

Test Data:

LibrarySystem.LibraryAdmin admin = system.createAdmin("password");

Expected: No errors

Result: Passed

Test T2:

Test Data:

admin.addBook("F. Scott Fitzgerald", "The Great Gatsby", "1c1s.1", LibrarySystem.Status.*available*);  
admin.addBook("William Shakespeare", "Hamlet", "1c1s.2", LibrarySystem.Status.*available*);  
admin.addBook("Charlotte Brontë", "Jane Eyre", "1c1s.3", LibrarySystem.Status.*available*);  
admin.addBook("Oscar Wilde", "The Picture of Dorian Gray", "2c1s.3", LibrarySystem.Status.*available*);  
admin.addBook("Geoffrey Chaucer", "The Canterbury Tales", "2c1s.1", LibrarySystem.Status.*available*);  
admin.addBook("Jane Austen", "Pride and Prejudice", "1c2s.1", LibrarySystem.Status.*available*);  
admin.addBook("George Orwell", "Nineteen Eighty-Four", "3c3s.2", LibrarySystem.Status.*available*);  
admin.addBook("Mark Twain", "Adventures of Huckleberry Finn", "2c2s.1", LibrarySystem.Status.*available*);  
admin.addBook("Fyodor Dostoyevsky", "Crime and Punishment", "2c1s.2", LibrarySystem.Status.*available*);  
admin.addBook("William Shakespeare", "Macbeth", "2c1s.4", LibrarySystem.Status.*available*);  
admin.addBook("Mark Twain", "Adventures of Huckleberry Finn", "2c2s.1", LibrarySystem.Status.*not\_available*);  
admin.addBook("William Shakespeare", "Hamlet", "3c3s.1", LibrarySystem.Status.*available*);

Expected: No errors

Result: Passed

Test T3:

Test Data:

admin.searchByAuthor("William Shakespeare");

Expected:

1- William Shakespeare : Hamlet

2- William Shakespeare : Macbeth

Choice:

1

Hamlet - 1c1s.2 - available

Hamlet - 3c3s.1 – available

Result: Passed

Test T4:

Test Data:

admin.updateInformation("Oscar Wilde","The Picture of Dorian Gray","2c1s.3", "3c1s.1", LibrarySystem.Status.*not\_available*);  
admin.updateInformation("Jane Austen", "Pride and Prejudice", "1c2s.1", "2c2s.3", LibrarySystem.Status.*reserved*);  
admin.updateInformation("William Shakespeare", "Macbeth", "2c1s.4", "2c3s.1",LibrarySystem.Status.*available*);

Expected: No error

Result: Passed

Test T5:

Test Data:

admin.deleteBook("Fyodor Dostoyevsky", "Crime and Punishment", "2c1s.2");  
admin.deleteBook("Jane Austen", "Pride and Prejudice", "1c2s.1");

Expected: No error

Result: Passed

Test T6:

user.searchByTitle("Hamlet");

Expected:

William Shakespeare - Hamlet - 1c1s.2 - available

William Shakespeare - Hamlet - 3c3s.1 – available

Result: Passed

**Question 4:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Double Hashing** | **BinaryTreeSearch**  **Chaining** | **Linear**  **Probing** | **LinkedList**  **Chaining** |
| **N = 100000** | **15 ms** | **72 ms** | **39 ms** | **76 ms** |
| **N = 500000** | **155 ms** | **139 ms** | **63 ms** | **144 ms** |
| **N = 1000000** | **372 ms** | **436 ms** | **78 ms** | **415 ms** |