Liam Hulsman-benson

CAB301 Assignment

***An algorithm that you designed for implementing the function “Display the top 10 most frequently borrowed movie DVDs by the members in the descending order of the frequency***

For a function that displayed the top 10 movies two methods were developed. The first method uses a max-heap sort to order the entire array then show the first 10 values. The second method uses an insertion sort on a list of 10 movies that have been previously sorted when they were placed in the array. The array contains at most 10 movies with the list resorted every time a movie is rented. Therefore, there is always a sorted top 10 array available in O(1) time.

The max-heap sort method starts by receiving a flattened array from MovieCollection, containing title and number of times rented for each movie, the tree is flattened using in-order traversal. Before heapify is run on it recursively. The max-heap sort method was selected as it returns the descending order of movies, does not use extra space and has an average and worst runtime of O(n\*log(n)) and can have a best runtime of O(n). Which is an efficient method of find the max values. After the max heap has been made, the first 10 elements are printed, to show the top 10 most popular movies. However, the following method that was developed for a top 10 order is much more efficient in general.

The insertion sort list method is the more efficient method as it prevents a sorting algorithm each time the top 10 list is requested. Instead the array is always available without any modifications when called, meaning a request is only O(1) time. The array is instead updated at the rental time of a movie. When a movie is rented, three options can occur:

* The movie could not already be on the list and not have more rentals than the 10th position, in this case nothing is done.
* The movie could not already be on the list but has more rentals than the 10th position. In this case the 10th movie is replaced with the new movie then the list is sorted using insertion sort.
* The movie could already be on the list, in this case insertion sort is called on the current list to see if the movie just rented can move up in the list, again using insertion sort.

Insertion sort was selected to sort the methods as the list is already ordered besides the movie currently being looked at, this allows for a run time of O(n). which is better than the first methods O(nlog(n)) run time. However it doesn’t correspond to n?

***Analysis of the time complexity of the algorithm***

In the heapify method, appendix A, heapify can be called recursively at most height of tree times, h, as it attempts to find the largest node. In a worst-case scenario this would be at the bottom of the tree, so h times, where h is from the current iterative node. Therefore running time is h\*O(12), O(12) can be simplified to a constant,c. Height can be simplified as well due to h <= log(n). This equality comes from log(the number of nodes in the tree) is equal to the height of the tree. Therefore, the simplified expression for the heapify function is, c\*log(n).

In the none recursive state, appendix C, heapify is called n-3 times, length of array – 1 to i > 1. Therefore overall worst-case runtime of max-heap sorting is (n-3)\*(c\*log(n)), which can be simplified to (n\*log(n)).

*As shown the lectures insertion sort has a runtime of O(n) in almost sorted arrays.*

***Your functional testing results - screenshots for each of the functional tests.***

|  |  |  |  |
| --- | --- | --- | --- |
|  | Test | Before | After |
| 1 | Invalid staff login |  |  |
| 2 | Valid staff login |  |  |
| 3 | Adding 3 movies to library, will be first, 2nd and 3rd movie in the library. Uses display movies from member actions to view all movies. |  |  |
| 4 | Register a member, with name “Liam H” will be first member and password 0000 |  | |
| 5 | Setting password for new member not using 4 digits. |  | |
| 6 | Get liam h number |  |  |
| 7 | Attempting login with not a real member |  |  |
| 8 | Login with real member but wrong password |  |  |
| 9 | Correct member login |  |  |
| 10 | Rent a movie, uses show members rented movies to verify it works. Rented 2 movies. Also shows the time rented has increased when showing all movies to members |  |  |
| 11 | Attempt to rent a movie with no copies or not a valid title |  | |
| 12 | Attempting to rent a movie that you already have | Refer to after pictures of test 10. |  |
| 13 | Return a movie you have | Refer to after pictures of test 12. |  |
| 14 | Return a movie you don’t have. | Refer to after pictures of test 13. |  |
| 15 | Rent movie 5 to make it most rented. |  |  |
| 16 | Rent and return movie 2 twice to make it most rented | Refer to after of test 15. |  |
| 17 | Most popular when no movies have been rented. Have to restart console to get this output. |  |  |

# Appendix

## A

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 | w |
| 7 |  |
| 8 |  |
| 9 |  |
|  |  |

## B

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 | *Temp* |
| 16 |  |
| 17 |  |
| 18 |  |

## C

|  |  |
| --- | --- |
| 1 |  |
| 2 |  |
|  |  |
|  |  |