CS 3310 Data and File Structure

Instructor Dr.Ajay K. Gupta

Name: Mariam Ghali

**Sorting based on Array and Linked list**

**Phase 1: objective**

1. The main goal of this assignment is to see the space and time complexity of sorting based on two different implementations
2. To practice bubble sort, insert sort, selection sort and merge sort.
3. The effect of the data structure
   1. The program should do the following:
      1. Generate N random lower case letters and store them in a linked-list and array
      2. Implement the four different kinds of sort such as : Bubble sort, insertion sort, selection sort, and marge sort
      3. For the insertion sort, we should use the binary search to avoid the higher number of the comparison
      4. Sort based on the letters of the last name then the alphabetical order
4. **Phase 2: Description**

The program consists of 4 classes:

1. The main method that will handle reading the input from the user, create and print to an output file, create a random lower-case character and save them in array and linkedlist, then do the sorting in the array based implementation. The three-different sorting: Bubble sort, insertion sort, selection sort. And a method for the specific way we want to sort. Instead of sorting alphabetically, sort it with my last name then alphabetically. And it will get the time consumer for each method.
2. Class for the merge sort that will take the array, first and last element as a parameters, make a sort method in the class that will recricevely spilt the array until it reach only one element and then call the merge sort in the same class what will compare and merge the right and left subarrays together
3. The charNode class that will create the node and the values of the linked list
4. The charLinkedList will create the linked list by having the insert method and it will sort in the 4 different methods. Bubble, insertion, selection and merge sort
5. **Pseudocode**
6. Main method

Open file

Write to file

Bubblesort()

Declare integer I to zero

Declare integr J to zero

Declare a char temp

For each time going through the loop

For each 2 values nect ot each other

IF the keys are in the wrong order based of the sort method

Swap the keys

END OF IF

Return the sorted array

Insertion sort()

Declare integer I to zero

Declare integr J to zero

Declare a char temp

For each time we go through the array or compare to elements

IF the keys are in the wrong order based of the sort method

Call binary seach method

For each number of position we got

Shift the elements

END OF IF

Return the sorted array

binarySearch()

Declare a mid point by adding the low and the high and divide by 2

IF low greater than or equal high

Return Call the sort function based of the new position

IF the item is equal to mid

Return the middle plus one

IF item is greater than value of the mide point

Return the value form the call function

ENF IF

Selection sort()

Declare integer I to zero

Declare integr J to zero

Declare a char temp

For each time we go through the loop “outer loop”

Declear the unsorted index

For each time we compare the the first elemnt with array

If the min index is greater that the second element

Swap the min element with the frst one

END OF IF

Return the sorted array

Sort()

If x > y

If x is special char

If y is special char

If special char x is greater than spical char y

Return true

Return false

Return false

Return true

END OF IF

ELSE

If y is special char

If y is special char

If special char y is greater than special char x

Return false

Return true

Return true

Return false

END OF IT

SPEICAL char()

Declare index equal to zero

Declare char array with the last name

For each time go through the array of the last name

If the letter of x and y is similar to the letter of the last name

Return the index

END OF IT

Return -1

Close the buffer

Close the file

1. In the charNode class

CharNode

Value

Next

1. The merger sort class

Sort()

Declare middle to zero

If left less that the right

Middle equal to the addition of left and right divide by 2

Recursive call for the left subarray

Recursive call for the right subarray

Merge sort the sorted halves

ENF OF IF

Return the sorted array

Merge()

Find the size of the two subarray

Crete an array for the each subarray

Copy the data to the left and the right subarray

Initial index of the merged subarray

WHILE The index of both left and right subarray are less than the size

If the first element of the left subarray greater than the element of right

Copy the value to the left

ELSE

Copy to the right

END OF WHILE LOOP

1. Linked list

Insert

IF head == null

Create a node called temp

Make head equal temp

Make tail equal temp

ELSE

Temp equal head

WHILE true

IF temp.next is null

Create a node for the next addition or next value

Set tail to the next temp node

Break at the end of the list

ELSE

Set temp to temp next node

binarySearch()

Declare a mid point by adding the low and the high and divide by 2

IF low greater than or equal high

Return Call the sort function based of the new position

IF the item is equal to mid

Return the middle plus one

IF item is greater than value of the mide point

Return the value form the call function

ENF IF

Selection sort()

Declare integer I to zero

Declare integr J to zero

Declare a node temp

For each time we go through the loop “outer loop”

Declear the unsorted index

For each time we compare the the first elemnt with array

If the min index is greater that the second element

Swap the min element with the frst one

END OF IF

Return the sorted array

Merge sort()

Call the merge sort from the class merge

**Thermotical analysis**

In the array implementation for the sorting. Arrays takes time complexity of order o(n) to insert all it is values where n is the number of the values. While linked list takes o(1) for insertion one at a time it depends on the number of the values we are going to insert.

In the ARRAY implementation, for the bubble sort it takes o(n^2) to sort the whole array. Since we have double for loop one for comparing each 2 elements next to each other and the other to continue comparing until it is sorted. For the insertion sort, it usually takes o(n^2) like bubble sort because we compare then search for the right place to save the value. But since we are using binary search instead of nested loop. Binary search will take o(log n) and the loop that go through the array take o(n). so the time complexity for the insertion search for array is o(nlogn)

For the selection search, it takes o(n^2) because it has to loop twice. One for compare the max or the min index “depends on the way we implemented” and the second one where is the right index should be. The merge sort will take (nlogn) and it appears of the best sort for the array implementation. N for go through the loop and log n because of when we merge the subarrays back to be one array.

In the linked list. It takes o(1) to insert of delete and element form the last. In the bubble sort will take also o(n^2) similar to the array because it is the same process and we have to go to each node twice, in the selection sort it takes also o(n^2) . n number of irritation and n number of comparisons. For the insertion sort it will take o(n log n) because of the binary search. And finally merge sort and it will be the best or “second preferable sort in this project ” of sorting a linked list because it takes o(n logn)

Sorting and special sort that sort the letters takes o(n) because it compares each char of the array with the char of the last name. The way that special char and sort works is it takes the letters of th array or the linked list and compare it with the last name. it makes priority to the char of the last name before the alphabetical order.

**The empirical analyses**

It agrees with the thermotical analysis by showing that the graph of the bubble sort of the array implementation is the highest values of the graph then the selection sort is the second highest values. Then the insertion sort and merge sort are kind of similar because both take o(n logn)