GIT Department of Computer Engineering
CSE 222/505 - Spring 2023
Homework # Report 6

Hamza KONAÇ
210104004202

1. SYSTEM REQUIREMENTS

This assignment is contunie of the homework6. Additionally features of the this assignment is the new classes that are perform insertion , selection,bubble and quick sort.

2. USE CASE AND CLASS DIAGRAMS

3. PROBLEM SOLUTION APPROACH

T(n) complexities of the algorithms:

Merge sort : O(n*logn)

- ⇨ For all cases merge sort's time complexities always be nlogn . it doesn't change for the best worst or average.

- ⇨ **Divide** step computes mid position of sub-array and it takes constant time O(1).
- ⇨ **Conquer** step recursively sort two sub arrays of approx n/2 elements each.
- ⇨ **Combine** step merges a total of n elements at each pass requiring at most n comparisons so it take O(n).
- ⇨ The algorithm requires approx logn passes to sort an array of n elements and so total time complexity is nlogn.

Insertion sort:

Best->O(n)

Average->O(n^2)

Worst->O(n^2)

→The insertion step is performed n − 1 times. In the worst case, all elements in the sorted subarray are compared to next value for each insertion, so the maximum number of comparisons is represented by the series 1+2+3 + . . .(n-2)+(n-1) which is O(n2)

Selection sort:

→Selection sort find smallest item in the subarray. Then swap value tu current index value's . Perform this for all values .so that T(n) will be Θ(n^2)

Buble sort:

→This sort algorithm shift element through last of the array.by use this algorithm biggest elemnt go to last index.if there is no change sort has done.Therefore best case will be O(n) and worst case will be O(n^2)

Quick sort:

→This algorithm split array 2 part and take a pivot value. Then by using this pivaot value get together the value that smaller than the pivot and get together the bigger value than the pivot.if array is already sorted this will be worst case for the quick sort. And if the left value's of the subarray always smaller than the right value's this will be best case.

Best case->nlogn

Average case-> nlogn

Worst case ->n^2

4. TEST CASES

Input→ “a ss ddd ffff ggggg hhhhhh jjjjjj kkkkkkk lllllllll”
Input →”aaaaaaaaa sssssss ddddddd ffffff gggggg hhhhh jjjj kkk ll i”
Input →”aaaaa ssss ddd ff g zzzzzz hhhhhhh jjjjjjj kkkkkkkkk lllllllll”
Input →”aaa sssssss ww oooo pppppppp q bbbbbb zzzzzzz ttttt”
Input →”aaaaaaaa sss d fffff gggggggggg ooooooo”

5. RUNNING AND RESULTS

Input→ “a ss ddd ffff ggggg hhhhhh jjjjjj kkkkkkk lllllllll”

```
The original (unsorted) map:
Letter: a - Count: 1 Words: [a]
Letter: s - Count: 2 Words: [ss,ss]
Letter: d - Count: 3 Words: [ddd,ddd,ddd]
Letter: f - Count: 4 Words: [ffff,ffff,ffff,ffff]
Letter: g - Count: 5 Words: [ggggg,ggggg,ggggg,ggggg,ggggg]
Letter: h - Count: 6 Words: [hhhhhh,hhhhhh,hhhhhh,hhhhhh,hhhhhh,hhhhhh]
Letter: j - Count: 7 Words: [jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj]
Letter: k - Count: 8 Words: [kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk]
Letter: l - Count: 9 Words: [lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll]


The sorted map:
Time taken to sort map  MERGE: 0.0014562
Time taken to sort map SELECTION: 0.0014421
Time taken to sort map INSERTION: 0.0012897
Time taken to sort map BUBBLE: 0.0012937
Time taken to sort map QUICK: 0.0018345
```

->for sorted arrays insertion and bubble sort’s time complexity is O(n) . Therefore insertion sort and bubble sort is the fastest. Selection sort T(n) is always O(n^2) but array size so small so that it dont give us riht runtime .quick sort’s worst case is sorted array therefore it run time is bigger than others although it’s complexity equals to O(nlogn)

Input→ “lllllllll kkkkkkkk jjjjjj hhhhhh ggggg ffff ddd ss a”

```
The original (unsorted) map:
Letter: l - Count: 9 Words: [lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll,lllllllll]
Letter: k - Count: 8 Words: [kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk,kkkkkkkk]
Letter: j - Count: 7 Words: [jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj,jjjjjjj]
Letter: h - Count: 6 Words: [hhhhhh,hhhhhh,hhhhhh,hhhhhh,hhhhhh,hhhhhh]
Letter: g - Count: 5 Words: [ggggg,ggggg,ggggg,ggggg,ggggg]
Letter: f - Count: 4 Words: [ffff,ffff,ffff,ffff]
Letter: d - Count: 3 Words: [ddd,ddd,ddd]
Letter: s - Count: 2 Words: [ss,ss]
Letter: a - Count: 1 Words: [a]


The sorted map:
Time taken to sort map  MERGE: 0.0014726
Time taken to sort map SELECTION: 0.0024376
Time taken to sort map INSERTION: 0.0015467
Time taken to sort map BUBBLE: 0.0013942
Time taken to sort map QUICK: 0.0020636
```

→this input worst case for all algorithms. But merge sort and selection sort has no worst case therefore there is no effect on their run time. Quick sort and insertion sort’s worst case is reverse sorted array’s . because of that rum time increased for quick sort ,insertion sort .

Input →”aaaaa ssss ddd ff g zzzzzz hhhhhhh jjjjjjj kkkkkkkkk lllllllll”

```
The original (unsorted) map:
Letter: a - Count: 5 Words: [aaaaa,aaaaa,aaaaa,aaaaa,aaaaa]
Letter: s - Count: 4 Words: [ssss,ssss,ssss,ssss]
Letter: d - Count: 3 Words: [ddd,ddd,ddd]
Letter: f - Count: 2 Words: [ff,ff]
Letter: g - Count: 1 Words: [g]
Letter: z - Count: 6 Words: [zzzzzz,zzzzzz,zzzzzz,zzzzzz,zzzzzz,zzzzzz]
Letter: h - Count: 7 Words: [hhhhhhh,hhhhhhh,hhhhhhh,hhhhhhh,hhhhhhh,hhhhhhh,hhhhhhh]
Letter: j - Count: 8 Words: [jjjjjjjj,jjjjjjjj,jjjjjjjj,jjjjjjjj,jjjjjjjj,jjjjjjjj,jjjjjjjj,jjjjjjjj]
Letter: k - Count: 9 Words: [kkkkkkkkk,kkkkkkkkk,kkkkkkkkk,kkkkkkkkk,kkkkkkkkk,kkkkkkkkk,kkkkkkkkk,kkkkkkkkk,kkkkkkkkk]
Letter: l - Count: 10 Words: [lllllllllll,lllllllllll,lllllllllll,lllllllllll,lllllllllll,lllllllllll,lllllllllll,lllllllllll,lllllllllll,lllllllllll]


The sorted map:
Time taken to sort map  MERGE: 0.0014968
Time taken to sort map SELECTION: 0.0015715
Time taken to sort map INSERTION: 0.0014073
Time taken to sort map BUBBLE: 0.0010783
Time taken to sort map QUICK: 0.0012203
```

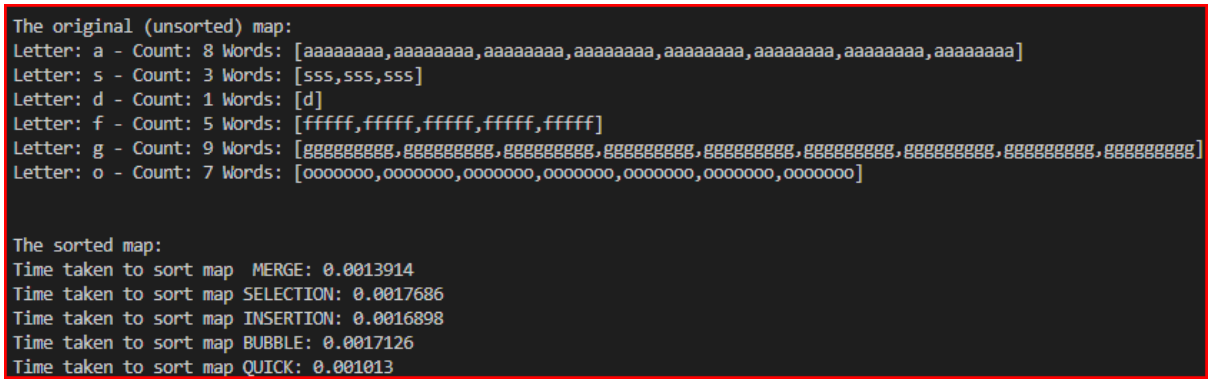
Input →”aaa sssssss ww oooo pppppppp q bbbbbb zzzzzzz ttttt”

```
The original (unsorted) map:
Letter: a - Count: 3 Words: [aaa,aaa,aaa]
Letter: s - Count: 7 Words: [sssssss,sssssss,sssssss,sssssss,sssssss,sssssss,sssssss]
Letter: w - Count: 2 Words: [ww,ww]
Letter: o - Count: 4 Words: [oooo,oooo,oooo,oooo]
Letter: p - Count: 8 Words: [pppppppp,pppppppp,pppppppp,pppppppp,pppppppp,pppppppp,pppppppp,pppppppp]
Letter: q - Count: 1 Words: [q]
Letter: b - Count: 6 Words: [bbbbbb,bbbbbb,bbbbbb,bbbbbb,bbbbbb,bbbbbb]
Letter: z - Count: 8 Words: [zzzzzzzz,zzzzzzzz,zzzzzzzz,zzzzzzzz,zzzzzzzz,zzzzzzzz,zzzzzzzz,zzzzzzzz]
Letter: t - Count: 6 Words: [tttttt,tttttt,tttttt,tttttt,tttttt,tttttt]


The sorted map:
Time taken to sort map  MERGE: 0.0033677
Time taken to sort map SELECTION: 0.0025712
Time taken to sort map INSERTION: 0.001529
Time taken to sort map BUBBLE: 0.0026033
Time taken to sort map QUICK: 0.0034104
```

→this sample shows us average time complexity of the all algorithm. There for results are close to each other

Input →”aaaaaaaa sss d fffff gggggggggg ooooooo”



→This sample ınput is the best case of the quick sort . So that it has fastest runtime. But ınput is not worst case for other algorithm except the merge and selection sort thay have same complexity for all situation.

==========

→For sorting the values some algorithms shift value other swap the values. Bubble sort, quick sort and selection sort swap the values.But there is important point in here . if the values are same in the array , buble sort doesn't change values adding order ,it skip this step. But quick sort and selection sort always swap the values even if they are be equal.therefore this situation create worst case for quick sort in my algorithm because of all values in the array always be element of the right or left subarray . this create worst case for quick sort algoritm.

İnsertion sort shift the values instead of the swap the values . Therefore best case occur if all values are equal in the array