GTU Department of Computer Engineering CSE 222/505 - Spring 2020
Homework 06 Report

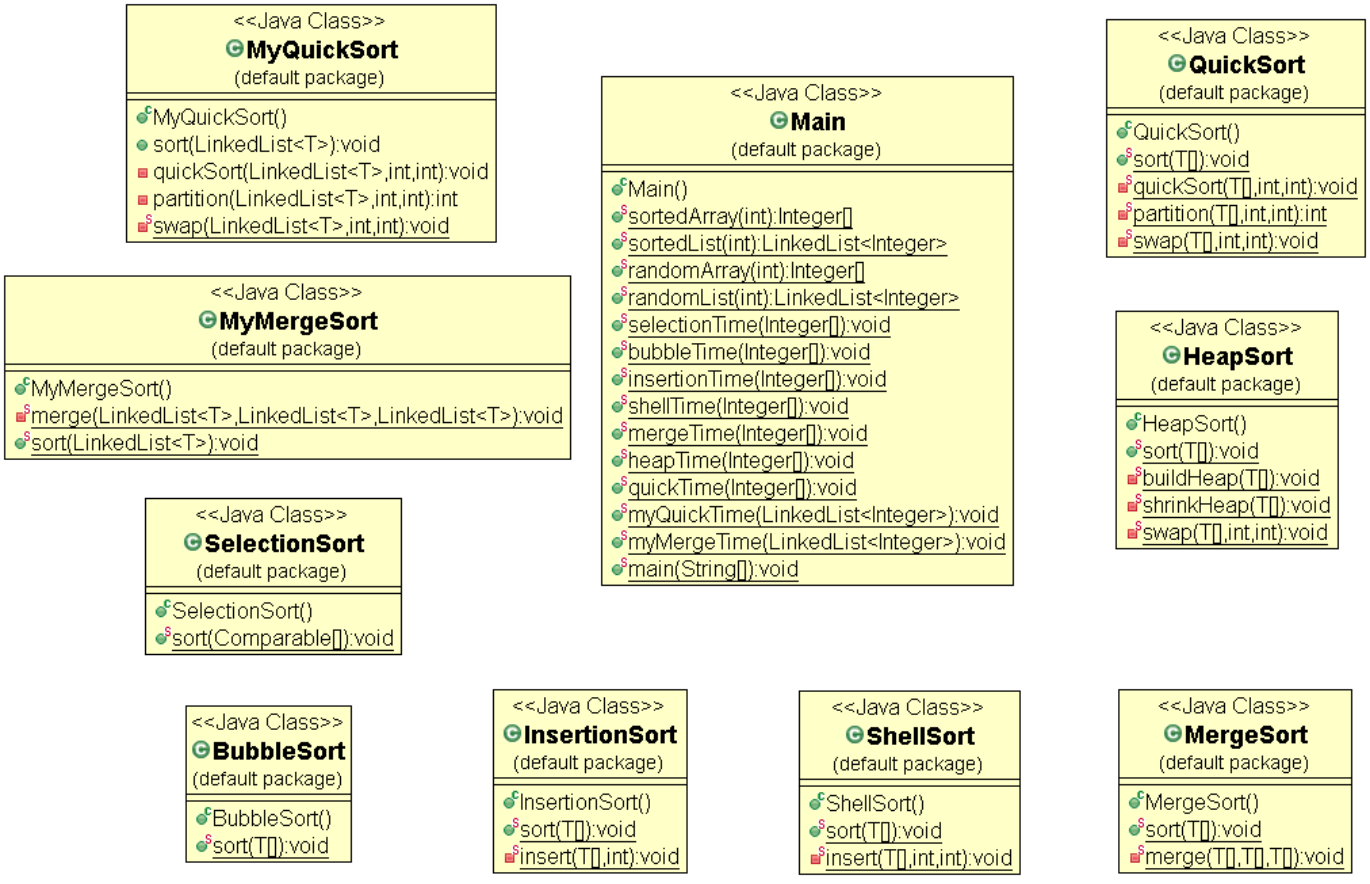
ESRA ERYILMAZ
171044046

- Q1 -

Q1 is in the 171044046.pdf file.

- Q2 -

1. CLASS DIAGRAMS



2. PROBLEM SOLUTION APPROACH

Merge sort is a recursive algorithm that continually splits a list in half. If the list is empty or has one item, it is sorted by definition (the base case). If the list has more than one item, we split the list and recursively invoke a merge sort on both halves. Once the two halves are sorted, the fundamental operation, called a merge, is performed. Merging is the process of taking two smaller sorted lists and combining them together into a single, sorted, new list.
While implementing merge sort I used linked list to keep data,
and also for traversing the list I used List Iterator.

A quick sort first selects a value, which is called the pivot value. Although there are many different ways to choose the pivot value, I choose the first item in the list. The role of the pivot value is to assist with splitting the list. The actual position where the pivot value belongs in the final sorted list, commonly called the split point, will be used to divide the list for subsequent calls to the quick sort.

The key process in quickSort is partition(). Target of partitions is, given a linked list and an element pivot, put pivot at its correct position in sorted linked list and put all smaller elements (smaller than pivot) before pivot, and put all greater elements (greater than pivot) after pivot.
While implementing quick sort I used linked list to keep data.
Since,I constantly use the get() method of the linked list which time complexity is O(n) ; It takes lots of time to compile it.

Merge Sort and QuickSort is a Divide and Conquer algorithm.

3. TEST CASES

Test Case ID	Test Method	Test Input	Test Output	Pass/Fail
T1	myQuick - sort() method	Linked list	void	Pass
T2	myMerge - sort() method	Linked list	void	Pass
T3	selection - sort() method	Array	void	Pass
T4	bubble - sort() method	Array	void	Pass
T5	insertion - sort() method	Array	void	Pass
T6	shell - sort() method	Array	void	Pass
T7	merge - sort() method	Array	void	Pass
T8	heap - sort() method	Array	void	Pass
T9	quick - sort() method	Array	void	Pass

I implement T1 and T2 methods.
Other tests methods in the book.

!! For quick sort implementation in the book and my implementation ; sorted array or linked list gives stack overflow error. So for quick sort I add that "If array/linked list is sorted than return without compare it". !!

4. RUNNING AND RESULTS

I ran the code according to the array/linked list size.
The numbers represent run times. (in milliseconds)

```
              TEST STARTING...
--------------------------------------------------------------------------

****** FOR RANDOM ******

                ____SIZE 10000____
       MyQuick|MyMerge|Select|Bubble|Insertion|Shell| Merge | Heap | Quick
Test 1) 3234    38      230     354     58      7       3       15      3
Test 2) 3348    9       199     324     40      2       1       2       9
Test 3) 3184    12      189     366     39      1       1       2       2
Test 4) 3046    11      188     369     38      1       1       1       1
Test 5) 3154    12      188     363     38      1       2       1       1
Test 6) 3267    12      189     366     39      1       1       1       1
Test 7) 3212    13      189     366     38      1       1       1       1
Test 8) 3086    14      188     366     38      1       8       1       1
Test 9) 3142    15      192     363     39      1       1       1       1
Test 10) 3102   15      191     376     39      1       1       1       1
Test 11) 3119   15      188     364     38      1       1       1       1
Test 12) 3153   14      189     367     45      1       1       1       1
Test 13) 3073   14      188     368     44      1       4       1       1
Test 14) 3103   12      188     366     44      1       2       1       1
Test 15) 3149   12      188     368     44      1       1       1       1
Test 16) 3184   12      189     367     47      1       1       1       1
Test 17) 3252   11      188     366     45      1       1       1       1
Test 18) 3253   17      190     367     45      1       1       1       1
Test 19) 3178   18      189     368     45      1       1       1       1
Test 20) 3105   15      189     369     45      1       1       1       1
--------------------------------------------------------------------------
                ____SIZE 40000____
       MyQuick|MyMerge|Select|Bubble|Insertion|Shell| Merge | Heap | Quick
Test 1) 74885   120     3026    6650    1047    8       7       8       4
Test 2) 71812   117     3028    6632    1043    8       7       8       4
Test 3) 76849   145     3025    6655    1063    8       7       8       5
Test 4) 53376   147     3027    6649    1083    8       7       8       5
Test 5) 54054   167     3147    7225    1133    9       7       8       5
Test 6) 74533   86      3068    6666    1074    8       176     8       4
Test 7) 76699   73      3046    6640    1071    8       7       8       4
Test 8) 73604   79      3029    6640    1071    8       7       8       5
Test 9) 72042   74      3028    6617    1037    8       7       8       4
Test 10) 72919  77      3028    6624    1048    8       7       8       5
Test 11) 76598  54      3034    6729    1085    8       7       8       5
Test 12) 77290  61      3027    6740    1072    8       7       8       5
Test 13) 76632  73      3207    6896    1085    8       7       8       4
Test 14) 84326  53      3051    6688    1062    8       8       8       4
Test 15) 76223  101     3504    7864    1402    11      10      8       6
Test 16) 75640  78      3033    6689    1063    8       7       8       4
Test 17) 77142  49      3039    6666    1076    8       7       8       5
Test 18) 79927  65      3117    7066    1105    8       7       8       5
Test 19) 76788  75      3053    6696    1106    8       7       8       5
Test 20) 74393  72      3030    6613    1056    8       7       8       5
--------------------------------------------------------------------------
--------------------------------------------------------------------------
                ____SIZE 100000____
       MyQuick|MyMerge|Select|Bubble|Insertion|Shell| Merge | Heap | Quick
Test 1) 622499  240     19069   40851   7601    24      19      24      13
Test 2) 611074  386     19286   41505   7548    24      20      24      13
Test 3) 605063  360     19985   41944   7672    24      19      24      12
Test 4) 645886  459     19718   41168   7520    24      20      23      12
Test 5) 576448  419     18933   40406   7440    25      19      23      13
Test 6) 695607  854     19796   43604   8648    26      24      25      14
Test 7) 707409  243     19930   40346   7490    24      19      23      13
Test 8) 600272  280     21470   40710   7637    24      20      24      13
Test 9) 612751  262     19089   41323   7671    24      20      24      13
Test 10) 553391 231     19009   40491   7544    24      21      24      13
Test 11) 594865 243     19084   40815   7667    24      20      24      13
Test 12) 609192 237     19552   41479   7582    24      21      24      13
Test 13) 617261 252     19170   41781   7594    24      21      26      13
Test 14) 569345 168     19441   40902   7480    24      21      24      13
Test 15) 681054 198     21522   46022   8511    27      24      28      15
Test 16) 667466 672     21532   44932   7608    38      33      60      74
Test 17) 626671 456     23512   55244   10139   78      52      59      24
Test 18) 944483 661     25536   59772   11488   48      53      52      24
Test 19) 887291 357     25662   58983   10451   31      623     38      18
Test 20) 858367 209     28062   54962   9614    27      24      31      15
--------------------------------------------------------------------------
```

For 150.000 and 180.000 size , Quick sort method takes hours for just one random.

So for the myQuick sort method, I compile for one random to see what happens. Then I removed myQuick sort method and compile all the remaining methods.

```
                ____SIZE 150000____
        MyQuick | MyMerge|Select|Bubble|Insertion|Shell| Merge | Heap | Quick
Test 1) 2532802 1366    62983   128206  22789   77      55      84      45
Test 2)          1096   65325   135247  21316   79      882     86      79
Test 3)          535    57686   125005  21110   67      51      69      26
Test 4)          500    58354   129389  32096   65      66      68      27
Test 5)          469    68381   149865  31823   62      131     64      27
Test 6)          501    64638   140391  28433   61      42      60      26
Test 7)          543    70916   155794  61408   60      126     56      26
Test 8)          410    64901   135563  26122   74      45      66      27
Test 9)          464    63223   139864  27391   63      49      87      30
Test 10)         463    66631   140246  28532   58      45      62      26
Test 11)         480    65720   140705  25874   64      49      81      61
Test 12)         534    78736   176885  30325   62      57      69      29
Test 13)         421    65529   143744  28220   61      48      64      28
Test 14)         473    64646   143233  28096   65      49      68      27
Test 15)         448    65004   145084  27479   60      43      68      30
Test 16)         480    65891   314694  24958   64      50      89      28
Test 17)         494    69737   146707  27472   57      43      59      26
Test 18)         396    68220   153692  28521   59      46      75      28
Test 19)         469    62018   145013  27491   65      49      63      27
Test 20)         485    65912   144026  25768   58      43      72      27
-------------------------------------------------------------------------
```

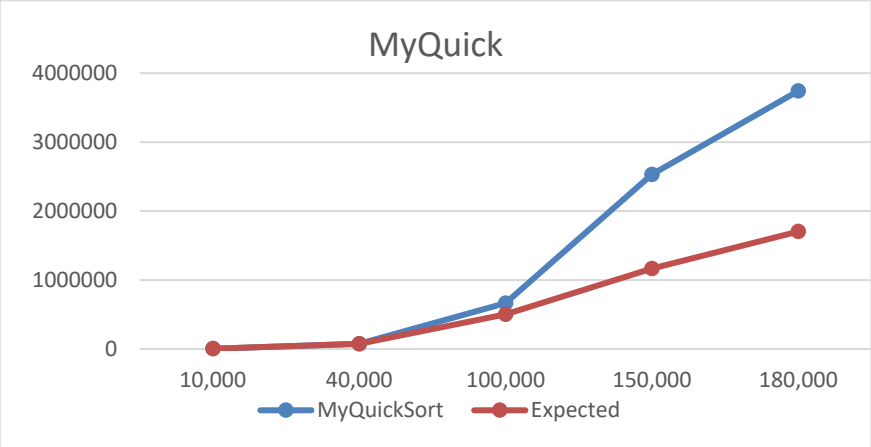
```
                ____SIZE 180000____
        MyQuick | MyMerge|Select|Bubble|Insertion|Shell| Merge | Heap | Quick
Test 1) 3741671 1730    65522   123343  24759   64      61      74      62
Test 2)          557    100466  211238  45832   159     63      115     57
Test 3)          764    99358   212088  38995   77      54      74      32
Test 4)          1908   107236  219261  45081   98      59      85      41
Test 5)          768    105664  237792  43199   102     62      75      34
Test 6)          695    108930  229247  53837   84      60      75      34
Test 7)          830    113543  239861  57593   79      60      74      34
Test 8)          697    110602  257772  45660   82      55      81      34
Test 9)          614    101547  205270  37723   75      486     76      33
Test 10)         1693   99314   206333  52724   80      58      74      33
Test 11)         1523   106763  199267  34849   57      42      101     61
Test 12)         1060   79954   165436  37028   57      39      52      23
Test 13)         1516   71469   155628  33907   59      39      52      23
Test 14)         1198   75557   187125  52532   86      44      61      27
Test 15)         894    77794   161528  31278   53      41      52      23
Test 16)         1484   78767   164010  35940   68      42      65      26
Test 17)         1206   88375   187367  37039   65      41      60      24
Test 18)         1200   84995   171011  38415   68      43      54      23
Test 19)         1096   80103   178901  42355   74      43      59      25
Test 20)         1279   84628   189599  35422   67      48      67      26
-------------------------------------------------------------------------
```

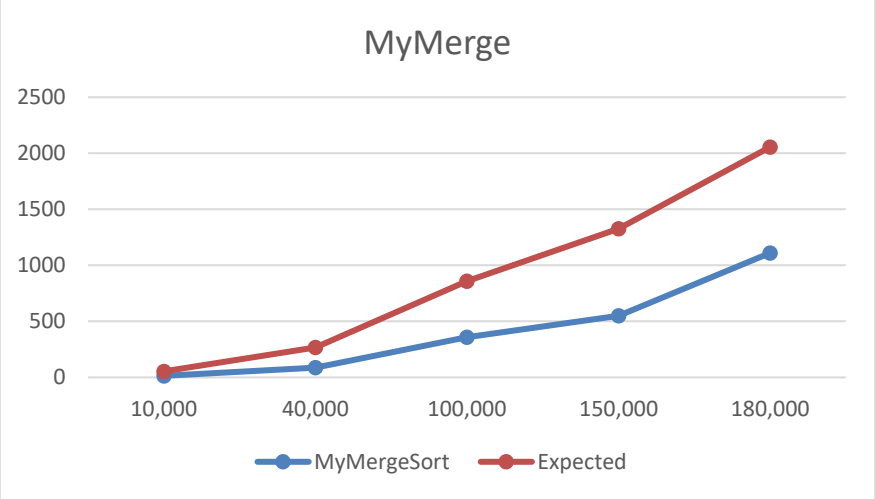
Graphs

For expected run time: (average)

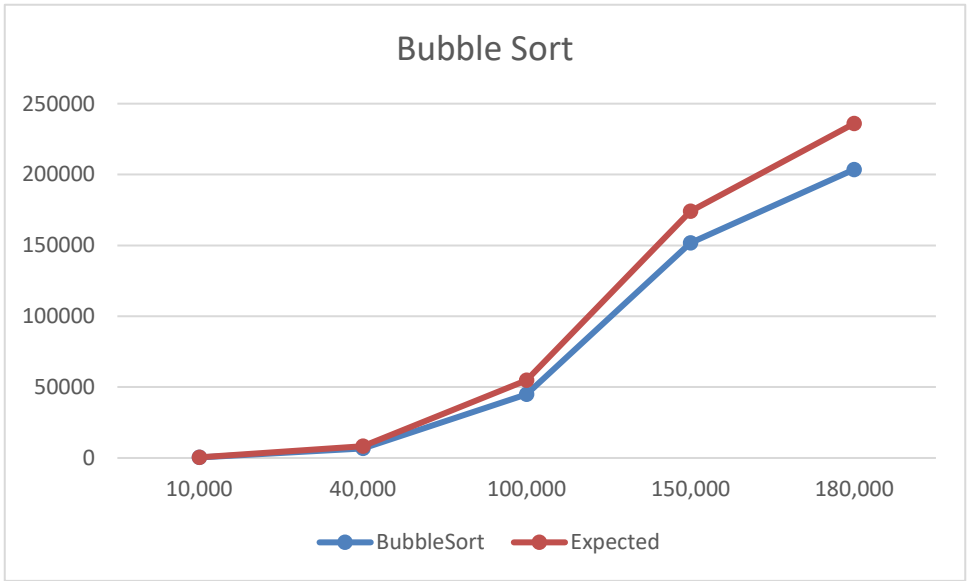
MyQuickSort : O(n^2logn)
MyMergeSort : O(nlogn)
SelectionSort : O(n^2)
BubbleSort : O(n^2)
InsertionSort : O(n^2)
ShellSort : O(n(logn)^2)
MergeSort : O(nlogn)
HeapSort : O(nlogn)
QuickSort : O(nlogn)

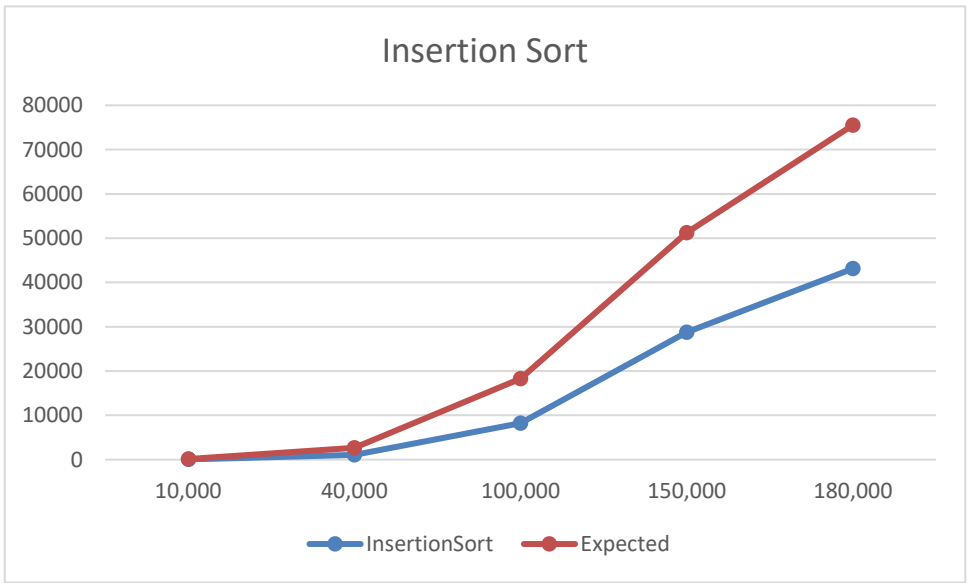
x -> size of array/linked list
y -> run time(in milliseconds)

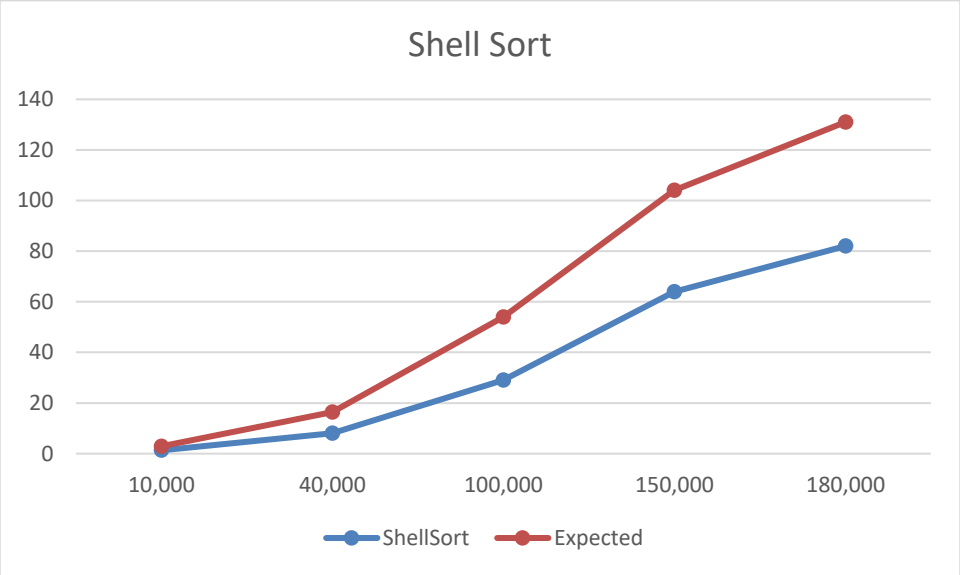


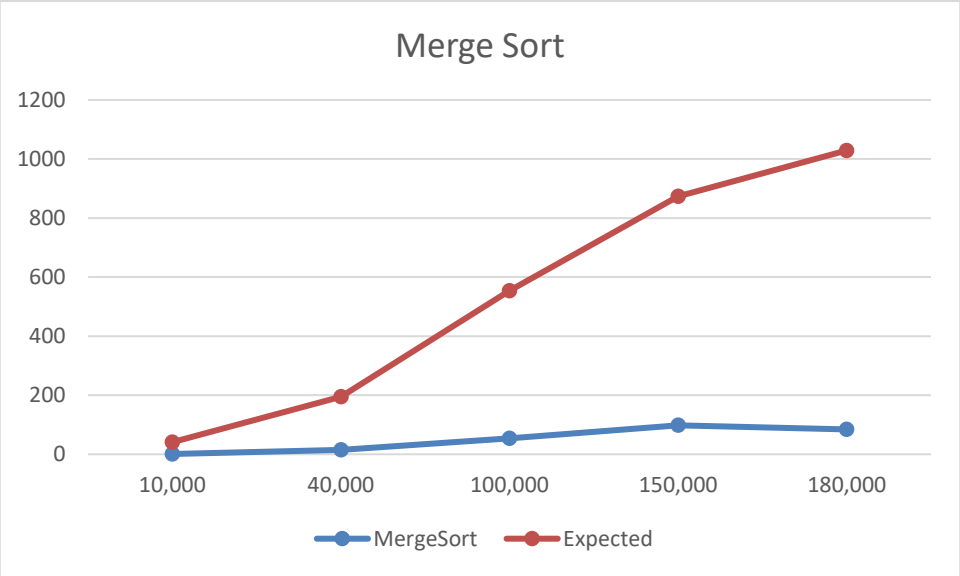


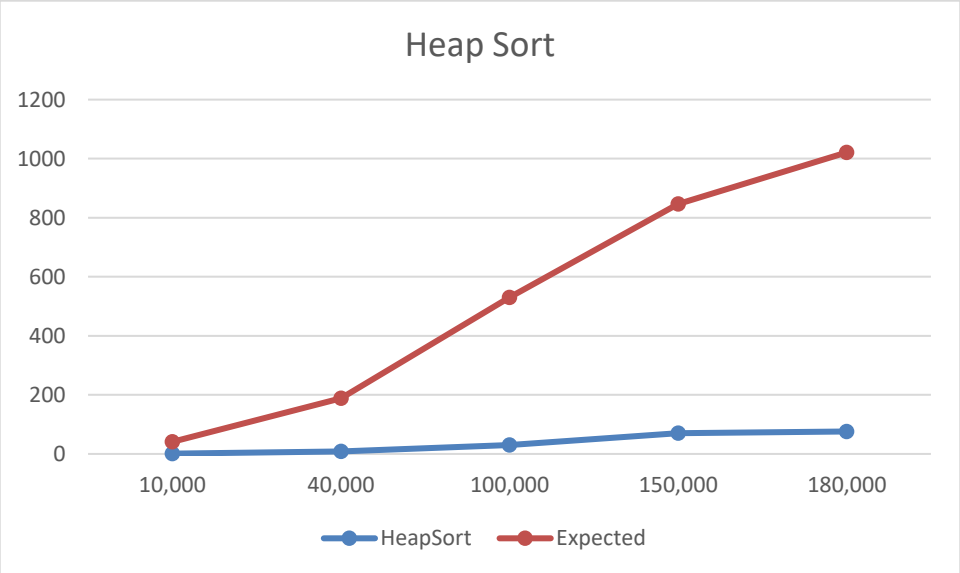

Selection Sort
140000
120000
100000
80000
60000
40000
20000
0
10,000
40,000
100,000
150,000
180,000
SelectionSort
Expected

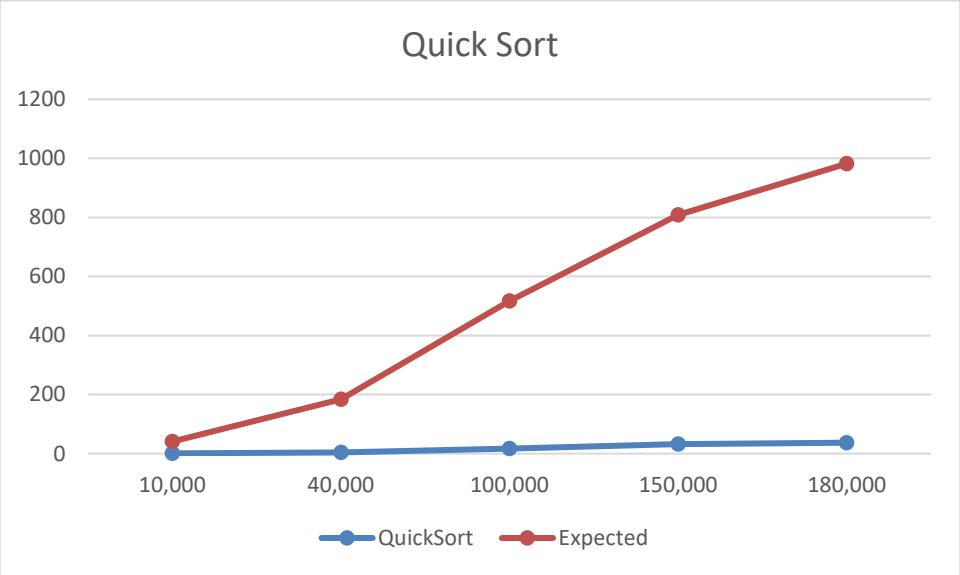

Bubble Sort
250000
200000
150000
100000
50000
0
10,000
40,000
100,000
150,000
180,000
BubbleSort
Expected


Insertion Sort
80000
70000
60000
50000
40000
30000
20000
10000
0
10,000
40,000
100,000
150,000
180,000
InsertionSort
Expected


Shell Sort
140
120
100
80
60
40
20
0
10,000
40,000
100,000
150,000
180,000
ShellSort
Expected

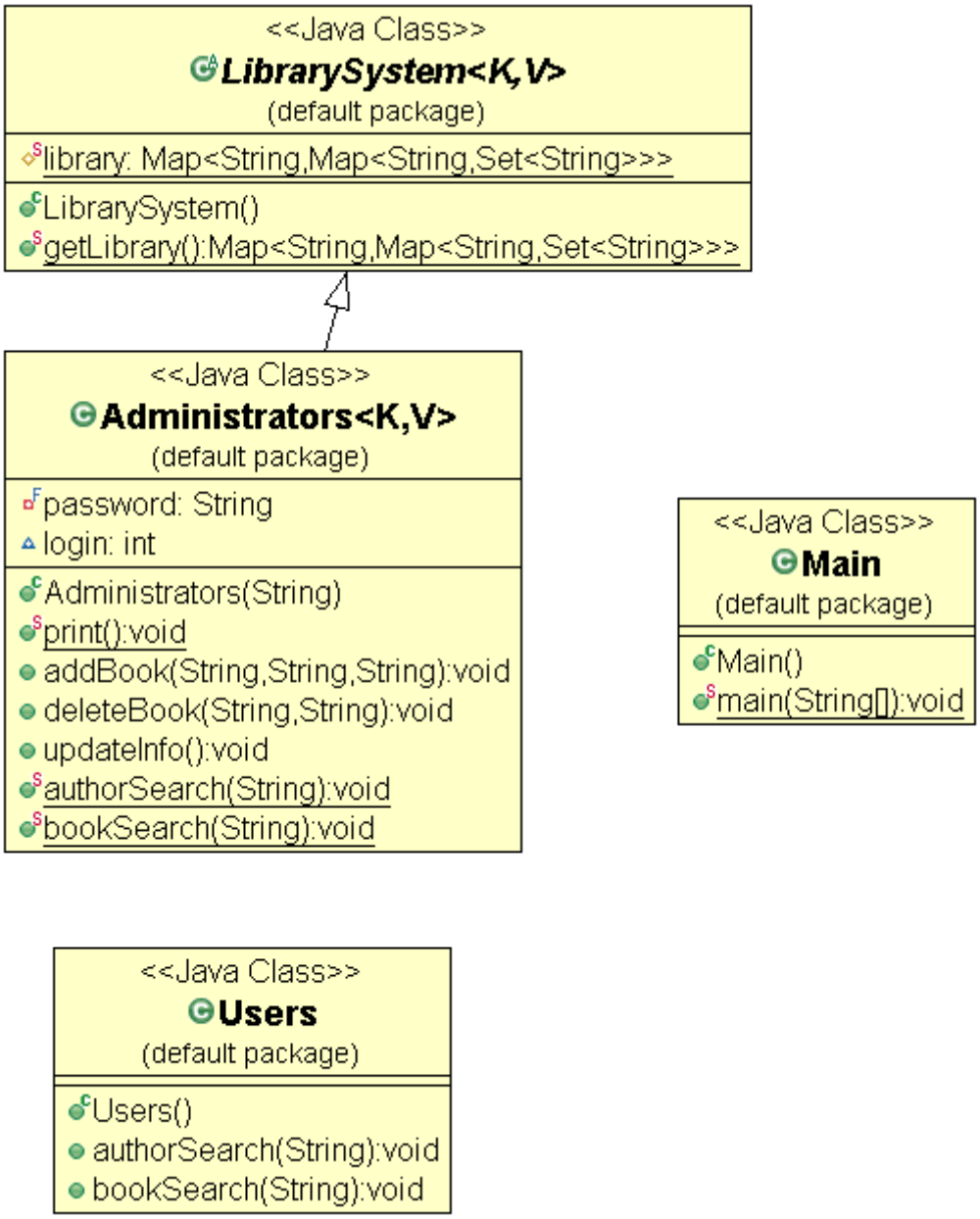

Merge Sort
1200
1000
800
600
400
200
0
10,000
40,000
100,000
150,000
180,000
MergeSort
Expected


Heap Sort
1200
1000
800
600
400
200
0
10,000
40,000
100,000
150,000
180,000
HeapSort
Expected


Quick Sort
1200
1000
800
600
400
200
0
10,000
40,000
100,000
150,000
180,000
QuickSort
Expected

- Q3 -

1. **CLASS DIAGRAMS**



2. PROBLEM SOLUTION APPROACH

I use my data structure like this: Nested map ,
-Outer
key -> author name
value -> inner map
-Inner
key -> book name
value -> (sets) location

I have four classes which are
LibrarySystem -> to keep data (nested map)
Administrators -> can perform all functionalities
Users -> searching books or authors
Main -> driver class

Administrator's constructor takes password. And inside Administrators class I keep 'int login' data to check if the password entered by the admin is correct. If correct then initialize login = 1 and 0 otherwise.

The user can only search by author name and by book title.
Since admin can do everything, I called the admin methods inside Users class.

I used the map and set of the java while designing the methods.
For example if I want to access library by author name ; I used containsKey() method from java Map class. It gives me that author exist or not.

3. TEST CASES

Test Case ID	Test Method	Test Input	Test Output	Pass/Fail
T1	addBook() method	Author, book and location as String	Void	Pass
T2	deleteBook() method	Author and book as String	Void	Pass
T3	updateInfo() method	No input (scanner will take input on console)	Void	Pass
T4	authorSearch() method	Author name as String	Prints all books of the author if there any exist.	Pass
T5	bookSearch() method	Book name as String	Prints book's author name and locations if there any exist.	Pass
T6	print() method	No input	Prints the informations in the library	Pass

4. RUNNING AND RESULTS

Test ID	Test Result
T1	Admin -> adds books ... [PRINT]: Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]} Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]} Yasar Kemal = {Ince Memed=[c7s7.3221]} Oguz Atay = {Tutunamayanlar=[c4s3.2312]}
T2	Admin -> deleteBook(Yasar Kemal,Ince Memed) [PRINT]: Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]} Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]} Yasar Kemal = {} Oguz Atay = {Tutunamayanlar=[c4s3.2312]}
T3	Admin -> updateInfo() What do you want to do ? 1)Add book 2)Remove book Choice 1 or 2 : 1 Author name: Shakespeare Book name: Hamlet Location: c3s3.4554 [PRINT]: Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]} Shakespeare = {Hamlet=[c3s3.4554]} Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]} Yasar Kemal = {} Oguz Atay = {Tutunamayanlar=[c4s3.2312]}
T4	User -> authorSearch(Dostoyevsky) All books of the author : [Pool Folk, Crime and Punishment] Enter book name: Crime and Punishment The location of the book of your choice: [c5s4.3222]
T5	User -> bookSearch(Hobbit) Author name: Tolkien Book locations: [[c1s1.1111, c1s2.2222]]
T6	[PRINT]: Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]} Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]} Yasar Kemal = {Ince Memed=[c7s7.3221]} Oguz Atay = {Tutunamayanlar=[c4s3.2312]}

Running command;

```
               TEST STARTING...
------------------------------------------------
Admin login successful

        Admin -> adds books ...
[PRINT]:
Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]}
Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]}
Yasar Kemal = {Ince Memed=[c7s7.3221]}
Oguz Atay = {Tutunamayanlar=[c4s3.2312]}

        Admin -> deleteBook(Yasar Kemal,Ince Memed)
[PRINT]:
Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]}
Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]}
Yasar Kemal = {}
Oguz Atay = {Tutunamayanlar=[c4s3.2312]}

        User -> bookSearch(Hobbit)
Author name: Tolkien
Book locations: [[c1s1.1111, c1s2.2222]]

        User -> authorSearch(Dostoyevsky)
All books of the author : [Pool Folk, Crime and Punishment]
Enter book name: Crime and Punishment
The location of the book of your choice:
[c5s4.3222]

        Admin -> updateInfo()
What do you want to do ?
1)Add book
2)Remove book
Choice 1 or 2 : 1
Author name: Shakespeare
Book name: Hamlet
Location: c3s3.4554

[PRINT]:
Tolkien = {Hobbit=[c1s1.1111, c1s2.2222]}
Shakespeare = {Hamlet=[c3s3.4554]}
Dostoyevsky = {Pool Folk=[c6s7.1332], Crime and Punishment=[c5s4.3222]}
Yasar Kemal = {}
Oguz Atay = {Tutunamayanlar=[c4s3.2312]}


------------------------------------------------
               TEST FINISHED...
```

- Q4 -

I could not implement it.