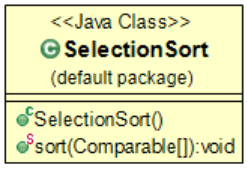
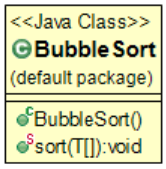
GIT Department of Computer Engineering
CSE 222/505 - Spring 2020
Homework 6 Report

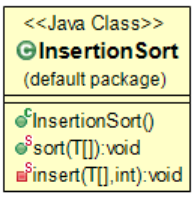
Şeyda ÖZER
171044023

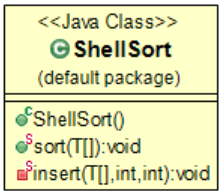
CLASS DIAGRAMS

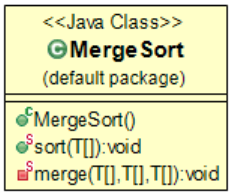
Q2:

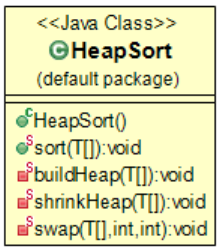


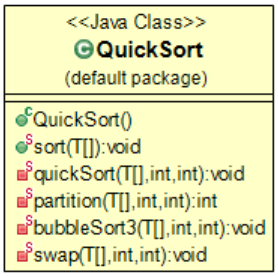


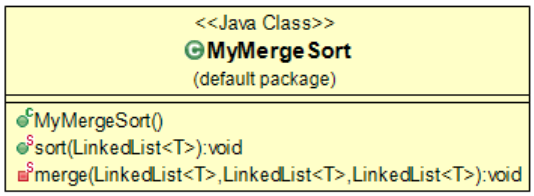


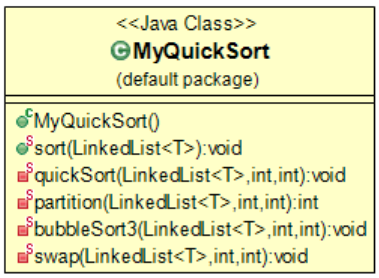




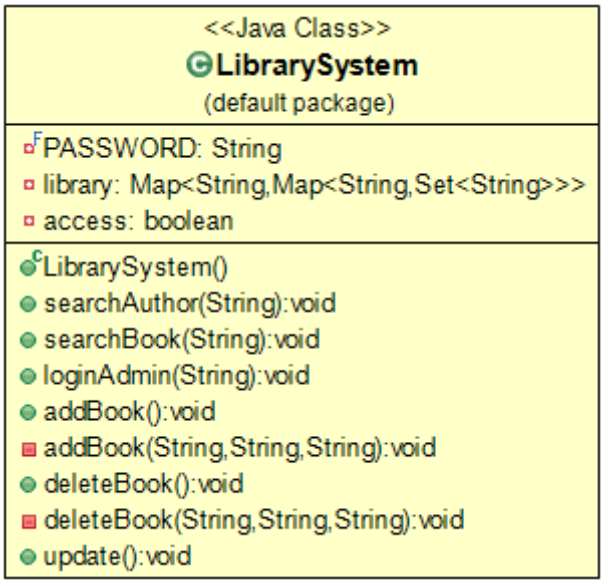






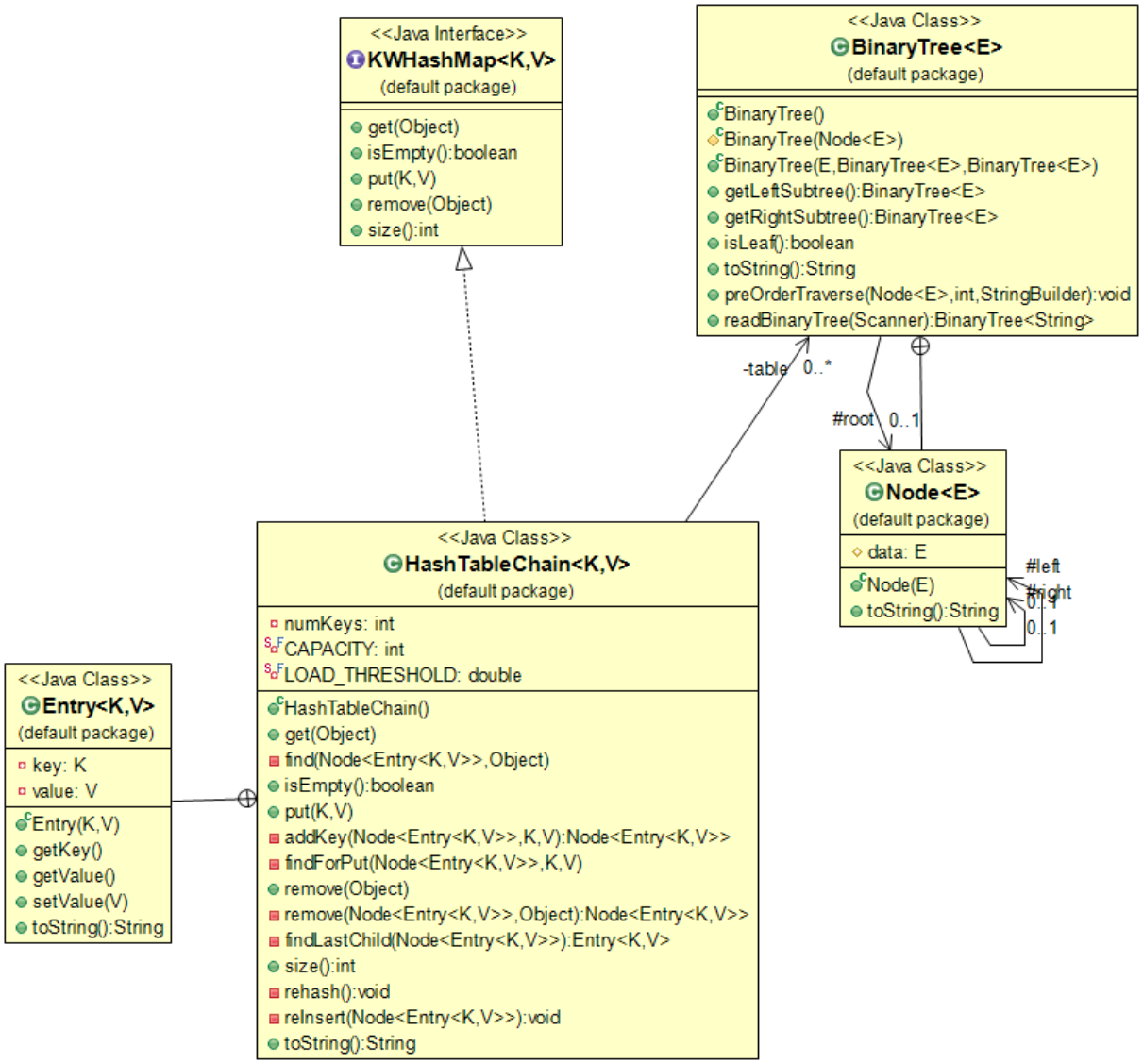


Q3:



Q4:

a)


<<Java Interface>>
KWHashMap<K,V>
(default package)
get(Object)
isEmpty():boolean
put(K,V)
remove(Object)
size():int
<<Java Class>>
BinaryTree<E>
(default package)
BinaryTree()
BinaryTree(Node<E>)
BinaryTree(E,BinaryTree<E>,BinaryTree<E>)
getLeftSubtree():BinaryTree<E>
getRightSubtree():BinaryTree<E>
isLeaf():boolean
toString():String
preOrderTraverse(Node<E>,int,StringBuilder):void
readBinaryTree(Scanner):BinaryTree<String>
-table
0..*
#root
0..1
<<Java Class>>
Node<E>
(default package)
data: E
Node(E)
toString():String
#left
#right
0..1
0..1
<<Java Class>>
HashTableChain<K,V>
(default package)
numKeys: int
CAPACITY: int
LOAD_THRESHOLD: double
HashTableChain()
get(Object)
find(Node<Entry<K,V>>,Object)
isEmpty():boolean
put(K,V)
addKey(Node<Entry<K,V>>,K,V):Node<Entry<K,V>>
findForPut(Node<Entry<K,V>>,K,V)
remove(Object)
remove(Node<Entry<K,V>>,Object):Node<Entry<K,V>>
findLastChild(Node<Entry<K,V>>):Entry<K,V>
size():int
rehash():void
reInsert(Node<Entry<K,V>>):void
toString():String
<<Java Class>>
Entry<K,V>
(default package)
key: K
value: V
Entry(K,V)
getKey()
getValue()
setValue(V)
toString():String

PROBLEM SOLUTION APPROACH

Q2:

For Merge sort with linked list:

n^2 operations are performed at each step. Since the table is split into halves each time, there are logn steps. So, T(n) = Q(n^2 logn)

For Quicksort with linked list:

T(n) = O(n^2 logn) as in the merge sort method.

When an array is used, elements can be accessed directly with an index. This time is constant. However, if the linked list is used, the element is reached in linear time. Therefore, java array should be used in sorting algorithms.

Since the expected time values were very large when creating a table, I simplified them with 1000.

Q3:
I put the final variable to the class for administrator password. Also I put a boolean variable. The variable is true, if the user enters the correct password. I check if the user is an administrator with this variable.

When searching for an author, I first check if the author is on the system. If the author is in the system, I print author's books on the screen. Then I print the locations of the book the user chooses.

While looking for a book in the system, I check all the authors. When the book is find, I print book's information on the screen.

When adding a book to the system, I first search the author in the system. If the author is not found, I add the author and author's book to the system. If the author is found, I check the status of the book.

When deleting a book to the system, I first search the author in the system. If the author is found, I delete the book.

If the admin wants to update a location of the book, the admin must enter the author's name and the title of the book. Then the locations of the book are displayed. The admin must choose a location and enter a new location instead.

Q4:

a) HashTableChain

We use binary tree instead of linked list in this assignment. Each table element references a binary tree that contains all the items that hash to same table index.

When searching for a key, the tree in the index (according to key) is checked. After finding the key, the desired operation can be done. In this assignment, while adding key to the tree, I first added to the left branch. When deleting, I deleted the rightmost branch when one of the branches of the tree was not empty.

TEST CASES

Q3 – Library Automation System

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass / Fail
1	Search a author that is in not the system	searchAuthor("Seyda")	This author is not found.	Message: No books for this author were found.	Pass
2	Search a book that is not in the system	searchBook("book1")	This book is not found.	Message: This book is not found!	Pass
3	Adding a book but the admin did not enter	addBook()	No change in the system.	No change.	Pass
4	Deleting a book but the admin did not enter admin	deleteBook()	No change in the system.	No change.	Pass
5	Updating a book but the admin did not enter	update()	No change in the system.	No change.	Pass
6	Adding a new book	Book() Book name: book1 Author Name: Seyda Location: c1s1	It is added.	It is added.	Pass
7	Adding a new book with same author and new location	Book() Book name: book2 Author Name: Seyda Location: c1s3	It is added.	It is added.	Pass
8	Adding the same book with same author and same location	Book() Book name: book1 Author Name: Seyda Location: c1s1	It is not added.	This book is already added. It is not added.	Pass
9	Adding a new book with new author	Book() Book name: book1 Author Name: Isra Location: c2s1	It is added.	It is added.	Pass
10	Adding the same book but different location	Book() Book name: book1 Author Name: Seyda Location: c1s2	It is added.	It is added.	Pass
11	Search a author that is in the system	searchAuthor("Seyda") book name: book1	Author is found and author's all books are printed. Then, whichever book the user chooses, the location(s) of that book will be displayed.	Author is found and author's all books are printed. Then, the locations of the selected book are displayed.	Pass

12	Search a book that is in the system	searchBook("book1")	It is found and display its informations.	It is found and display its informations.	Pass
13	Delete a book that has one copy	deleteBook() Book name: book1 Author Name: Isra Location: c2s1	It is deleted.	It is deleted.	Pass
14	Delete a book that has more than one copy	deleteBook() Book name: book1 Author Name: Seyda Location: c1s1	The copy is deleted.	The copy is deleted.	Pass
15	Delete a book that is not in the system.	deleteBook() Try1: (wrong name) Book name: book3 Author name: Seyda Location: c1s2 Try2: (wrong location) Book name: book1 Author name: Seyda Location: c1s1 Try3: (wrong author) Book name: book1 Author name: Melek Location: c1s2	It is not found.	This book is not found.	Pass
16	Update locations of the book	update() Book name: book2 Author Name: Seyda old location: c1s3 new location: c1s1	The admin enters title and author name. Later the admin choose a location and enter new location.	The admin enters title and author name. Later the admin choose a location and enter new location.	Pass

Q4 – Hash Table with Chaining

Test Case #	Test Case Description	Test Data	Expected Result	Actual Result	Pass / Fail
1	Test the table is empty	isEmpty()	The map is empty.	The map is empty.	Pass
2	Put new entry with new key	put(1, "A")	It is added.	It is added.	Pass
3	Test the table is not empty	isEmpty()	The table is not empty.	The table is not empty.	Pass
4	Put new entry with old key	put(1, "a")	The key's value is changed to "a".	The key's value is changed to "a".	Pass
5	Get an value	get(2)	Gets the value associated with 2	Gets B	Pass
6	Get an value but its key is not the map	get(4)	Returns null	Returns null	Pass

7	Remove a key that is in the table	remove(3)	It is removed.	It is removed.	Pass
8	Remove a key that is not in the table	Remove(4)	It is not removed.	Returns null	Pass
9	Put the keys that have the same hash code.	Put(key, value1) Put(key, value2)	They are added.	I couldn't test it because I couldn't find the keys that have the same hash code.	?
10	rehash()			I could not test the rehash method because I could not create a situation that requires it.	?

RUNNING AND RESULTS

Q3 – Library Automation System

Test Case 1:

Test Data:

```
library.searchAuthor("Seyda");
```

Result:

```
No books for this author were found.
```

Test Case 2:

Test Data:

```
library.searchBook("book1");
```

Result:

```
This book is not found!
```

Test Case 3:

Test Data:

```
library.addBook();
```

Result:

The method does not work because admin is not logged into the system.

Test Case 4:

Test Data:

```
library.deleteBook();
```

Result:

The method does not work because admin is not logged into the system.

Test Case 5:

Test Data:

```
library.update();
```

Result:

The method does not work because admin is not logged into the system.

NOTE: Admin login was done before other tests.

```
library.loginAdmin("12345");
```

Test Case 6:

Test Data:

```
library.addBook();
```

```
Adding book process
Enter the book informations:
The book's name:
book1
The author's name:
Seyda
The location:
c1s1
```

Result:

```
This book is added.
```

Test Case 7:

Test Data:

```
library.addBook();
```

```
Adding book process
Enter the book informations:
The book's name:
book2
The author's name:
Seyda
The location:
c1s3
```

Result:

```
This book is added.
```

Test Case 8:

Test Data:

```
library.addBook();
```

```
Adding book process
Enter the book informations:
The book's name:
book1
The author's name:
Seyda
The location:
c1s1
```

Result:

```
This book is already added.
```

Test Case 9:

Test Data:

```
library.addBook();
```

```
Adding book process
Enter the book informations:
The book's name:
book1
The author's name:
Isra
The location:
c2s1
```

Result:

```
This book is added.
```

Test Case 10:

Test Data:

```
library.addBook();
```

```
Adding book process
Enter the book informations:
The book's name:
book1
The author's name:
Seyda
The location:
c1s2
```

Result:

```
This book is added.
```

Test Case 11:

Test Data:

```
library.searchAuthor("Seyda");
```

```
The author's books:
1 - book2
2 - book1

Enter the title of the book.
book1
```

Result:

```
The locations of the book:
- c1s2
- c1s1
```

Test Case 12:

Test Data:

```
library.searchBook("book1");
```

Result:

```
Informations of the book(book1) :
Author name: Seyda
Location(s):
- c1s2
- c1s1

Informations of the book(book1) :
Author name: Isra
Location(s):
- c2s1
```

Test Case 13:

Test Data:

```
library.deleteBook();
```

```
Deleting book process
Enter the book informations:
The book's name:
book1
The author's name:
Isra
The location:
c2s1
```

Result:

```
This book is deleted.
```

Test Case 14:

Test Data:

```
library.deleteBook();
```

```
Deleting book process
Enter the book informations:
The book's name:
book1
The author's name:
Seyda
The location:
c1s1
```

Result:

```
This book is deleted.
```

Test Case 15:

Test Data:

```
library.deleteBook();
```

(wrong name)

```
Deleting book process
Enter the book informations:
The book's name:
book3
The author's name:
Seyda
The location:
c1s2
```

(wrong location)

```
Deleting book process
Enter the book informations:
The book's name:
book1
The author's name:
Seyda
The location:
c1s1
```

(wrong author name)

```
Deleting book process
Enter the book informations:
The book's name:
book1
The author's name:
Melek
The location:
c1s2
```

Result:

```
This book is not found.
```

Test Case 16:

Test Data:

```
library.update();
```

```
Please enter the name and author of the book you want to update, respectively.
book2
Seyda
Location(s) of the book:
- c1s3
Please enter the location you want to remove and add, respectively.
c1s3
c1s1
This location is updated.
```

Result:

The location is updated.

```
Informations of the book(book2) :
Author name: Seyda
Location(s):
- c1s1
```

Q4 – Hash Table with Chaining

Test Case 1:

Test Data:

```
if(cMap.isEmpty())
    System.out.println("The map is empty.");
else
    System.out.println("The map is not empty.");
```

Result:

```
The map is empty.
```

Test Case 2:

Test Data:

```
cMap.put(1, "A");
```

Result:

```
[1- A]
  null
  null
```

Test Case 3:

Test Data:

```
if(cMap.isEmpty())
    System.out.println("The map is empty.");
else
    System.out.println("The map is not empty.");
```

Result:

```
The map is not empty.
```

Test Case 4:

Test Data:

```
cMap.put(1, "a");
```

Result:

```
[1- a]
  null
  null
```

The map is:

```
[1- a]
  null
  null

[2- C]
  null
  null

[3- D]
  null
  null
```

Test Case 5:

Test Data:

```
System.out.println(cMap.get(2));
```

Result:

```
The value of key(2): C
```

Test Case 6:

Test Data:

```
System.out.println(cMap.get(4));
```

Result:

```
The value of key(4): null
```

Test Case 7:

Test Data:

```
System.out.println(cMap.remove(3));
```

Result:

```
The value of removed key(3): D
```

The map:

```
[1- a]
  null
  null

[2- C]
  null
  null
```

Test Case 8:

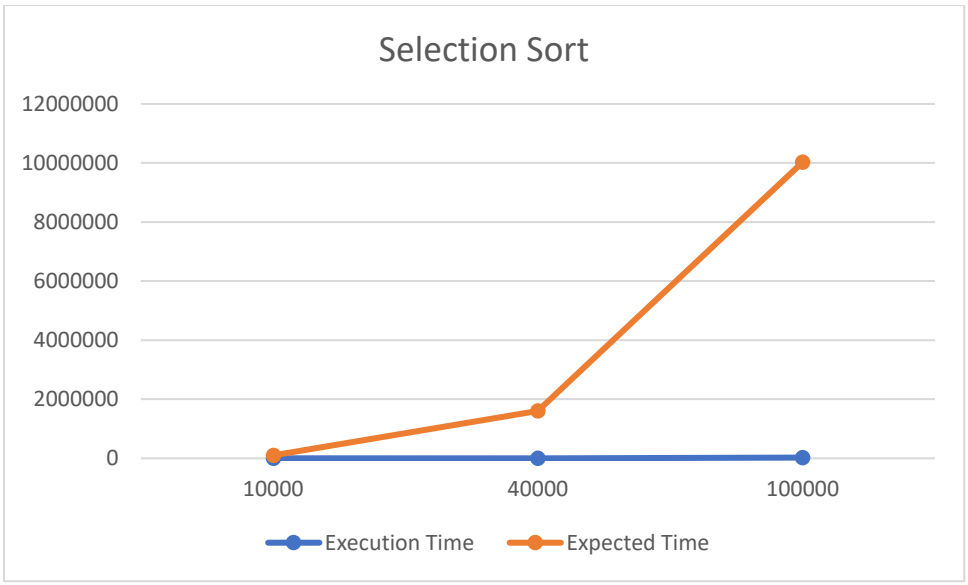
Test Data:

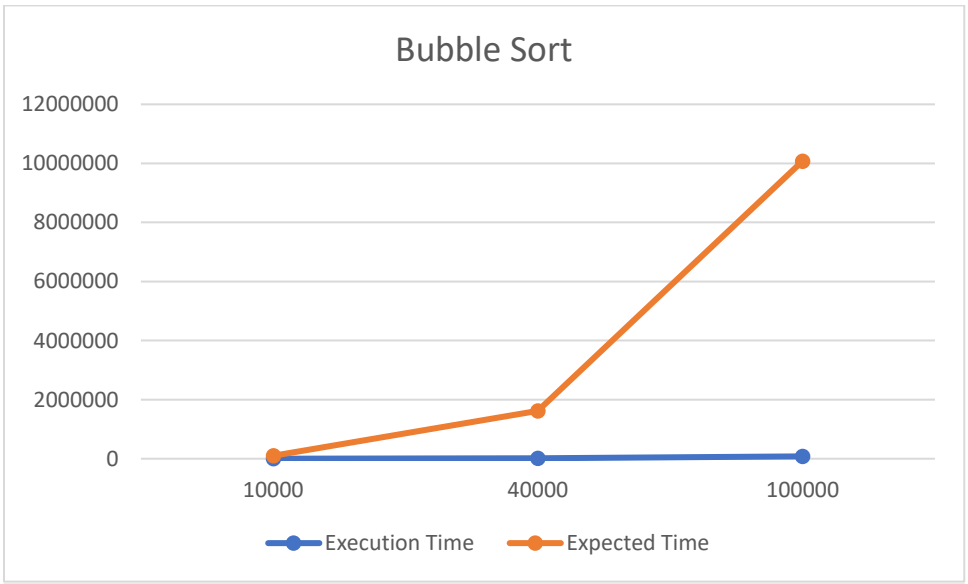
```
System.out.println(cMap.remove(4));
```

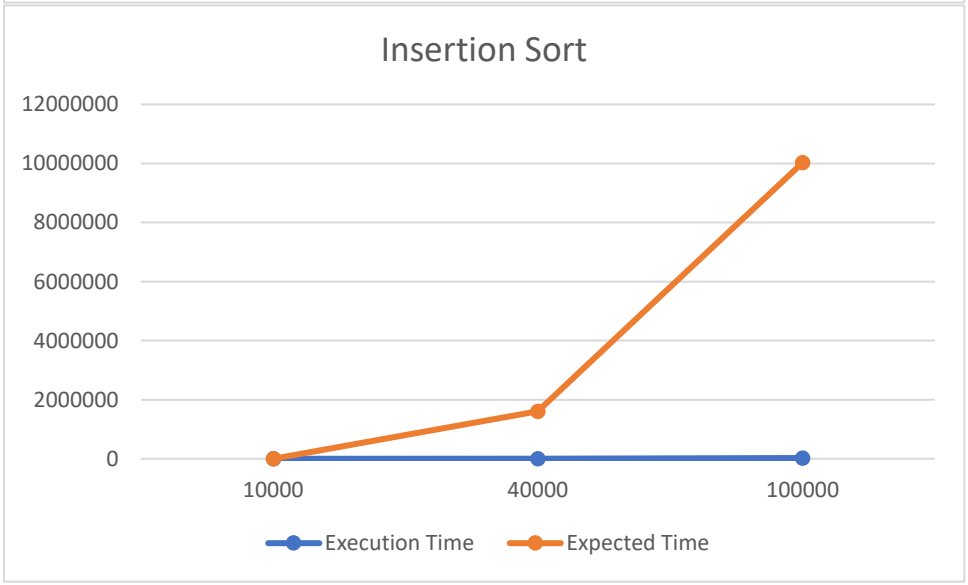
Result:

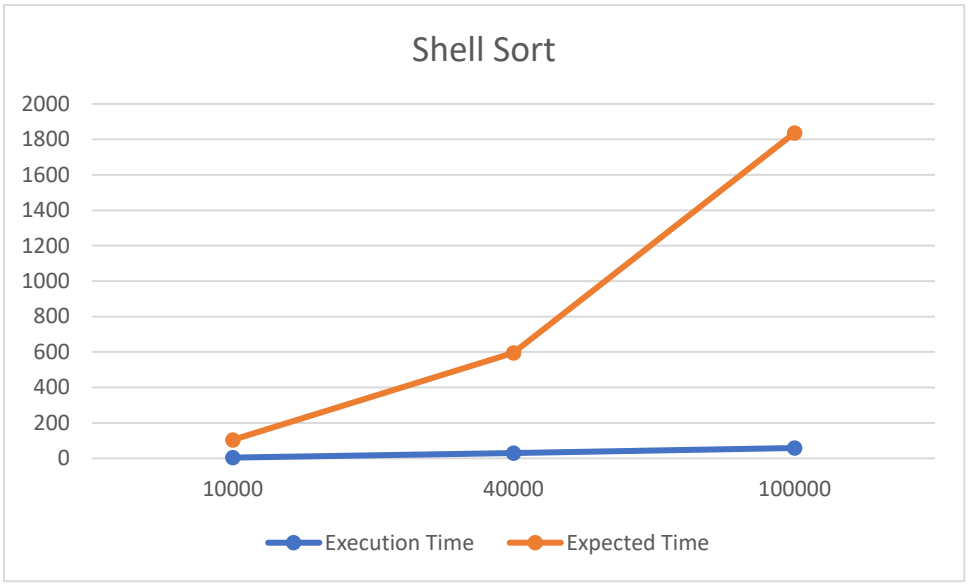
```
The value of removed key(4): null
```

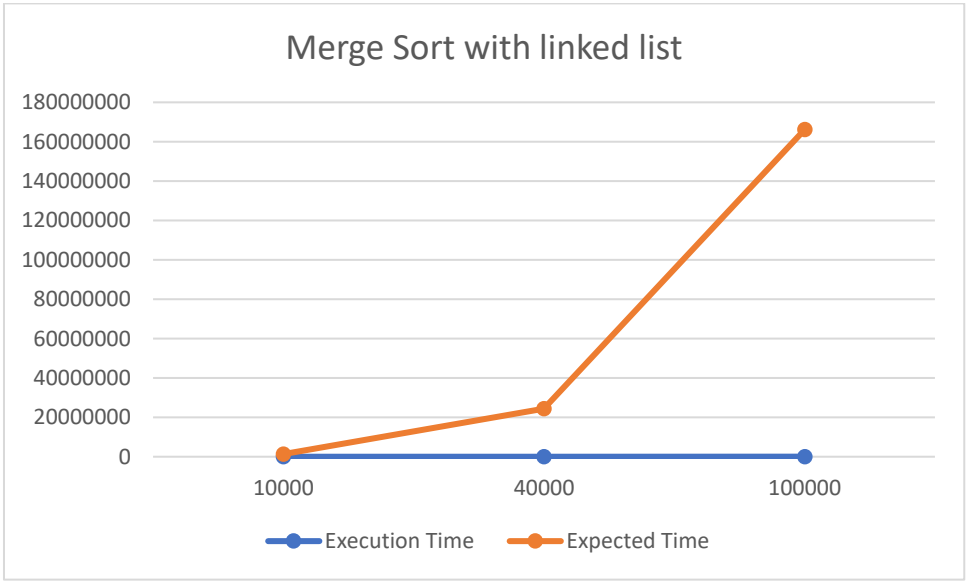
Q2:

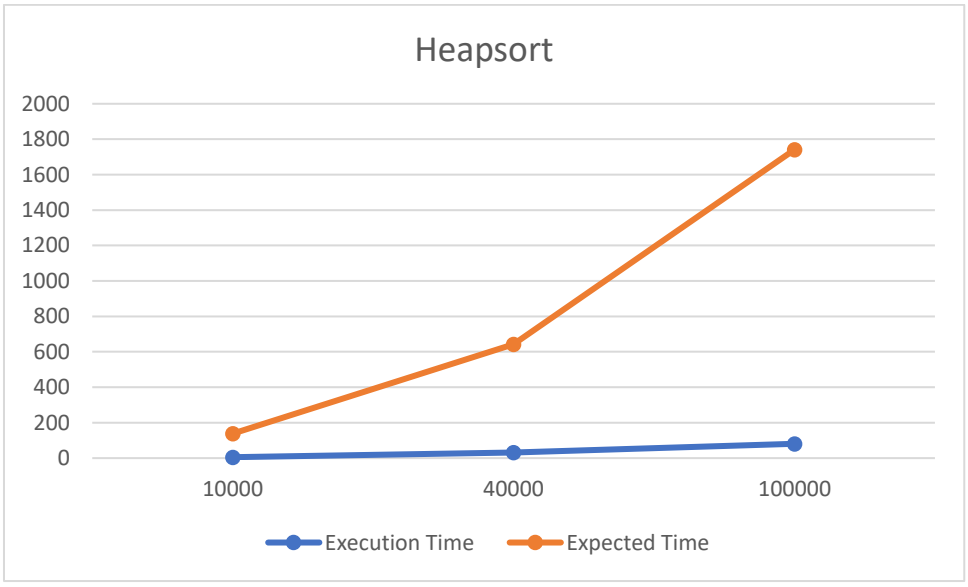

Selection Sort
12000000
10000000
8000000
6000000
4000000
2000000
0
10000
40000
100000
Execution Time
Expected Time

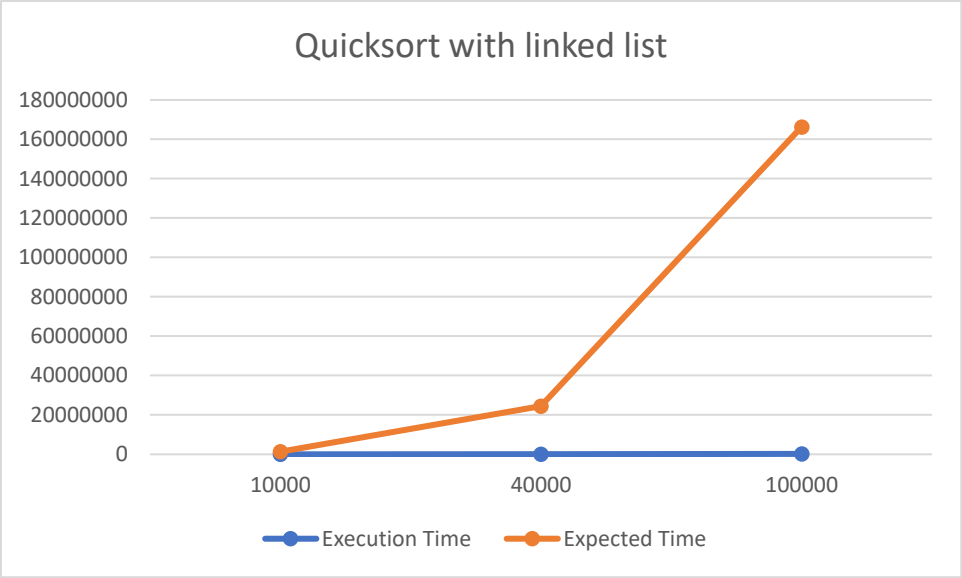

Bubble Sort
12000000
10000000
8000000
6000000
4000000
2000000
0
10000
40000
100000
Execution Time
Expected Time


Insertion Sort
12000000
10000000
8000000
6000000
4000000
2000000
0
10000
40000
100000
Execution Time
Expected Time


Shell Sort
2000
1800
1600
1400
1200
1000
800
600
400
200
0
10000
40000
100000
Execution Time
Expected Time


Merge Sort with linked list
180000000
160000000
140000000
120000000
100000000
80000000
60000000
40000000
20000000
0
10000
40000
100000
Execution Time
Expected Time


Heapsort
2000
1800
1600
1400
1200
1000
800
600
400
200
0
10000
40000
100000
Execution Time
Expected Time



Running Times:

	Selection Sort	Bubble Sort	Insertion Sort	Shell Sort	Merge Sort	Heap sort	Quick sort	MyMerge Sort	MyQuick sort
10000(sorted)	163	2	1	6	6	40	5	273	1012
10000	287	549	198	7	7	6	7	348	1891
10000	290	497	113	15	5	4	20	324	1924
10000	286	489	119	2	4	3	13	353	2139
10000	281	486	104	3	4	3	2	329	2088
10000	280	536	117	3	9	3	2	330	2087
10000	281	456	98	3	6	2	2	330	2129
10000	284	550	123	3	3	3	2	331	2105
10000	283	604	151	4	3	3	2	330	2150
10000	280	531	125	2	2	3	2	331	2123
10000	280	469	101	2	6	3	2	332	2127
10000	282	558	134	3	4	3	2	332	2092
10000	281	610	158	3	11	4	2	333	2112
10000	282	831	228	3	3	4	2	334	2088
10000	281	477	101	3	4	3	2	332	2092
10000	280	838	294	4	10	4	2	332	2119
10000	280	850	243	4	3	4	3	333	2192
10000	285	834	305	4	4	4	2	338	2215
10000	282	838	299	4	12	4	3	335	2109
10000	280	833	242	4	4	4	3	338	1894
10000	290	645	119	3	3	4	2	335	1898
///////									
40000(sorted)	912	3	2	23	57	56	15	4390	19271
40000	2342	14599	4202	47	21	30	25	5496	38906
40000	2357	14950	5822	19	16	22	24	5442	47386
40000	3472	23325	7936	31	26	35	33	8044	55681
40000	4190	23816	7519	27	22	32	14	8499	54716
40000	3326	22260	7534	29	21	31	14	8242	53595
40000	3314	22406	7226	29	58	32	14	7877	54170

40000	3141	22027	7423	28	20	31	14	7940	54756
40000	3406	22088	6667	28	19	28	14	8041	53348
40000	3213	22231	7254	36	61	30	14	7940	55032
40000	3693	28270	6322	31	23	29	15	8731	55350
40000	3636	22744	7611	29	22	29	14	7847	54727
40000	3243	20381	5869	31	22	32	15	8005	54843
40000	3227	22099	7195	29	23	31	18	7866	53908
40000	3349	21968	6522	29	32	29	15	7914	54585
40000	3290	22271	7221	28	22	29	14	8099	55553
40000	3298	21876	6501	32	22	32	16	8099	53840
40000	3321	22519	7276	27	27	29	14	8202	53832
40000	3143	21826	6474	29	23	29	19	7829	54392
40000	3188	21915	7170	28	23	27	14	7871	53765
40000	3209	21742	6553	30	21	29	14	7912	56951
//////									
100000 sorted	18505	13	15	79	87	276	98	81470	410784
100000	86874	297627	92976	238	169	266	352	101410	314990
100000	20759	68650	22156	40	33	45	20	24097	172335
100000	18583	63527	19763	32	34	38	19	22104	162872
100000	17722	60836	20067	34	41	37	19	22412	171407
100000	18957	62395	20687	40	29	39	20	24668	178292
100000	18564	63953	20694	34	26	36	19	22239	181001
100000	18323	66861	22221	37	27	39	18	25774	168851
100000	19343	62088	21227	34	42	37	18	23379	181091
100000	20482	64933	20712	40	30	41	21	23323	191001
100000	18801	62841	20777	33	27	36	19	22383	189262
100000									
100000									
100000									
100000									
100000									
100000									
100000									
100000									
100000									
100000									
////////									
150000 sorted	10919	7	9	73	42	131	40	45577	191404
150000	43785	154544	51743	85	57	71	69	61168	412810
150000									
150000									
150000									
150000									
150000									
150000									
150000									
150000									
150000									
150000									

150000									
150000									
150000									
150000									
150000									
150000									
150000									
150000									
150000									
180000 sorted	17512	6	12	63	55	151	63	111195	630450
//////////	97661	315839	99072	171	83	130	83	136697	
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									
180000									