**COMP 5511**

**Assignment 3 – Theory Portion**

## Group Members

Aida Sharif Rohani (Group leader) - 21341669

Edip Tac - 26783287

Faezeh Mobasheri - 26821022

Milan Jetha - 40013982

Q1)a)

Since we have to traverse the heap to find the max value, the complexity is O(n). The reason is that, we will have to traverse half of the heap since we can only proceed either left or right. This means that we are traversing n/2 elements, which still gives a complexity of O(n).

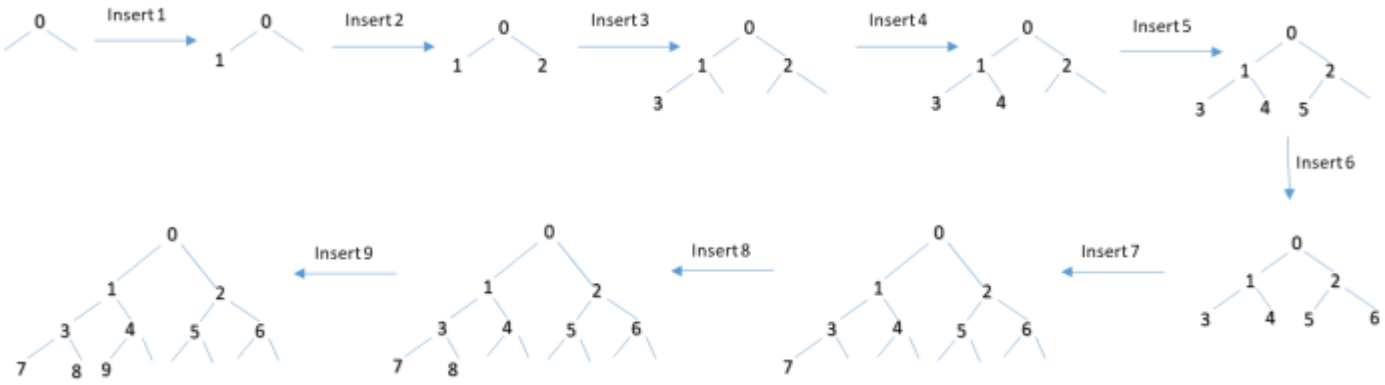
When compared to the conventional binary search tree, the complexity is the same, i.e. you will have to check on average half of the tree just like the case of finding the max element in the min-heap. This will give the complexity of O(n). However, if we are working with a balanced binary tree that is also sorted, the complexity will be O(log n). Since the sorted and balanced binary tree will always be traversed through only 1 path (either left or right child) and has a height of log(n), it would take O(log n) to find the max element.

Q1)b)

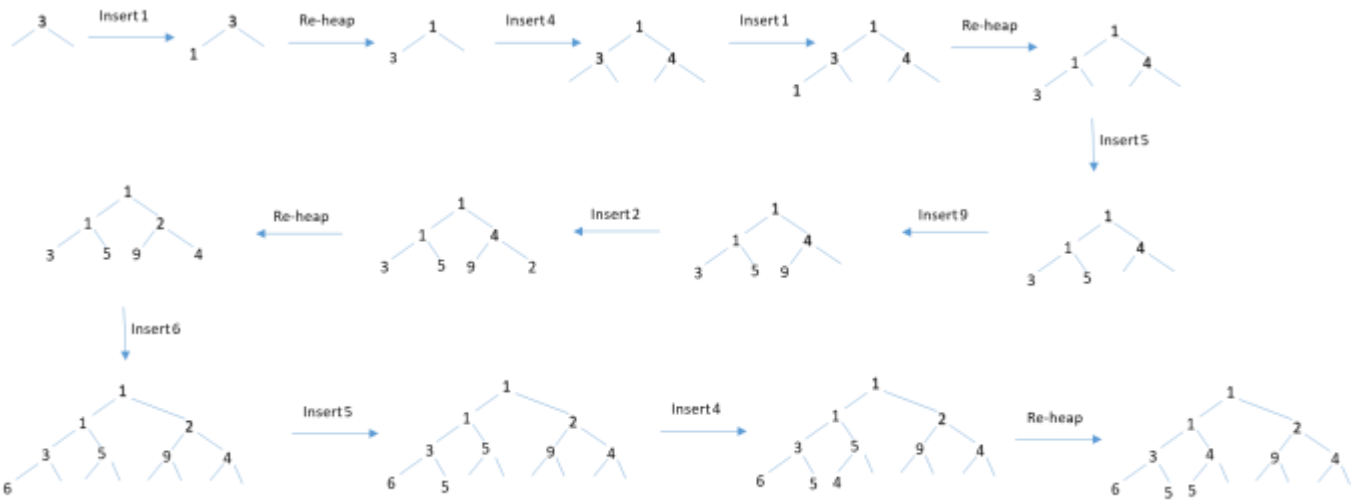
| **Structure** | **Insert** | **Delete-Min** | **Remove** | **Find-Min** |
|---|---|---|---|---|
| Binary Heap | O(log n) | O(log n) | O(log n) | O(1) |
| Ordered Array | O(n) | O(1) | O(n) | O(1) |
| Ordered List | O(n) | O(1) | O(1) | O(1) |

Since the most frequently used functions for the event queue would be the insert and delete-min operations. Since the Binary Heap has both of the operations with the complexity of O(log n) it would be a better design compared to the others that have one of those key operations as O(n). When the game is being played with a large number of events, the O(log n) method will perform better instead of bottlenecking the program at either the Insert or the Delete-Min function.

Q2) a)



Q2)b)




Q3)a)

Division method is good when we are dealing only with integers and the size of the hash-table is a prime number. This method consists of only doing a single division so it is fast to implement. However, it is not very suitable when dealing with a large set of data since the larger the set of data, since the chances of having collisions will be very high using the division method, and a more complex technique would be required. The reason for this is that in a large set of data, the chances of having numbers that end with the same digits is very high. Using the division method, this would definitely result in a large number of collisions.

So, the division method is a good choice when dealing with a reasonable sized set of data (not too large) and when the table size is a prime number and the table is going to be accessed very often. Since the division method of hashing is very fast, programs that require very frequent access (so hashing is done very frequently) to the table can use it.

Q3)b)

In order to make the best hash function, whenever the postal codes are entered remove the space between the two segments of the postal code.

This will result in the postal code having the format LDLDLD. For clarification purposes, we can refer to the postal codes as having the following identification: $L_1D_1L_2D_2L_3D_3$.

Furthermore, since the L values are always letters between A-Z, we can convert those into digits as well where A will be given the value 1 and Z will be given the value 26. The values of D are already numbers and this would therefore convert the entire postal code to a digit.

After the conversion we can see that the smallest postal code would be A0A0A0 and the largest would be Z9Z9Z9 and the size of the hash table must be able to support this size.

Now that we know the size of the table we need a hash function that has the least possible amount of collisions. To do this, we can see that the number possible postal codes we can get out of the chosen representation would be 26*10*26*10*26*10. The hash function should therefore match the postal codes to only 1 specific number to have no collisions at all. Since the digits can only be from 0-9 and the letters can only be from 1 to 26, we can use them as base values to calculate the hash.

We could thus take the RADIX values of each integer value as base 10 and the RADIX value of each letter as base 26.

The possible hash could therefore be:

$h(L_1D_1L_2D_2L_3D_3) = (D_1*10^0)+ (D_2*10^1)+ (D_3*10^2)+(\{[\text{Int value of } L_1]*26^0\}+\{[\text{Int value of } L_2]*26^1\}+\{[\text{Int value of } L_3]*26^2\}*10^3)$

Q4)

K=5

Result:

BPlease enter key value:

5

Heap sort of original text file = [0.24 0.97 1.04 1.05 1.28 1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 3.66 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

removed Heap sort of original text file = [1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 3.66 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

Heap sort of original text file = [1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

Shrunk list from Heap sort = [1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

5 minimum GPAs from Heap Sort = [0.24 0.97 1.04 1.05 1.28 ]

Selection sort of original text file = [0.24, 0.97, 1.04, 1.05, 1.28, 1.36, 1.47, 1.55, 1.56, 1.59, 1.78, 1.89, 1.99, 2.0, 2.09, 2.17, 2.56, 2.78, 2.88, 2.89, 2.89, 3.09, 3.12, 3.46, 3.47, 3.52, 3.56, 3.66, 3.66, 3.75, 3.78, 3.79, 3.81, 3.81, 3.97, 3.98, 4.0, 4.02, 4.12, 4.26]

5 minimum GPAs from Selection Sort = [0.24, 0.97, 1.04, 1.05, 1.28]

Shrunk array from selection sort = [1.36, 1.47, 1.55, 1.56, 1.59, 1.78, 1.89, 1.99, 2.0, 2.09, 2.17, 2.56, 2.78, 2.88, 2.89, 2.89, 3.09, 3.12, 3.46, 3.47, 3.52, 3.56, 3.66, 3.66, 3.75, 3.78, 3.79, 3.81, 3.81, 3.97, 3.98, 4.0, 4.02, 4.12, 4.26]

K=9

Result:

BPlease enter key value:

9

BPlease enter key value:

9

Heap sort of original text file = [0.24 0.97 1.04 1.05 1.28 1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 3.66 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

removed Heap sort of original text file = [1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 3.66 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

Heap sort of original text file = [1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

Shrunk list from Heap sort = [1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

9 minimum GPAs from Heap Sort = [0.24 0.97 1.04 1.05 1.28 1.36 1.47 1.55 1.56 ]

Selection sort of original text file = [0.24, 0.97, 1.04, 1.05, 1.28, 1.36, 1.47, 1.55, 1.56, 1.59, 1.78, 1.89, 1.99, 2.0, 2.09, 2.17, 2.56, 2.78, 2.88, 2.89, 2.89, 3.09, 3.12, 3.46, 3.47, 3.52, 3.56, 3.66, 3.66, 3.75, 3.78, 3.79, 3.81, 3.81, 3.97, 3.98, 4.0, 4.02, 4.12, 4.26]

9 minimum GPAs from Selection Sort = [0.24, 0.97, 1.04, 1.05, 1.28, 1.36, 1.47, 1.55, 1.56]

Shrunk array from selection sort = [1.59, 1.78, 1.89, 1.99, 2.0, 2.09, 2.17, 2.56, 2.78, 2.88, 2.89, 2.89, 3.09, 3.12, 3.46, 3.47, 3.52, 3.56, 3.66, 3.66, 3.75, 3.78, 3.79, 3.81, 3.81, 3.97, 3.98, 4.0, 4.02, 4.12, 4.26]

K=13

Result:

BPlease enter key value:

13

Heap sort of original text file = [0.24 0.97 1.04 1.05 1.28 1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 2.0 2.09 2.17 2.56 2.78 2.88 2.89 3.66 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

removed Heap sort of original text file = [2.0 2.09 2.17 2.56 2.78 2.88 2.89 3.66 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

Heap sort of original text file = [2.0 2.09 2.17 2.56 2.78 2.88 2.89 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

Shrunk list from Heap sort = [2.0 2.09 2.17 2.56 2.78 2.88 2.89 2.89 3.09 3.12 3.46 3.47 3.52 3.56 3.66 3.66 3.75 3.78 3.79 3.81 3.81 3.97 3.98 4.0 4.02 4.12 4.26 ]

13 minimum GPAs from Heap Sort = [0.24 0.97 1.04 1.05 1.28 1.36 1.47 1.55 1.56 1.59 1.78 1.89 1.99 ]

Selection sort of original text file = [0.24, 0.97, 1.04, 1.05, 1.28, 1.36, 1.47, 1.55, 1.56, 1.59, 1.78, 1.89, 1.99, 2.0, 2.09, 2.17, 2.56, 2.78, 2.88, 2.89, 2.89, 3.09, 3.12, 3.46, 3.47, 3.52, 3.56, 3.66, 3.66, 3.75, 3.78, 3.79, 3.81, 3.81, 3.97, 3.98, 4.0, 4.02, 4.12, 4.26]

13 minimum GPAs from Selection Sort = [0.24, 0.97, 1.04, 1.05, 1.28, 1.36, 1.47, 1.55, 1.56, 1.59, 1.78, 1.89, 1.99]

Shrunk array from selection sort = [2.0, 2.09, 2.17, 2.56, 2.78, 2.88, 2.89, 2.89, 3.09, 3.12, 3.46, 3.47, 3.52, 3.56, 3.66, 3.66, 3.75, 3.78, 3.79, 3.81, 3.81, 3.97, 3.98, 4.0, 4.02, 4.12, 4.26]

Q6)

| | **Number of Comparison** | | |
|---|---|---|---|
| **Searching Name\Searching Type** | **Binary Search** | **Dictionary search** | **Hashing** |
| Azevedo, Ana | 8 | 153 | 1 |
| Silva, Rui | 8 | 1746 | 1 |
| Boussebough, Imane | 10 | 266 | 1 |
| Terracina, Giorgio | 10 | 1869 | 1 |
| Lefebvre, Peter | Not found | Not Found (1042) | 5 |
| Houghten, Sher | Not found | Not Found (796) | 6 |
| Revez, Peter | 10 | 1563 | 1 |