University of Windsor

COMP-8547-1-R-2021S

Advanced Computing Concepts
Summer 2021
Assignment 1

Submitted By
Anubha Sharma
Master of Applied Computing
Student Id: 110037181

Submitted To
Dr. Abedalrhman Alkhateeb

I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work."

Anubha
110037181

Question 1: Within a Java class, write a method that creates n random strings of length 10 and inserts them in a hash table. The method should compute the average time for each insertion.

Answer:
Please refer the package ***assignmentSolution,*** class ***AssignmentSolutionOne*** and method ***assignmentQuestionOneSolution*** for the coded solution.

The Steps followed for resolving are as follows

A. Generate A list of Random Strings [1].
    1. Created a list of ASCII values of the characters from which the random number needs to be generated.
    2. Choose a random index using Math.random() = selected Index.
    3. Convert the ASCII value at selected Index into character and concat with help of StringBuilder.
    4. Repeat steps 2,3,4 to generate a random string of length 10.
    5. Repeat step 4 to generate a list of n number of Random Strings.
B. Insert the list generated in Hash Table.

Result Obtained: The average time to insert 15 elements is 1446 ns.

Result Screenshot

```
List of random strings to be added in HashTable are :
OGmY4FTqSH
FQGb07w89c
tAgTSeiZ0O
6hjM4d1UAu
p0XBnd0HpP
iWQcT88HD3
ZynEvROsIs
SxfxYgORnF
dG8UMxyWz8
ExVinFCJEw
YQbdUlXTpW
bttICrxoaK
EvDBmGfbnG
VSVuVhMPam
vqNvcHm7vn

Details of insertion in Hashtable are as follows:
Key: 0          Value: OGmY4FTqSH               Time Taken: 5700 ns
Key: 1          Value: FQGb07w89c               Time Taken: 1000 ns
Key: 2          Value: tAgTSeiZ0O               Time Taken: 900 ns
Key: 3          Value: 6hjM4d1UAu               Time Taken: 800 ns
Key: 4          Value: p0XBnd0HpP               Time Taken: 900 ns
Key: 5          Value: iWQcT88HD3               Time Taken: 800 ns
Key: 6          Value: ZynEvROsIs               Time Taken: 700 ns
Key: 7          Value: SxfxYgORnF               Time Taken: 800 ns
Key: 8          Value: dG8UMxyWz8               Time Taken: 2500 ns
Key: 9          Value: ExVinFCJEw               Time Taken: 900 ns
Key: 10         Value: YQbdUlXTpW               Time Taken: 700 ns
Key: 11         Value: bttICrxoaK               Time Taken: 700 ns
Key: 12         Value: EvDBmGfbnG               Time Taken: 700 ns
Key: 13         Value: VSVuVhMPam               Time Taken: 3800 ns
Key: 14         Value: vqNvcHm7vn               Time Taken: 800 ns
Avg Time Taken for 15 elements is 1446 ns
```

Question 2: Write another method that finds n random strings in the hash table. The method should delete the string if found. It should also compute the average time of each search.

Answer:
Please refer to the package *assignmentSolution* class *AssignmentSolutionTwo* and method *assignmentQuestionTwoSolution* for the solution.

The Steps followed for resolving are as follows:

A. Generate a Hash Table with i random strings inserted, using the steps used in Question 1.
B. Again, using step A of Question 1, generate a list of n random strings.
C. Find the list of String generates in step B in Hash table generated in step A. If the String is found then delete the string

Result Obtained: The average time to search 10 elements is 1390 ns.

Output Screenshot

```
List of random strings to be added in HashTable are :
ltmsJOXH4g
34jWConNuJ
ZnOl7HRkfw
7ebDEKiSa6
qCNPJbjXyk

Details of insertion in Hashtable are as follows:
Key: 0          Value: ltmsJOXH4g               Time Taken: 6000 ns
Key: 1          Value: 34jWConNuJ               Time Taken: 1200 ns
Key: 2          Value: ZnOl7HRkfw               Time Taken: 1100 ns
Key: 3          Value: 7ebDEKiSa6               Time Taken: 1000 ns
Key: 4          Value: qCNPJbjXyk               Time Taken: 1200 ns
Avg Time Taken for 5 elements is 2100 ns

List of random strings to be found in HashTable are :
[11FYuynJHD, od4hcuVSfW, PW4dmtNqgR, b31KPyeDyD, v9IMY3jeLB, cHCryeIwxq, O6eJQuPxgp, cxh9A3XntQ, HEZAjZFljx, dlhsixnvou]

Starting the Search

Time Taken for searching 11FYuynJHD is 5900 ns
Time Taken for searching od4hcuVSfW is 1000 ns
Time Taken for searching PW4dmtNqgR is 900 ns
Time Taken for searching b31KPyeDyD is 700 ns
Time Taken for searching v9IMY3jeLB is 700 ns
Time Taken for searching cHCryeIwxq is 1100 ns
Time Taken for searching O6eJQuPxgp is 1000 ns
Time Taken for searching cxh9A3XntQ is 800 ns
Time Taken for searching HEZAjZFljx is 1000 ns
Time Taken for searching dlhsixnvou is 800 ns
Avg Time Taken for 10 elements is 1390 ns
```

Question 3: Repeat #1 and #2 with $n = 2^i$, $i = 1, \ldots, 20$. Place the numbers in a table and compare the results for Cuckoo, QuadraticProbing and SeparateChaining. Comment on the times obtained and compare them with the complexities as discussed in class.

Answer 3.1:
Please refer to the package *assignmentSolution* class AssignmentSolutionThreeInsertion and method solution for the solution.

The Steps followed for resolving are as follows:

A. Optimizing 2^n (class: AssignmentSolutionThreeInsertion, method randomListGenerator)
   1. For i=1, compute 2^n. Add the result in the list.
   2. For i++, (i<=20), Compute list[i-1] * 2. Add the result in the list.
B. Optimizing the generation of random list. (class: AssignmentSolutionThreeInsertion, method randomListGenerator)
   1. Generate a list of random strings for a value of i and save it as previous result.
   2. Now when creating a list of random strings for i+1, generate a list only for i and concat the result with previous result. (2^n+1 = 2^n + 2^n).
C. For Each n = 2^i, i = 1, …,20, generate a list of random string using step A, Step B and Question 1, Step A.
D. Insert the list in new Hash Table, Cuckoo Hash Table, Quadratic Hash Table, Separate Chaining Hash Table.

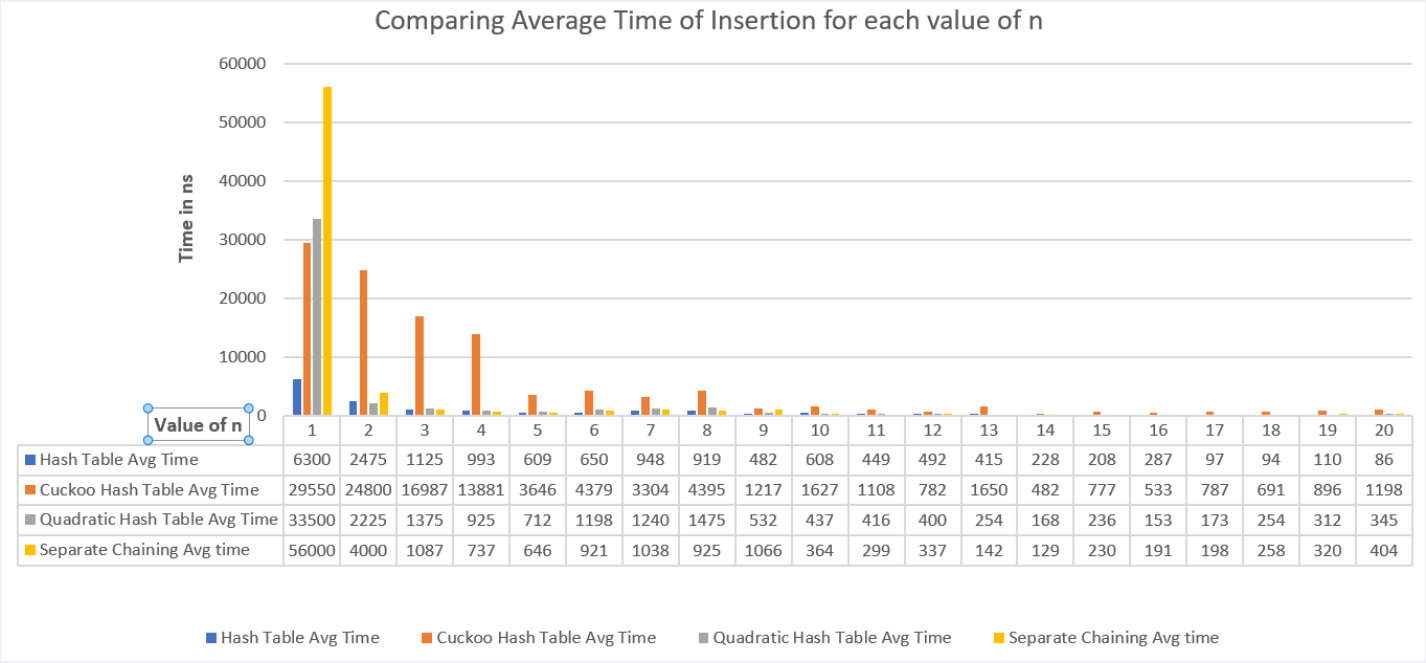
Result Obtained: As the value of n increases, the average time taken by each collection decreases. Among Cuckoo Hash table, Quadratic Probing Hash Table and Separate Chaining Hash Table, the performance of Quadratic Probing Hash Table is the best.

A snippet of output

For complete result, kindly refer Assignment3.1.log file in the jar file

```
[Console output redirected to file:D:\8547_Assignment3.1_202105_132736.log]
2021-05-20 13:27:36.707

Printing Entries for 2 power 1
----------------------------------
                                    HashMap                 Cuckoo                  Quadratic                       SeparateChaining
Time taken for entry number : 0 is:      9400                    40400                   48100                   109300
Time taken for entry number : 1 is:      3200                    18700                   18900                   2700

Time taken by each collection is:       6300ns                  29550ns                         33500 ns                        56000 ns
2021-05-20 13:27:36.737

Printing Entries for 2 power 2
----------------------------------
                                    HashMap                 Cuckoo                  Quadratic                       SeparateChaining
Time taken for entry number : 0 is:      6900                    17500                   2300                    3300
Time taken for entry number : 1 is:      1200                    15000                   1000                    1600
Time taken for entry number : 2 is:      900                     52900                   4800                    1200
Time taken for entry number : 3 is:      900                     13800                   800                     9900

Time taken by each collection is:       2475ns                  24800ns                         2225 ns                 4000 ns
2021-05-20 13:27:36.737

Printing Entries for 2 power 3
----------------------------------
                                    HashMap                 Cuckoo                  Quadratic                       SeparateChaining
Time taken for entry number : 0 is:      2600                    12100                   1900                    2200
Time taken for entry number : 1 is:      1100                    12300                   1000                    900
Time taken for entry number : 2 is:      1000                    9600                    700                     700
Time taken for entry number : 3 is:      700                     12000                   1200                    1000
Time taken for entry number : 4 is:      800                     13800                   600                     1100
Time taken for entry number : 5 is:      900                     57700                   4600                    1000
Time taken for entry number : 6 is:      1400                    9100                    500                     800
Time taken for entry number : 7 is:      500                     9300                    500                     1000
```

```
Time taken for entry number : 1048545 is:        0                    800                  300                  300
Time taken for entry number : 1048546 is:        100                  1500                 400                  900
Time taken for entry number : 1048547 is:        100                  1000                 700                  500
Time taken for entry number : 1048548 is:        100                  1100                 400                  300
Time taken for entry number : 1048549 is:        100                  1100                 400                  600
Time taken for entry number : 1048550 is:        0                    900                  700                  800
Time taken for entry number : 1048551 is:        0                    1000                 100                  200
Time taken for entry number : 1048552 is:        0                    1000                 300                  500
Time taken for entry number : 1048553 is:        0                    1000                 200                  300
Time taken for entry number : 1048554 is:        100                  1000                 100                  300
Time taken for entry number : 1048555 is:        100                  1000                 400                  600
Time taken for entry number : 1048556 is:        100                  1200                 400                  600
Time taken for entry number : 1048557 is:        100                  1000                 100                  300
Time taken for entry number : 1048558 is:        0                    1200                 100                  400
Time taken for entry number : 1048559 is:        0                    1200                 100                  400
Time taken for entry number : 1048560 is:        0                    1000                 200                  900
Time taken for entry number : 1048561 is:        0                    1100                 100                  500
Time taken for entry number : 1048562 is:        0                    1000                 100                  300
Time taken for entry number : 1048563 is:        0                    900                  100                  600
Time taken for entry number : 1048564 is:        200                  1300                 200                  500
Time taken for entry number : 1048565 is:        100                  1400                 500                  500
Time taken for entry number : 1048566 is:        100                  1700                 200                  300
Time taken for entry number : 1048567 is:        100                  1300                 400                  300
Time taken for entry number : 1048568 is:        100                  1200                 400                  600
Time taken for entry number : 1048569 is:        100                  700                  400                  600
Time taken for entry number : 1048570 is:        0                    1100                 0                    500
Time taken for entry number : 1048571 is:        0                    1700                 400                  600
Time taken for entry number : 1048572 is:        0                    1100                 600                  700
Time taken for entry number : 1048573 is:        0                    1000                 300                  900
Time taken for entry number : 1048574 is:        0                    900                  400                  600
Time taken for entry number : 1048575 is:        100                  1000                 200                  300

Time taken by each collection is:       86ns                 1198ns               345 ns               404 ns
2021-05-20 13:28:50.364
```



Value of n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Hash Table Avg Time	6300	2475	1125	993	609	650	948	919	482	608	449	492	415	228	208	287	97	94	110	86
Cuckoo Hash Table Avg Time	29550	24800	16987	13881	3646	4379	3304	4395	1217	1627	1108	782	1650	482	777	533	787	691	896	1198
Quadratic Hash Table Avg Time	33500	2225	1375	925	712	1198	1240	1475	532	437	416	400	254	168	236	153	173	254	312	345
Separate Chaining Avg time	56000	4000	1087	737	646	921	1038	925	1066	364	299	337	142	129	230	191	198	258	320	404

Answer 3.2
Please refer to the package *assignmentSolution* class AssignmentSolutionThreeSearchAndDelete and method solution for the solution.

The Steps followed for resolving are as follows:

A. Using question 3.1 generate Hash table, cuckoo hash table, Quadratic Hash Table and Separate Chaining Hash Table and insert a list of j random strings.
B. For Each n = 2^i, i = 1, …,20, generate a list of random string using Answer 3.1 step A, Answer 3.1 Step B and Question 1, Step A.
C. Search the list in Hash Table, Cuckoo Hash Table, Quadratic Hash Table, Separate Chaining Hash Table generated in step A

Result Obtained: As the value of n increases, the average time taken by each collection decreases. However, Quadratic Hash Table has performed the best as the average time taken to search 2^20 entries are only 87 ns as compared to cuckoo hash table and separate chaining hash table, which have taken 391ns and 1293 ns for searching the same number of entries.

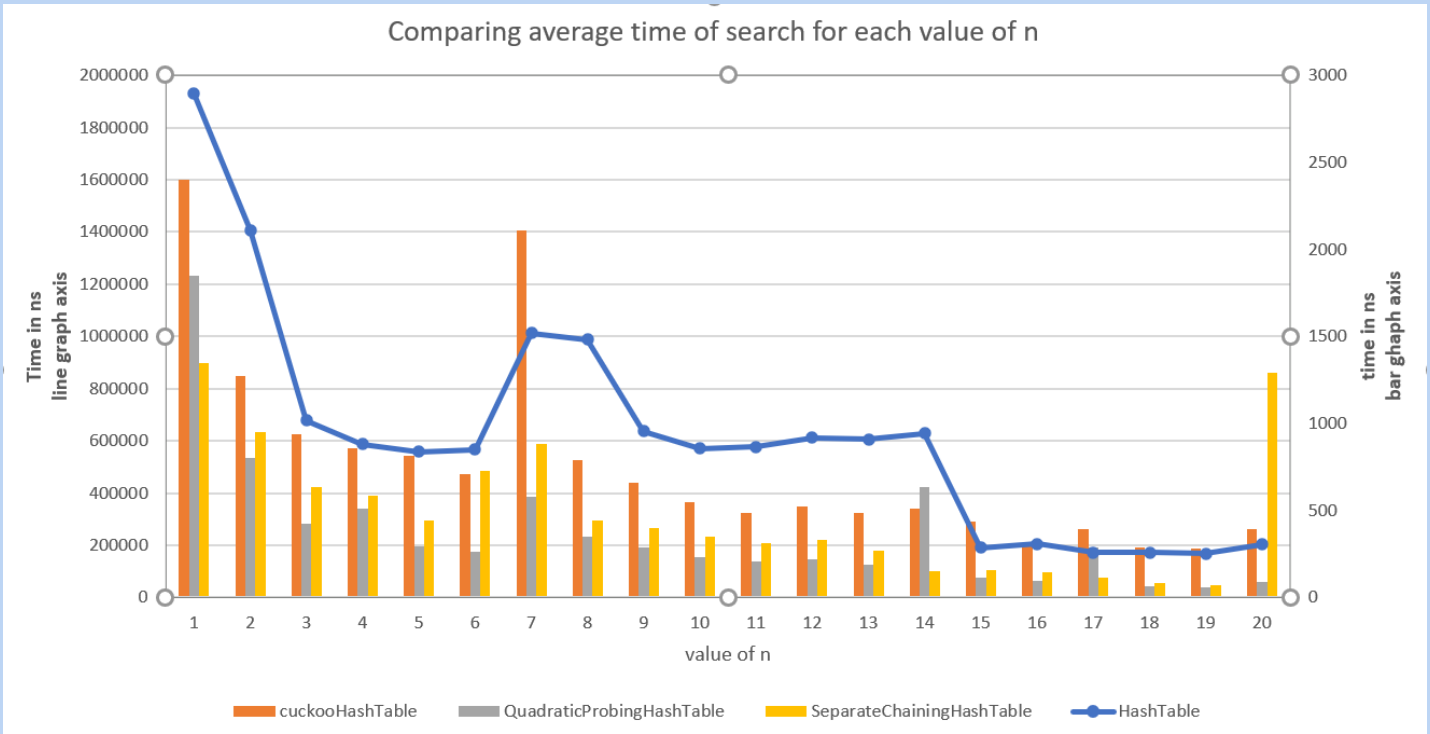
Output Screenshot

For complete result, kindly refer Assignment3.2.log file in the jar file

```
[Console output redirected to file:D:\8547_Assignment3.2_202105_164918.log]

Printing Entries for 2 power 1
----------------------------------
                                HashMap                 Cuckoo                  Quadratic                       SeparateChaining
Time taken for entry number : 0 is:      2428500                 3600                    3300                    2300
Time taken for entry number : 1 is:      1429500                 1200                    400                     400

Time taken by each collection is:       1929000ns               2400ns                  1850 ns                 1350 ns
2021-05-20 16:49:18.939

Printing Entries for 2 power 2
----------------------------------
                                HashMap                 Cuckoo                  Quadratic                       SeparateChaining
Time taken for entry number : 0 is:      2484500                 2000                    1600                    1900
Time taken for entry number : 1 is:      1327600                 1100                    400                     700
Time taken for entry number : 2 is:      940900          1100                    800                     500
Time taken for entry number : 3 is:      878200          900                     400                     700

Time taken by each collection is:       1407800ns               1275ns                  800 ns                  950 ns
2021-05-20 16:49:18.949

Printing Entries for 2 power 3
----------------------------------
                                HashMap                 Cuckoo                  Quadratic                       SeparateChaining
Time taken for entry number : 0 is:      917200          1500                    1200                    1900
Time taken for entry number : 1 is:      815500          800                     300                     500
Time taken for entry number : 2 is:      746200          1200                    200                     300
Time taken for entry number : 3 is:      639700          600                     400                     500
Time taken for entry number : 4 is:      645100          800                     400                     600
Time taken for entry number : 5 is:      562600          800                     300                     400
Time taken for entry number : 6 is:      559500          900                     300                     500
Time taken for entry number : 7 is:      537800          900                     300                     400

Time taken for entry number : 1048546 is:       218800          400             0               100
Time taken for entry number : 1048547 is:       218700          400             100             200
Time taken for entry number : 1048548 is:       218700          400             100             100
Time taken for entry number : 1048549 is:       219700          400             0               200
Time taken for entry number : 1048550 is:       218300          1400            100             100
Time taken for entry number : 1048551 is:       218700          400             0               100
Time taken for entry number : 1048552 is:       218500          400             100             200
Time taken for entry number : 1048553 is:       227200          400             100             100
Time taken for entry number : 1048554 is:       209600          400             100             200
Time taken for entry number : 1048555 is:       210700          300             100             100
Time taken for entry number : 1048556 is:       211500          500             0               100
Time taken for entry number : 1048557 is:       210900          300             0               100
Time taken for entry number : 1048558 is:       211000          400             100             100
Time taken for entry number : 1048559 is:       206400          400             100             100
Time taken for entry number : 1048560 is:       210800          300             100             100
Time taken for entry number : 1048561 is:       211100          300             0               200
Time taken for entry number : 1048562 is:       211300          400             100             100
Time taken for entry number : 1048563 is:       211500          400             1000            200
Time taken for entry number : 1048564 is:       210900          500             100             100
Time taken for entry number : 1048565 is:       210900          400             0               100
Time taken for entry number : 1048566 is:       211100          300             100             100
Time taken for entry number : 1048567 is:       211300          400             0               200
Time taken for entry number : 1048568 is:       211300          300             200             100
Time taken for entry number : 1048569 is:       212100          400             0               100
Time taken for entry number : 1048570 is:       211400          300             100             0
Time taken for entry number : 1048571 is:       211200          400             100             200
Time taken for entry number : 1048572 is:       210900          300             0               0
Time taken for entry number : 1048573 is:       208200          400             100             100
Time taken for entry number : 1048574 is:       211000          400             100             200
Time taken for entry number : 1048575 is:       210900          400             100             100

Time taken by each collection is:       203508ns                391ns           87 ns           1293 ns
2021-05-20 16:57:37.485
```

n	HashTable	cuckooHashTable	QuadraticProbingHashTable	SeparateChainingHashTable
1	1929000	2400	1850	1350
2	1407800	1275	800	950
3	677950	937	425	637
4	588100	856	512	587
5	558659	815	296	443
6	566645	709	265	728
7	1011986	2108	578	885
8	987964	788	347	441
9	636654	657	286	399
10	570505	547	235	347
11	576492	485	210	312
12	612965	525	219	330
13	605024	485	187	268
14	627934	513	635	152
15	190520	437	115	155
16	204747	313	98	148
17	172135	396	286	117
18	172224	285	64	81
19	168768	284	61	73
20	203508	391	87	1293

Data for Average time taken for searching 2^n entries in each collection



Question 4: Use the Java classes BinarySearchTree, AVLTree, RedBlackBST, SplayTree given in class. For each tree:

a) Insert 100,000 integer keys, from 1 to 100,000 (in that order). Find the average time for each insertion. Note: you can add the following VM arguments to your project: -Xss16m. This will help increase the size of the recursion stack.
b) b. Do 100,000 searches of random integer keys between 1 and 100,000. Find the average time of each search.
c) c. Delete all the keys in the trees, starting from 100,000 down to 1 (in that order). Find the average time of each deletion.

Answer:

Please refer to the package *assignmentSolution* class AssignmentSolutionFour and method main for the solution.

Result Obtained: I have commented on the result obtained collectively in question 6.

Output (Time in ns)

```
[Console output redirected to file:D:\Pro
Binary Search Tree
Insertion
---------
Worst case time :11076100
Avg Time Taken 757248
Search
---------
Worst case time :4444200
average time taken: 95472
Deletion
---------
Worst case time :34029600
average time taken: 1071680

AVL Tree
Insertion
---------
Worst case time :1228600
Avg Time Taken 727
Search
---------
Worst case time :119200
average time taken: 686
Deletion
---------
Worst case time :118000
average time taken: 634
```

```
Red-Black Tree
Insertion
---------
Worst case time :1531600
Avg Time Taken 1018
Search
---------
Worst case time :113500
average time taken: 872
Deletion
---------
Worst case time :221400
average time taken: 1631

Splay Tree
Insertion
---------
Worst case time :43100
Avg Time Taken 345
Search
---------
Worst case time :14007300
average time taken: 1099
deletion
---------
Worst case time :100600
average time taken: 368
```

Question 5. Use the Java classes BinarySearchTree, AVLTree, RedBlackBST, SplayTree given in class. For each tree:

a. Insert 100,000 integer keys, from 1 to 100,000 (in that order). Find the average time for each insertion. Note: you can add the following VM arguments to your project: -Xss16m. This will help increase the size of the recursion stack.

b. Do 100,000 searches of random integer keys between 1 and 100,000. Find the average time of each search.

c. Delete all the keys in the trees, starting from 100,000 down to 1 (in that order). Find the average time of each deletion.

Answer 5:
Please refer to the package *assignmentSolution* class AssignmentSolutionFive and method main for the solution

Result Obtained: I have commented on the result obtained collectively in question 6.

Output (Time in ns)

```
Binary Search Tree
Insertion
---------
Worst case time :856200
Avg Time Taken 287
Deletion
---------
Worst case time :25500
average time taken: 126

AVL Tree
Insertion
---------
Worst case time :299900
Avg Time Taken 270
Deletion
---------
Worst case time :47400
average time taken: 187

Red-Black Tree
Insertion
---------
Worst case time :383000
Avg Time Taken 355
Deletion
---------
symbol table does not contain 4462
symbol table does not contain 2198
symbol table does not contain 851
symbol table does not contain 1411
symbol table does not contain 8106
symbol table does not contain 1980
symbol table does not contain 9703
```

```
symbol table does not contain 7013
symbol table does not contain 3288
symbol table does not contain 7546
symbol table does not contain 6200
symbol table does not contain 7558
symbol table does not contain 8845
symbol table does not contain 1548
symbol table does not contain 6401
symbol table does not contain 1048
symbol table does not contain 4947
symbol table does not contain 1779
symbol table does not contain 8727
symbol table does not contain 7808
symbol table does not contain 7003
symbol table does not contain 3182
symbol table does not contain 1927
Worst case time :62286800
average time taken: 6923

Splay Tree
Insertion
---------
Worst case time :25800
Avg Time Taken 307
Deletion
---------
Worst case time :27100
average time taken: 136
```

Question 6: Draw a table that contains all the average times found in #4 and #5. Comment on the results obtained and compare them with the worst-case and average-case running times of each operation for each tree. Which tree will you use in your implementations for real problems? Note: you decide on the format of the table (use your creativity to present the results in the best possible way). =
Answer 6:

Average time and worst-case time for question no. 4

Time in ns

Type Of Tree	Insertion Average Time	Insertion Worst Time	Searching Average Time	Searching Worst Time	Deletion Average Time	Deletion Worst Time
Binary Search Tree	757248	11076100	95472	4444200	1071680	34029600
AVL Tree	727	1228600	686	119200	634	118000
Red Black tree	1018	1531600	872	113500	1631	221400
Splay Tree	345	43100	1099	14007300	368	100600

Average time and worst-case time for question no. 5

Time in ns

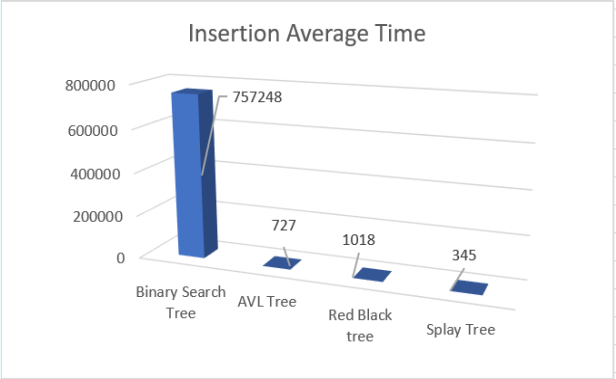
Type Of Tree	Random Insertion Average Time	Random Insertion Worst Time	Random Deletion Average Time	Random Deletion Worst Time
Binary Search Tree	287	856200	126	25500
AVL Tree	270	299900	187	47400
Red Black tree	355	383000	6923	62286800
Splay Tree	307	25800	136	27100

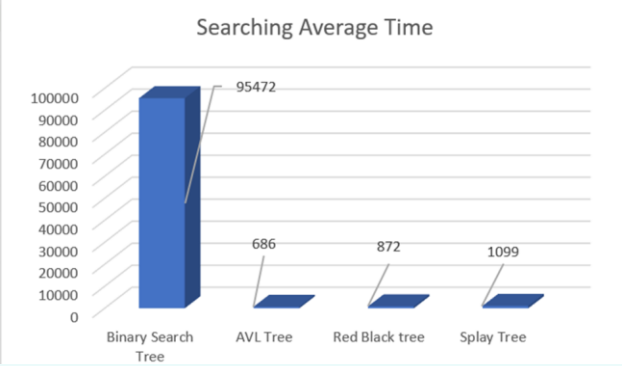
Looking at the average **insertion** time (in order), we can see that both average time and worst time for insertion is the greatest for binary search tree and Splay Tree has performed the best. Comparing the same with the insertion time taken for randomly generated values, AVL search tree has taken the least average time followed by Splay Tree and Red Black Tree and Binary Search Tree. But, when we compare the average times with the worst-case scenarios, the trend is not the same. Although, the binary search tree has still has taken maximum time**, Splay tree** has performed the best as it took only 43100 ns and 25800 ns for insertion in order and random insertion respectively.

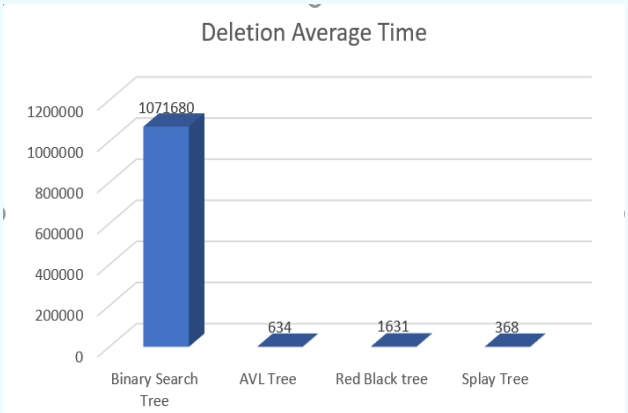
For **Searching** time too, the performance of Binary search tree is the lowest as both average search time and worst-case time are the highest. For the search operation **AVL** tree has performed the best followed by Red Black tree and Splay Tree in that order. However, when we see the worst-case scenario, Red-Black tree has taken the least amount of time.

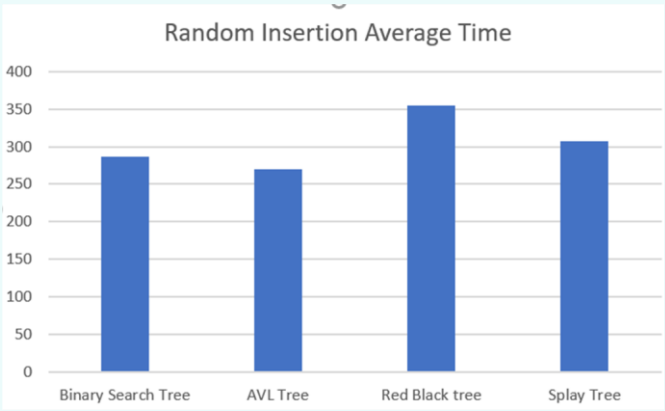
Similarly, for the **deletion** operation too, Binary search tree has performed the least and **AVL** tree has the best performance when the insertion was in order. However, while we were performing the deletion operation randomly, **Binary search tree and Splay Tree** has performed the best and performance of red black tree was the lowest. And, the same can be said about the worst-case times as well.

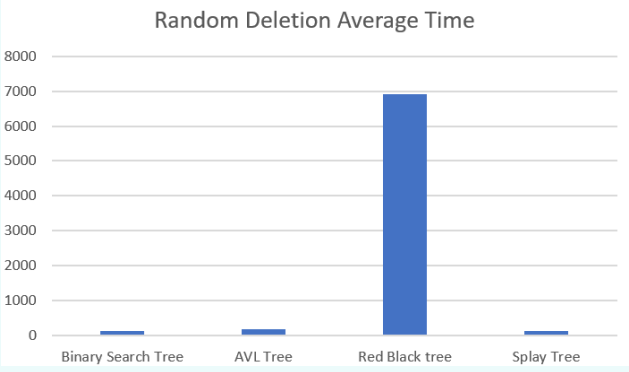
All these observations, have been plotted in graphs below and can be verified. In My Opinion, **AVL tree and splay tree** should be used for real time problems. Although, others have performed better in some of the scenarios, but the performance of the AVL tree and splay tree has been consistent.











References:

1. MrLore (StackExchange user name), “Answer to Question, ‘Generating a random number between multiple ranges’”. https://stackoverflow.com/questions/15591173/generating-a-random-number-between-multiple-ranges/15591239.