Info6305: Program Structure & Algorithms

HashTable Report

Abhishek Dongare
NUID:001356287

HashTable: Hash table (hash map) is a data structure that implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found.

Ideally, the hash function will assign each key to a unique bucket, but most hash table designs employ an imperfect hash function, which might cause hash collisions where the hash function generates the same index for more than one key. Such collisions must be accommodated in some way.

In a well-dimensioned hash table, the average cost (number of instructions) for each lookup is independent of the number of elements stored in the table. Many hash table designs also allow arbitrary insertions and deletions of key-value pairs, at (amortized[2]) constant average cost per operation.[3][4]

In many situations, hash tables turn out to be on average more efficient than search trees or any other table lookup structure. For this reason, they are widely used in many kinds of computer software, particularly for associative arrays, database indexing, caches, and sets.

Hashing: The idea of hashing is to distribute the entries (key/value pairs) across an array of buckets. Given a key, the algorithm computes an index that suggests where the entry can be found:

```
index = f(key, array_size)

hash = hashfunc(key)
index = hash % array_size
```

Linear Probing: Linear probing is a component of open addressing schemes for using a hash table to solve the dictionary problem. In the dictionary problem, a data structure should maintain a collection of key–value pairs subject to operations that insert or delete pairs from the collection or that search for the value associated with a given key. In open addressing solutions to this problem, the data structure is an array T (the hash table) whose cells T[i] (when nonempty) each store a single key–value pair. A hash function is used to map each key into the cell of T where that key should be stored, typically scrambling the keys so that keys with similar values are not placed near each other in the table. A hash collision occurs when the hash function maps a key into a cell that is already occupied by a different key. Linear probing is a strategy for resolving collisions, by placing the new key into the closest following empty cell.[3][4]

Search

To search for a given key x, the cells of T are examined, beginning with the cell at index h(x) (where h is the hash function) and continuing to the adjacent cells h(x) + 1, h(x) + 2, ..., until finding either an empty cell or a cell whose stored key is x. If a cell containing the key is found, the search returns the value from that cell. Otherwise, if an empty cell is found, the key cannot be in the table, because it would have been placed in that cell in preference to any later cell that has not yet been searched. In this case, the search returns as its result that the key is not present in the dictionary.[3][4]

Insertion

To insert a key–value pair (x,v) into the table (possibly replacing any existing pair with the same key), the insertion algorithm follows the same sequence of cells that would be followed for a search, until finding either an empty cell or a cell whose stored key is x. The new key–value pair is then placed into that cell.[3][4]

If the insertion would cause the load factor of the table (its fraction of occupied cells) to grow above some preset threshold, the whole table may be replaced by a new table, larger by a constant factor, with a new hash function, as in a dynamic array. Setting this threshold close to zero and using a high growth rate for the table size leads to faster hash table operations but greater memory usage than threshold values close to one and low growth rates. A common choice would be to double the table size when the load factor would exceed 1/2, causing the load factor to stay between 1/4 and 1/2.

Double Hashing: Double hashing with open addressing is a classical data structure on a table {\displaystyle T} T. Let {\displaystyle n} n be the number of elements stored in {\displaystyle T} T, then {\displaystyle T} T's load factor is {\displaystyle \alpha ={\frac {n}{|T|}}} \alpha = \frac{n}{|T|}.

Double hashing approximates uniform open address hashing. That is, start by randomly, uniformly and independently selecting two universal hash functions {\displaystyle h_{1}} h_{1} and {\displaystyle h_{2}} h_{2} to build a double hashing table {\displaystyle T} T.

All elements are put in {\displaystyle T} T by double hashing using {\displaystyle h_{1}} h_{1} and {\displaystyle h_{2}} h_{2}. Given a key {\displaystyle k} k, determining the {\displaystyle (i+1)} (i+1)-st hash location is computed by:

{\displaystyle h(i,k)=(h_{1}(k)+i\cdot h_{2}(k))\mod |T|.} h(i,k) = (h_1(k) + i \cdot h_2(k)) \mod |T|.

Let {\displaystyle T} T have fixed load factor {\displaystyle \alpha :1>\alpha >0} \alpha: 1 > \alpha > 0. Bradford and Katehakis[1] showed the expected number of probes for an unsuccessful search in {\displaystyle T} T, still using these initially chosen hash functions, is {\displaystyle {\frac {1}{1-\alpha }}} \frac{1}{1-\alpha} regardless of the distribution of the inputs. More precisely, these two uniformly, randomly and independently chosen hash functions are chosen from a set of universal hash functions where pairwise independence suffices.

Previous results include: Guibas and Szemerédi[2] showed {\displaystyle {\frac {1}{1-\alpha }}} \frac{1}{1-\alpha} holds for unsuccessful search for load factors {\displaystyle \alpha <0.319} \alpha < 0.319. Also, Lueker and Molodowitch[3] showed this held assuming ideal randomized functions. Schmidt and Siegel[4] showed this with {\displaystyle k} k-wise independent and uniform functions (for {\displaystyle k=c\log n} k = c \log n, and suitable constant {\displaystyle c} c).

Load Factor:
A critical statistic for a hash table is the load factor, defined as

LoadFactor = N / K.

where

n is the number of entries occupied in the hash table.
k is the number of buckets.

As the load factor grows larger, the hash table becomes slower, and it may even fail to work (depending on the method used). The expected constant time property of a hash table assumes that the load factor is kept below some bound. For a fixed number of buckets, the time for a lookup grows with the number of entries and therefore the desired constant time is not achieved. As a real-world example, the default load factor for a HashMap in Java 10 is 0.75, which "offers a good tradeoff between time and space costs."[10]

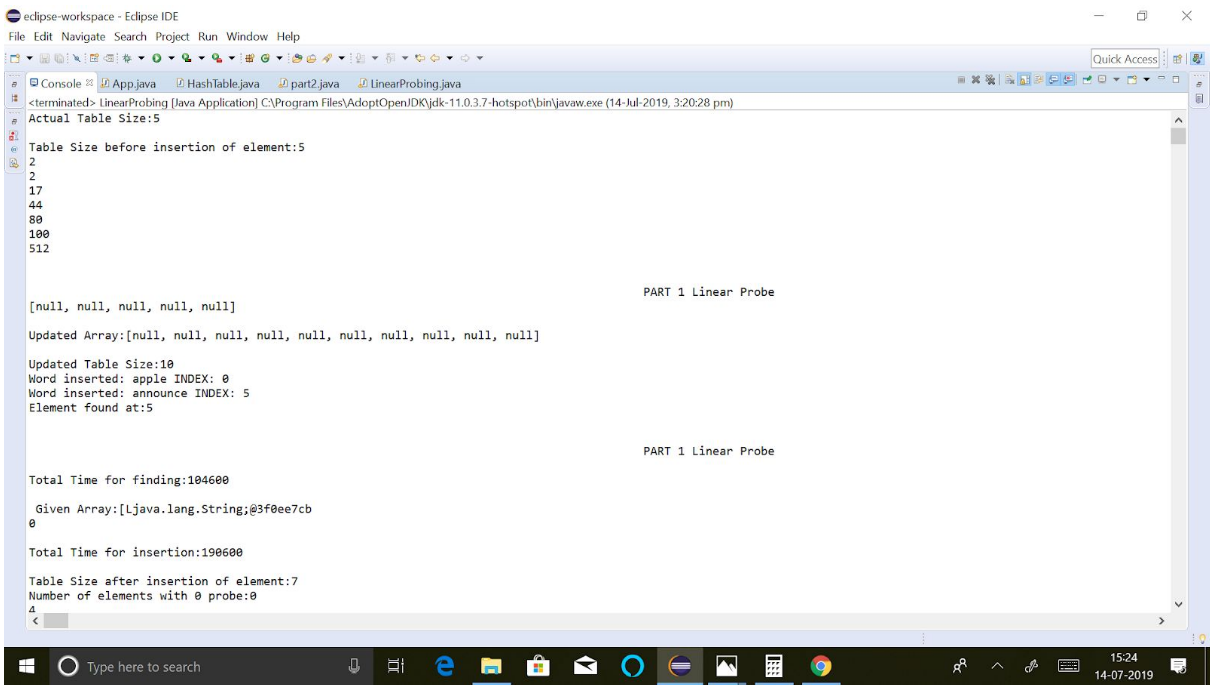
Second to that, one can examine the variance of number of entries per bucket. For example, two tables both have 1,000 entries and 1,000 buckets; one has exactly one entry in each bucket, the other has all entries in the same bucket. Clearly the hashing is not working in the second one.

A low load factor is not especially beneficial. As the load factor approaches 0, the proportion of unused areas in the hash table increases, but there is not necessarily any reduction in search cost. This results in wasted memory.

Empirical Analysis:

Linear probing:

Load Factor: 0.1

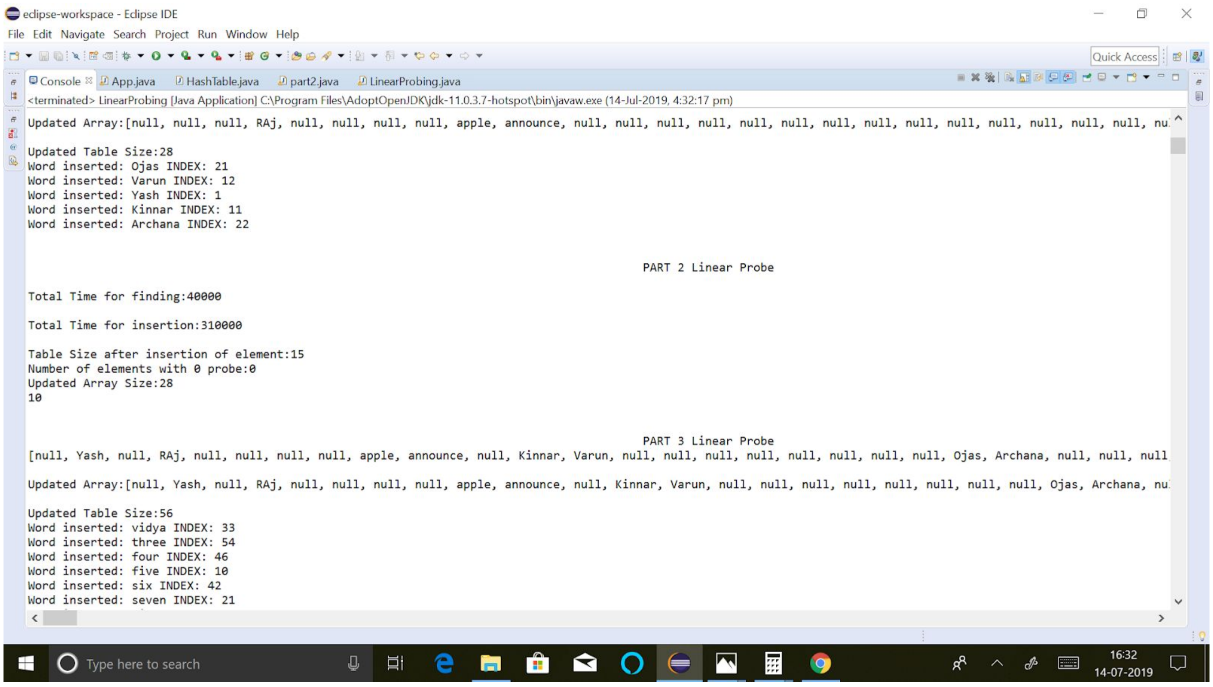


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 3:20:28 pm)
Actual Table Size:5

Table Size before insertion of element:5
2
2
17
44
80
100
512


                                                                                        PART 1 Linear Probe
[null, null, null, null, null]

Updated Array:[null, null, null, null, null, null, null, null, null, null]

Updated Table Size:10
Word inserted: apple INDEX: 0
Word inserted: announce INDEX: 5
Element found at:5


                                                                                        PART 1 Linear Probe

Total Time for finding:104600

 Given Array:[Ljava.lang.String;@3f0ee7cb
0

Total Time for insertion:190600

Table Size after insertion of element:7
Number of elements with 0 probe:0
4
```

Time for insertion: 190600 Nano seconds.

Time for finding: 104600 Nano seconds.

Load Factor: 0.2

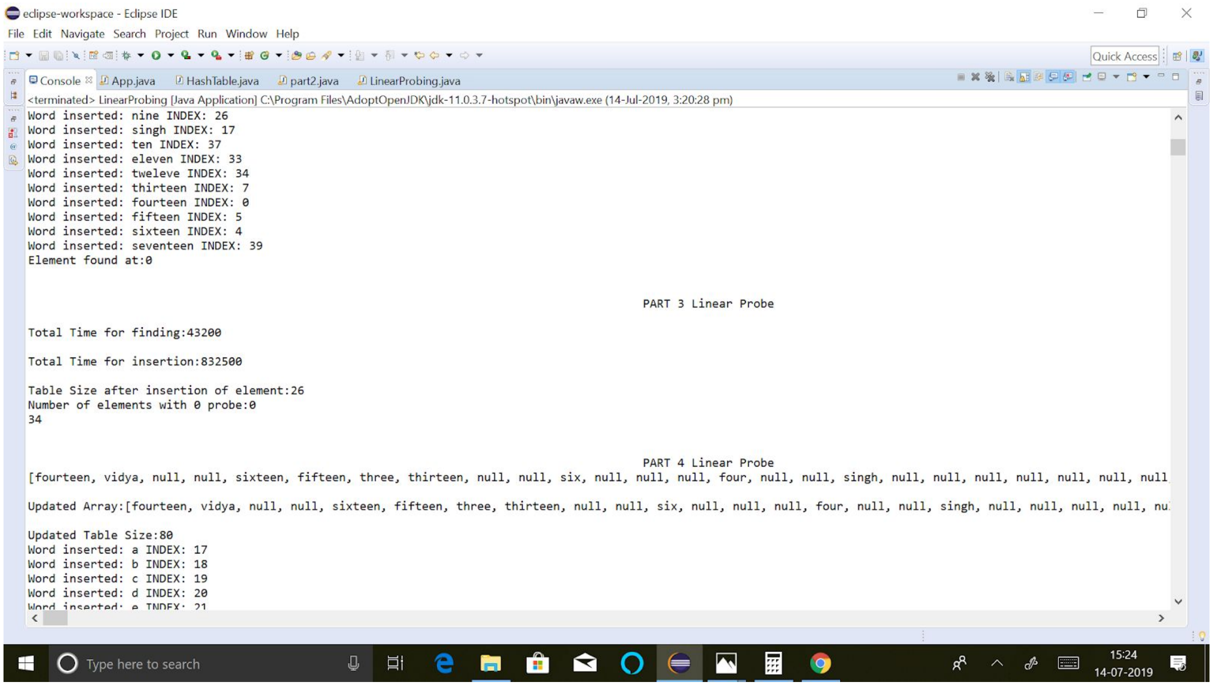


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 4:32:17 pm)
Updated Array:[null, null, null, RAj, null, null, null, null, apple, announce, null, null, null, null, null, null, null, null, null, null, null, null, null, null, nu

Updated Table Size:28
Word inserted: Ojas INDEX: 21
Word inserted: Varun INDEX: 12
Word inserted: Yash INDEX: 1
Word inserted: Kinnar INDEX: 11
Word inserted: Archana INDEX: 22


                                                                                         PART 2 Linear Probe

Total Time for finding:40000

Total Time for insertion:310000

Table Size after insertion of element:15
Number of elements with 0 probe:0
Updated Array Size:28
10


                                                                                         PART 3 Linear Probe
[null, Yash, null, RAj, null, null, null, null, apple, announce, null, Kinnar, Varun, null, null, null, null, null, null, null, null, Ojas, Archana, null, null, null

Updated Array:[null, Yash, null, RAj, null, null, null, null, apple, announce, null, Kinnar, Varun, null, null, null, null, null, null, null, null, Ojas, Archana, nu

Updated Table Size:56
Word inserted: vidya INDEX: 33
Word inserted: three INDEX: 54
Word inserted: four INDEX: 46
Word inserted: five INDEX: 10
Word inserted: six INDEX: 42
Word inserted: seven INDEX: 21
```

Time for insertion:310000 Nano seconds.

Time for finding: 40000 Nano seconds.

Load Factor: 0.3

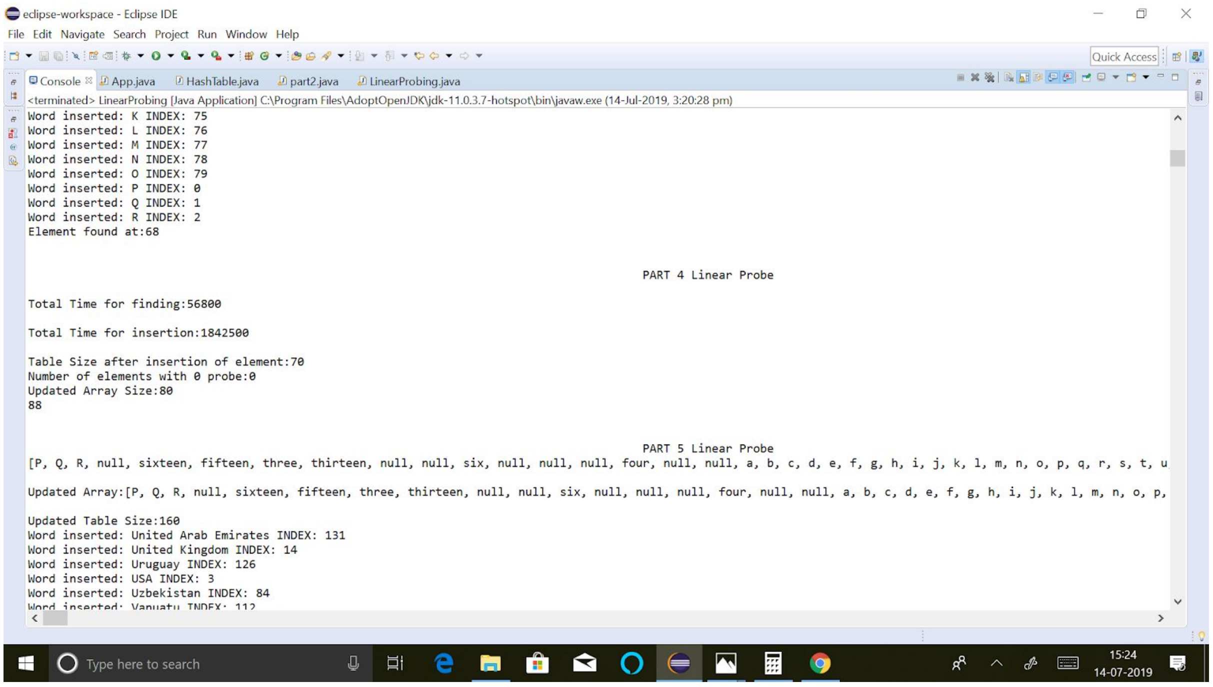


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 3:20:28 pm)
Word inserted: nine INDEX: 26
Word inserted: singh INDEX: 17
Word inserted: ten INDEX: 37
Word inserted: eleven INDEX: 33
Word inserted: tweleve INDEX: 34
Word inserted: thirteen INDEX: 7
Word inserted: fourteen INDEX: 0
Word inserted: fifteen INDEX: 5
Word inserted: sixteen INDEX: 4
Word inserted: seventeen INDEX: 39
Element found at:0


                                                                                         PART 3 Linear Probe

Total Time for finding:43200

Total Time for insertion:832500

Table Size after insertion of element:26
Number of elements with 0 probe:0
34


                                                                                         PART 4 Linear Probe
[fourteen, vidya, null, null, sixteen, fifteen, three, thirteen, null, null, six, null, null, null, four, null, null, singh, null, null, null, null, null, null, null

Updated Array:[fourteen, vidya, null, null, sixteen, fifteen, three, thirteen, null, null, six, null, null, null, four, null, null, singh, null, null, null, null, nu

Updated Table Size:80
Word inserted: a INDEX: 17
Word inserted: b INDEX: 18
Word inserted: c INDEX: 19
Word inserted: d INDEX: 20
Word inserted: e INDEX: 21
```

Time for insertion: 832500Nano seconds.

Time for finding: 43200 Nano seconds.

Load Factor: 0.4

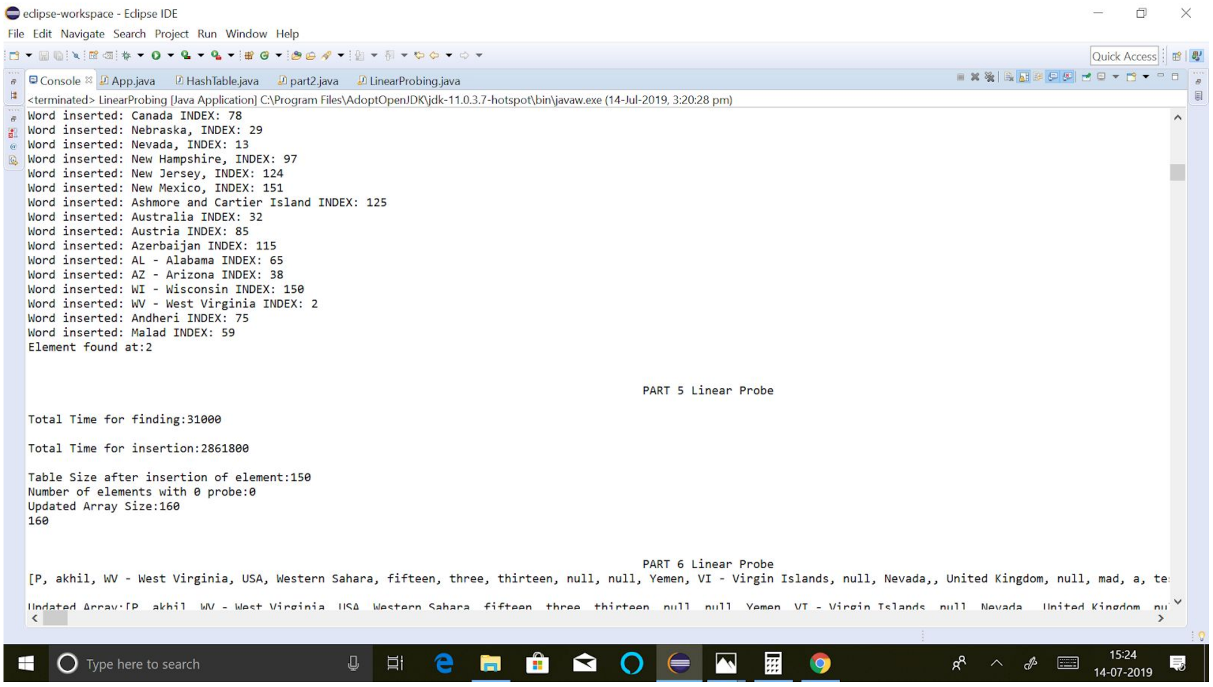


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 3:20:28 pm)
Word inserted: K INDEX: 75
Word inserted: L INDEX: 76
Word inserted: M INDEX: 77
Word inserted: N INDEX: 78
Word inserted: O INDEX: 79
Word inserted: P INDEX: 0
Word inserted: Q INDEX: 1
Word inserted: R INDEX: 2
Element found at:68


                                                                                    PART 4 Linear Probe

Total Time for finding:56800

Total Time for insertion:1842500

Table Size after insertion of element:70
Number of elements with 0 probe:0
Updated Array Size:80
88


                                                                                    PART 5 Linear Probe
[P, Q, R, null, sixteen, fifteen, three, thirteen, null, null, six, null, null, null, four, null, null, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u

Updated Array:[P, Q, R, null, sixteen, fifteen, three, thirteen, null, null, six, null, null, null, four, null, null, a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p,

Updated Table Size:160
Word inserted: United Arab Emirates INDEX: 131
Word inserted: United Kingdom INDEX: 14
Word inserted: Uruguay INDEX: 126
Word inserted: USA INDEX: 3
Word inserted: Uzbekistan INDEX: 84
Word inserted: Vanuatu INDEX: 112
```

Time for insertion: 1842500 Nano seconds.

Time for finding: 56800 Nano seconds.

Load Factor: 0.5

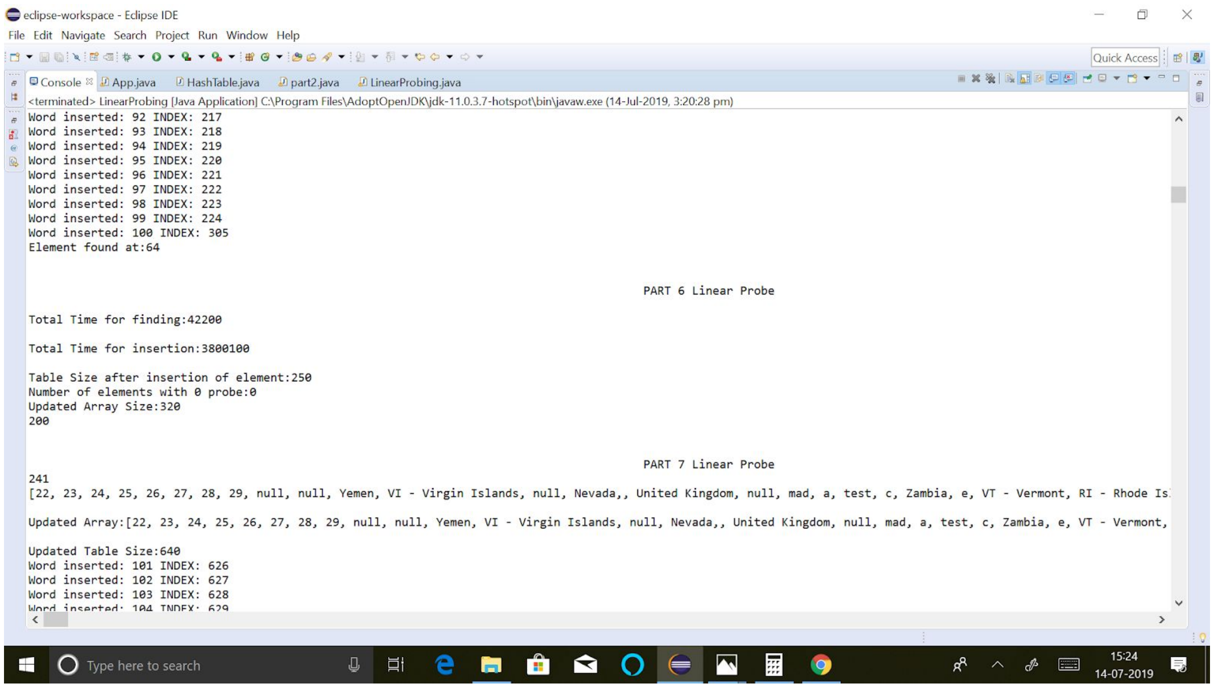


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 3:20:28 pm)
Word inserted: Canada INDEX: 78
Word inserted: Nebraska, INDEX: 29
Word inserted: Nevada, INDEX: 13
Word inserted: New Hampshire, INDEX: 97
Word inserted: New Jersey, INDEX: 124
Word inserted: New Mexico, INDEX: 151
Word inserted: Ashmore and Cartier Island INDEX: 125
Word inserted: Australia INDEX: 32
Word inserted: Austria INDEX: 85
Word inserted: Azerbaijan INDEX: 115
Word inserted: AL - Alabama INDEX: 65
Word inserted: AZ - Arizona INDEX: 38
Word inserted: WI - Wisconsin INDEX: 150
Word inserted: WV - West Virginia INDEX: 2
Word inserted: Andheri INDEX: 75
Word inserted: Malad INDEX: 59
Element found at:2


                                                                                    PART 5 Linear Probe

Total Time for finding:31000

Total Time for insertion:2861800

Table Size after insertion of element:150
Number of elements with 0 probe:0
Updated Array Size:160
160


                                                                                    PART 6 Linear Probe
[P, akhil, WV - West Virginia, USA, Western Sahara, fifteen, three, thirteen, null, null, Yemen, VI - Virgin Islands, null, Nevada,, United Kingdom, null, mad, a, te

Updated Array:[P, akhil, WV - West Virginia, USA, Western Sahara, fifteen, three, thirteen, null, null, Yemen, VI - Virgin Islands, null, Nevada,, United Kingdom, nu
```

Time for insertion: 2861800 Nano seconds.

Time for finding: 31000 Nano seconds.

Load Factor: 0.6

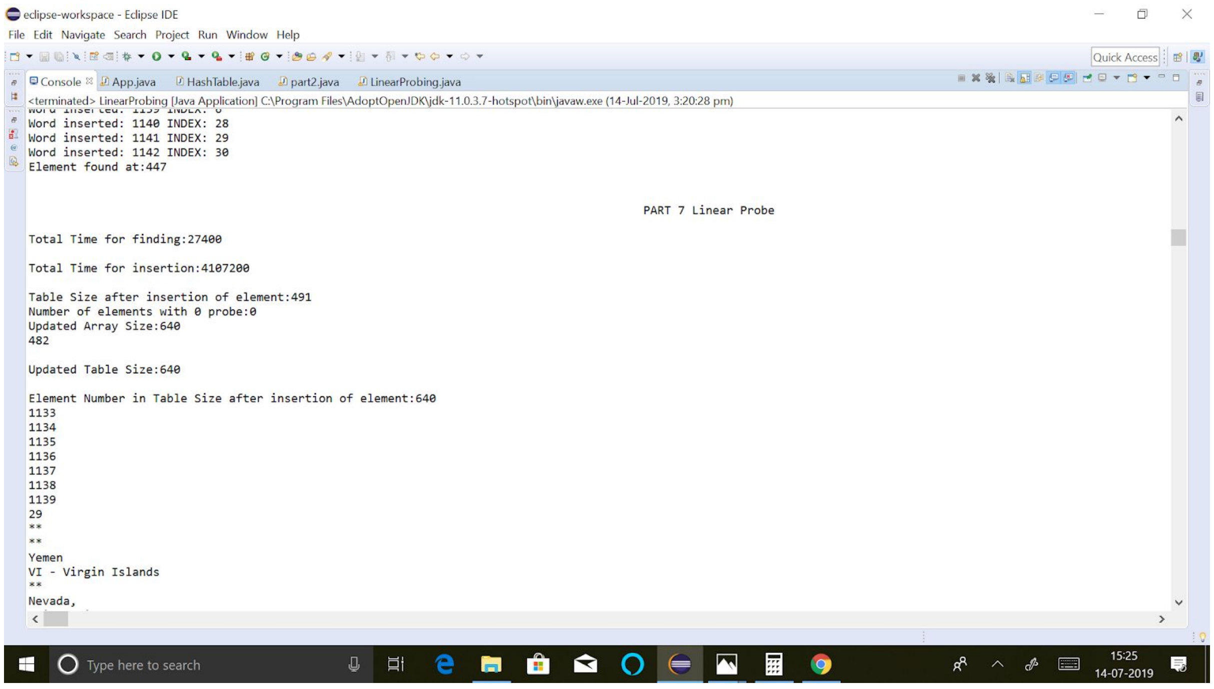


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 3:20:28 pm)
Word inserted: 92 INDEX: 217
Word inserted: 93 INDEX: 218
Word inserted: 94 INDEX: 219
Word inserted: 95 INDEX: 220
Word inserted: 96 INDEX: 221
Word inserted: 97 INDEX: 222
Word inserted: 98 INDEX: 223
Word inserted: 99 INDEX: 224
Word inserted: 100 INDEX: 305
Element found at:64



                                                                                         PART 6 Linear Probe


Total Time for finding:42200

Total Time for insertion:3800100

Table Size after insertion of element:250
Number of elements with 0 probe:0
Updated Array Size:320
200



                                                                                         PART 7 Linear Probe
241
[22, 23, 24, 25, 26, 27, 28, 29, null, null, Yemen, VI - Virgin Islands, null, Nevada,, United Kingdom, null, mad, a, test, c, Zambia, e, VT - Vermont, RI - Rhode Is

Updated Array:[22, 23, 24, 25, 26, 27, 28, 29, null, null, Yemen, VI - Virgin Islands, null, Nevada,, United Kingdom, null, mad, a, test, c, Zambia, e, VT - Vermont,

Updated Table Size:640
Word inserted: 101 INDEX: 626
Word inserted: 102 INDEX: 627
Word inserted: 103 INDEX: 628
Word inserted: 104 INDEX: 629
```

Time for insertion: 3800100 Nano seconds.

Time for finding: 42200 Nano seconds.

Load Factor: 0.75

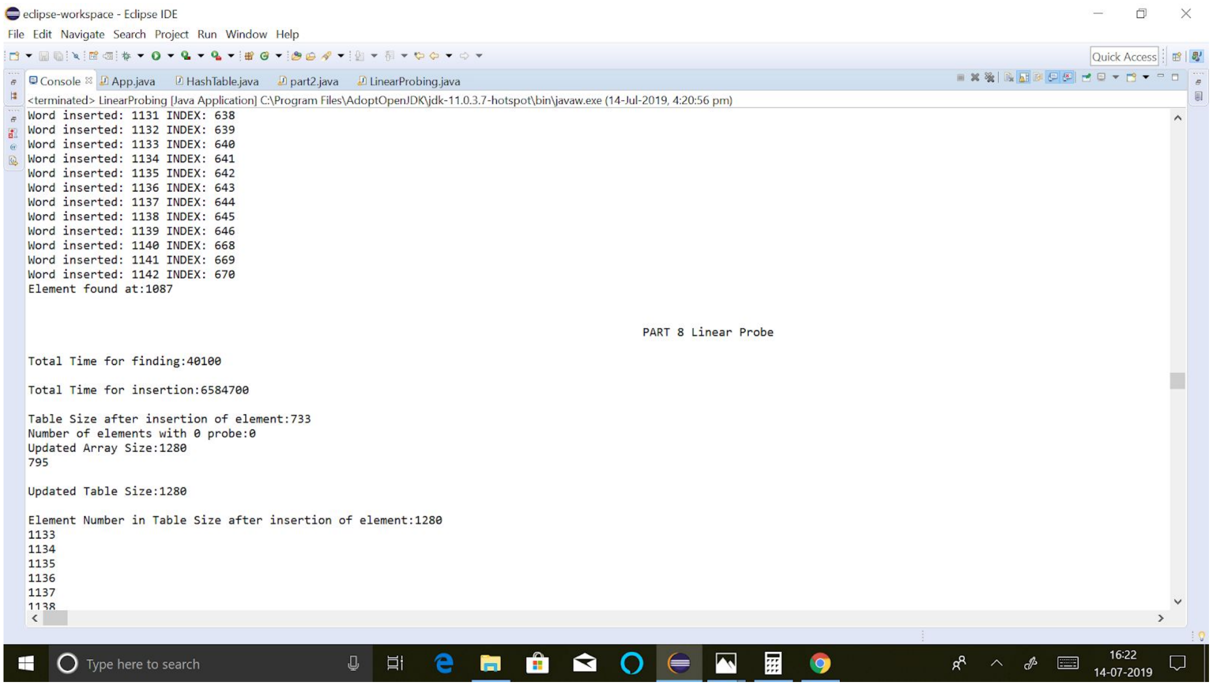


```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 3:20:28 pm)
Word inserted: 1139 INDEX: 0
Word inserted: 1140 INDEX: 28
Word inserted: 1141 INDEX: 29
Word inserted: 1142 INDEX: 30
Element found at:447



                                                                                         PART 7 Linear Probe


Total Time for finding:27400

Total Time for insertion:4107200

Table Size after insertion of element:491
Number of elements with 0 probe:0
Updated Array Size:640
482

Updated Table Size:640

Element Number in Table Size after insertion of element:640
1133
1134
1135
1136
1137
1138
1139
29
**
**
Yemen
VI - Virgin Islands
**
Nevada,
```

Time for insertion: 4107200 Nano seconds.

Time for finding: 27400 Nano seconds.

Load Factor:0.8



```
<terminated> LinearProbing [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 4:20:56 pm)
Word inserted: 1131 INDEX: 638
Word inserted: 1132 INDEX: 639
Word inserted: 1133 INDEX: 640
Word inserted: 1134 INDEX: 641
Word inserted: 1135 INDEX: 642
Word inserted: 1136 INDEX: 643
Word inserted: 1137 INDEX: 644
Word inserted: 1138 INDEX: 645
Word inserted: 1139 INDEX: 646
Word inserted: 1140 INDEX: 668
Word inserted: 1141 INDEX: 669
Word inserted: 1142 INDEX: 670
Element found at:1087


                                                                                         PART 8 Linear Probe


Total Time for finding:40100

Total Time for insertion:6584700

Table Size after insertion of element:733
Number of elements with 0 probe:0
Updated Array Size:1280
795

Updated Table Size:1280

Element Number in Table Size after insertion of element:1280
1133
1134
1135
1136
1137
1138
```

Time for insertion: 6584700 Nano seconds.
Time for finding: 40100 Nano seconds.

Load Factor	Insertion Time	Finding time
0.1	190600	104600
0.2	310000	40000
0.3	832500	43200
0.4	1842500	56800
0.5	2861800	31000
0.6	3800100	42200
0.75	4107200	**27400**
0.8	6584700	40100

Double Hash:
Double hashing with open addressing is a classical data structure on a table {\displaystyle T} T. Let {\displaystyle n} n be the number of elements stored in {\displaystyle T} T, then {\displaystyle T} T's load factor is {\displaystyle \alpha ={\frac {n}{|T|}}} \alpha = \frac{n}{|T|}.

Double hashing approximates uniform open address hashing. That is, start by randomly, uniformly and independently selecting two universal hash functions {\displaystyle h_{1}} h_{1} and {\displaystyle h_{2}} h_{2} to build a double hashing table {\displaystyle T} T.

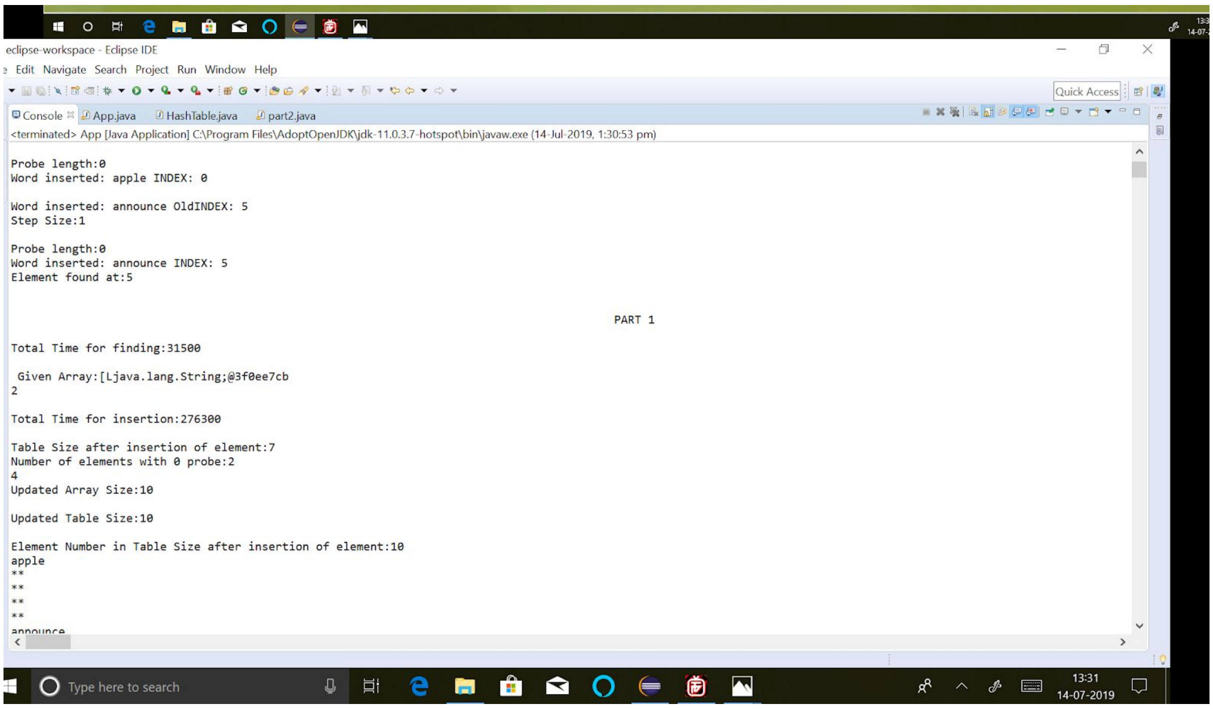
All elements are put in {\displaystyle T} T by double hashing using {\displaystyle h_{1}} h_{1} and {\displaystyle h_{2}} h_{2}. Given a key {\displaystyle k} k, determining the {\displaystyle (i+1)} (i+1)-st hash location is computed by:

{\displaystyle h(i,k)=(h_{1}(k)+i\cdot h_{2}(k))\mod |T|.} h(i,k) = (h_1(k) + i \cdot h_2(k)) \mod |T|.

Let {\displaystyle T} T have fixed load factor {\displaystyle \alpha :1>\alpha >0} \alpha: 1 > \alpha > 0. Bradford and Katehakis[1] showed the expected number of probes for an unsuccessful search in {\displaystyle T} T, still using these initially chosen hash functions, is {\displaystyle {\frac {1}{1-\alpha }}} \frac{1}{1-\alpha} regardless of the distribution of the inputs. More precisely, these two uniformly, randomly and independently chosen hash functions are chosen from a set of universal hash functions where pair-wise independence suffices.

Previous results include: Guibas and Szemerédi[2] showed {\displaystyle {\frac {1}{1-\alpha }}} \frac{1}{1-\alpha} holds for unsuccessful search for load factors {\displaystyle \alpha <0.319} \alpha < 0.319. Also, Lueker and Molodowitch[3] showed this held assuming ideal randomized functions. Schmidt and Siegel[4] showed this with {\displaystyle k} k-wise independent and uniform functions (for {\displaystyle k=c\log n} k = c \log n, and suitable constant {\displaystyle c} c).

Load Factor: 0.1 Number of elements with 0 Probes: 2

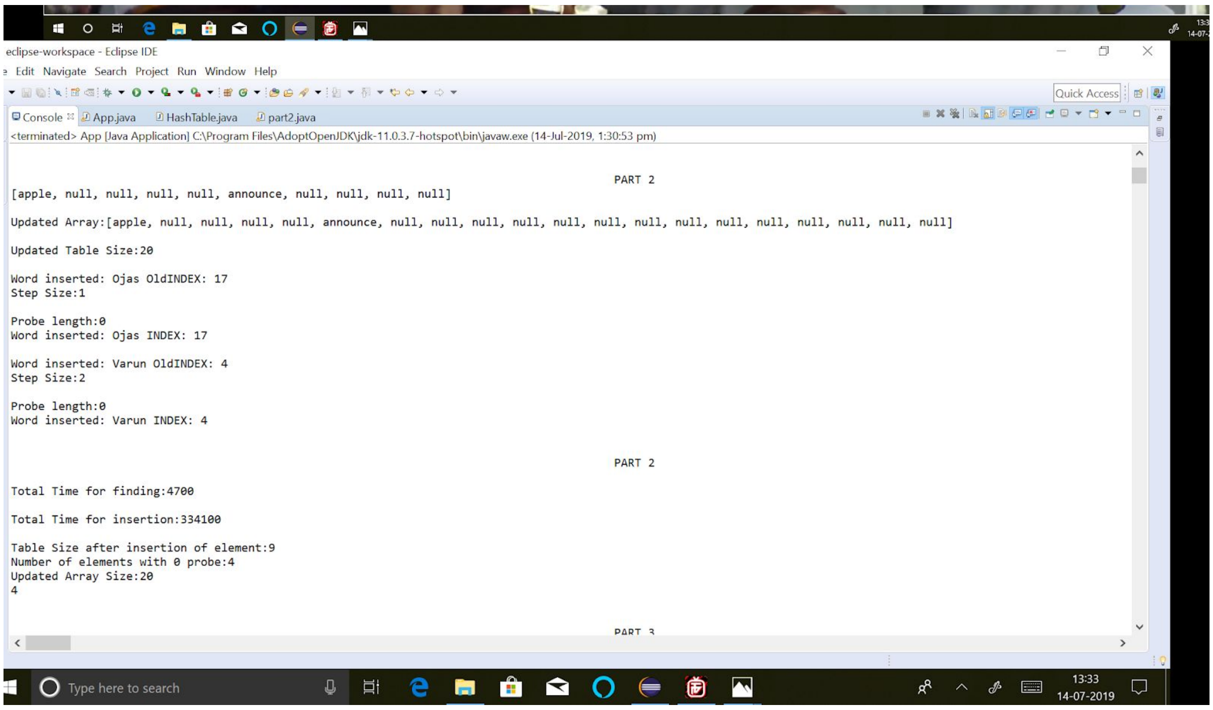


```
Probe length:0
Word inserted: apple INDEX: 0

Word inserted: announce OldINDEX: 5
Step Size:1

Probe length:0
Word inserted: announce INDEX: 5
Element found at:5

                                                                                         PART 1

Total Time for finding:31500

 Given Array:[Ljava.lang.String;@3f0ee7cb
2

Total Time for insertion:276300

Table Size after insertion of element:7
Number of elements with 0 probe:2
4
Updated Array Size:10

Updated Table Size:10

Element Number in Table Size after insertion of element:10
apple
**
**
**
**
announce
```

Time for insertion: 276300 Nano seconds.

Time for finding: 31500 Nano seconds.

Load Factor: 0.2 Number of elements with 0 Probes: 2

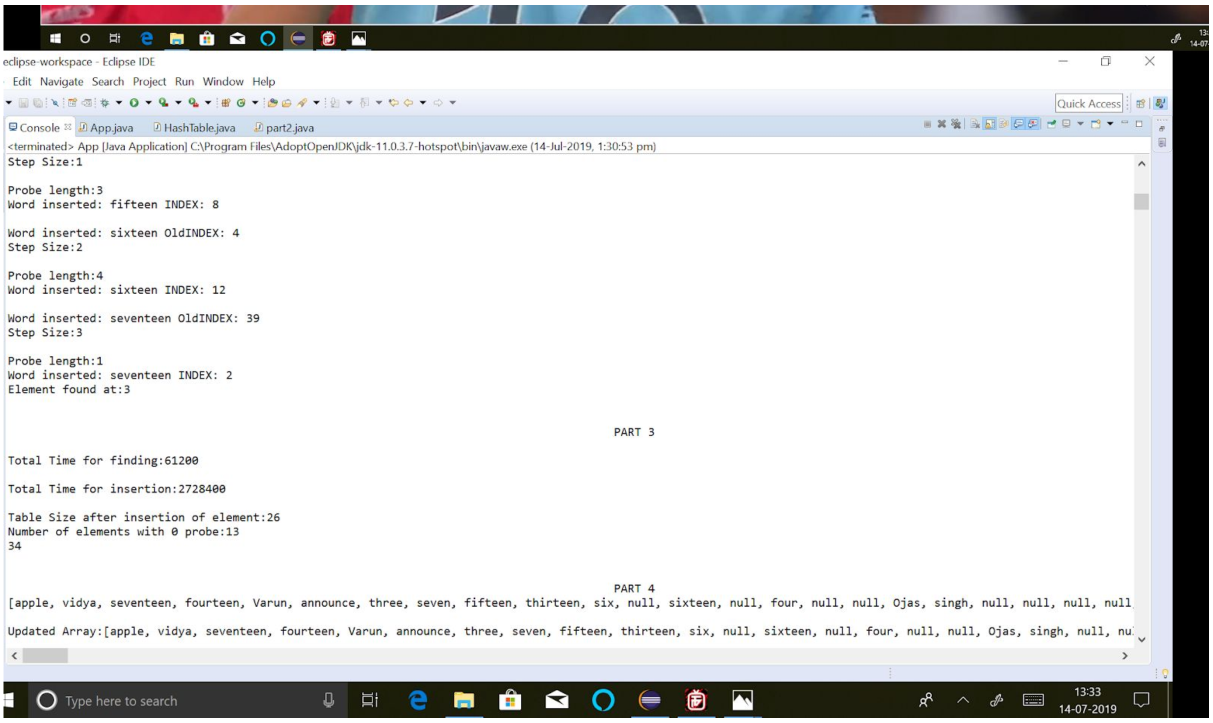


```
                                                                                         PART 2
[apple, null, null, null, null, announce, null, null, null, null]

Updated Array:[apple, null, null, null, null, announce, null, null, null, null, null, null, null, null, null, null, null, null, null, null]

Updated Table Size:20

Word inserted: Ojas OldINDEX: 17
Step Size:1

Probe length:0
Word inserted: Ojas INDEX: 17

Word inserted: Varun OldINDEX: 4
Step Size:2

Probe length:0
Word inserted: Varun INDEX: 4

                                                                                         PART 2

Total Time for finding:4700

Total Time for insertion:334100

Table Size after insertion of element:9
Number of elements with 0 probe:4
Updated Array Size:20
4

                                                                                         PART 3
```

Time for insertion: 334100 Nano seconds.

Time for finding: 4700 Nano seconds.

Load Factor: 0.3 Number of elements with 0 Probes: 9

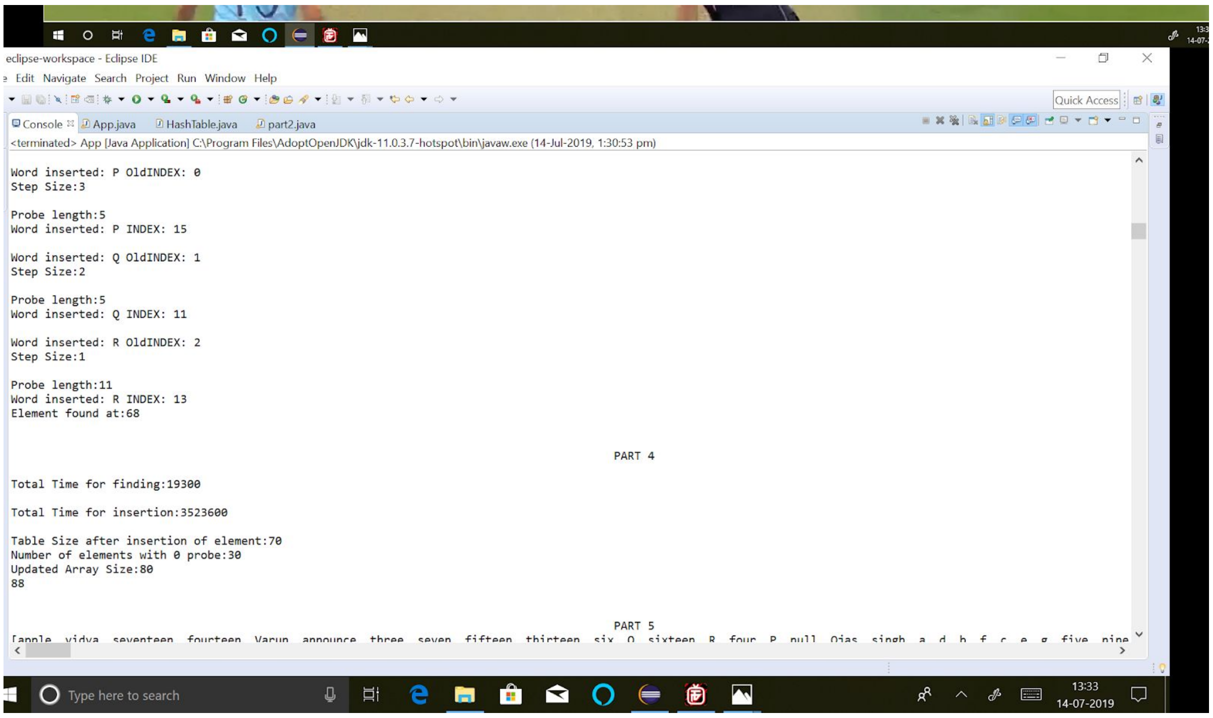


```
<terminated> App [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 1:30:53 pm)
Step Size:1

Probe length:3
Word inserted: fifteen INDEX: 8

Word inserted: sixteen OldINDEX: 4
Step Size:2

Probe length:4
Word inserted: sixteen INDEX: 12

Word inserted: seventeen OldINDEX: 39
Step Size:3

Probe length:1
Word inserted: seventeen INDEX: 2
Element found at:3

                                                                                  PART 3

Total Time for finding:61200

Total Time for insertion:2728400

Table Size after insertion of element:26
Number of elements with 0 probe:13
34

                                                                                  PART 4
[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, null, sixteen, null, four, null, null, Ojas, singh, null, null, null, null

Updated Array:[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, null, sixteen, null, four, null, null, Ojas, singh, null, nu
```

Time for insertion: 2728400 Nano seconds.

Time for finding: 61200 Nano seconds.

Load Factor: 0.4 Number of elements with 0 Probes: 17

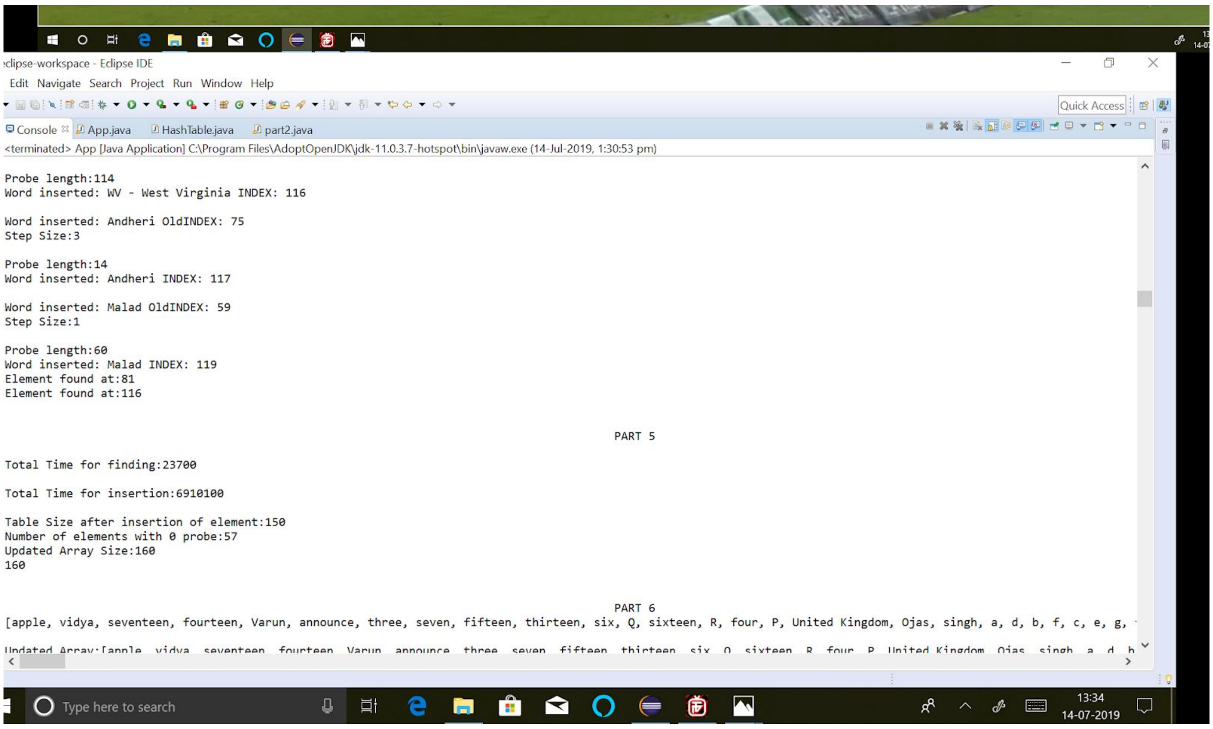


```
<terminated> App [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 1:30:53 pm)

Word inserted: P OldINDEX: 0
Step Size:3

Probe length:5
Word inserted: P INDEX: 15

Word inserted: Q OldINDEX: 1
Step Size:2

Probe length:5
Word inserted: Q INDEX: 11

Word inserted: R OldINDEX: 2
Step Size:1

Probe length:11
Word inserted: R INDEX: 13
Element found at:68

                                                                                  PART 4

Total Time for finding:19300

Total Time for insertion:3523600

Table Size after insertion of element:70
Number of elements with 0 probe:30
Updated Array Size:80
88

                                                                                  PART 5
[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, Q, sixteen, R, four, P, null, Ojas, singh, a, d, b, f, c, e, g, five, nine
```

Time for insertion: 3523600 Nano seconds.

Time for finding: 19300 Nano seconds.

Load Factor: 0.5 Number of elements with 0 Probes: 27

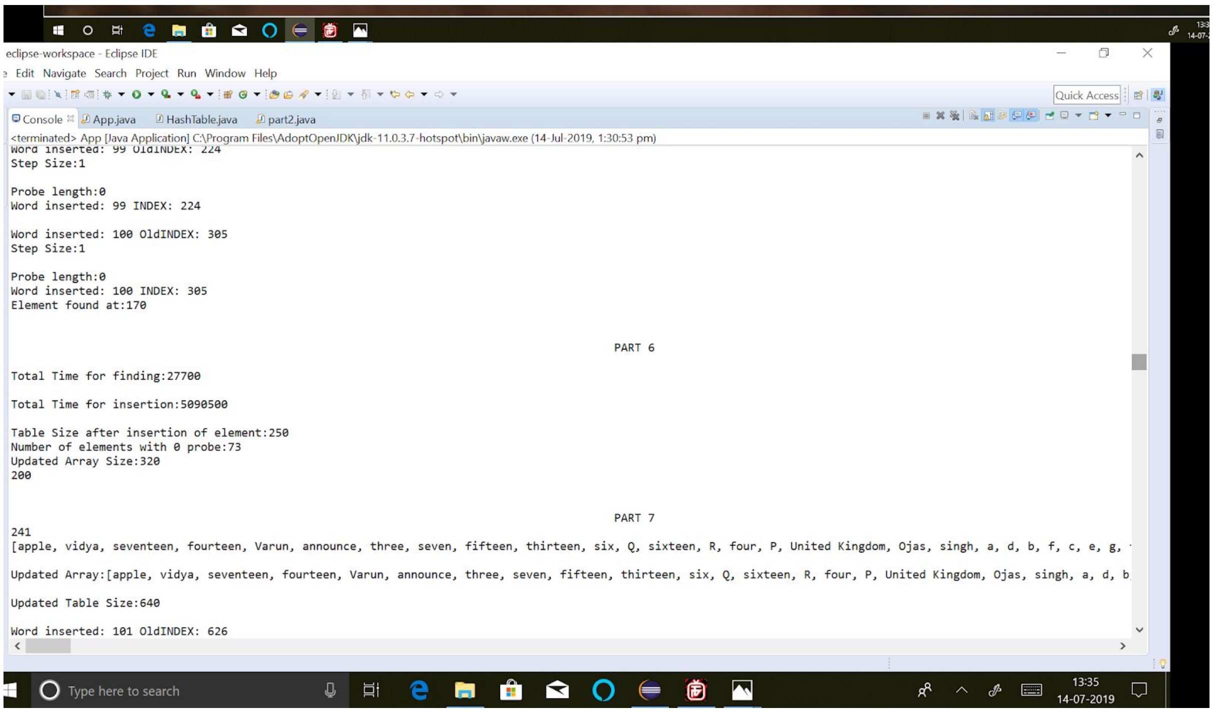

eclipse-workspace - Eclipse IDE
Edit Navigate Search Project Run Window Help
Quick Access
Console
App.java
HashTable.java
part2.java
<terminated> App [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 1:30:53 pm)
Type here to search
13:34
14-07-2019

```
Probe length:114
Word inserted: WV - West Virginia INDEX: 116

Word inserted: Andheri OldINDEX: 75
Step Size:3

Probe length:14
Word inserted: Andheri INDEX: 117

Word inserted: Malad OldINDEX: 59
Step Size:1

Probe length:60
Word inserted: Malad INDEX: 119
Element found at:81
Element found at:116


                                                                  PART 5

Total Time for finding:23700

Total Time for insertion:6910100

Table Size after insertion of element:150
Number of elements with 0 probe:57
Updated Array Size:160
160


                                                                  PART 6
[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, Q, sixteen, R, four, P, United Kingdom, Ojas, singh, a, d, b, f, c, e, g,

Updated Array:[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, Q, sixteen, R, four, P, United Kingdom, Ojas, singh, a, d, b
```

Time for insertion: 6910100 Nano seconds.

Time for finding: 23700 Nano seconds.

Load Factor: 0.6 Number of elements with 0 Probes: 16

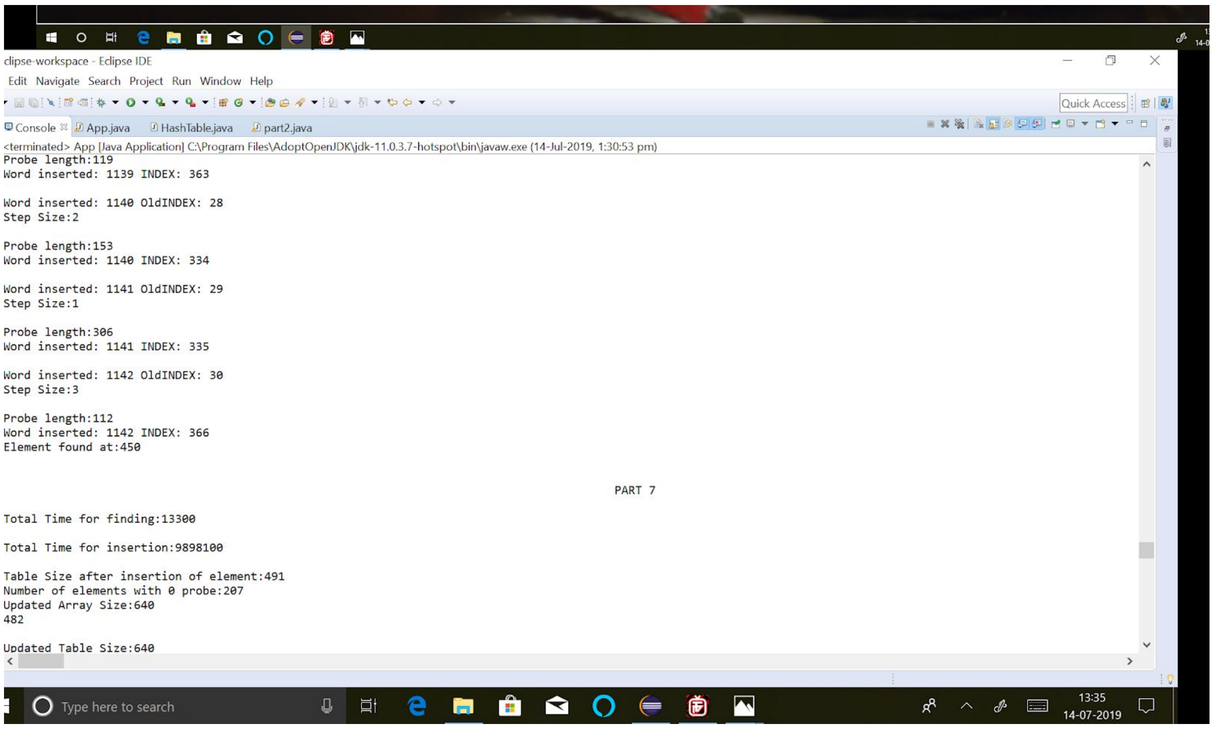

eclipse-workspace - Eclipse IDE
Edit Navigate Search Project Run Window Help
Quick Access
Console
App.java
HashTable.java
part2.java
<terminated> App [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 1:30:53 pm)
Type here to search
13:35
14-07-2019

```
Word inserted: 99 OldINDEX: 224
Step Size:1

Probe length:0
Word inserted: 99 INDEX: 224

Word inserted: 100 OldINDEX: 305
Step Size:1

Probe length:0
Word inserted: 100 INDEX: 305
Element found at:170


                                                                  PART 6

Total Time for finding:27700

Total Time for insertion:5090500

Table Size after insertion of element:250
Number of elements with 0 probe:73
Updated Array Size:320
200


                                                                  PART 7
241
[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, Q, sixteen, R, four, P, United Kingdom, Ojas, singh, a, d, b, f, c, e, g,

Updated Array:[apple, vidya, seventeen, fourteen, Varun, announce, three, seven, fifteen, thirteen, six, Q, sixteen, R, four, P, United Kingdom, Ojas, singh, a, d, b

Updated Table Size:640

Word inserted: 101 OldINDEX: 626
```

Time for insertion: 5090500 Nano seconds.

Time for finding: 27700 Nano seconds.

Load Factor: 0.75 Number of elements with 0 Probes: 134



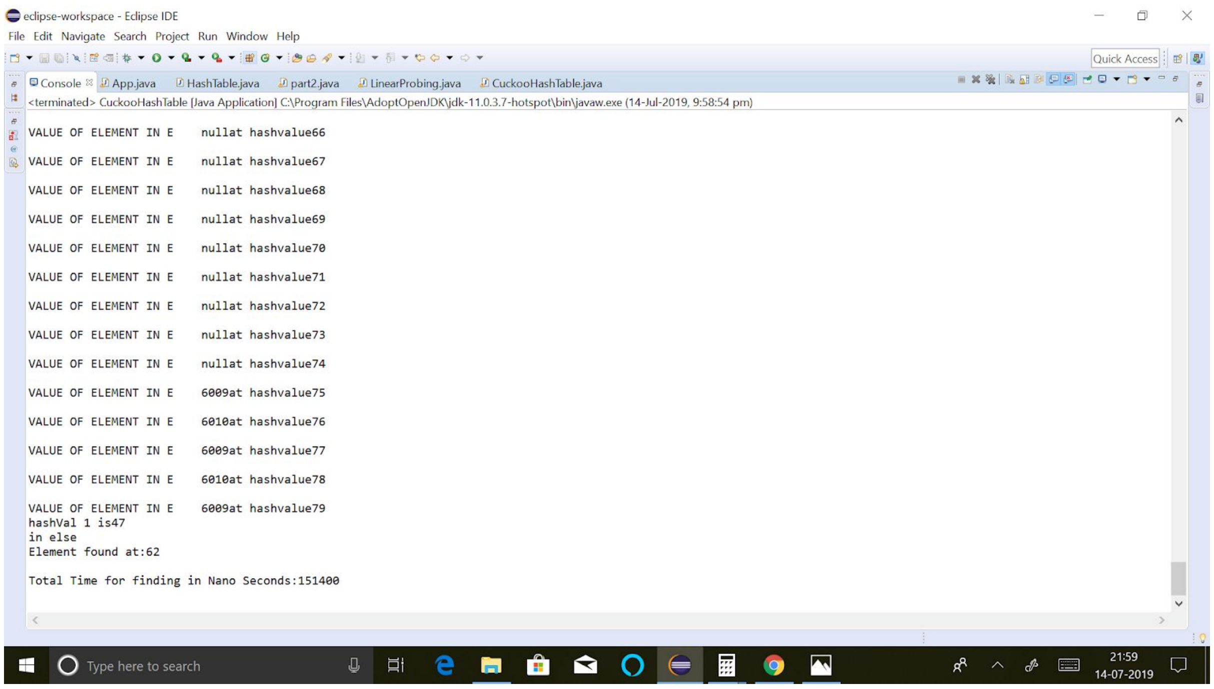
```
<terminated> App [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 1:30:53 pm)
Probe length:119
Word inserted: 1139 INDEX: 363

Word inserted: 1140 OldINDEX: 28
Step Size:2

Probe length:153
Word inserted: 1140 INDEX: 334

Word inserted: 1141 OldINDEX: 29
Step Size:1

Probe length:306
Word inserted: 1141 INDEX: 335

Word inserted: 1142 OldINDEX: 30
Step Size:3

Probe length:112
Word inserted: 1142 INDEX: 366
Element found at:450


                                                                                    PART 7


Total Time for finding:13300

Total Time for insertion:9898100

Table Size after insertion of element:491
Number of elements with 0 probe:207
Updated Array Size:640
482

Updated Table Size:640
```

Time for insertion: 9898100 Nano seconds.

Time for finding: 13300 Nano seconds.

Load Factor	Insertion Time	Finding time
0.1	276300	31500
0.2	334100	4700
0.3	2728400	61200
0.4	3523600	19300
0.5	6910100	23700
0.6	5050500	27700
0.75	9898100	**13300**
0.8	6584700	40100

CUCKOO HASHING:



```
<terminated> CuckooHashTable [Java Application] C:\Program Files\AdoptOpenJDK\jdk-11.0.3.7-hotspot\bin\javaw.exe (14-Jul-2019, 9:58:54 pm)

VALUE OF ELEMENT IN E    nullat hashvalue66

VALUE OF ELEMENT IN E    nullat hashvalue67

VALUE OF ELEMENT IN E    nullat hashvalue68

VALUE OF ELEMENT IN E    nullat hashvalue69

VALUE OF ELEMENT IN E    nullat hashvalue70

VALUE OF ELEMENT IN E    nullat hashvalue71

VALUE OF ELEMENT IN E    nullat hashvalue72

VALUE OF ELEMENT IN E    nullat hashvalue73

VALUE OF ELEMENT IN E    nullat hashvalue74

VALUE OF ELEMENT IN E    6009at hashvalue75

VALUE OF ELEMENT IN E    6010at hashvalue76

VALUE OF ELEMENT IN E    6009at hashvalue77

VALUE OF ELEMENT IN E    6010at hashvalue78

VALUE OF ELEMENT IN E    6009at hashvalue79
hashVal 1 is47
in else
Element found at:62

Total Time for finding in Nano Seconds:151400
```

It can be seen from the above observation that time required to find an element using Cuckoo Hash is more. Hence Cuckoo hash takes lot of time in linear probing.

Conclusion:

Number of elements with 0 probes indicate the number of elements which were added to their respective indices after resizing. This number specifies the efficiency of the hashtable after resizing. Thus at load factor 0.75, we find the element with the least time and also the number of elements with 0 probe is maximum at 0.75. Thus for the above empirical analysis , 0.75 is the breakeven point of the algorithm.