Name : Kiran Tulsulkar NUID : 001057882

EECE 7205 : Fundamentals of Computer Engineering

Assignment 3

Question 1 :

Problem 1 (30 Points)

Assume we have a hash table consisting of $m = 11$ slots, and suppose nonnegative integer key values are hashed into the table using the following hash function *h1*():

```
int h1(int key) {
 int x = (key + 5) * (key + 5);
 x = x / 16;
 x = x + key;
 x = x % 11;
 return x;
}
```

The sequence of 12 integer key values listed below are to be inserted in the table, in the order given. Suppose that collisions are resolved by using linear probing. Show the probe sequence (the sequence of slots to be probed until an empty one, if any, is available) for each key. In addition, show the final contents of the hash table after all keys are inserted.

Solution :

Key Value	*Hash Function Index*	*Probe Sequence*	*Final Index*
43	0	-	0
23	6	-	6
1	3	-	3
0	1	-	1
15	7	-	7
31	2	-	2
4	9	-	9
7	5	-	5
11	5	6,7,8	8
3	7	8,9,10	10
5	0	1,2,3,4	4
9	10	0,1,2,3,4,5,6,7,8,9,10	No Index

Name : Kiran Tulsulkar   NUID : 001057882

Final Hash Table :

Index	0	1	2	3	4	5	6	7	8	9	10
Key Values	43	0	31	1	5	7	23	15	11	4	3

Name : Kiran Tulsulkar NUID : 001057882

Question 2 :

Problem 2 (40 Points)

Write a C++ program to study how the hash table size affects the collision rate. The following are the requirements of this program:

a. Generate 1000 random integers that represent the birthdays of people who were born between January 1, 2000 and December 31, 2004 in the format *mmddyy*. An example of such integers is 112303 for someone who was born on November 23, 2003. For simplicity, assume that all people were born before the 28th day of their birth month and do not worry about duplicate birthdays.

b. Define four hash tables with the following sizes: $m_1 = 97$, $m_2 = 98$, $m_3 = 100$, and $m_4 = 101$.

c. Simulate storing the 1000 random birthdays that you generate in step *a* in each of the four hash tables using hash function $h(k) = k \bmod m_i$. Where k is the birthday integer and m_i is the size of table i (i = 1, 2, 3, 4). Assume that collision is resolved by chaining. You do not need to generate the actual chains in your program instead store in each slot of the tables the number of birthdays that are mapped to that slot. Note that, at the end, if a slot of a table has the value x, this means there was (x-1) collisions on that slot.

d. For each one of the four tables, calculate the minimum, maximum, mean, and variance of the collision numbers stored in the table from step c.

e. Comments on the results you calculated in step *d* by explaining how the hash table size affects the collision rate.

Solution :

```
-bash-4.2$ ./as32
Enter Size of Hash table = 97
Minimum collisions = 2
Maximum collisions = 17
Mean of collisions = 9.30928
Variance of collisions = 11.8013
```

```
-bash-4.2$ ./as32
Enter Size of Hash table = 98
Minimum collisions = 1
Maximum collisions = 19
Mean of collisions = 9.20408
Variance of collisions = 19.6318
```

```
-bash-4.2$ ./as32
Enter Size of Hash table = 100
Minimum collisions = 179
Maximum collisions = 219
Mean of collisions = 9.95
Variance of collisions = 1795.96
```

Name : Kiran Tulsulkar   NUID : 001057882

```
-bash-4.2$ ./as32
Enter Size of Hash table = 101
Minimum collisions = 1
Maximum collisions = 49
Mean of collisions = 9.49505
Variance of collisions = 151.227
```

1. Every key i.e birthdays shares a common factor with the size of hash table which is either 97,98,100 and 101 which will be hashed to index that is a multiple of this factor.
2. Therefore, to minimize collisions, it is important to reduce the number of common factors between *size of table* and the keys i.e *birthdays.*
3. This can be done by selecting size of hash table which has very few common factors such as a prime number.
4. Therefore 97 has minimum variance among the four.

Name : Kiran Tulsulkar   NUID : 001057882

Question 3 :

Problem 3 (30 Points)

If the root of a complete tree starts at index 1 and given the index *i* of a node, prove the heap indices calculations as shown below:

```
PARENT(i)
1   return ⌊i/2⌋

LEFT(i)
1   return 2i

RIGHT(i)
1   return 2i + 1
```

Solution :

Assume :

A = { A,B,C,D,E,F,G,H,I,J}

index = { 1,2,3,4,5,6,7,8,9,10}

Calculations :

root = 1 implies A is at index 1

So according to the function left child = 2*i = 2 and Right Child = 2*i +1 = 3

Then consider root at i=2 => Left child = 2*2 = 4 and right child = 2*2 +1 = 5

Then consider root at i=4 => Left child = 8 and Right child = 9

Then consider root at i=5 => Left child = 1

Then consider root at i=3 => Left Child = 2*3 and Right Child = 2*3 + 1 = 7

Tree :

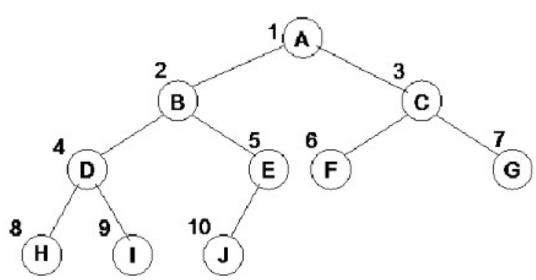

Image Source : http://agikarasugi2021ds.blogspot.com/