# 7205 HW3

**Name: Xuebao Zhao  NUID: 002108354**

## Q1:

| Key Value | Probe Sequence |
|---|---|
| **43** | 0 |
| **23** | 6 |
| **1** | 3 |
| **0** | 1 |
| **15** | 7 |
| **31** | 2 |
| **4** | 9 |
| **7** | 5 |
| **11** | 5->6->7->8 |
| **3** | 7->8->9->10 |
| **5** | 0->1->2->3->4 |
| **9** | 10->0->1->2->3->4->5->6->7->8->9 |

| | Final Hash Table Contents |
|---|---|
| **0** | 43 |
| **1** | 0 |
| **2** | 31 |
| **3** | 1 |
| **4** | 5 |
| **5** | 7 |
| **6** | 23 |
| **7** | 15 |
| **8** | 11 |
| **9** | 4 |
| **10** | 3 |

## Q2:

Use three arrays to store months, days, and years respectively. Fill month array with random numbers from 1 to 12. Fill days array with random numbers from 1 to 27. And fill years array with random number from 0 to 4. Combine these values to generate date random numbers with this formula:

$$date = months \times 10000 + days \times 100 + years$$

Create 4 hash tables (hash1, hash2, hash3, and hash4) to store the number of collisions. Iterate over the values in the date array and use the corresponding hash function

$h(k) = k \ mod \ m_i$. where $m_i$ is the sizes of the hash tables. Add one to the value of the array element whose index is $h(k)$ in the hash table. At the end, if a slot of the table has the value x, this means there was (x-1) collisions on that slot.

Calculate the minimum, maximum, mean, and variance of the collision values stored in the table.

Results:

```
-bash-4.2$ g++ -std=c++11 -o q2.out q2_hash.cpp
-bash-4.2$ ./q2.out

---------------- m = 97 -----------------
Minimum: 2
Maximum: 22
Mean: 9.30928
Variance: 14.894
----------------------------------------

---------------- m = 98 -----------------
Minimum: 0
Maximum: 30
Mean: 9.2449
Variance: 43.5727
----------------------------------------

---------------- m = 100 -----------------
Minimum: 0
Maximum: 221
Mean: 9.95
Variance: 1892.87
----------------------------------------

---------------- m = 101 -----------------
Minimum: 0
Maximum: 51
Mean: 9.51485
Variance: 209.772
----------------------------------------
```

Comments:

Compare these 4 sets of values. The best results were obtained when m = 97.

Because the minimum of the collision values is 2 while the others are 0. A minimum value of 0 means that there is no conflict in this slot or even no stored value. This is a waste of table space and will affect the performance of the hash table.

And when m = 97, the maximum is the smallest. It means that it has the smallest conflict rate than others.

When m = 97, The variance is the smallest, too. It means hash collisions are more evenly distributed across slots.

It turns out that we should make sure the size of hash table is a prime number, this will produce the most scattered remainder and minimize hash collisions.

Because the data represents a range of dates, the format is special. And this led to some poor results. When m = 100, the values stored in the hash table are almost all 0. Because the year is from 2000 to 2004, the last two digits of keys are 00, 01, 02, 03, 04. After calculating by the hash function (h(k) = key mod 100), all values are stored into the first five slots. As a result, the 'Maximum' and 'Variance' are huge.

## Q3:

In a complete binary tree, all levels except the lowest level are completely filled from the left to right. So, the number of nodes of levels in a complete binary tree can be thought of as a geometric sequence with common ratio 2. For example, the number of nodes in the first layer is $2^0 = 1$. The number of nodes in the second layer is $2^1 = 2$. The number of nodes in the third layer is $2^2 = 4$. So, the number of nodes in the $k^{th}$ layer is $2^{k-1}$.

And the index can be calculated by using the sum formula of geometric sequence:

$$S_n = \frac{a_1(1-q^n)}{1-q} \qquad q \neq 1$$

Where q = 2, a1 = 1.
So, the index of the first node of the K layer will be:

$$\frac{1 \times (1-2^{k-1})}{1-2} + 1 = 2^{k-1}$$

And the index of the last node of the K layer will be:

$$\frac{1 \times (1-2^{k})}{1-2} = 2^{k} - 1$$

①
If a node is root. Then it doesn't have a parent.
Assume that one node (not root) is the $m^{th}$ node at layer $k$. Its index will be: $2^{k-1} + m - 1$.
If the node is the left child. Then its parent node is the $(m+1)/2^{th}$ node at layer $k-1$. The index will be:

$$Parent = 2^{k-2} + (m+1)/2 - 1 = 2^{k-2} + (m-1)/2 = \frac{1}{2}Child$$

If the node is the right child. Then its parent node is the $m/2^{th}$ node at layer $k-1$. The index will be:

$$Parent = 2^{k-2} + m/2 - 1 = 2^{k-2} + (m-2)/2 = \frac{1}{2}(Child - 1)$$

Combined the above two points, we can conclude that:

$$Parent = \frac{1}{2}\lfloor Child \rfloor$$

②

Assume that the parent node is the $m^{th}$ node at layer $k$. Its index will be: $2^{k-1} + m - 1$. Then its left child node is the $2m - 1^{th}$ node at layer $k + 1$. And the right child node is the $2m^{th}$ node at layer $k + 1$. The indexes can be calculated:

Left:

$$Left = 2^k + 2m - 2 = 2 \times (2^{k-1} + m - 1) = 2Parent$$

Right:

$$Right = 2^k + 2m - 1 = 2 \times (2^{k-1} + m - 1) + 1 = 2Parent + 1$$