Hashing Lab

Values:

**15 54 13 10 135 114 49 174 27 24**

Calculate the hash index with module, Hash(key) = entry % 13; continue through all values until everything is placed into an element.

Linear probing

|  |  |
| --- | --- |
| 0 | 13 |
| 1 | 27 |
| 2 | 15 |
| 3 | 54 |
| 4 | 24 |
| 5 | 135 |
| 6 | 174 |
| 7 | Null |
| 8 | Null |
| 9 | Null |
| 10 | 10 |
| 11 | 114 |
| 12 | 49 |

If a value has landed on an occupied key, then iterate to following key.

(HashValue + 1) % 13

Quadratic probing

|  |  |
| --- | --- |
| 0 | 13 |
| 1 | 49 |
| 2 | 15 |
| 3 | 54 |
| 4 | 27 |
| 5 | 135 |
| 6 | 174 |
| 7 | Null |
| 8 | Null |
| 9 | Null |
| 10 | 10 |
| 11 | 114 |
| 12 | 24 |

When collision occurs, quadratic probing continues with

I + j^2. I is the original hash key and j is the incremented if collision is true.

Linear quotient/double hashing

|  |  |
| --- | --- |
| 0 | 13 |
| 1 | 27 |
| 2 | 15 |
| 3 | 49 |
| 4 |  |
| 5 | 135 |
| 6 | 54 |
| 7 |  |
| 8 | 114 |
| 9 |  |
| 10 | 10 |
| 11 | 174 |
| 12 | 24 |

**LQHashing:**

1. ip = pk % N

2. q=pk / N

if (q%N != 0)

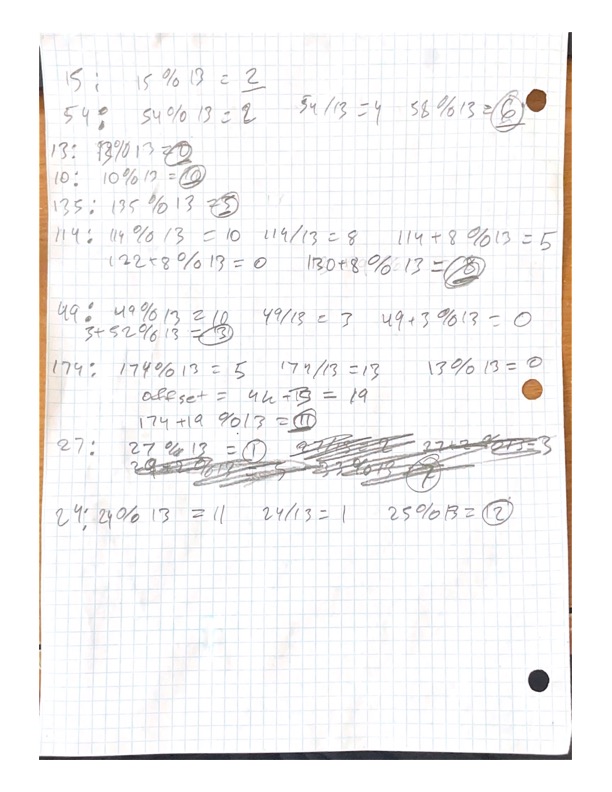
offset = q

else

offset = 4k+3 prime

3. While collisions:

ip’ = (ip + offset) % N

****4. Set Array[ip]=key

|  |  |
| --- | --- |
| **0** |  |
| **1** | **89** |
| **2** | **46** |
| **3** | **45** |
| **4** | **48** |
| **5** | **26** |
| **6** | **25** |
| **7** | **65** |
| **8** | **74** |
| **9** |  |
| **10** | **32** |
| **11** | **33** |
| **12** | **34** |
| **13** | **76** |
| **14** |  |
| **15** |  |
| **16** | **60** |
| **17** |  |
| **18** |  |
| **19** | **63** |
| **20** |  |
| **21** | **21** |

**Separate Chaining**

**Bucket hashing of 10 elements (N=10) ip = (pk) % N**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **0** | **10** |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** | **13** |  |  |  |
| **4** | **54** | **114** | **174** | **24** |
| **5** | **15** | **135** |  |  |
| **6** |  |  |  |  |
| **7** | **27** |  |  |  |
| **8** |  |  |  |  |
| **9** | **49** |  |  |  |

**My values: 43 46 89 48 32 63 45 33 60 26 74 25 76 34 65**

**I will do double hashing.**

**QHashing:**

1. ip = pk % N

2. q=pk / N

if (q%N != 0)

offset = q

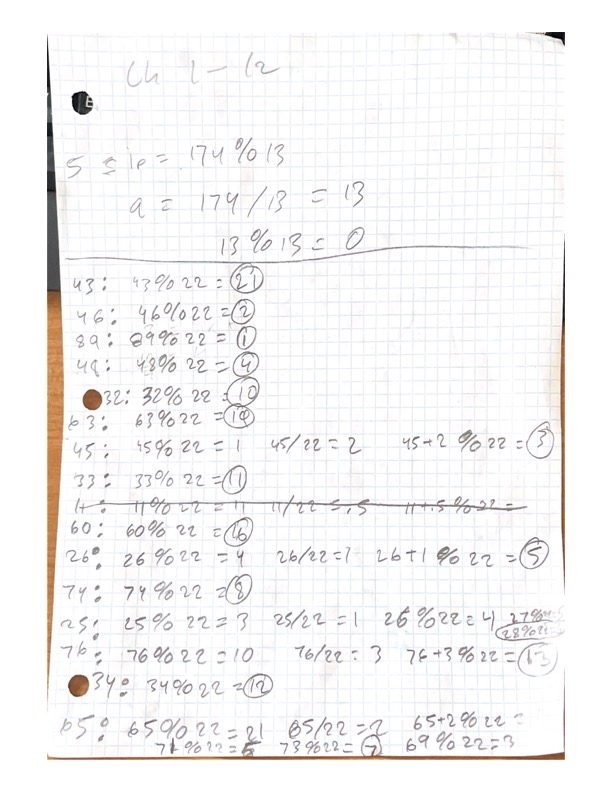
else

offset = 4k+3 prime

3. While collisions:

ip’ = (ip + offset) % N

4. Set Array[ip]=key



Write-up:

What an interesting lab, the Hash Map concept is unique to any other data structure I have worked with. Collisions are the hardest take away from Hash Maps. Probing linearly and quadratically is not difficult to understand but duplicating a hash on top of a hash came with lots of struggles. In fact, I required myself to write out all the work to ensure full comprehension. In the customized section I choose double hashing also known as linear quotient, again to make I understand. Separating chaining can utilize all the previous algorithms such as linked list, queue, and stack. To summarize double hashing the first placement is dictated by module. When a collision occurs, addition to element depends on quotient of the value and size of array. If the quotient is equivalent to the size of the array or zero than you would need the increment by a prime number greater or equal to the size of your array.