1. The Reason for Adding 1 in Double Hashing

In double hashing, the secondary hash function $g(k)$ is often defined as $g(k)=(k$ $\bmod m)+1$. The purpose of adding 1 is to ensure that the secondary hash function never returns 0 . If $g(k)$ could return 0 , then the probing function $p(k, i)=(h(k)+i g(k)) \bmod n$ might not change positions as $i$ increases, leading to the algorithm being stuck at the initial collision position.

Example:

Suppose $h(k)=k \bmod 10$ and $g(k)=k \bmod 5$, and consider the key $k=10$. For $k=10$ , $h(10)=0$. If $g(10)=0$, then the probing function $p(10, i)$ would return 0 for any value of $i$, causing the algorithm to enter an infinite loop. However, if we use $g(k)=(k \bmod 5)+1$, then $g(10)=1$, ensuring that the probing function explores new positions as $i$ increases.

2. Why Linear Probing and Double Hashing Always Find an Empty Spot

Linear probing and double hashing can always find an empty spot for a newly inserted key, provided the size $s$ of the hash table is less than its capacity $n$, and $n$ is a prime number. For linear probing, if $c$ is relatively prime to $n$, the probing sequence will cover all positions in the hash table. This is because being relatively prime ensures that the sequence does not cycle before covering all positions.

For double hashing, since $g(k)$ is non-zero and $m$ and $n$ are relatively prime, the probing sequence will visit all possible indices before reaching $n$ probes.

3. How Quadratic Probing and Double Hashing Reduce Clustering

Both quadratic probing and double hashing reduce clustering by providing different probing sequences.

- Quadratic Probing: Uses the square of the probe number $i$ to increase the probing distance as the number of probes increases. This avoids clustering, as keys that collide will be dispersed along different paths in the hash table.

- Double Hashing: Creates a unique probing sequence for each key using a second hash function. This reduces the likelihood of different keys sharing the same probing path, thereby reducing clustering.

4.

(a) Linear Probing c=2

Forward Insertion: Keys 14, 27, 33, 3, 18, 13, 37, 15, 22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Initial Position h(k) | Probing Sequence | Final Position | Collisions |
| 14 | 3 | 3 | 3 | 0 |
| 27 | 5 | 5 | 5 | 0 |
| 33 | 0 | 0 | 0 | 0 |
| 3 | 3 | 3, 5, 7 | 7 | 2 |
| 18 | 7 | 7, 9 | 9 | 1 |
| 13 | 2 | 2 | 2 | 0 |
| 37 | 4 | 4 | 4 | 0 |
| 15 | 4 | 4, 6 | 6 | 1 |
| 22 | 0 | 0, 2, 4, 6, 8 | 8 | 4 |

Reverse Insertion: Keys 22, 15, 37, 13, 18, 3, 33, 27, 14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Initial Position h(k) | Probing Sequence | Final Position | Collisions |
| 22 | 0 | 0 | 0 | 0 |
| 15 | 4 | 4 | 4 | 0 |
| 37 | 4 | 4, 6 | 6 | 1 |
| 13 | 2 | 2 | 2 | 0 |
| 18 | 7 | 7 | 7 | 0 |
| 3 | 3 | 3 | 3 | 0 |
| 33 | 0 | 0, 2 | 2 | 1 |
| 27 | 5 | 5, 7, 9 | 9 | 2 |
| 14 | 3 | 3, 5, 7, 9, 0, 2, 4, 6, 8 | 8 | 8 |

(b) Quadratic Probing c\_1=0,c\_2=1

Forward Insertion: Keys 14, 27, 33, 3, 18, 13, 37, 15, 22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Initial Position h(k) | Probing Sequence | Final Position | Collisions |
| 14 | 3 | 3 | 3 | 0 |
| 27 | 5 | 5 | 5 | 0 |
| 33 | 0 | 0 | 0 | 0 |
| 3 | 3 | 3, 4, 7 | 4 | 1 |
| 18 | 7 | 7 | 7 | 0 |
| 13 | 2 | 2, 3 | 3 | 1 |
| 37 | 4 | 4, 5, 8 | 5 | 1 |
| 15 | 4 | 4, 5, 8, 1, 6, 3, 2 | 1 | 6 |
| 22 | 0 | 0, 1, 4, 9, 6, 5, 8, 3, 10, 9, 10 | 10 | 10 |

Reverse Insertion: Keys 22, 15, 37, 13, 18, 3, 33, 27, 14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Initial Position h(k) | Probing Sequence | Final Position | Collisions |
| 22 | 0 | 0 | 0 | 0 |
| 15 | 4 | 4 | 4 | 0 |
| 37 | 4 | 4, 5, 8 | 5 | 1 |
| 13 | 2 | 2 | 2 | 0 |
| 18 | 7 | 7 | 7 | 0 |
| 3 | 3 | 3 | 3 | 0 |
| 33 | 0 | 0, 1 | 1 | 1 |
| 27 | 5 | 5, 6, 9 | 6 | 1 |
| 14 | 3 | 3, 4, 7 | 7 | 2 |

(c) Double Hashing $g(k)=(k \bmod 3)+1$

Forward Insertion: Keys 14, 27, 33, 3, 18, 13, 37, 15, 22

Double Hashing uses the formula $p(k, i)=(h(k)+i g(k)) \bmod 11$, where $g(k)=(k$ $\bmod 3)+1$

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Initial Position h(k) | Probing Sequence | Final Position | Collisions |
| 14 | 3 | 3 | 3 | 0 |
| 27 | 5 | 5 | 5 | 0 |
| 33 | 0 | 0 | 0 | 0 |
| 3 | 3 | 3, 4 | 4 | 1 |
| 18 | 7 | 7 | 7 | 0 |
| 13 | 2 | 2 | 2 | 0 |
| 37 | 4 | 4, 5 | 5 | 1 |
| 15 | 4 | 4, 5, 6 | 6 | 2 |
| 22 | 0 | 0, 1 | 1 | 1 |

Reverse Insertion: Keys 22, 15, 37, 13, 18, 3, 33, 27, 14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Key | Initial Position h(k) | Probing Sequence | Final Position | Collisions |
| 22 | 0 | 0 | 0 | 0 |
| 15 | 4 | 4 | 4 | 0 |
| 37 | 4 | 4, 5 | 5 | 1 |
| 13 | 2 | 2 | 2 | 0 |
| 18 | 7 | 7 | 7 | 0 |
| 3 | 3 | 3 | 3 | 0 |
| 33 | 0 | 0, 1 | 1 | 1 |
| 27 | 5 | 5, 6 | 6 | 1 |
| 14 | 3 | 3, 4 | 4 | 1 |

In this method, the probing process for each key takes into account the increase in $i g(k)$. A collision occurs each time the probing function $p(k, i)$ returns an already occupied position. The tables above show the initial position, probing sequence, final position, and the number of collisions for each key.