

A good hash function should have following properties  
1) Efficiently computable.  
2) Should uniformly distribute the keys (Each table position equally likely for each key)

Collision Handling:

1.Chaining:

2.Open Addressing:

Search : O(1) [Average] O(n) [Worst case]

Insertion : O(1) [Average] O(n) [Worst Case]

Deletion : O(1) [Average] O(n) [Worst Case]

A hash function h defined h(key)=key mod 7, with linear probing, is used to insert the keys 44, 45, 79, 55, 91, 18, 63 into a table indexed from 0 to 6. What will be the location of key 18?

Ans:-

keys 44, 45, 79, 55, 91, 18, 63 h(key)= key mod 7 h(44) = 44mod7 = 2 h(45) = 45mod7 = 3 h(79) = 79mod7 = 2 but 2 is already filled by 44, linear probing is applied but 3 is also filled by 45. So, 79 will occupy 4. h(55) = 55mod7 = 6 h(91) = 91mod7 = 0 h(18) = 18mod7 = 4 but 4 is occupied by 79 so, it will occupy 5. h(63) = 63mod7 = 0. 0 is also occupied so, it will occupy 1. So, option (C) is correct.

Insert the characters of the string **K R P C S N Y T J M** into a hash table of size 10. Use the hash function

h(x) = ( ord(x) – ord("a") + 1 ) mod10

If linear probing is used to resolve collisions, then the following insertion causes collision

Ans:-

(a) The hash table with size 10 will have index from 0 to 9. hash function = h(x) = ((ord(x) - ord(A) + 1)) mod 10 So for string K R P C S N Y T J M: K will be inserted at index : (11-1+1) mod 10 = 1 R at index: (18-1+1) mod 10 = 8 P at index: (16-1+1) mod 10 = 6 C at index: (3-1+1) mod 10 = 3 S at index: (19-1+1) mod 10 = 9 N at index: (14-1+1) mod 10 = 4 Y at index (25-1+1) mod 10 = 5 T at index (20-1+1) mod 10 = 0 J at index (10-1+1) mod 10 = 0 // first collision occurs. M at index (13-1+1) mod 10 = 3 //second collision occurs. Only J and M are causing the collision. (b) Final Hash table will be:

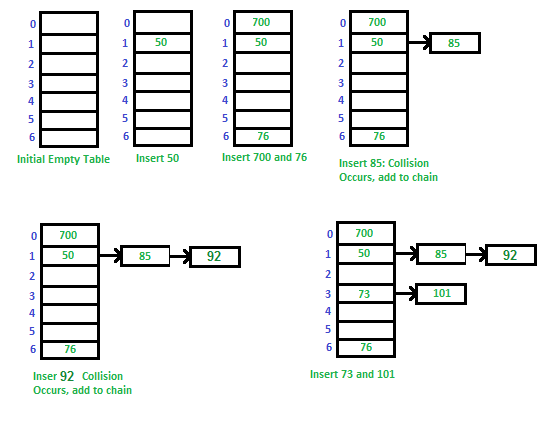
Consider a 13 element hash table for which f(key)=key mod 13 is used with integer keys. Assuming linear probing is used for collision resolution, at which location would the key 103 be inserted, if the keys 661, 182, 24 and 103 are inserted in that order?

Ans: 661 mod 13 = 11 182 mod 13 = 0 24 mod 13 = 11, already filled, so after linear probing it will get index 12 103 mod 13 = 12, already filled, so after linear probing it will get index 1 https://www.geeksforgeeks.org/wp-content/uploads/fffgggg.png So, option (B) is correct.

**How to handle Collisions?**  
There are mainly two methods to handle collision:  
1) Separate Chaining  
2) Open Addressing

1. ) Separate Chaining

each cell of hash table point to a linked list of records that have same hash function value.



**Disadvantages:**  
1) Cache performance of chaining is not good as keys are stored **using a linked list.** Open addressing provides better cache performance as everything is stored in the same table.  
2) Wastage of Space (Some Parts of hash table are never used)  
3) If the chain becomes long, then search time can become O(n) in the worst case.  
4) Uses extra space for links.

2. Open Addressing

Open Addressing is done following ways:

***a) Linear Probing:*** In linear probing, we linearly probe for next slot.



**Clustering:** The main problem with linear probing is clustering, many consecutive elements form groups and it starts taking time to find a free slot or to search an element.

**Linear probing** **has the best cache performance but suffers from clustering**. One more advantage of Linear probing is easy to compute.

**Quadratic probing** lies between the two in terms of cache performance and clustering.

**Double hashing has poor cache performance but no clustering.** Double hashing requires more computation time as two hash functions need to be computed.