**MY DSA DOCUMENT**

**1.HASH TABLE**

**1.What is hashing and why do we do hashing**

[**https://www.youtube.com/watch?v=wWgIAphfn2U&feature=youtu.be**](https://www.youtube.com/watch?v=wWgIAphfn2U&feature=youtu.be)

**2.A simple example of hashing.**

In this we create a big 2 dimensional matrix with big size(max).then we keep the index in that matrix as 1 which has same value as given array to make it as present.

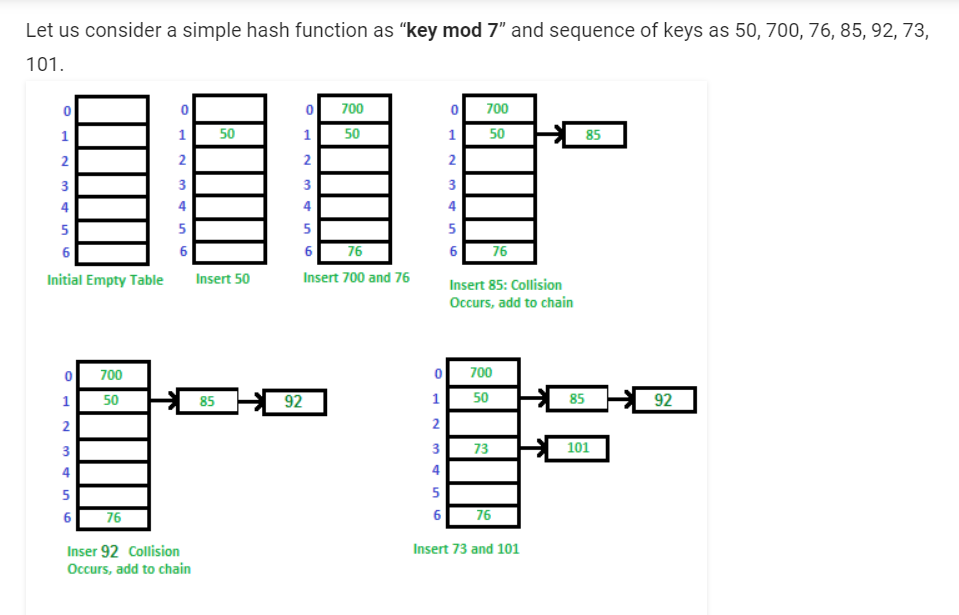
<https://drive.google.com/file/d/1AhJWjhnI5beY4TN80e-oqfhjiDqGysFx/view?usp=sharing>

**3.Collision handling-**

**(a)Chaining**-The idea is to make each cell of hash table point to a linked list of records that have same hash function value. Chaining is simple, but requires additional memory outside the table.

**(b)Open addressing**-In open addressing, all elements are stored in the hash table itself. Each table entry contains either a record or NIL. When searching for an element, we one by one examine table slots until the desired element is found or it is clear that the element is not in the table.

(a) **Separate Chaining:**  
The idea is to make each cell of hash table point to a linked list of records that have same hash function value.



**Code**- <https://www.geeksforgeeks.org/c-program-hashing-chaining/>

**Advantages:**  
1) Simple to implement.  
2) Hash table never fills up, we can always add more elements to the chain.  
3) Less sensitive to the hash function or load factors.  
4) It is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.

**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.

(b) **Open addressing-**

(i) linear probing-If slot hash(x) % S is full, then we try (hash(x) + 1) % S

If (hash(x) + 1) % S is also full, then we try (hash(x) + 2) % S

If (hash(x) + 2) % S is also full, then we try (hash(x) + 3) % S

For eg-Let us consider a simple hash function as “key mod 7” and sequence of keys as 50, 700, 76, 85, 92, 73, 101.



Challenges in Linear Probing :

1.Primary Clustering: One of the problems with linear probing is Primary clustering, many consecutive elements form groups and it starts taking time to find a free slot or to search an element. 

2.Secondary Clustering*:*Secondary clustering is less severe, two records do only have the same collision chain(Probe Sequence) if their initial position is the same.

(ii) quadratic probing-let hash(x) be the slot index computed using hash function.

If slot hash(x) % S is full, then we try (hash(x) + 1\*1) % S

If (hash(x) + 1\*1) % S is also full, then we try (hash(x) + 2\*2) % S

If (hash(x) + 2\*2) % S is also full, then we try (hash(x) + 3\*3) % S

..................................................

..................................................

(iii) double hashing-We use another hash function hash2(x) and look for i\*hash2(x) slot in i’th rotation.

let hash(x) be the slot index computed using hash function.

If slot hash(x) % S is full, then we try (hash(x) + 1\*hash2(x)) % S

If (hash(x) + 1\*hash2(x)) % S is also full, then we try (hash(x) + 2\*hash2(x)) % S

If (hash(x) + 2\*hash2(x)) % S is also full, then we try (hash(x) + 3\*hash2(x)) % S

..................................................

..................................................

**Comparison of above three:**   
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.