

Open Addressing

Like separate chaining, open addressing is a method for handling collisions. In Open Addressing, all elements are stored in the hash table itself. So at any point, the size of the table must be greater than or equal to the total number of keys (Note that we can increase table size by copying old data if needed).

Insert(k): Keep probing until an empty slot is found. Once an empty slot is found, insert k .

Search(k): Keep probing until slot's key doesn't become equal to k or an empty slot is reached.

Delete(k): **Delete operation is interesting.** If we simply delete a key, then the search may fail. So slots of deleted keys are marked specially as "deleted". The insert can insert an item in a deleted slot, but the search doesn't stop at a deleted slot.

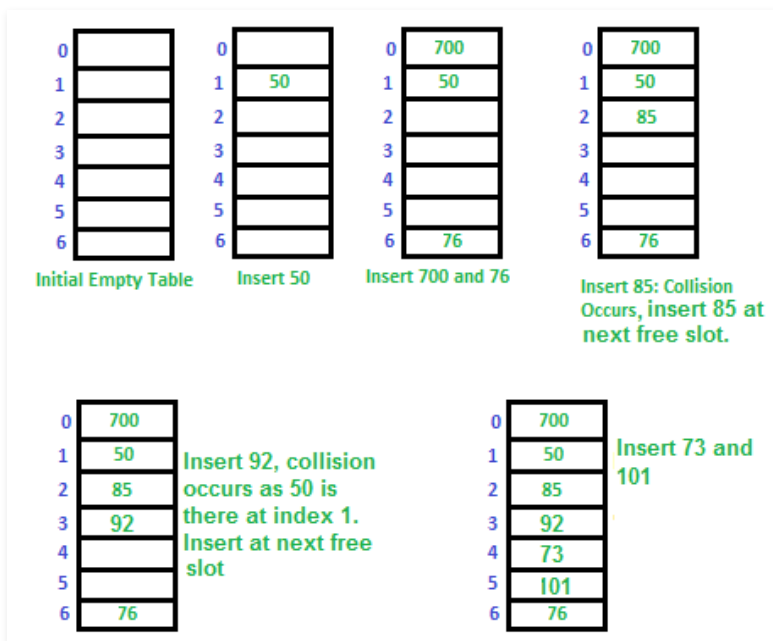
Open Addressing is done in the following ways:

a) Linear Probing: In linear probing, we linearly probe for next slot. For example, the typical gap between two probes is 1 as seen in the example below.

Let **hash(x)** be the slot index computed using a hash function and **S** be the table size

If slot $\text{hash}(x) \% S$ is full, then we try $(\text{hash}(x) + 1) \% S$
 If $(\text{hash}(x) + 1) \% S$ is also full, then we try $(\text{hash}(x) + 2) \% S$
 If $(\text{hash}(x) + 2) \% S$ is also full, then we try $(\text{hash}(x) + 3) \% S$

Let us consider a simple hash function as "key mod 7" and a 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 for an element.
2. **Secondary Clustering:** Secondary clustering is less severe, two records only have the same collision chain (Probe Sequence) if their initial position is the same.

b) Quadratic Probing We look for i^2 'th slot in i 'th iteration.



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

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

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

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

c) Double Hashing We use another hash function $\text{hash2}(x)$ and look for $i*\text{hash2}(x)$ slot in i 'th rotation.

let $\text{hash}(x)$ be the slot index computed using hash function.
 If slot $\text{hash}(x) \% S$ is full, then we try $(\text{hash}(x) + 1*\text{hash2}(x)) \% S$
 If $(\text{hash}(x) + 1*\text{hash2}(x)) \% S$ is also full, then we try $(\text{hash}(x) + 2$
 If $(\text{hash}(x) + 2*\text{hash2}(x)) \% S$ is also full, then we try $(\text{hash}(x) + 3$

See [this](#) for step by step diagrams.

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.

S.No.	Separate Chaining	Open Addressing
1.	Chaining is Simpler to implement.	Open Addressing requires more computation.
2.	In chaining, Hash table never fills up, we can always add more elements to chain.	In open addressing, table may become full.



S.No.	Separate Chaining	Open Addressing
3.	Chaining is Less sensitive to the hash function or load factors.	Open addressing requires extra care to avoid clustering and load factor.
4.	Chaining is mostly used when it is unknown how many and how frequently keys may be inserted or deleted.	Open addressing is used when the frequency and number of keys is known.
5.	Cache performance of chaining is not good as keys are stored using linked list.	Open addressing provides better cache performance as everything is stored in the same table.
6.	Wastage of Space (Some Parts of hash table in chaining are never used).	In Open addressing, a slot can be used even if an input doesn't map to it.
7.	Chaining uses extra space for links.	No links in Open addressing

Performance of Open Addressing:

Like Chaining, the performance of hashing can be evaluated under the assumption that each key is equally likely to be hashed to any slot of the table (simple uniform hashing)

m = Number of slots in the hash table

n = Number of keys to be inserted in the hash table

Load factor $\alpha = n/m$ (< 1)



Expected time to search/insert/delete $< 1/(1 - \alpha)$

So Search, Insert and Delete take $(1/(1 - \alpha))$ time