Collision in Hashing

In this, the hash function is used to find the index of the array. The hash value is used to create an index for the key in the hash table. The hash function may return the same hash value for two or more keys. When two or more keys have the same hash value, a collision happens. To handle this collision, we use collision resolution techniques.

Collision Resolution Techniques

There are two types of collision resolution techniques.

1. Separate chaining (open hashing)
2. Open addressing (closed hashing)

Separate chaining: This method involves making a linked list out of the slot where the collision happened, and then adding the new key to the list. Separate chaining is the term used to describe how this connected list of slots resembles a chain. It is more frequently utilized when we are unsure of the number of keys to add or remove.

Time complexity

- Its worst-case complexity for searching is o(n).
- Its worst-case complexity for deletion is o(n).

Advantages of separate chaining

- It is easy to implement.
- The hash table never fills full, so we can add more elements to the chain.
- It is less sensitive to the function of the hashing.

Disadvantages of separate chaining

- In this, the cache performance of chaining is not good.
- Memory wastage is too much in this method.
- It requires more space for element links.

Open addressing: To prevent collisions in the hashing table, open addressing is employed as a collision-resolution technique. No key is kept anywhere else besides the hash table. As a result, the hash table's size is never equal to or less than the number of keys. Additionally known as closed hashing.

The following techniques are used in open addressing:

- Linear probing
- Quadratic probing
- Double hashing

Linear probing: This involves doing a linear probe for the following slot when a collision occurs and continuing to do so until an empty slot is discovered.
The worst time to search for an element in linear probing is O. The cache performs best with linear probing, but clustering is a concern. This method's key benefit is that it is simple to calculate.

Disadvantages of linear probing:

- The main problem is clustering.
- It takes too much time to find an empty slot.

Quadratic probing: When a collision happens in this, we probe for the i2-nd slot in the ith iteration, continuing to do so until an empty slot is discovered. In comparison to linear probing, quadratic probing has a worse cache performance. Additionally, clustering is less of a concern with quadratic probing.

Double hashing: In this, you employ a different hashing algorithm, and in the ith iteration, you look for (i * hash 2(x)). The determination of two hash functions requires more time. Although there is no clustering issue, the performance of the cache is relatively poor when using double probing.

Recursion: Introduction to Recursion

Any function which calls itself is called recursion. **A recursive method solves a problem by calling a copy of itself to work on a smaller problem.** Each time a function calls itself with a slightly simpler version of the original problem. This sequence of smaller problems must eventually converge on a base case.

Working of recursion

We can define the steps of the recursive approach by summarizing the above three steps:

- **Base case**: A recursive function must have a terminating condition at which the process will stop calling itself. Such a case is known as the base case. In the absence of a base case, it will keep calling itself and get stuck in an infinite loop. Soon, the recursion depth* will be exceeded and it will throw an error.
- **Recursive call (Smaller problem):** The recursive function will invoke itself on a smaller version of the main problem. We need to be careful while writing this step as it is crucial to correctly figure out what your smaller problem is.
- **Self-work**: Generally, we perform a calculation step in each recursive call. We can achieve this calculation step before or after the recursive call depending upon the nature of the problem.