**1. What is a Hash Function?**

A [hash function](https://www.geeksforgeeks.org/dsa/hash-functions-and-list-types-of-hash-functions/) is an algorithm that converts an input into a fixed-size hash code, which determines the index for storing or retrieving data in a hash table.

**2. What is the time complexity of inserting and retrieving an element from a hashmap or dictionary?**

The time complexity for pushing and retrieving the elements from a Hash Map O(1).

**3. What are the properties of a good hash function?**

Some properties of good hash functions are:

* **Uniform distribution**: A good hash function spreads values evenly across the hash table to avoid clustering.
* **Efficiency**: It should compute hash values quickly with minimal computational overhead.
* **Deterministic**: The same input should always produce the same hash value.
* **Minimizing collisions**: It should reduce the chances of different inputs mapping to the same hash value.

**4. What is a perfect hash function?**

A perfect hash function is a hash function that maps distinct keys to distinct indices without any collisions. It guarantees an O(1) lookup time in a hash table, but finding a perfect hash function may be computationally expensive.

**5. What are Collisions in Hashing?**

A Hashing collision occurs when two different keys produce the same hash code. This is resolved using techniques like [open addressing](https://www.geeksforgeeks.org/dsa/open-addressing-collision-handling-technique-in-hashing/) or [separate chaining.](https://www.geeksforgeeks.org/dsa/separate-chaining-collision-handling-technique-in-hashing/)

**6. What is Open Addressing and separate chaining ?**

Open addressing and separate chaining are [collision resolution methods](https://www.geeksforgeeks.org/dsa/collision-resolution-techniques/) where in Open Addressing, the algorithm searches for the next available slot in the hash table upon a collision. while, Separate chaining stores multiple elements at the same index by using linked lists or other data structures to resolve collisions.

**7. What is Hash Code in Hashing?**

A hash code is the output of a hash function, used as an index to store or retrieve data in a hash table.

**8. What is the time complexity of search, insert, and delete operations in a Hash Table?**

The time complexity for these operations is O(1) on average, but it can degrade to O(n) in the worst case with many collisions.

**9. Difference between Hashing and Self-Balancing BST?**

**Hashing** and **self-balancing binary search trees (BSTs)**are used for efficient data storage and retrieval but they serve different purposes.

Hashing offers **O(1) average time** for search, insert, and delete, making it ideal for quick lookups in applications like caches, dictionaries, and symbol tables. However, it does not maintain key order and can degrade to **O(n)** in the worst case due to collisions.

Self-balancing BSTs (e.g. AVL, Red-Black Trees) guarantee **O(log n)** for all operations and preserve elements in **sorted order**, enabling range queries, ordered traversals, and predecessor/successor lookups. While slightly slower on average than hashing, they are preferred when **ordering and structured access** are required.

**10. Explain the concept of Load Factor in Hash Tables.**

The [load factor in Hash Table](https://www.geeksforgeeks.org/dsa/load-factor-and-rehashing/) is the ratio of elements to slots in a hash table. A high load factor indicates more collisions, leading to slower performance. Hash tables resize when this threshold is exceeded.

**11. Why does chaining perform better than linear probing in scenarios with high load factors?**

* Chaining stores all colliding elements in a linked list at that index, allowing constant-time insertion and flexible storage even when the table is full.
* Linear probing shifts elements to the next available spot, leading to clustering and performance degradation at high load factors due to long probe sequences.

**12. What is the purpose of rehashing in a hash table?**

Rehashing is the process of resizing a hash table when the load factor exceeds a certain threshold. It involves creating a new, larger table and re-inserting the elements from the old table using their hash codes. Rehashing ensures that the table maintains efficient performance as it grows.

**13. How would you implement a hash table with separate chaining?**

Create a hash table where each index holds a linked list to store elements that hash to the same index. Implement basic operations like insert(), get(), and remove().

**14. What is cuckoo hashing?**

[Cuckoo hashing](https://www.geeksforgeeks.org/dsa/cuckoo-hashing/)is a collision resolution technique that uses two hash functions to store keys in a hash table.

**15. What are bloom filters, and how do they relate to hashing?**

A [bloom filter](https://www.geeksforgeeks.org/python/bloom-filters-introduction-and-python-implementation/)is a probabilistic data structure used to test whether an element is a member of a set. It uses multiple hash functions to map elements to a bit array, allowing for fast membership testing, but with a small probability of false positives.