1. Conditions for Double Hashing

Double hashing is a collision resolution technique used in hashing. Certain conditions must be satisfied for double hashing to work properly. The second hash function should never produce a value of zero because it is used to find the next position. The size of the hash table should be a prime number to ensure that all table locations can be visited. The two hash functions must be independent of each other. These conditions help avoid infinite loops and ensure proper distribution of keys in the hash table.

2. Double Hashing Technique

Double hashing uses two different hash functions to resolve collisions. When a collision occurs at a hash location, the second hash function is used to calculate the step size for probing. The new position is calculated repeatedly until an empty slot is found. This method reduces clustering and distributes elements evenly across the hash table. Double hashing is more efficient than linear and quadratic probing.

3. Hash Functions Used in Double Hashing

Two hash functions are used in double hashing. The first hash function determines the initial position in the hash table. The second hash function determines the step size used during probing. The general form of these functions ensures that all positions in the hash table can be explored if needed. Using two different functions improves efficiency and reduces collision chances.

4. Significance of Load Factor in Hashing

The load factor indicates how full a hash table is. It is calculated as the ratio of the number of stored elements to the total number of available slots. Load factor plays an important role in hashing performance. A high load factor increases collisions and slows down operations. A low load factor improves speed but wastes memory. Maintaining a balanced load factor ensures efficient hashing.

5. Advantages of Double Hashing

Double hashing has many advantages. It greatly reduces clustering compared to linear and quadratic probing. It provides better distribution of keys in the hash table. It improves search, insertion, and deletion performance. Double hashing also uses table space more efficiently and avoids long probe sequences.

6. Steps for Rehashing

Rehashing is the process of creating a new hash table when the existing table becomes too full. First, a new larger hash table is created. Then, all existing elements are removed from the old table. Next, each element is inserted into the new table using a new hash function. Rehashing reduces collisions and improves performance by lowering the load factor.

7. Time and Space Complexity of Rehashing

The time complexity of rehashing depends on the number of elements stored in the hash table. All elements must be reinserted into the new table, so the time complexity is linear. The space complexity increases temporarily because a new hash table is created. However, rehashing improves long-term performance by reducing collisions and maintaining efficiency.