String Hashing

Comparing two strings t and s of length n and m is a well-known and trivial task that can be done in O(min(n, m)). But It would be really nice if we can get something as fast as O(1) !.

Hence we use the concept of string hashing, where we map each string to a unique integer in some range and then compare the two mappings of two strings. Please note that this is not a deterministic algorithm and may fail sometimes but we will try to minimize the probability of failure.

Hash Function

A hash function is any function that can be used to map a string of arbitrary size to an integer in a fixed range [0, m). The values returned by a hash function are called hash values, hash codes, hash sums, or simply hashes.

To achieve a good hashing mechanism, It is important to have a good hash function with the following basic requirements:

1. Collisions are resistant. Collisions occur when pairs of elements are mapped to the same hash value. These should be avoided.
   Note: Irrespective of how good a hash function is, collisions are bound to occur.

2. Fast to compute: The hash of a string of length n should be computable in at most O(n) time else the whole purpose of hashing the string will be defeated.

3. Easy to compute: It should be easy to compute and must not become an algorithm in itself.

4. Uniform distribution: It should provide a uniform distribution across the hash table and should not result in clustering.

Notice, the opposite direction doesn't have to hold. If the hashes are equal (hash(s)=hash(t)), then the strings do not necessarily have to be equal. For example consider the hash function of a string hash(s) = (s[0] - ‘a’ + 1). Then the hash of the string “coding” and “code” are the same as their first characters are also the same.