4.1 Hashing

Hashing is the change of a line of character into a normally more limited fixed-length worth or key that addresses the first string.
Hashing is utilized to list and recover things in a data set since it is quicker to discover the thing utilizing the briefest hashed key than to discover it utilizing the first worth. It is likewise utilized in numerous encryption calculations.
A hash code is produced by utilizing a key, which is an exceptional worth.
Hashing is a strategy where given key field esteem is changed over into the location of capacity area of the record by applying a similar procedure on it.

The benefit of hashing is that permits the execution season of fundamental activity to stay consistent in any event, for the bigger side.

4.2 Why we need Hashing?

Assume we have 50 workers, and we need to give 4 digit key to every representative (with respect to security), and we need subsequent to entering a key, direct client guide to a specific position where information is put away.
In the event that we give the area number as per 4 digits, we should hold 0000 to 9999 locations since anyone can utilize anybody as a key. There is a great deal of wastage.
To tackle this issue, we use hashing which will deliver a more modest estimation of the record of the hash table relating to the key of the client.

4.3 Universal Hashing

Leave H alone a limited assortment of hash works that map a given universe U of keys into the reach {0, 1. m-1}. Such an assortment is supposed to be all inclusive if for each pair of particular keys k,l∈U, the quantity of hash capacities h∈ H for which h(k)= h(l) is all things considered |H|/m. As such, with a hash work haphazardly browsed H, the possibility of an impact between particular keys k and l is close to the opportunity 1/m of a crash if h(k) and h(l)were arbitrarily and autonomously looked over the set {0,1,. m-1}.

4.4. Rehashing

On the off chance that any stage the hash table turns out to be almost full, the running time for the activities of will begin taking an excess of time, embed activity may fall flat in such circumstance, the most ideal arrangement is as per the following:

1. Make another hash table twofold in size.
2. Output the first hash table, register new hash worth and addition into the new hash table.
3. Free the memory involved by the first hash table.

Model: Consider embeddings the keys 10, 22, 31,4,15,28,17,88 and 59 into a hash table of length m = 11 utilizing open tending to with the essential hash work h' (k) = k mod m .Illustrate the aftereffect of embeddings these keys utilizing straight examining, utilizing quadratic testing with c1=1 and c2=3, and utilizing twofold hashing with h2(k) = 1 + (k mod (m-1)).

Arrangement: Using Linear Probing the last condition of hash table would be:

0	22
1	88
2	/
3	/
4	4
5	15
6	28
7	17
8	59
9	31
10	10

Using Quadratic Probing with c1=1, c2=3, the final state of hash table would be h (k, i) = (h' (k) +c1*i+ c2 *i2) mod m where m=11 and h' (k) = k mod m.

0	22
1	88
2	/
3	17
4	4
5	/
6	28
7	59
8	15
9	31
10	10

Using Double Hashing, the final state of the hash table would be:

0	22
1	/
2	59
3	17
4	4
5	15
6	28
7	88
8	/
9	31
10	10

4.5 Hash Tables

It is an assortment of things which are put away so as to make it simple to discover them later.

Each position in the hash table is called opening, can hold a thing and is named by a number worth beginning at 0.

The planning between a thing and a space where the thing should be in a hash table is known as a Hash Function. A hash Function acknowledges a key and returns its hash coding, or hash esteem.

Expect we have a bunch of whole numbers 54, 26, 93, 17, 77, 31. Our first hash work needed to be as "leftover portion strategy" just takes the thing and separation it by table size, returning remaining portion as its hash esteem for example

h item = item % (size of table)

Let us say the size of table = 11, then

54 % 11 = 10 26 % 11 = 4 93 % 11 = 5

17 % 11 = 6 77 % 11 = 0 31 % 11 = 9

ITEM	HASH VALUE
54	10
26	4
93	5
17	6
77	0
31	9

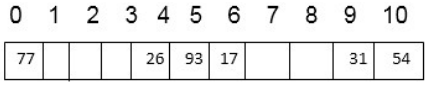

Fig: Hash Table

Now when we need to search any element, we just need to divide it by the table size, and we get the hash value. So we get the O (1) search time.

Now taking one more element 44 when we apply the hash function on 44, we get (44 % 11 = 0), But 0 hash value already has an element 77. This Problem is called as Collision.

Collision: According to the Hash Function, two or more item would need in the same slot. This is said to be called as Collision.

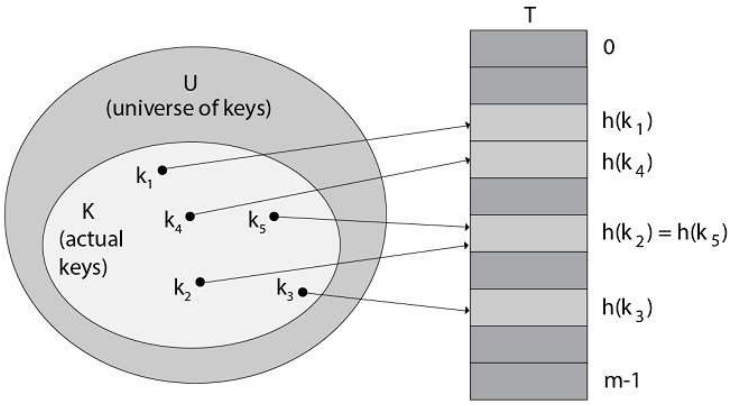

Figure: using a hash function h to map keys to hash-table slots. Because keys K2 and k5 map to the same slot, they collide.

4.6. Why use HashTable?

1. If U (Universe of keys) is large, storing a table T of size [U] may be impossible.
2. Set k of keys may be small relative to U so space allocated for T will waste.

So Hash Table requires less storage. Indirect addressing element with key k is stored in slot k with hashing it is stored in h (k) where h is a hash fn and hash (k) is the value of key k. Hash fn required array range.