# HASHING.

→ Mainly used to implement dictionaries & sets. (key : value) pair.

→ Search, Insert and Delete all in O(1) time.

→ All values are unique, no duplicates allowed.
If same inserted again, previous will be overwritten.

→ Not useful for : Sorted order. } Usually AVL or RedBlack tree used.
Finding closest value. }
Prefix Searching. } Trie used.

## ① → Applications

→ Hash Table → DS
Hashing → Technique.

→ Second most used DS after arrays.

→ a) Implementing Dictionaries.
b) Cryptography
c) Indexing in Databases.
d) Caches.
e) Symbol Tables in compilers
etc.

## ② Direct Address Table

→ Idea is a boolean array here all values 0 initially

```
delete(i) { table[i] = 0;}

insert(i) { table[i] = 1;}

Search(i) { return table[i];}
```

But there are problems with this DS (i.e with DA Table)

1) Can't handle large numbers;

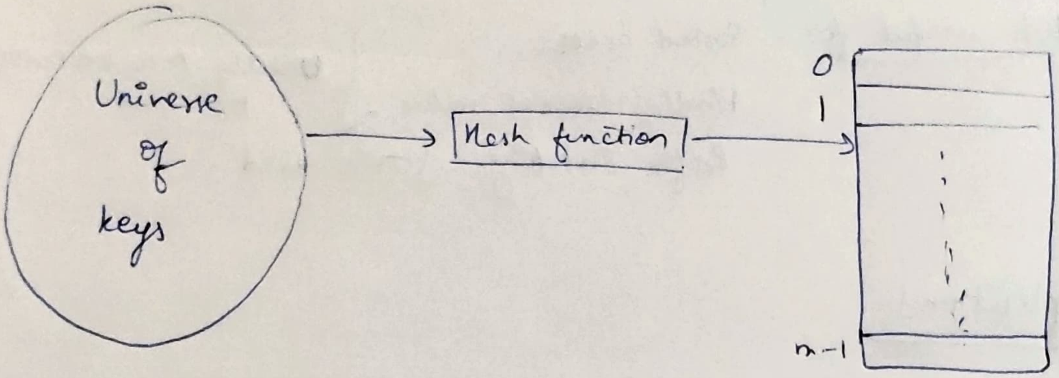
2) Can't handle floating point numbers:

3) " " strings as keys.

} This is where hashing comes in.

---

## ③ Hash functions



⇒ Requirements

- Should be able to map a large key to same small key
- Should generate values from 0 to m-1
- Should be fast
- Should uniformly distribute large keys in Hash Table slots

⇒ Examples.

1) $\underline{h(\text{large\_key}) = \text{large\_key} \% m}$

m → generally taken as prime number closest to m.

eg: m = no: of phone no:

large_key = phone no:

→ Not the best, best used in academics sometimes.

2) String hash fn

3) Universal hash. fn

- most used.
- Have various hash fns.

---