Hashing:

Why ?

Searching:

linear search: o(n)

Binary search:O(logn), demerit is that the data needs to be sorted order

Hash can search an item in O(1) well almost when no collision

Hashing is the fastest method of searching.

Problem: If known values , can you use an array to search? But if the size is

Not known then it could be bad on the space

Hashing mapping: one-to-one, one-to-many, many-to-one, manay to many

A hash function takes the value needs to be hashed, and gives us an index in the hash table where it is mapped/stored.

Ideal hashing: one-to-one, with drawback, that I need storage that could be unused. But we can improve it by trading some time complexity. The approach is

To create a many-to-one mapping, thus a hash function which can do that.

An improved hash function that can save us memory is saved.

Use f(x)=x%10;

The problem with is that there is a hash collision, two values ended up in a same key in the hashtable, how do we resolve collision .

Collision resolution technique:

Closed hashing: The size of the hash table is constant.

Open Hashing: The size of the hash table keeps expanding.

Open Hashing:

Chaining:

Closed hashing:

Open addressing

Leaner probing

Quadratic probing

Double hashing

Open Hashing : chaining:

* A hash function resolves a index in the table, which holds a chain of objects ,most preferably in ascending order.
* The hash function should be designed in a way to distribute the elements in an index.

Closed Hashing: Open addressing: Linear probing.

> drawbacks, while deleting , we should rehash

Hashing functions:

Different hash fucntions could be:

> Mod

> Midsquare

> Folding

quizzes: