**Hashing**

Why hashing?

When we want to map some data(information) to a certain key, we use hashing. It is not always possible to use separate array for hashing, hence we use hashmaps.

Applications

* Play with element’s index

We have two options to do this:

* Traverse the whole array and access the element’s

For each element’s accessing, we have to incur O(n) time complexity.

* Map indices to their elements

This requires little extra memory, but optimizes the time complexity to O(1) {unordered\_map} or O(log(n)) {ordered\_map}.

Second option is better.

Another problem with option one

Storing indices of such large elements is not possible as we can declare only array of size 108 (that too global).

Here is where **hashing** comes into picture

**Hashing**

Converting elements into smaller elements using special functions known as Hash functions.

Example: We have to map the following elements with their indices



(0) (1) (2)

Let us consider a hash function

h(x) = x%10

After applying hash function on each element we get,



(0) (1) (2)

|  |  |
| --- | --- |
| Key | Value |
| 7 | 0 |
| 8 | 1 |
| 9 | 2 |

It might happen that while compressing elements, some keys result in the same values. This is called **collision**.

Collision Handling

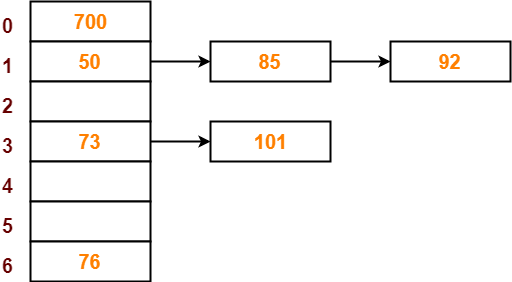
The methods to solve this problem comes under collision handling. There are mainly two methods

1. Separate Chaining
2. Open Addressing

**Separate Chaining**

If a collision occurs, create a chain of values at the same key using a linked list.

Example



In the above example, a chain of values is being made at keys *1, 3.*

Search Time Complexity: O(n)

Load Factor

Average amount of load at each key is called load factor.

Let the number of elements: *n*

Let the number of key on which values are to be mapped: *b*

Therefore the load factor of this combination is *n/b.*

Example:

In the given array, the number of keys on which we need to map the elements be 3.



Ans:

Number of elements = 6

Number of keys = 3

Load factor = 6/3 = 2.

**Open Addressing**

If a collision occurs, do probing.

Probing

In probing, we use a second argument probe number in the hash function. Probe number depends on the key, hence it is written as P(k).

There are three types of probing:

1. Linear Probing:

Probe number is a linear function of key.

Example: P(k) = ak + b.

1. Quadratic Probing

Probe number is a quadratic function of key.

Example: P(k) = ak2 + bk + c.

1. Double Hashing

In double hashing, we use a secondary hash function.

Example: P(k,x) = k\*h2(x), h2(x) is a secondary hash function.