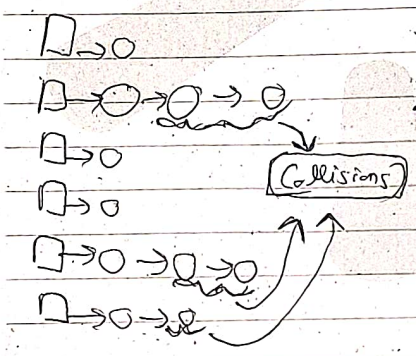
(Explanation)

We have to design HashSet without any inbuilt data structure. We have to build it from scratch.

Basic approach, we are not going to use the space as large as the number of keys, we will be having a smaller array bucket of lets say some size.

We are going to have a hash function to insert values into the array.

In case there is a collision, we add it to the chain in the same bucket



[This is called separate chaining, a way of handling collisions.]

Another way of handling @ Collisions.

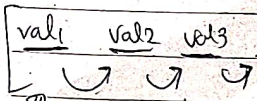
1) Open Addressing → In open Addressing we search for a slot to insert a key using probing techniques.

Linear Probing

$$Index = (h(K) + i) \quad (i = 0 \dots N) \; \& \; N$$

we keep on trying to find the slot until we find any empty.

This results in primary clustering.



[popelating the consecutive entries making primary clustering]

If we get numbers consecutively pointing to this index

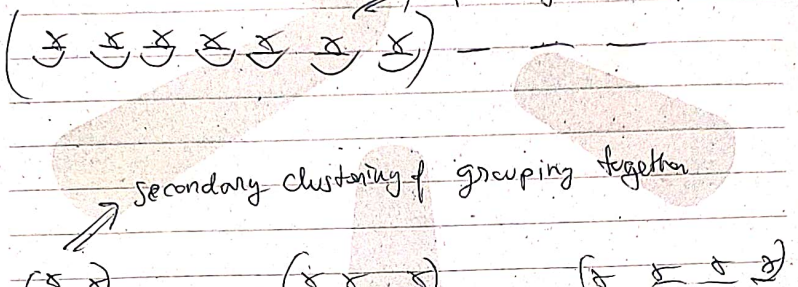
Quadratic Probing :

$$\text{Index} = \left(h(k) + i^2\right) \% N$$

$i = 0, 1, 2, \ldots$

in case of quadratic probing we are moving to 1, then 4, 9 like this in case of collision

this will remove primary clustering but another type of clustering will be introduced called secondary clustering.



Last way of Probing is
(Double Hashing)

$$\text{Index} = \left[h_1(k) + i \times h_2(k)\right] \% N$$

in case of ~~quadratic~~ double hashing. Chances of clustering reduce a lot and its considered best in probing terms to reduce collision.

(clustering is avoided in this because the step of jump $h_2(k)$ is different for every key determined by this h_2 function.

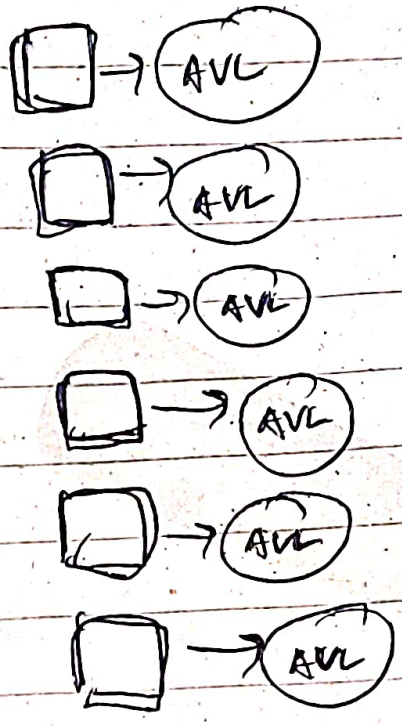
(Lastly) 2. (2 Choice Hashing)

$h_1(\text{key})$ and $h_2(\text{key})$ are two different hash values

you choose b/w the two.

* If either is empty choose any
* If both are occupied, place the element in bucket having fewer elements

We will go with the AVL tree Approach, Each bucket will be called an AVL tree



AVL is self balancing, will try to write the code myself