Insert Delete GetRandom O(1)

First thought of my brain.

getRandom( ) → Probability of getting each number is proportional to its count in the data structure.

(For code watch All O(1) datastructure, DLL implementation)

→ this made me think, that means maximum freq one should be printed always,

remove( ) } → these operations can take place.
insert( ) }

but for random() → I always output the one with maximum frequency

Test Case where it was wrong

[100 → 2
  10 → 1
  1 → 1] → val to frequency

I do 4 times getRandom call()

My code will output 100 always.

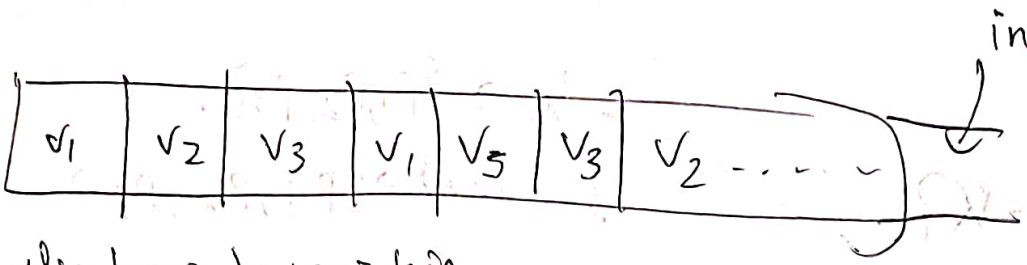
(Probability of 100 = 1)

But ideally probability of 100 = 2/4 = 1/2]

But in this 4 calls, ideally 100 appears 2 times, 10 appears 1 time, 1 appears one.

Then I saw the soln.

Then my brain understood, I have to maintain an array like data structure



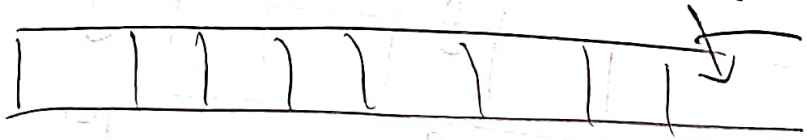
we also have to maintain.

{ Key → List<Positions> or Set<Positions> } → this is important for removal.

insert() → is simple → { add to the end of list } → insert into the set of positions for that key the last position = (size-1)

remove(K) →

Case 1



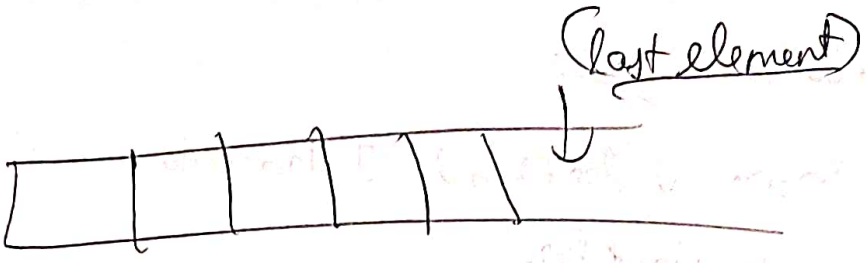
K == last element,

very easy, do a arr.remove(arr.size() - 1).

① →

② → remove it from the set of positions map.

(Case 2)



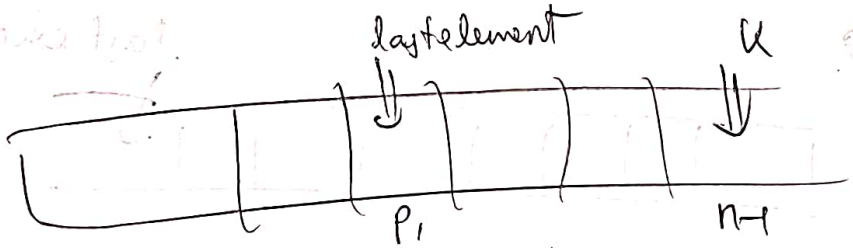
remove (K)

{lastElement != K}

K → {P1, P2, P3}.
lastelement → (Pj, Pk, Pi)

(Step 1) From this map find the position of K. → P1
You already know lastElement's position.
= (size - 1) ⇒ Pj

(Step 2) Swap values (P1, Pj)



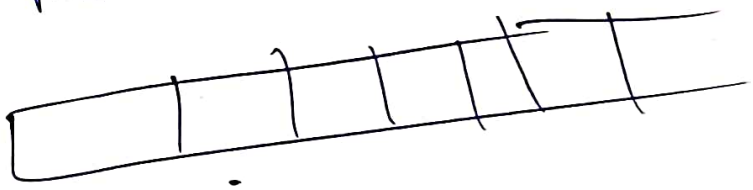
update these positions in the map. →

K → n-1, P2, P3
lastelement → (P1, Pk, Pi)

Now observe carefully,
this is the Case 1 scenario, we saw in last page.
So do the same now.

Imp Dont forget to remap the i position

↳ this is where i ~~was~~ was wrong again,



i

Since in the first approach, I saw testcase, where

100 → 2 times, 10 → 1 time, 1 → 1 time

So I was maintaining a pointer to round robin across the map. i = (i+1) % N after each getRandom() call.

This is wrong I need to let the random function take effect

So correct is.

```
Random rand = new Random()
index = rand.nextInt(arr.size())
```

↙

{0, ..., arr.size()-1}