Creation of HashMap

Sunday, 27 June 2021 5:42 PM

Functions of HashMap. (Key, value)

O(1) put
- → present update
- → absent → Insertion

O(1) remove
- → present → value along with key and remove that pair from HashMap.
- → Absent → null

O(1) get
- → present → value return
- → Absent → null

O(1) containskey →
- present → true
- Absent → false

] check presence of key.

O(n) keySet → set of keys present in HashMap.

O(n) display → print HashMap.

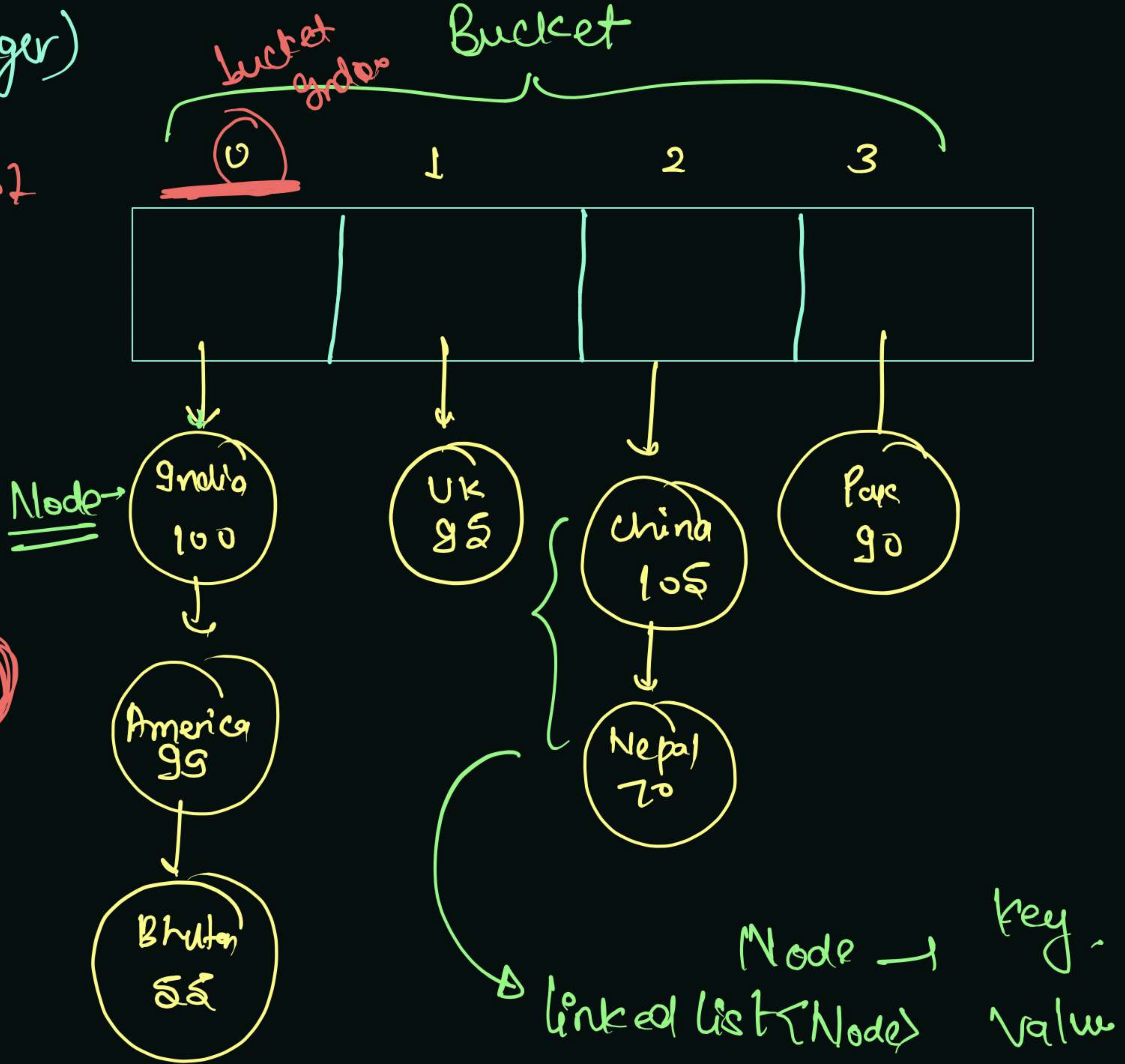
HashMap - key value (String, Integer)

Array of LinkedList of Nodes

"India"	→	100
"pak"	→	90
"Uk"	→	95
"China"	→	105
"Uganda"	→	50
"Nigeria"	→	20
"England"	→	65
"America"	→	99
Nepal	→	70
Bhutan	→	55

put - India → 5 1 0

get - India → 0



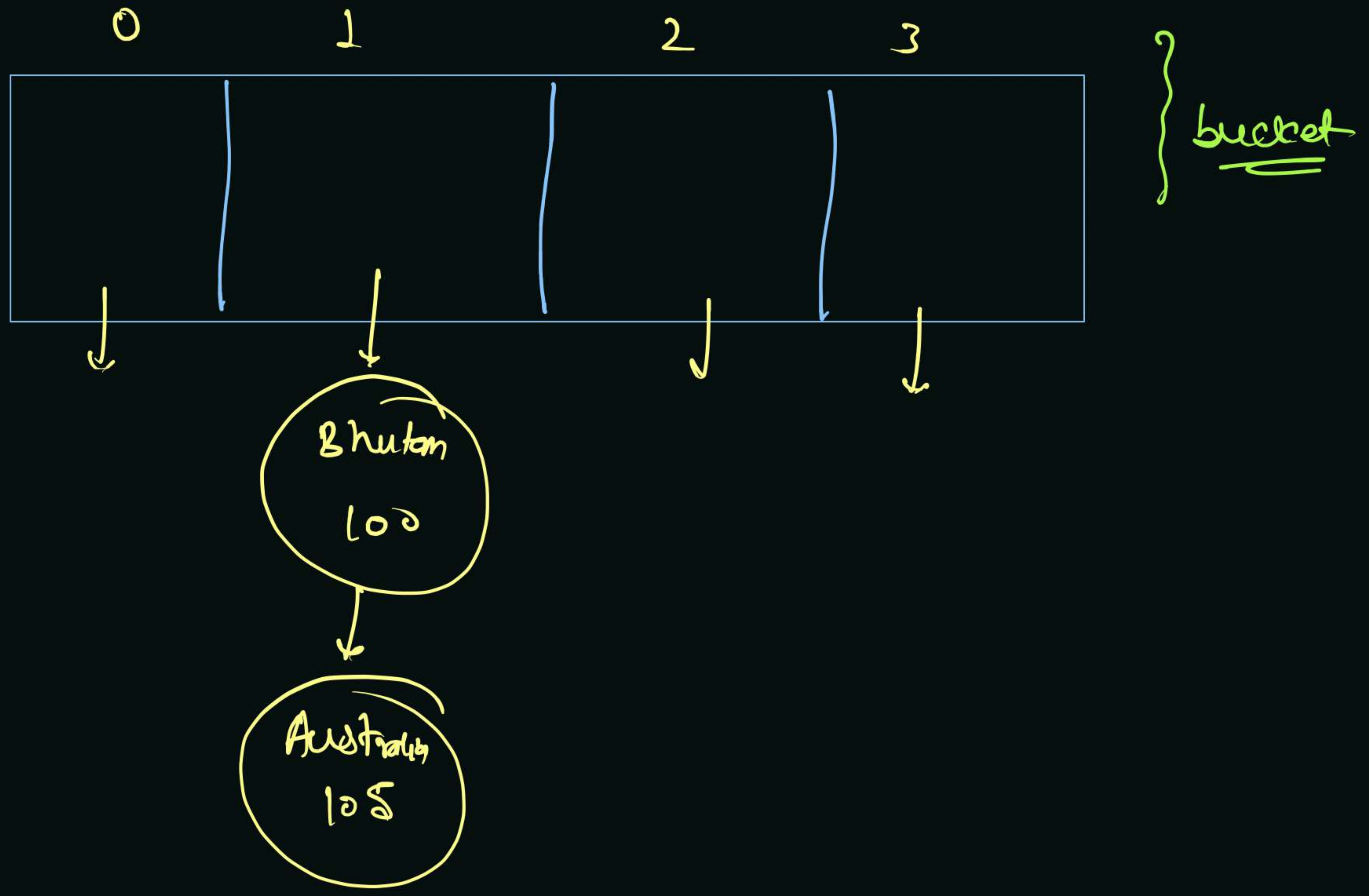
put → "Bhutan" → 100

Bucket Index = 1

"Australia" → 105

Bucket Index = 1

remove ("Bhutan")



Initial size of bucket = $\boxed{4}$

complexity

- removeFirst in Arraylist → No. of node present in single bucket
- remove first in Linkedlist → O(1)

Complexity for

addLast → O(1)

put in HashMap → India 110.

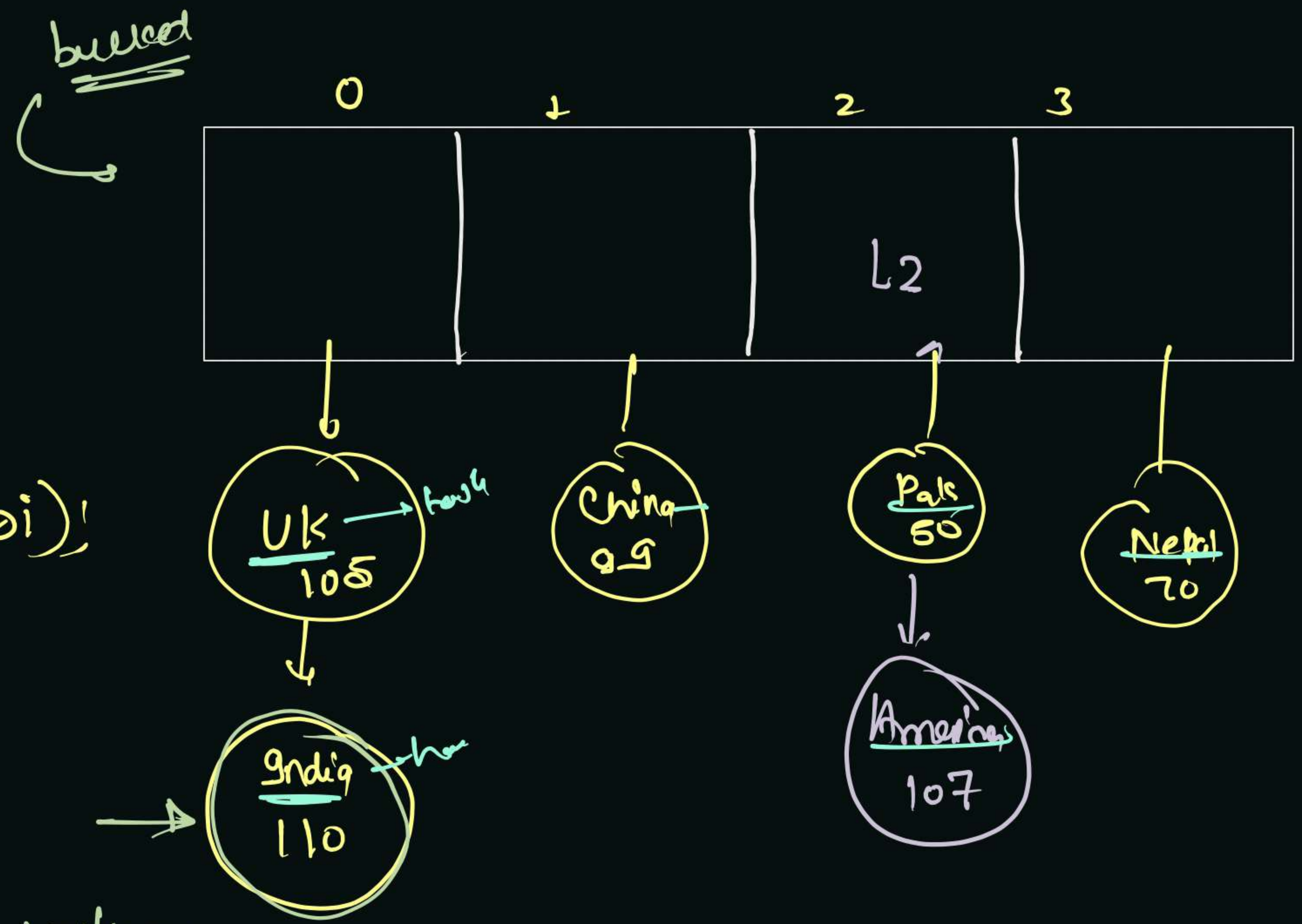
put India - 110
put India - 115

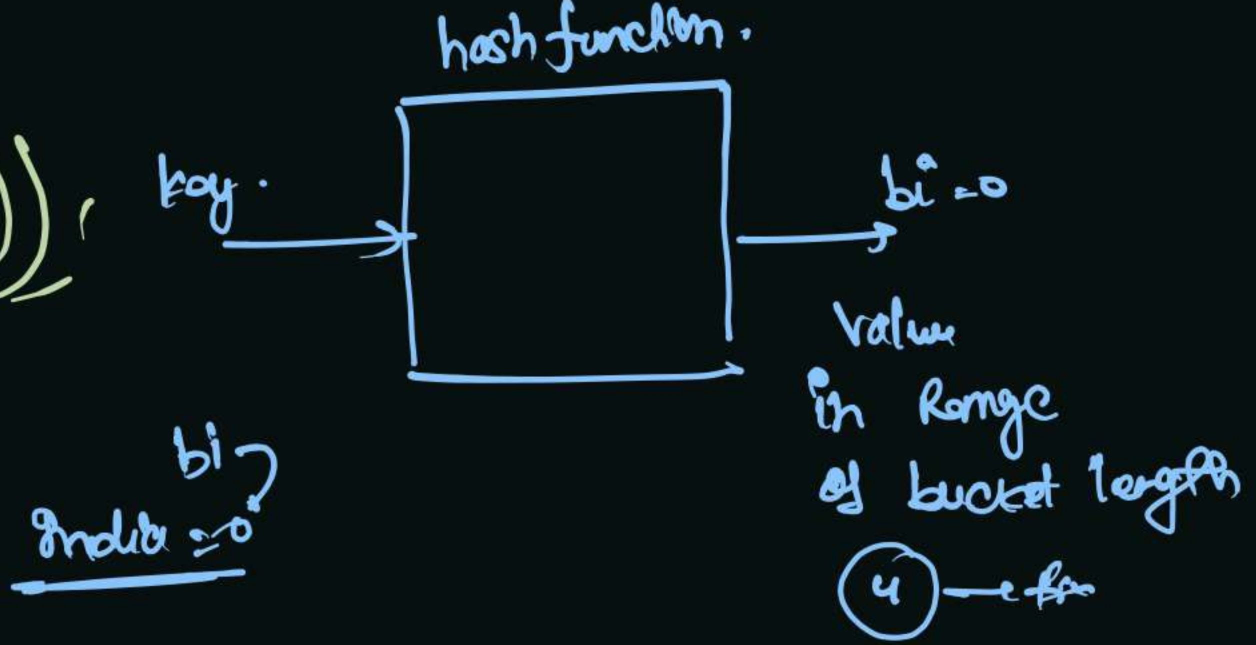
```
put(String key, int value) {
    int bi = hashfunction(key);
    int di = search within bucket(key, bi);
        → Present _ Node Indx In the linked
        → Absent - -1
    if(di != -1) {
        // update.
        bucket[bi].get(di).value = value;
    } else {
        // Insert
        bucket[bi]. addLast( new Node( key, value));
    }
}
```

put "America" 107

bi → 2 for America

di = $\boxed{-1}$





Code → (public)

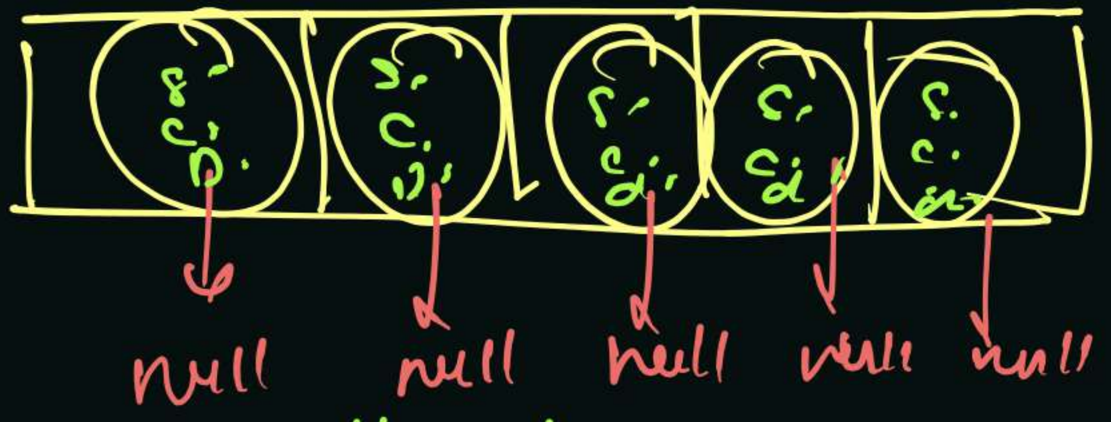
1. put ✓
2. get
3. remove
4. contains key.
5. key set → ArrayList<String>
6. Display

private.

1. hash function ✓
2. search in bucket ✓

Node class → key
value

bucket
size



ArrayList<Integer>[] arr = new ArrayList[5];

s.c. p.

m OpenBoard ??

within loop.

Initialise > [arr[i] = new ArrayList<>();

Hash Code. ⟶ Uniquely define.

↓ ⟶ class create by user]

Object

↳ Extend by Every class of java.

Hash code of Object class

[Hashcode is working as address Not Exactly but in similar pattern.

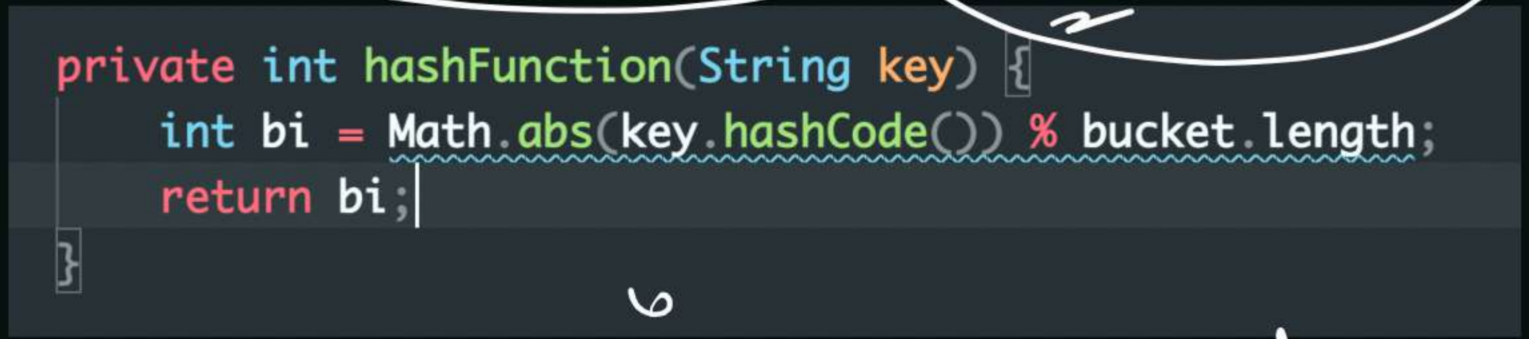
Car { name; }

c1 = new Car()

c1.name = 'A'

c2 = new Car()

c2.name = 'A'



```
private int hashFunction(String key) {
    int bi = Math.abs(key.hashCode()) % bucket.length;
    return bi;
}
```

If type of key is available in java class then hashcode is also available

→ to make it positive

absolute String.hashCode() ⟶ Integer] result

"India" ⟶ 48

45

-∞ to +∞

Integer

0 → bucket.length-1

result % bucket.length;

0 - bucket.length-1

Hashfunction →

① Hash function is based on hashcode.

② Hashcode is impleted ed on Every class of Java.

③ Object class is Extended by user defined class too, so it have presence of hashcode but uniqueness in not necessarily happend in this Sceneria.

④ If we r want to make key of that class then they have make a standard Hashcode function from override of hashcode for uniquely identify the keys.

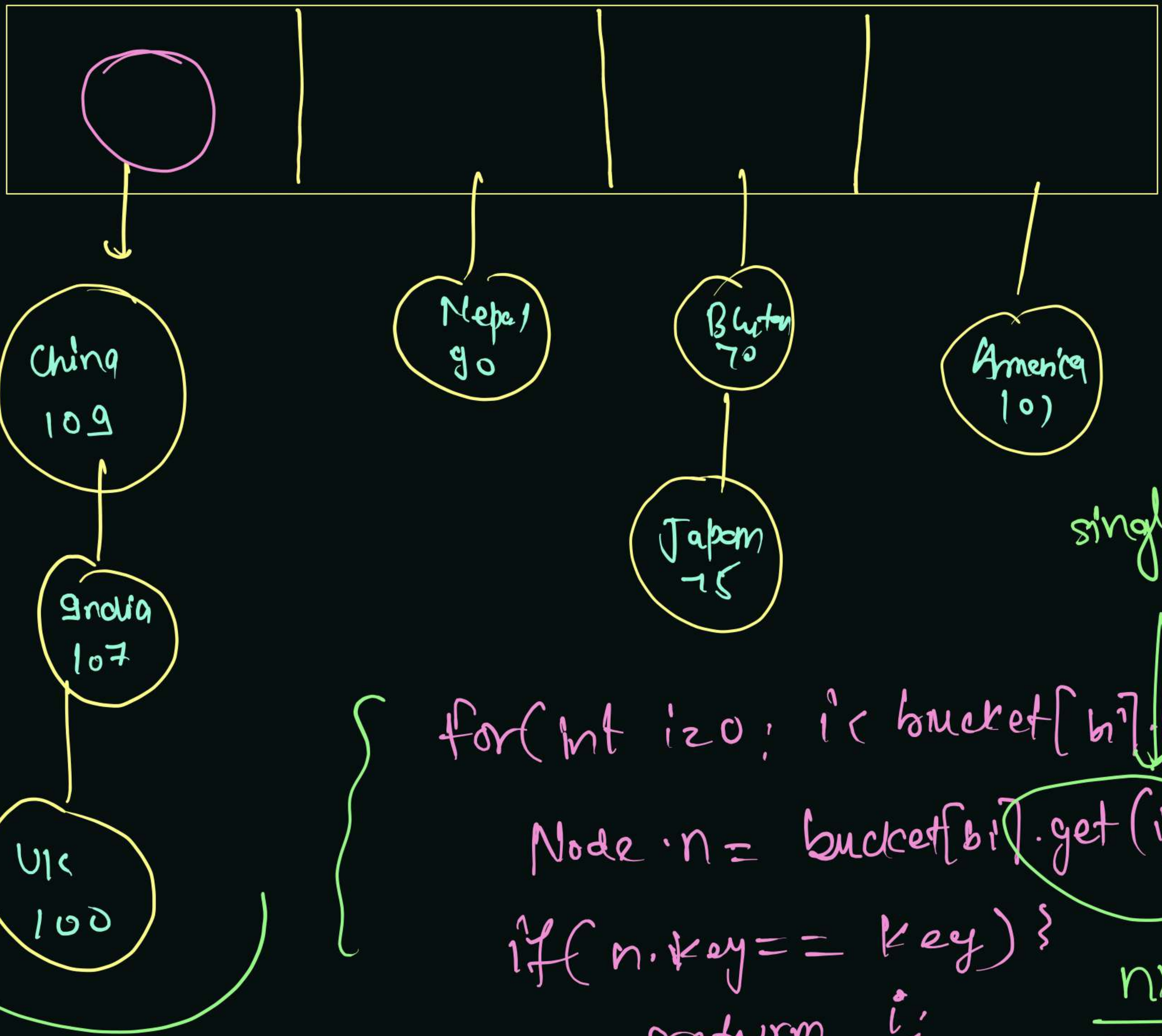
$-\infty <$ hashcode() $< \infty$
any integer value and diff for diff obj.

ⓐ Make it positive
ⓑ wrap it in range of bucket length.

SearchWithinBucket(
    String key, int bi){

key = "India"
bi = 0

check in linked list
present at bucket index



How to optimise this traversal's complexity.

```
for(int i=0; i< bucket[bi].size(); i++){
    Node n = bucket[bi].get(i);
    if(n.key == key){
        return i;
    }
} return -1;
```

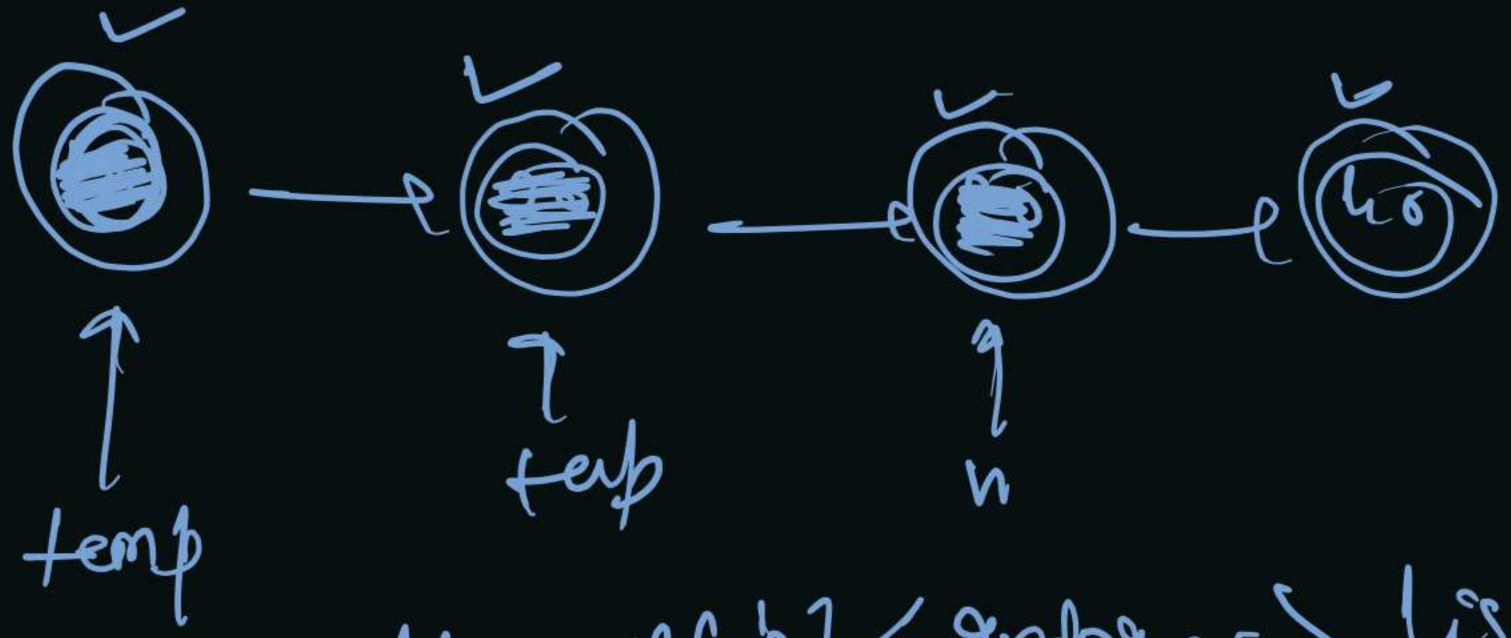
single → O(N)

n x O(n)
O(n^2)

where n → no. of nodes present in single bucket

Iterator loop on linked.

```
O(n)
int di = -1;
for( Node n: bucket [bi]) {
    di++;
    if( n.keys.equals(key) == True) {
        return di;
    }
}
return -1;
```



```
LinkedList<Integer> list
        = new LinkedList<>();

for(int val: list) {
    syso( val);
}
```

get in HashMap-

- present → value return
- Absent → null return

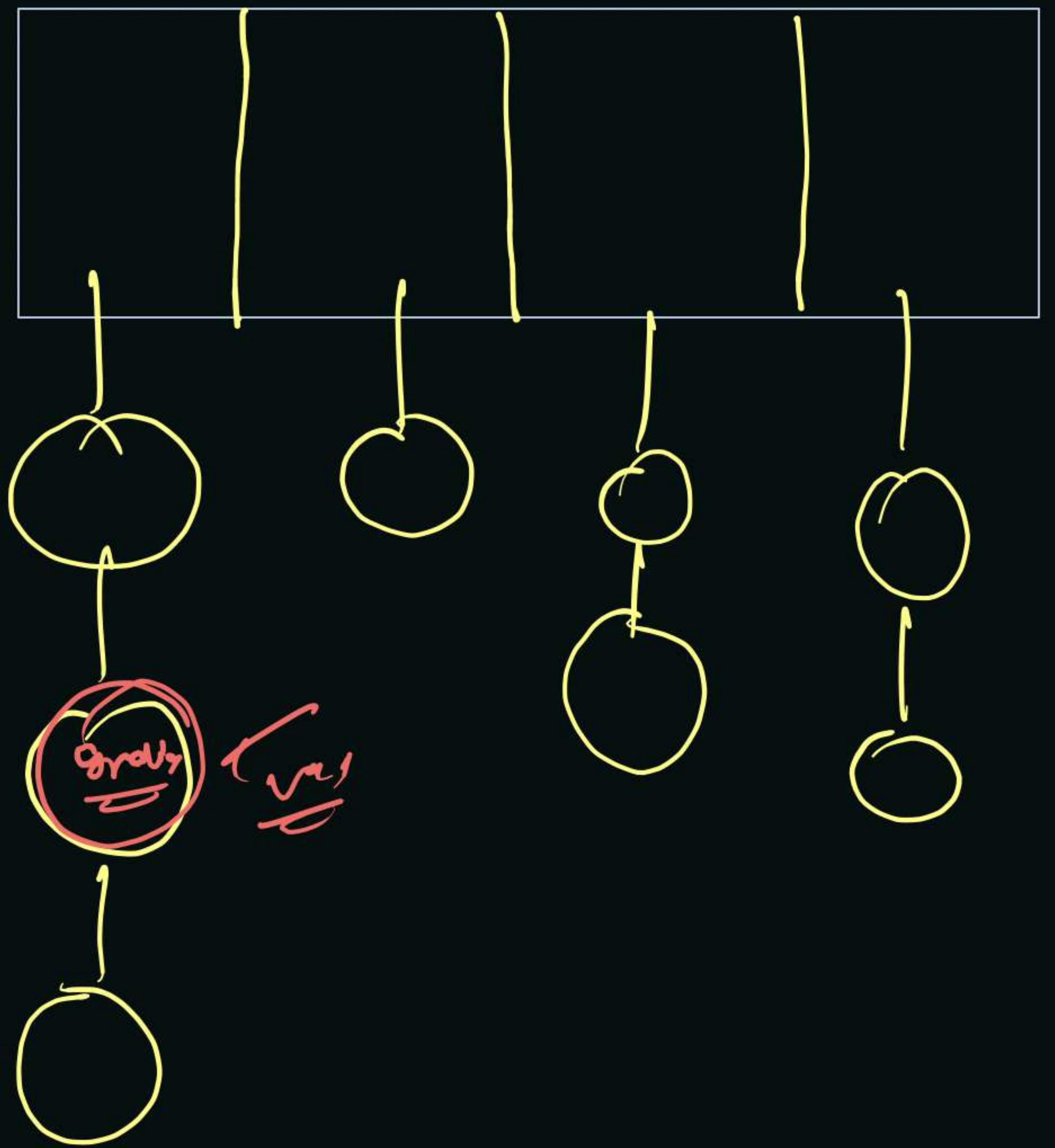
int

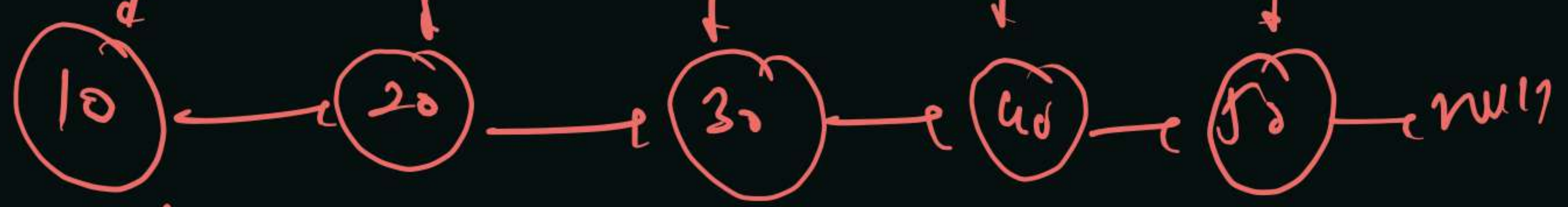
```
bi = hashFunction(key);

di = searchWithinBucket(key, bi);

if( di == -1) {
    // Absent
    return null -1;
} else {
    // present
    return bucket[bi].~~get~~(di).value;
}
```

pres = bi=1





ite

get(i) → O(n²)

Node temp = head;

```
while(temp != null){
    temp = temp.next
}
```

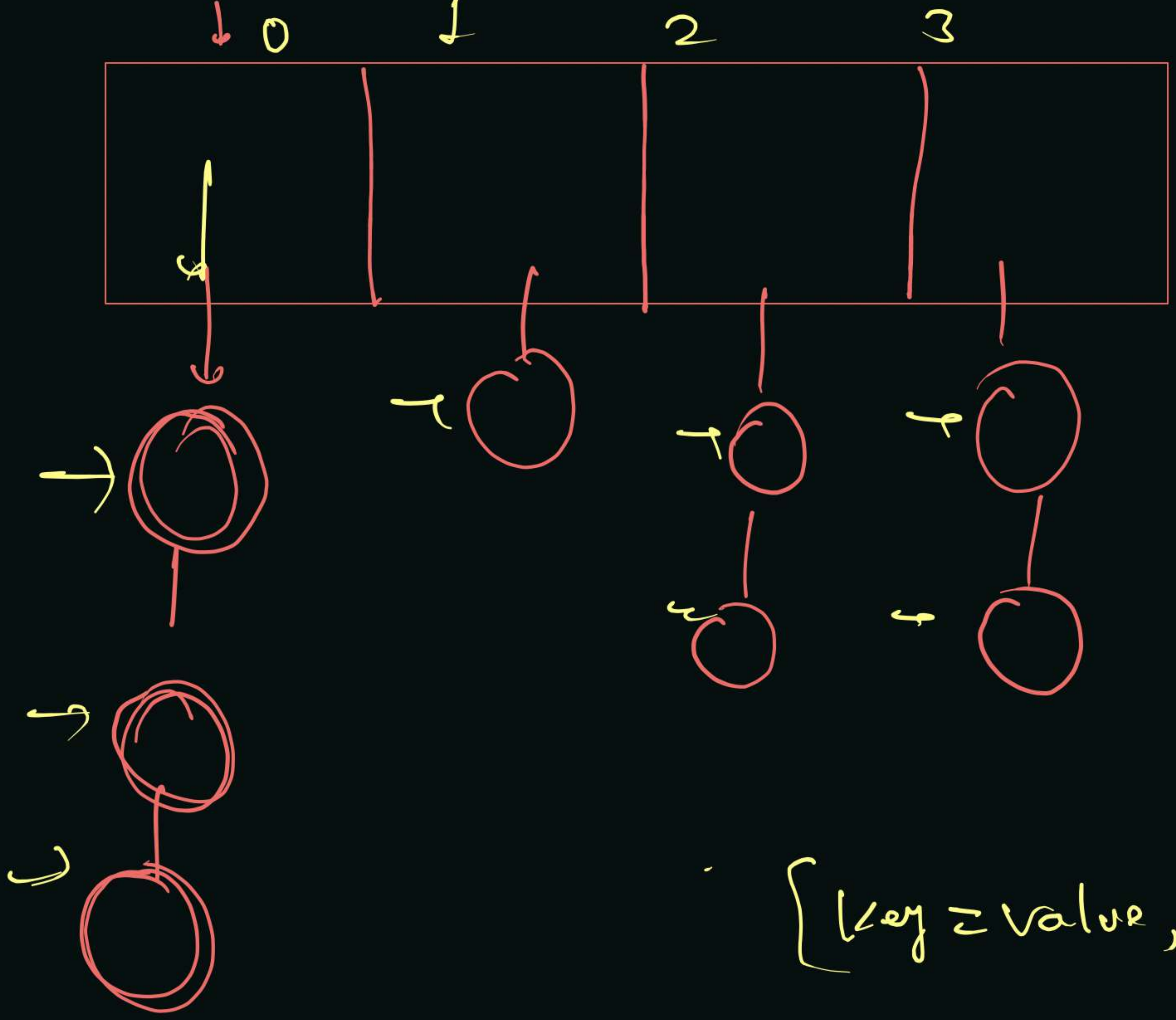
O(n)

keySet

Arraylist <String> keys
= new ArrayList<>();

bi

nodes ~~of~~ linked
for. Each loop.



{key = value, ——, }

keys → { ______________________________ };

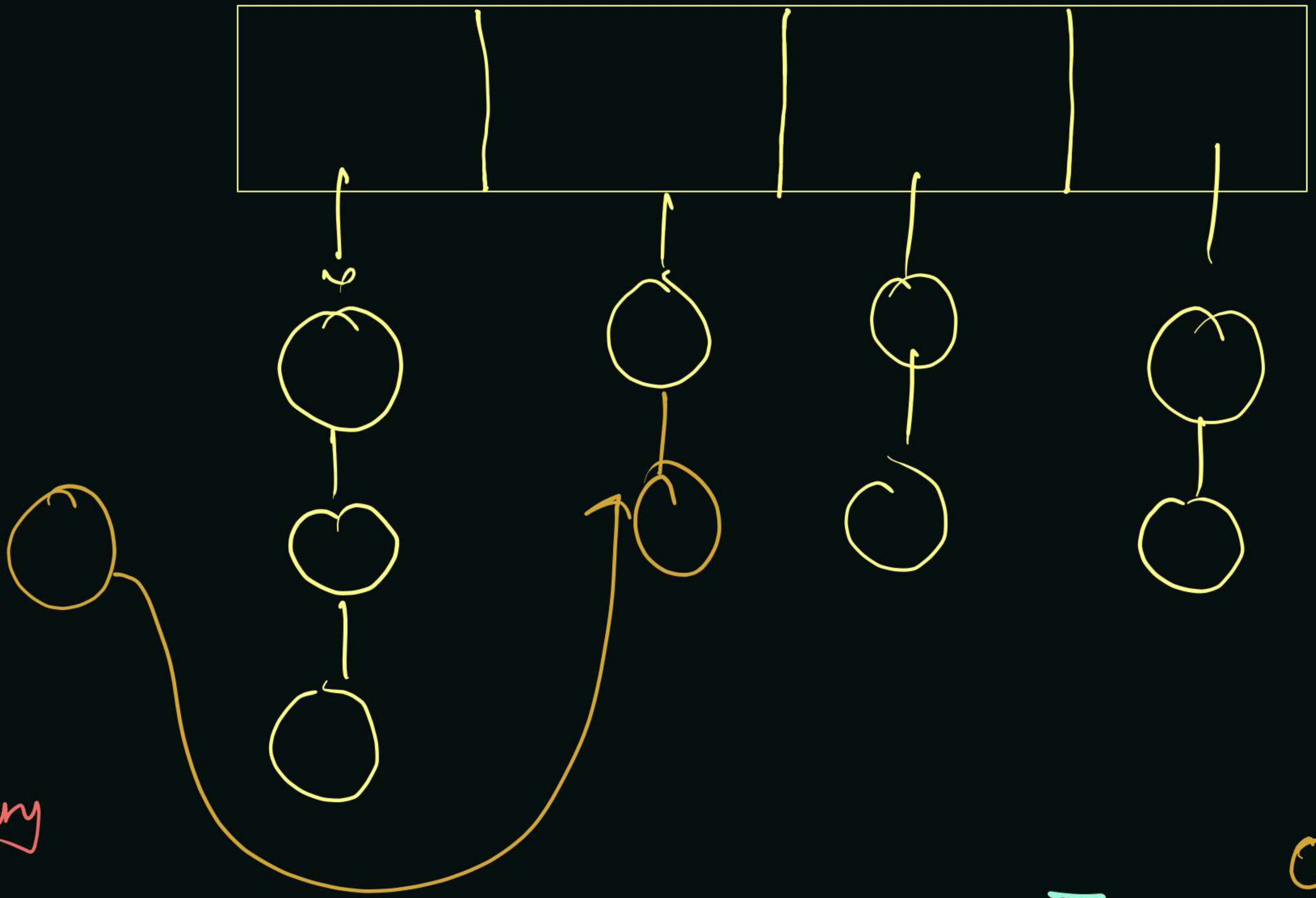
return keys.

Loading Factor

average → 'λ' = $\dfrac{\text{total Nodes}}{\text{bucket length}}$

→ No. of nodes present in single bucket

$\lambda = \dfrac{8}{4} = \boxed{2}$



Maintain λ on Every put operation →

$\lambda = \dfrac{9}{4} = 2.25$, $\boxed{\lambda > 2}$

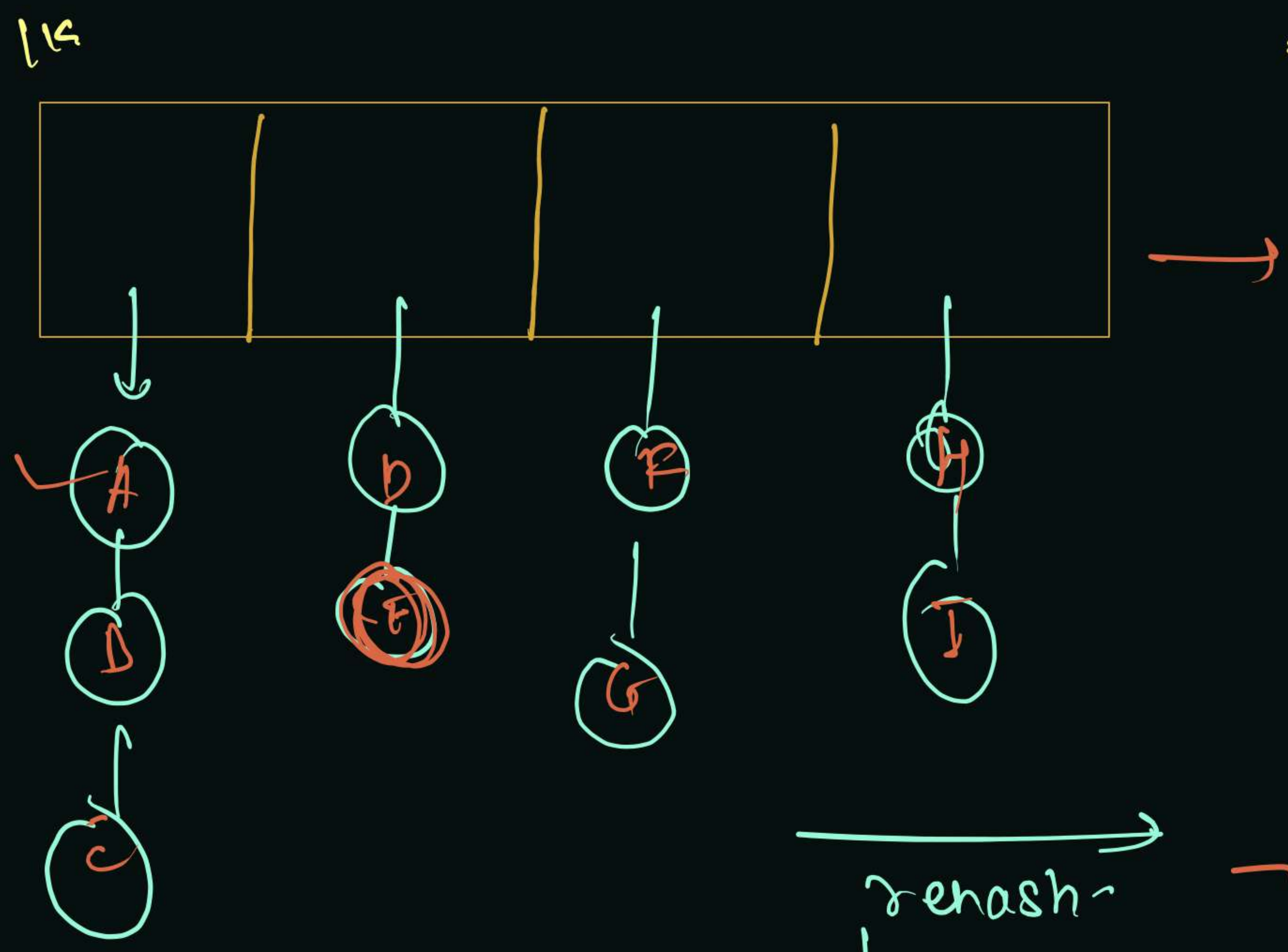
rehash( ); ⟶ rehash help in Maintainence in λ'.

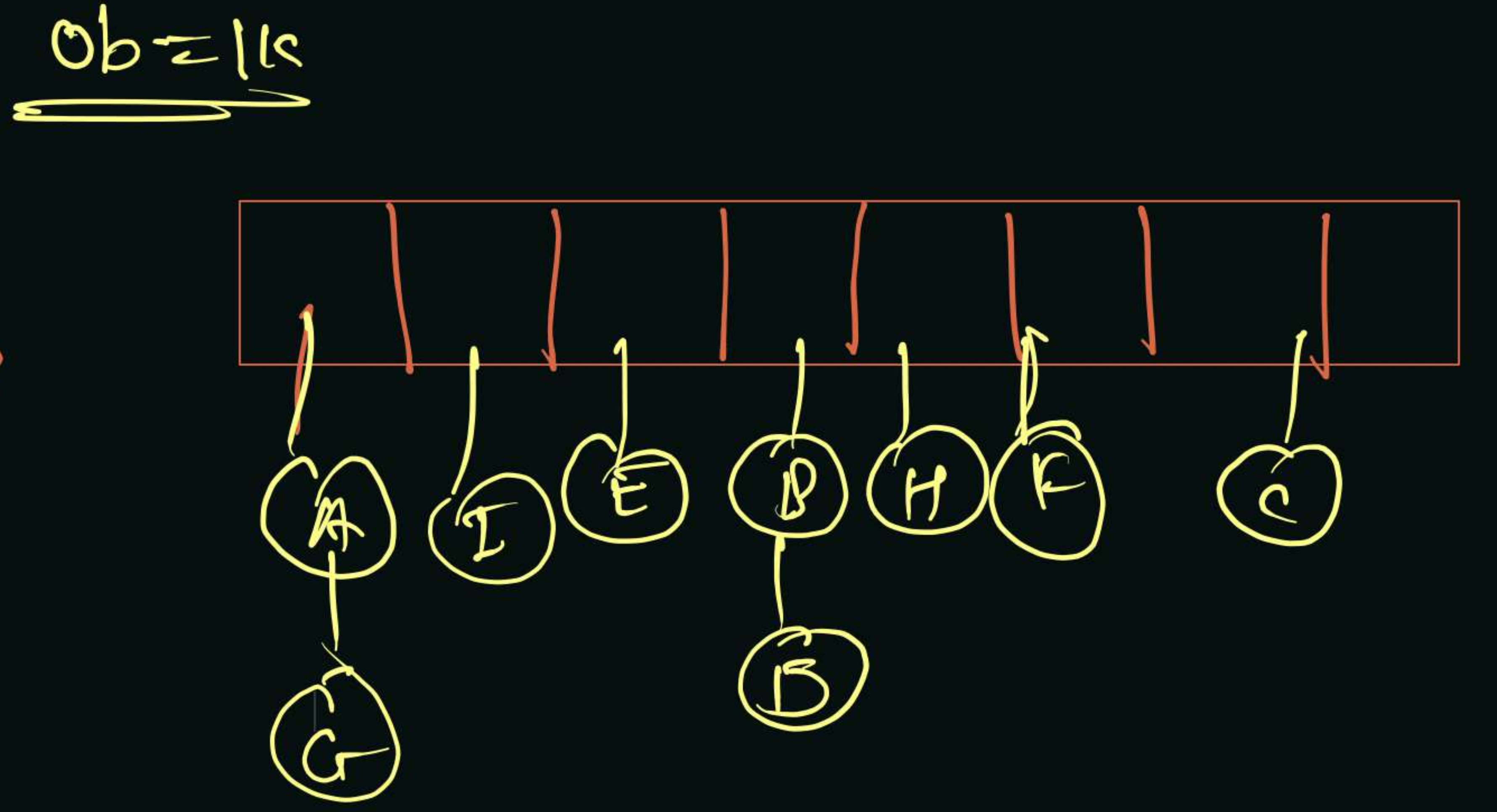
put, remove, get, contains key,] → no. of nodes present in single bucket

$O(\lambda)$

average

hashing





$\lambda = \frac{8}{4} = 2$

put $\longrightarrow$ ○

$\lambda = \frac{9}{4} = 2.25$

rehash

bucket length is twice from current length.

travel on every node and call for put again

$\lambda = \frac{9}{8} = 1.125$

$\boxed{\lambda < 2}$

Fine