

Control time

$F, 60$   
 $B, 65$   
 $C, 75$   
 $E, 85$   
 $G, 95$   
 $H, 77$   
 $I, 39$   
 $J, 37$   
 $K, 73$

$12/4 = 3$   
 $12/8 = 1.5$

$0(n)$   
 $12/8 = 1.5$

HashMap

$D, 85$   
 $A, 60$   
 $B, 65$   
 $C, 75$

$0(n)$   
 $12/8 = 1.5$

hashFun

$ASIC \times 4$

```
public void put(K key, V value) {  
    int idx = Hashfun(key);  
    Node temp = bukt.get(idx);  
    while (temp != null) {  
        if (temp.key == key) {  
            temp.value = value;  
            return;  
        }  
        temp = temp.next;  
    }  
    temp = bukt.get(idx);  
    Node node = new Node();  
    node.key = key;  
    node.value = value;  
    node.next = temp;  
    bukt.set(idx, node);  
    size++;  
}
```

```
public void put(K key, V value) {  
    // ...  
}
```

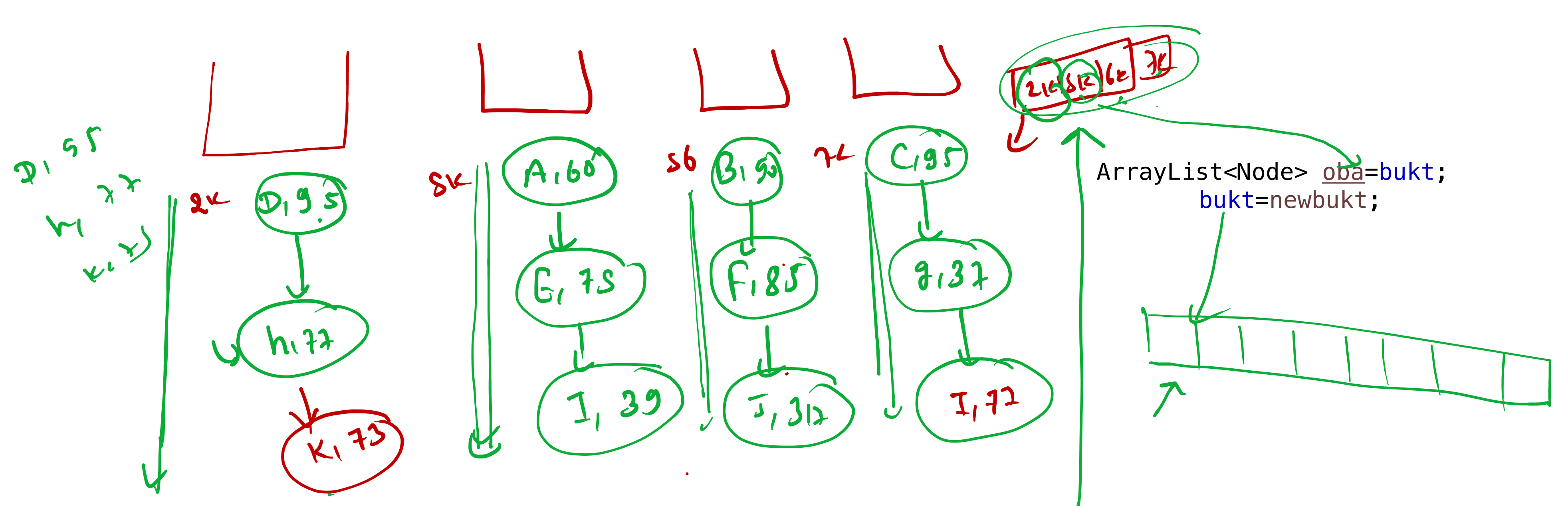
```
class Node {  
    Key -  
    value -  
    next -  
}
```

pre-

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

$1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5$

```
public V remove(K key) {  
    int idx = Hashfun(key);  
    Node curr = bukt.get(idx);  
    Node prev = null;  
    while (curr != null) {  
        if (curr.key.equals(key)) {  
            break;  
        }  
        prev = curr;  
        curr = curr.next;  
    }  
    if (curr == null) {  
        return null;  
    }  
    else if (prev == null) {  
        bukt.set(idx, curr.next);  
        size--;  
    }  
    else {  
        prev.next = curr.next;  
        size--;  
    }  
}
```



```
for (Node temp : oba) {  
    while (temp != null) {  
        put(temp.key, temp.value);  
        temp = temp.next;  
    }  
}
```

1 2 3 4 5

7 8

10 11 12

15 16 17

20 21

```
int ans = 0;  
for (int key : map.keySet()) {  
    if (map.get(key)) {  
        int c = 0;  
        while (map.containsKey(key)) {  
            c++;  
            key++;  
        }  
        ans = Math.max(ans, c);  
    }  
}
```

Int	boolean
2	F
5	F
10	T
4	F
1	T
3	F
7	T
11	F
8	F
15	T
16	F
20	T