Hash map introduction

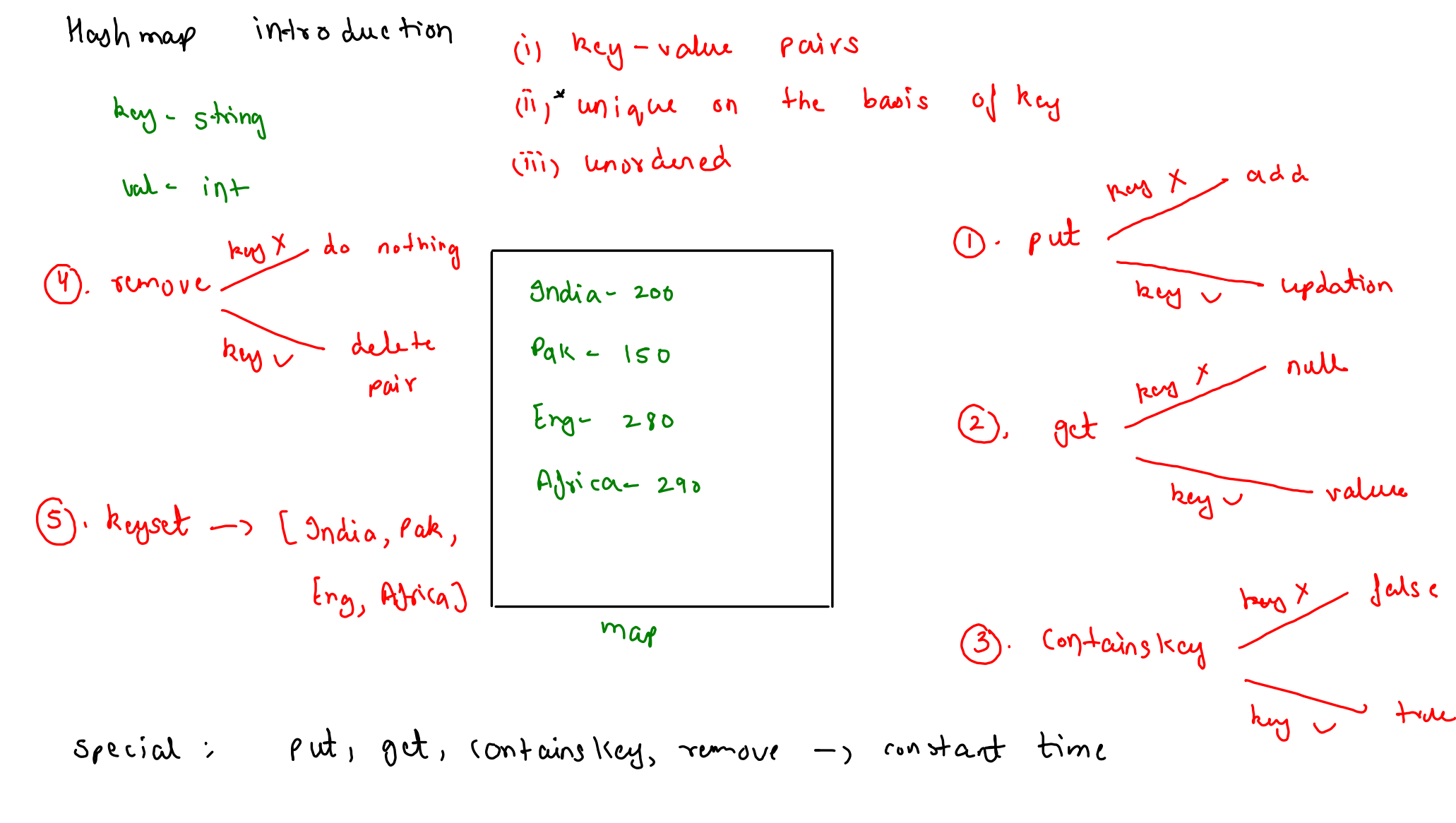
(i) key - value pairs

(ii)* unique on the basis of key

(iii) unordered

key - string

val - int



1. put
   - key X → add
   - key ✓ → updation
2. get
   - key X → null
   - key ✓ → value
3. containsKey
   - key X → false
   - key ✓ → true
4. remove
   - key X → do nothing
   - key ✓ → delete pair
5. keyset → [India, Pak, Eng, Africa]

special : put, get, containsKey, remove → constant time

Highest Frequency Character

Str : b c d a b m g m b a c a d a a

key vs value

char vs int

freq. map

b - 3

c - 2

d - 2

a - 4 → hfc

m - 2

g - 1

```java
//build freq map
for(int i=0; i < str.length();i++){
    char ch = str.charAt(i);

    if(map.containsKey(ch) == false) {
        map.put(ch,1);
    }
    else {
        int nf = map.get(ch) + 1;
        map.put(ch,nf);
    }
}
```

Str : b c d a b m g m b a c a d a a

b - 3
c - 2
d - 2
a - 5
m - 2
g - 1

keyset -> [g, m, a, c, d, b]

```java
//find out the hfc
char hfc = '\0';
int hf = Integer.MIN_VALUE;


for(char key : map.keySet())  {
    int val = map.get(key);

    if(val > hf) {
        hfc = key;
        hf = val;
    }
}
```

hfc = ~~g~~ ~~m~~ a

hf = ~~-∞~~ ~~2~~ ~~2~~ 5

Get Common Elements - 1

Hashmap : key vs value

Hashset : keys

hashset using a1

~~5~~

~~2~~

6

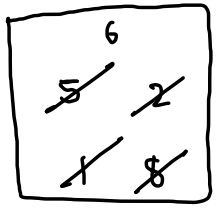
~~1~~

~~8~~

a1 : 5 2 6 5 1 8 1

a2 : 2 5 8 1 1 3 2 ↑

ans: 2 5 8 1



hs

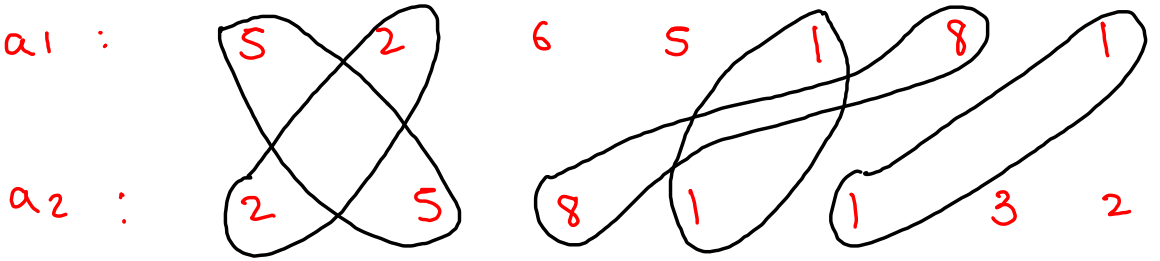
a1 : 5 2 6 5 1 8 1

a2 : 2 5 8 1 1 3 2 ↑

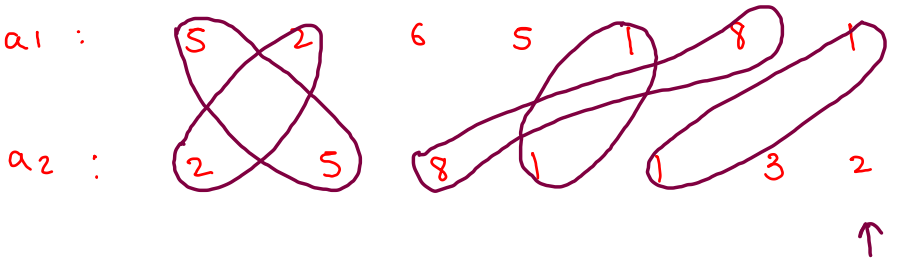
ans : 2 5 8 1

```java
public static void gce1(int[]a1,int[]a2) {
    HashSet<Integer>hs = new HashSet<>();

    //build hs using a1
    for(int i=0; i < a1.length;i++) {
        hs.add(a1[i]);
    }

    //print common elements in order a2
    for(int i=0; i < a2.length;i++) {
        if(hs.contains(a2[i]) == true) {
            System.out.println(a2[i]);
            hs.remove(a2[i]);
        }
    }
}
```

Get Common Elements - 2



ans : 2 5 8 1 1




```java
//populate map using a1
for(int i=0; i < a1.length;i++) {
    if(map.containsKey(a1[i]) == false) {
        map.put(a1[i],1);
    }
    else {
        int nf = map.get(a1[i]) + 1;
        map.put(a1[i],nf);
    }
}
```

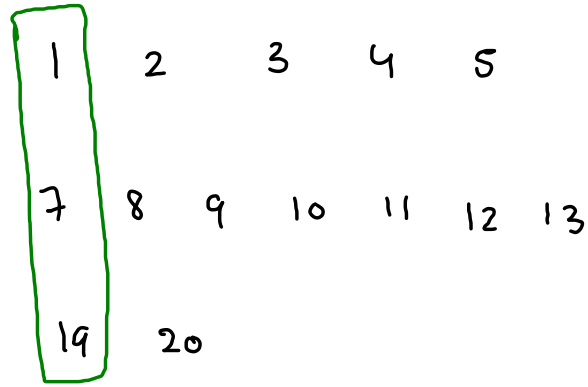
ans 2 5 8 1 1

5 - ~~2~~ 1
2 - ~~1~~ 0
6 - 1
1 - ~~2~~ ~~1~~ 0
8 - ~~1~~ 0

```java
//print common elements in order a2
for(int i=0; i < a2.length;i++) {
    if(map.containsKey(a2[i]) == true && map.get(a2[i]) > 0) {
        System.out.println(a2[i]);
        int nf = map.get(a2[i]) - 1;
        map.put(a2[i],nf);
    }
}
```

Longest Consecutive Sequence Of Elements

3 5 9 11 1 13 2 10 12 19 7 4 8 20



3 5 9 11 1 13 2 10 12 19 7 4 8 20

(i) find seq. start → (a) assume every ele is a seq. start

(b) discard the ele which can-not be a seq.

3 - ~~T~~ F

5 - ~~T~~ F

9 - ~~T~~ F

11 - ~~T~~ F

1 - T

13 - ~~T~~ F

2 - ~~T~~ F

10 - ~~T~~ F

12 - ~~T~~ F

19 - T

7 - T

4 - ~~T~~ F

8 - ~~T~~ F

20 - ~~T~~ F

3 5 9 11 1 13 2 10 12 19 7 4 8 20

(ii) find the best ans out of all sequences

3 - T F

5 - T F

9 - T F

11 - T F

1 - T

13 - T F

2 - T F

10 - T F

12 - T F

19 - T

7 - T

4 - T F

8 - T F

20 - T F

3 5 9 11 1 2 10 12 19 7 4 8 20

```java
//1. find the seq start points
HashMap<Integer,Boolean>map = new HashMap<>();

//a. assume all elements as seq start
for(int i=0; i < arr.length;i++) {
    map.put(arr[i],true);
}

//b. discard elements which can't be a seq start
for(int i=0; i < arr.length;i++) {
    if(map.containsKey(arr[i]-1) == true) {
        map.put(arr[i],false);
    }
}
```

3 - ~~T~~ F 2 - ~~T~~ F 4 - ~~T~~ F

5 - ~~T~~ F 10 - ~~T~~ F 8 - ~~T~~ F

9 - ~~T~~ F 12 - ~~T~~ F 20 - ~~T~~ F

11 - ~~T~~ F 19 - T

1 - T 7 - 1

```java
//2. find the longest consecutive seq
int olen = 0;
int st = 0;

for(int i = 0; i < arr.length;i++) {
    if(map.get(arr[i]) == true) {
        int len = 0;

        while(map.containsKey(arr[i] + len) == true) {
            len++;
        }

        if(len > olen) {
            olen = len;
            st = arr[i];
        }
    }
}
```

~~5~~ 6

olen = ~~0~~

st = ~~0~~ ~~1~~ 7

arr[i] = 7

len = 0 ~~1~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6

7 8 9 10 11 12

```java
//2. find the longest consecutive seq
int olen = 0;
int st = 0;

for(int i = 0; i < arr.length;i++) {
    if(map.get(arr[i]) == true) {
        int len = 0;

        while(map.containsKey(arr[i] + len) == true) {
            len++;
        }

        if(len > olen) {
            olen = len;
            st = arr[i];
        }

    }
}
```

(1) 3 - F

(1) 5 - F

(1) 9 - F

(1) 11 - F

(5) 1 - T

(1) 2 - F

(1) 10 - F

(1) 12 - F

(2) 19 - T

(6) 7 - T

(1) 4 - F

(1) 8 - F

(1) 20 - F

( ) -> no. of operation