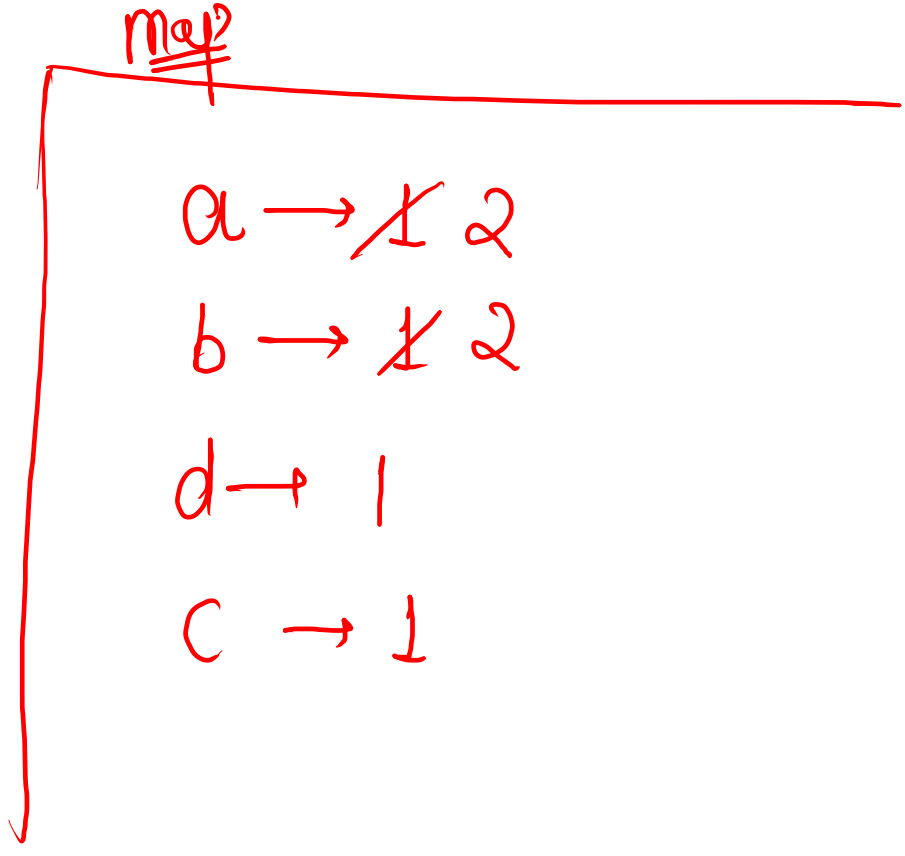
Character and it's Frequency

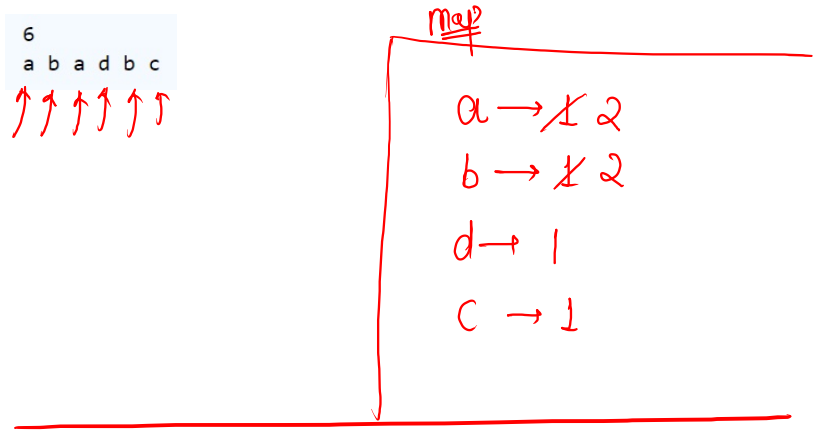
6

a b a d b c

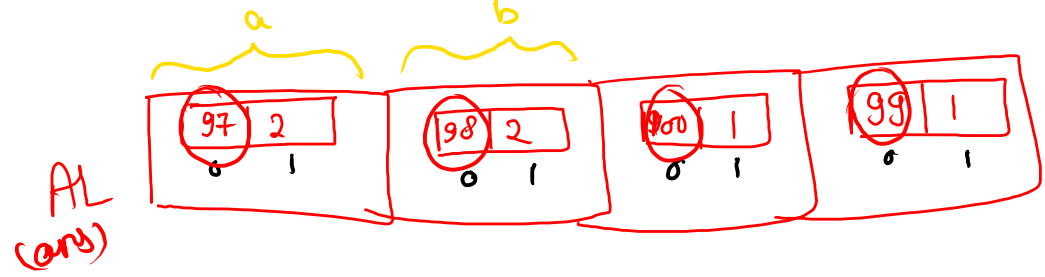


```java
public static void countFreq(ArrayList<Character> arr) {
    HashMap<Character, Integer> map = new HashMap<>();
    for (int i = 0; i < arr.size(); i++) {
        if ( !map.containsKey( arr.get(i) ) ) {
            map.put( arr.get(i), 1 );
        } else {
            int freq = map.get( arr.get(i) );
            map.put( arr.get(i), freq + 1 );
        }
    }

    ArrayList<int[]> ans = new ArrayList<>();
    for (Map.Entry<Character,Integer> entry : map.entrySet()) {
        char key = entry.getKey();
        int val = entry.getValue();
        int[] ar = new int[]{key, val};
        ans.add(ar);
    }

    Collections.sort(ans, (a, b) -> {
        return a[0] - b[0];
    });

    for (int i = 0; i < ans.size(); i++) {
        int[] rem = ans.get(i);
        System.out.println( (char)rem[0] + " " + rem[1] );
    }
}
```

6
a b a d b c

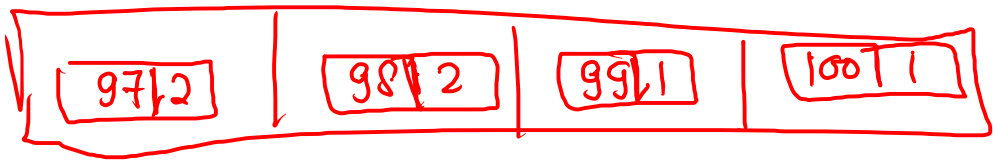


key = a b c
val = 2 2 1



AL (ans)

Sorted



a, 2 b, 2 c, 1 d, 1

Code

```java
public static void countFreq(ArrayList<Character> arr) {
    HashMap<Character, Integer> map = new HashMap<>();
    for (int i = 0; i < arr.size(); i++) {
        if ( !map.containsKey( arr.get(i) ) ) {
            map.put( arr.get(i), 1 );
        } else {
            int freq = map.get( arr.get(i) );
            map.put( arr.get(i), freq + 1 );
        }
    }

    ArrayList<Character> ans = new ArrayList<>();
    for (Map.Entry<Character,Integer> entry : map.entrySet()) {
        char key = entry.getKey();
        ans.add(key);
    }

    Collections.sort(ans);

    for (int i = 0; i < ans.size(); i++) {
        System.out.println( ans.get(i) + " " + map.get( ans.get(i) ) );
    }
}
```

key (sorted)

get value from HM

Longest Substring Without Repeating Characters 6

str = "ababc"

substring

a ← 1
ab ← 2
aba
abab
ababc

b ← 1
ba ← 2
bab
babc

a ← 1
ab ← 2
abc ← 3

b ← 1
bc ← 2

c ← 1

approch → $O(N^2)$ → TLE

Code

```java
public static void twoSumHM(int[] arr, int n, int tar) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        map.put( arr[i], i );
    }

    // val1 + val2 == tar
    int[] ar = new int[2];
    for (int i = 0; i < n; i++) {
        int val1 = arr[i];
        int val2 = tar - val1;
        if (map.containsKey(val2)) {
            ar[0] = i;
            ar[1] = map.get(val2);  // idx
            Arrays.sort(ar);
            System.out.println(ar[0] + " " + ar[1]);
            return;
        }
    }
}
```

⇒ HashSet (valdataType)

adv

↳ it always store values in sorted order

↳ all fⁿ have complexity of O(1)

↳ it will also override already present value

Syntex :-

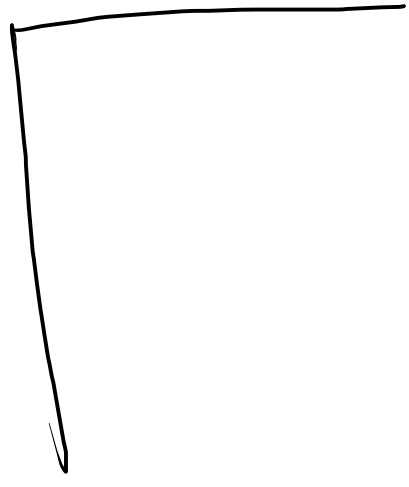
```
HashSet < KeyDataType> set = new HashSet<>();
```

→ HashSet fn

- set.add(val); → to add
- set.contains(val); → to check set
  return boolean DT

OR

- Set.size();
- set.isEmpty();
- set.remove(val); → to delete.



→ add and remove function

```
public static void main(String[] args) {
    HashSet<Integer> set = new HashSet<>();
    set.add(1);
    set.add(3);
    set.add(5);
    set.add(2);
    set.add(4);
    set.add(5);
    set.remove(5);
    System.out.println(set);
}
```

1
3
~~5~~
2
4

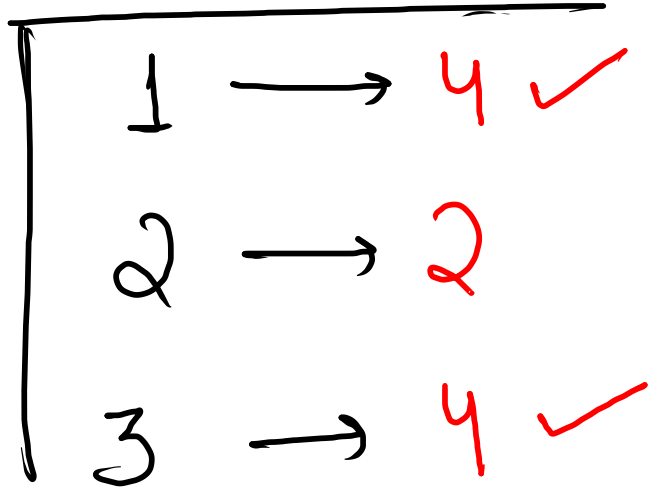
→ 4

[1, 2, 3, 4]

Unique Number of Occurrences

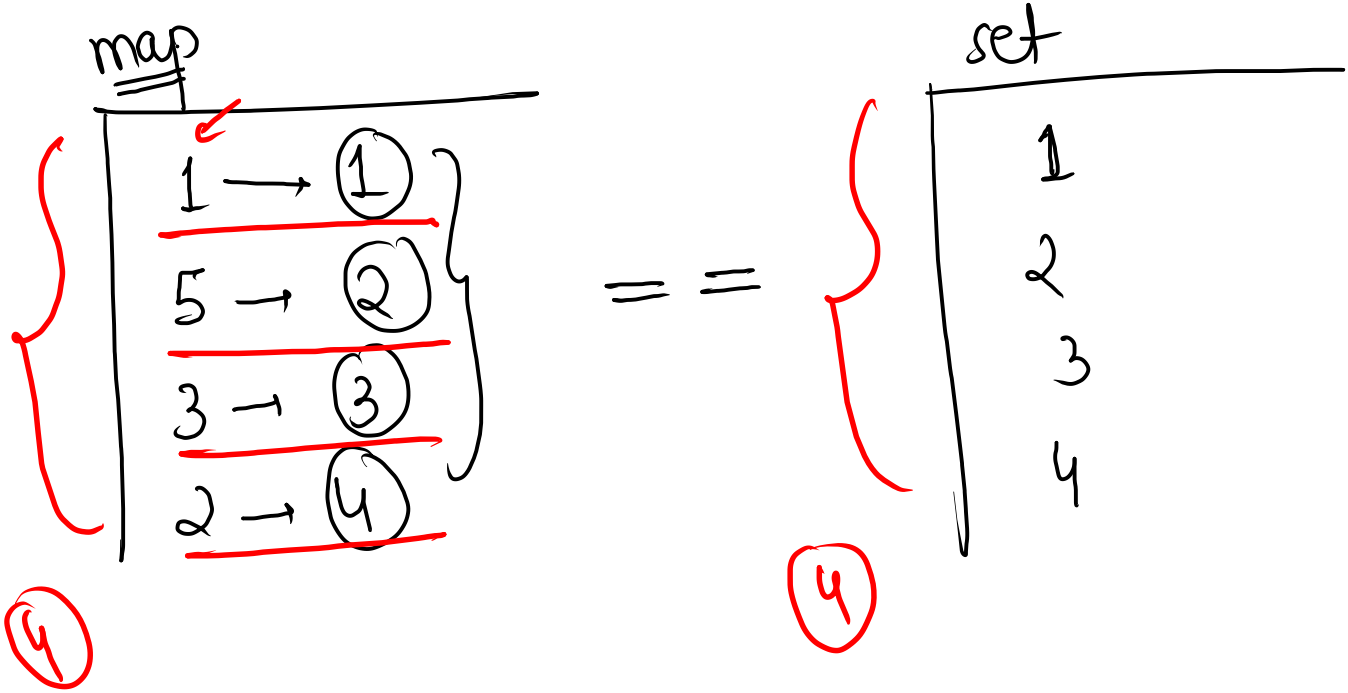
arr = [1, 1, 1, 1, 2, 2, 3, 3, 3, 3]

Dry run



False

arr = [1, 5, 5, 3, 3, 3, 2, 2, 2, 2]



Unique Number of Occurrences

Code

```
public static void uniqueNumber(int[] arr, int n) {
    HashMap<Integer, Integer> map = new HashMap<>();
    for (int i = 0; i < n; i++) {
        if ( map.containsKey( arr[i] ) ) {
            map.put( arr[i], map.get(arr[i]) + 1 );
        } else {
            map.put(arr[i], 1);
        }
    }

    HashSet<Integer> set = new HashSet<>(map.values());
    //  for (Integer i : map.values()) {
    //      set.add(i);
    //  }

    if (set.size() == map.size()) System.out.println(true);
    else System.out.println(false);
}
```

Longest Substring Without Repeating Characters 6

str = "ababc"

substring

a ← 1
ab ← 2
aba
abab
ababc

b ← 1
ba ← 2
bab
babc

a ← 1
ab ← 2
abc ← 3

b ← 1
bc ← 2

c ← 1

approch → $O(N^2)$ → TLE

Str = "abcabcde abc"

left ↑ right ↑

ans = ~~3~~ ✗ 5

range :- left to right

hashset

a
b
c
d
e

right

psudo code

1) if char at right is not present in set
   ↳ right++ and add char in set

2)
   ↳ left++ and remove left char from set

if no repetation then right++

else left++

T.C = O(N)

operations = 2 x N

Code

```java
public static int longestSubstringWithoutDuplication(String str) {
    HashSet<Character> set = new HashSet<>();
    int left = 0;
    int right = 0;
    int ans = -1;
    while (right < str.length()) {
        if (set.contains(str.charAt(right)) ) {
            set.remove( str.charAt(left) );
            left++;
        } else {
            set.add(str.charAt(right));
            right++;
            ans = Math.max( ans, right - left );
        }
    }
    return ans;
}
```

duplication

faith :- in b/w left and right there will no duplication ever

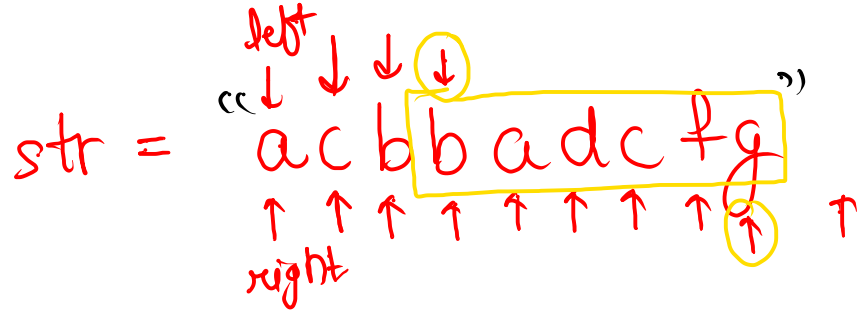
Note:- ↳ right will be ↑ing until we find duplication
right will add char in set

↳ left will be ↑ing until we remove duplication
left will remove char from set

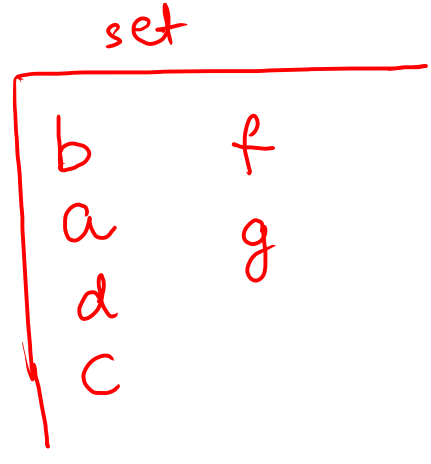
```java
public static int longestSubstringWithoutDuplication(String str) {
    HashSet<Character> set = new HashSet<>();
    int left = 0;
    int right = 0;
    int ans = -1;
    while (right < str.length()) {
        if (set.contains(str.charAt(right)) ) {
            set.remove( str.charAt(left) );
            left++;
        } else {
            set.add(str.charAt(right));
            right++;
            ans = Math.max( ans, right - left );
        }
    }
    return ans;
}
```

a

b



ans = ~~X~~ ~~X~~ ~~2~~ ~~3~~ ~~4~~ ~~5~~ 6



Length of Longest Palindrome

str = "abccccddeeee"

e

str = cbbaeabbc

palindrome = a

= a, b

= a, b, c

= cc, cbc, cac

= ccc, cbc, cac

e, a, b

edccccde → even, odd

edccecde

9

logic

↳ keep checking occurence

if single, ignore

if couple, use in answer

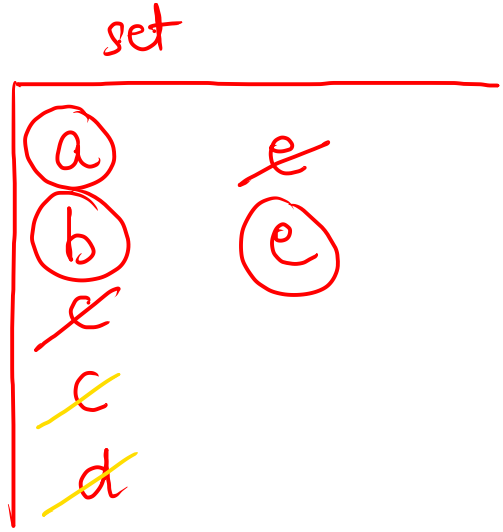
dry run

str = "ab cccc dd eec"

ans = e d c c c c d e

a, b, e

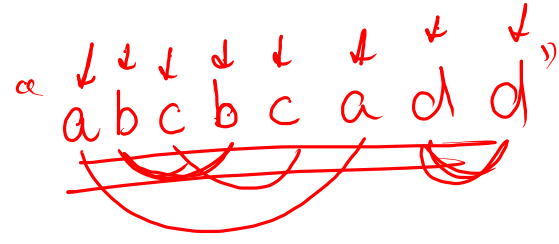
or

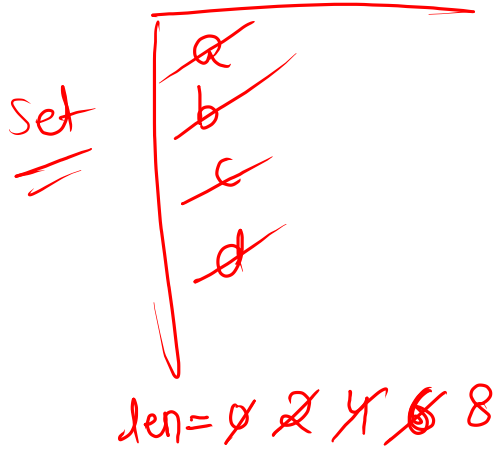


Note:-

Hashset will be containing those char which are not used in our ans till now.

```java
public static int createPalindrome(String str) {
    int n = str.length();
    int len = 0;
    HashSet<Character> set = new HashSet<>();
    for (int i = 0; i < n; i++) {
        char ch = str.charAt(i);
        if ( set.contains(ch) ) {
            set.remove(ch);
            len += 2;
        } else {
            set.add(ch);
        }
    }
    if (!set.isEmpty()) {
        len++;
    }
    return len;
}
```





if no repetation then right++

else left++

$T.C = \underline{\underline{O(N)}}$

operations = $2 \times N$