(19) 2 Sum problem:

arr[] = [2, (6), 5, (8), 11] target = 14

- 6 + 8 = 14

- Brute force approach
  - two loops
  - For every element check whether the elements remaining part to make sum is present or not
    ex: ~~6 8~~

Using Hashing: (Better approach)
- Save all the elements in hashmap
- ~~6 8~~ 14 - 6 = (8) map

Best approach:
Using two pointers
- sort

[2, 6, 5, 8, 11]

~~2 FTP~~

[2 5 6 8 11]

2 + 11 = 13 < 14

5 + 11 = 16 > 14

2 (5) 6 (8) 11

8 + 5 = (14) ✓

(20) 4 Sum problem:

Brute force approch:
four loops

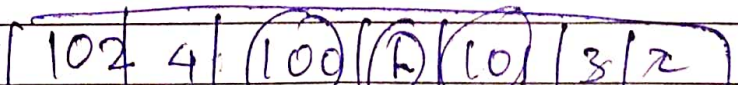
Using Hashmaps:
Using 3 loops + 1 Hashing

Best approach:
- Using two pointer approach
- Sort
- two loops + two pointer approach

(21) Longest consecutive sequence

Brute force:
- Sort array
- Find longest consecutive sequence

Better approach:
Using hashset



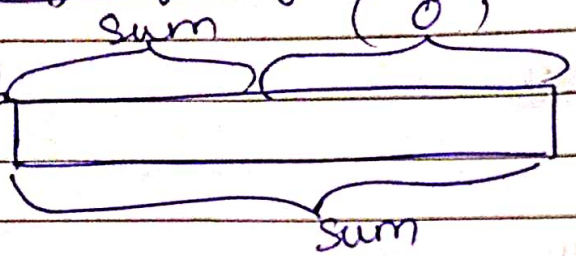
101? ✓ 99? ✗ 100? ✓ 1? ✓
find starting point

max
(1, 2, 3, 4) (4)
(100, 101, 102) (3) ⇒ (4)

⇒ Use hashset

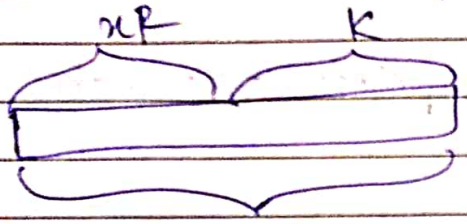
(22) Length of Longest Subarray with zero sum:

Better approach



Store hashmap with {sum_val, index}.
If we get sum=0, max = i+1;

(23) ~~Count the number of subarray with given XOR K~~



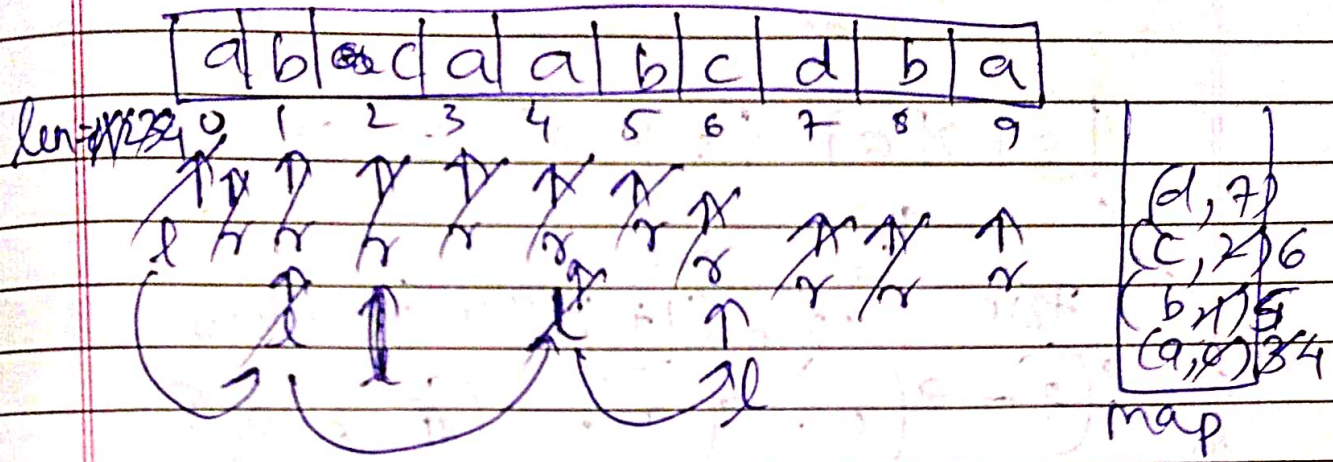
$XR \wedge K = XOR$

$XR \wedge XOR = K$

Store hashmap with {~~sum~~ XOR_val, index}
if we get XOR_val to be K, max = i+1;

(24) Longest Substring without repeating character:

~~Brute force~~ approach:



Code:

```
~~Map<char, int>~~ vector<int> mpp(256, -1);
~~int~~ int l=0, r=0;
while(r<n)
    if(mp[s[r]] != -1) left =
        max(mp[s[right]]+1, left);

    mpp[s[right]] = right;
    len = max(len, right - left + 1);
    right++;
}
return len;
}
```