3 Sum

nums = [-1, 0, 1, 2, -1, -4]

we need to find
a, b, c where
$a+b+c=0$

$-1+0+1=0$ [triplet]
$2+-1+-1=0$

we need to find unique triplets
$i!=j!=k$
should not be the same
number must be at three different indexes

(1) Brute force
find all possible combinations

```
for (i -0 to n) {
    for (j= i+1 to n) {
        for (k=j+1 to n) {
            if (a+b+c==0) {
                as store
                ↓
                will store in set
                not in unordered because
                we want it to be
                sorted even that if
                there are duplicates
                we know all the
                permutations given it'll
                be the same after being
                sorted
```

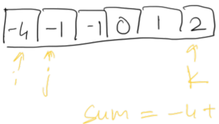
TC: $O(n^3 \times \log(\text{unique triplets}))$
↓ because we use set

(2) Better (Hashing)
$a+b+c=0$
$a=-(b+c)$ = target

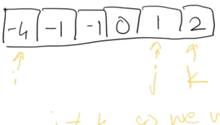
```
for (i=0 to n) {
    tar = -nums[i]
    set<int> s;
    for (j=i+1 to n){
        tofind = tar-nus[j]
        if (s.find (tofind) !=s.end()){
            found
        }
        s.insert[nums[j]
    }
}
```

TC: $O(n^2 \times \log(\text{unique Triplets}))$

(3) Optimized (2 pointer approach)
(1) nums => sort
[-4, -1, -1, 0, 1, 2]



Sum = $-4+(-1)+2=-3$ ↑
we need to increase sum
so j++



$j \neq k$ so we need to increase i now

```
for (i=0 to n) {
    if(i>0) && nums[i] == nums[i-1]) continue;
                                ↑ to avoid repeated vals in sorted array
    j=i+1, k=n-1
    while (j<k) {
        sum = nums[i]+ nus[j]+nums[k]
        if (sum < 0)
            j++
        else (sum>0)
            k--
        else {
            N[i], N[j]N[k] -> ans
            while (j<k && nums[j] == nums[j-1]
                j++      // to avoid repeated values
        }
    }
}
```

TC: $O(n\log n + n^2)$