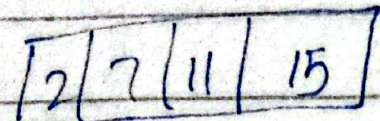
Pair Sum

Return pair in sorted array with target sum.



target = 9

2 + 7 = 9 → sum = target

↓

This is called Pair Sum.

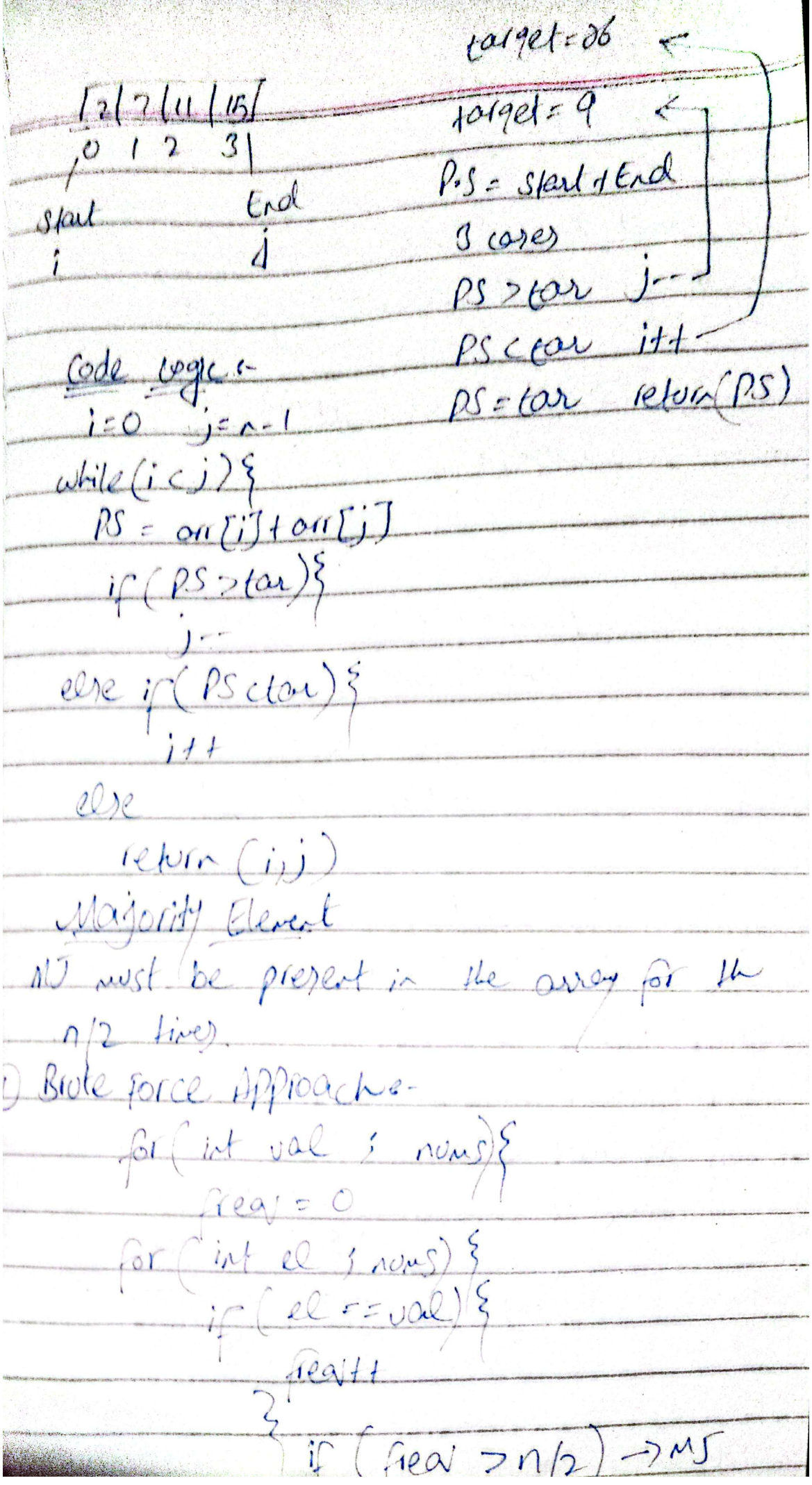
Brute force Approach

* we will be getting all the unique pairs.

```
for(int i=0; i<n; i++){
    for(j=i+1; j<n; j++){
        (i, j)
    if(arr[i] + arr[j] == target){
        vector.PB(i)
        vector.PB(j)
        return vector
    }
```

More optimized approach as the array is already in sorted form.

We will use 2 pointers approach here.



Code logic :-

```
i=0  j=n-1
while(i<j){
  PS = arr[i]+arr[j]
  if(PS>tar){
      j--
  else if(PS<tar){
      i++
  else
      return (i,j)
```

Majority Element

NJ must be present in the array for the n/2 times.

1) Brute force Approach :-

```
for(int val : nums){
    freq = 0
  for(int el : nums){
      if(el == val){
          freq++
      }
      if(freq > n/2) → MS
```

Slightly Optimized

* Sort first
* As the array is sorted, so we will count the existence of contiguous numbers to check the Majority element

```
for(i=1; i<n; i++){
  if(num[i] == num[i-1]){
      freq++
  else
  freq=1;
  ans=nums[i]
  if (freq > n/2){
      return ans.
  }
```

Most Optimized :-

Moore's voting Algo

MJ = n/2

if Same element then freq++

If different element then freq--

```
for(int i=0; i<n; i++){ freq=0
  if(freq==0){          ans=0.
    ans = nums[i]
  if (ans == num[i]){
      freq++
  else freq--}
```

Moore's algo variation

If there is no majority element, then return -1

ans -> freq > n/2

yes -> ans

NO -> -1