Two pointer Technique

1. Sort the array 2. Take l and r as two pointers to the left and right of the array. Find sum of l and r. 3. If sum > 0 then decrement r. 4. If sum falls below 0 then increment l. 5. Adjust l and r based on sum result and find pairComplexity: Time --> O(nlogn) [nlogn(sort)+n(scan using two pointer technique)] Space --> O(1)

In [11]:
```
def pairSumCloseToZero(arr):
    arr = sorted(arr)
    l = 0
    r = len(arr)-1
    current_closest = arr[l]+arr[r]
    current_sum = arr[l]+arr[r]
    current_min_index = l
    current_max_index = r
    while(l < r):
        if current_sum > 0:
            r -= 1
        else:
            l +=1
        if (l<r):
            current_sum = arr[l]+arr[r]
            if abs(current_sum) < abs(current_closest):
                current_closest = current_sum
                current_min_index = l
                current_max_index = r
    print("pair {0} and {1} have closest to zero {2}".format(arr[current_min_index],a
```

In [12]:
```
arr = [2,10,45,-54,53]
pairSumCloseToZero(arr)
#ans pair -54 and 53 have closest to zero -1
```

```
pair -54 and 53 have closest to zero -1
```

In [14]:
```
arr = [2,10,45,-54,54]
pairSumCloseToZero(arr)
#ans pair -54 and 54 have closest to zero 0
```

```
pair -54 and 54 have closest to zero 0
```

In [15]:
```
arr = [2,10,45,-54]
pairSumCloseToZero(arr)
#ans pair -54 and 45 have closest to zero -9
```

```
pair -54 and 45 have closest to zero -9
```

In [ ]: