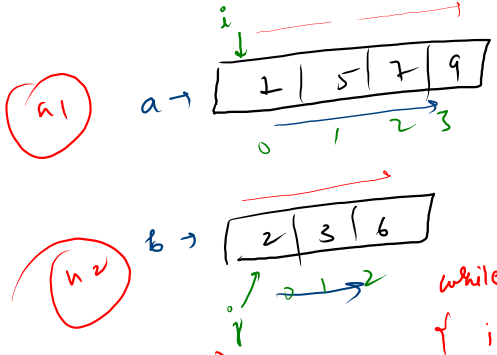
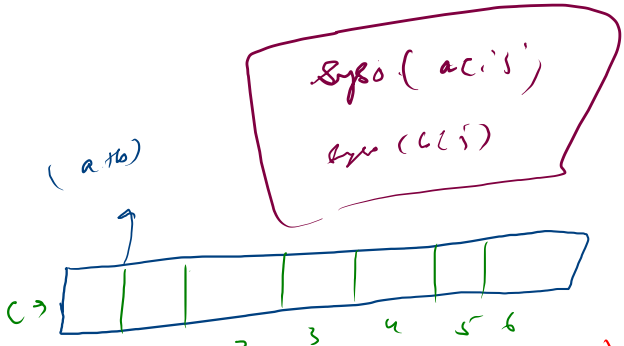
Mergesort

Merge two sorted array





O(max(n1, n2))

↓

linear T.C.

```
while(i<n1 && j<n2)
{
  if(a[i] > b[j])
      c[k] = b[j]; k++; j++
  else
      c[k] = a[i]; i++, k++;
}
```

```
while(i<n1)
{
  c[k] = a[i];
    i++
    k++;
}
```

```
while(j<n2)
{
  c[k] = b[j];
  j++
  k++;
}
```

Merge sort → Divide and conquer

↳ ascending order

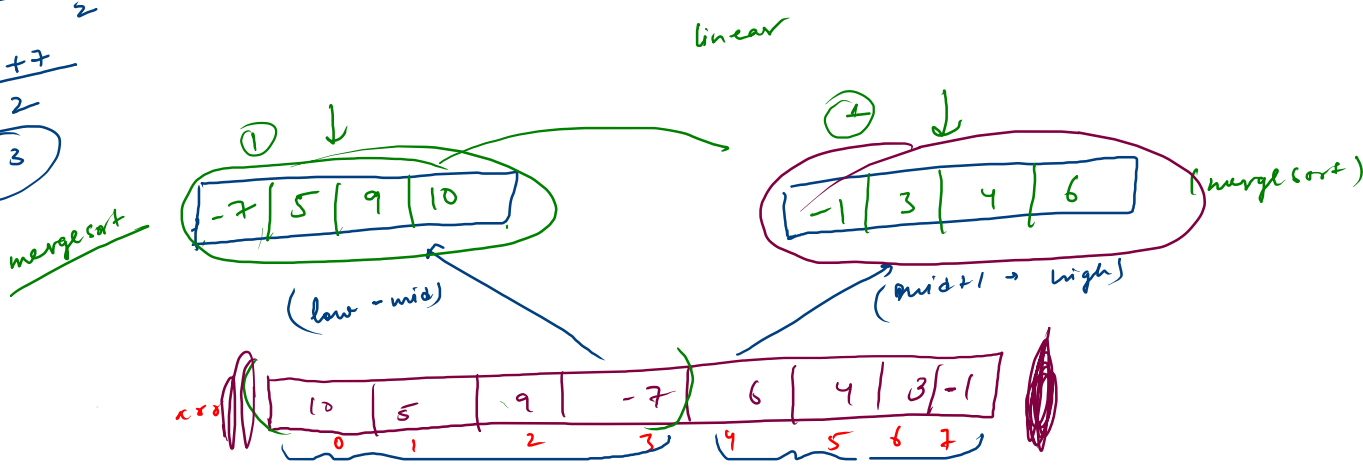
low = 0

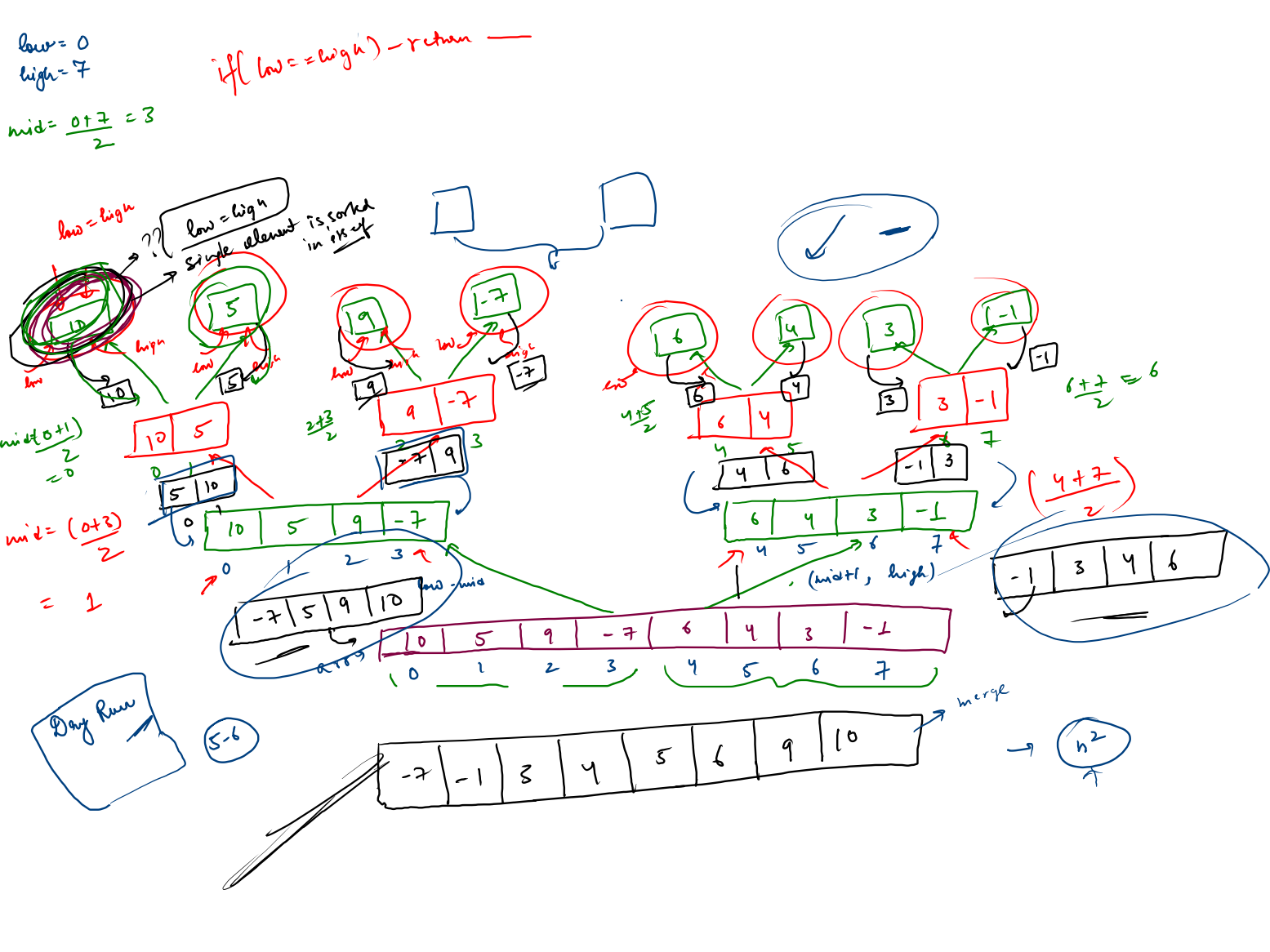
high = arr.length − 1 = 7

① mid = $\frac{low + high}{2}$

$= \frac{0+7}{2}$

$= 3$




low = 0
high = 7
if(low == high) - return
mid = 0+7 / 2 = 3
low = high
??
low = high
single element
is sorted in itself
10
5
9
-7
6
4
3
-1
mid = (0+1) / 2 = 0
mid = (0+3) / 2 = 1
2+3 / 2
4+5 / 2
6+7 / 2 = 6
(4+7) / 2
low - mid
(mid+1, high)
-7 5 9 10
-1 3 4 6
-7 -1 3 4 5 6 9 10
merge
n^2
Day Runs
5-6

```java
public static int [] mergeTwoSortedArray(int a[],int b[]){
    int n1 = a.length;
    int n2 = b.length;

    int res[] = new int[n1+n2];

    int i=0,j=0,k=0;
    while(i<n1 && j<n2){
        if(a[i] > b[j]){
            res[k] = b[j];
            k++;
            j++;
        }else{
            res[k] = a[i];
            i++;
            k++;
        }
    }

    while(i<n1){
        res[k] = a[i];
        i++;
        k++;
    }

    while(j<n2){
        res[k] = b[j];
        j++;
        k++;
    }

    return res;
}
```


```java
public static int [] mergeSort(int arr[],int low, int high){
    if(low == high){
        int base[] = new int[1];
        base[0] = arr[low];
        return base;
    }

    int mid = (low+high)/2;

    int a[] = mergeSort(arr,low,mid);
    int b[] = mergeSort(arr,mid+1,high);

    return mergeTwoSortedArray(a,b);
}
```

S.C. = O(n)

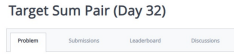
T.C. = O(n log n) < O(n²)

$T(n) = T(n/2) + T(n/2) + n + K$

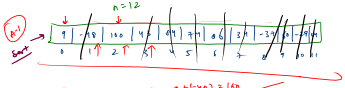
$T(n) = 2T(n/2) + n + K$

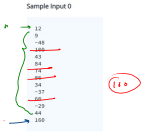
T.C. ⇒ O(n log n)

Target Sum Pair (Day 32)



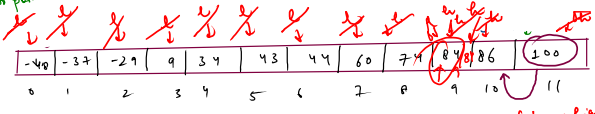
1. You are given an array(arr) of distinct integers and a target.
2. You have to print all the pairs having their sum equal to the target.





Arrays.sort(arr) → ascending

two pointers = low, high



→ sort (ascending)

Two pointer approach

low = 0
high = 11

sum = -48 + 100 = 52
= a+b

(a+b) < target

-48+100 = 52

low ++

(a+b) > target

86+100 > 160

186 > 160

high - -

100+60 = 160

(a+b == target)

print
low ++
high - -

60 100
74 86

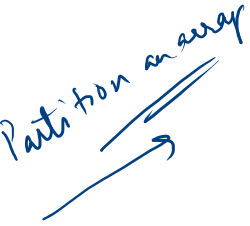
low ++
high - -
low - -
high ++

while(low < high)

-48+100 = 52 < 160
-37+100 = 63 < 160
-29+100 = 71 < 160
9+100 = 109 < 160
34+100 = 134 < 160
43+100 = 143 < 160
44+100 = 144 < 160
60+100 = 160 = 160
74+86 = 160 = 160
84+84 ← valid pair

here (nlogn + n)
↓ sorting, ↓ traverse

T.C. = O(nlogn)




```java
public class Solution {
    public static void targetSum(int arr[],int n, int target){
        Arrays.sort(arr);
        int low = 0, high = n-1;
        while(low<high){
            int sum = arr[low] + arr[high];

            if(sum == target){
                System.out.println(arr[low] + " " + arr[high]);
                low++;
                high--;
            }else if(sum > target){
                high--;
            }else{
                low++;
            }
        }
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT.
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr[] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }

        int target = scn.nextInt();

        targetSum(arr,n, target);
    }
}
```

Partition An Array (Day 32)

Problem | Submissions | Leaderboard | Discussions

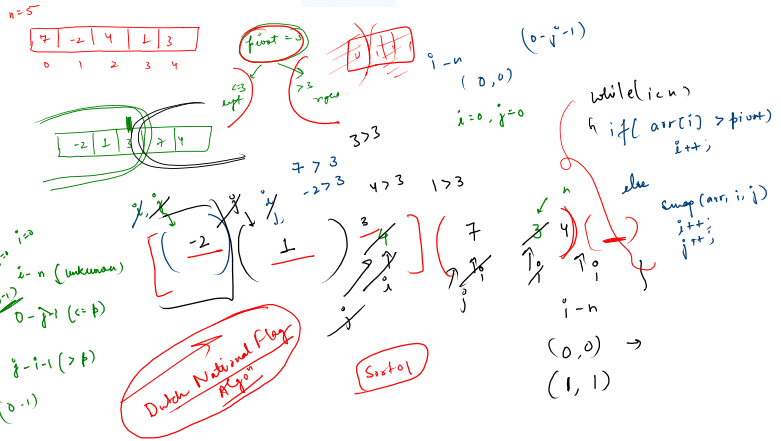
1. You are given an array(arr) of integers and a pivot.
2. You have to re-arrange the given array in such a way that all elements smaller or equal to pivot lie on the left side of pivot and all elements greater than pivot lie on its right side.
3. You have to achieve this in linear time.

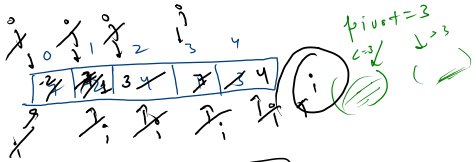
Sample Input 0

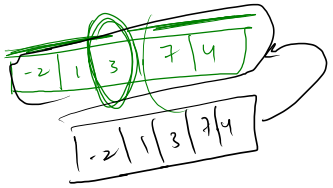
```
5
7 -2 4 1 3
3
```

Sample Output 0

```
-2 1 3 7 4
```







```
7 > 3
-2 > 3
4 > 3
1 > 3
3 > 3
```

```
i=0, j=0
while (i<n) {
  if( arr[i] > pivot) i++;
  else {
      swap(arr, i, j)
      i++;
      j++;
  }
}
```

```java
public class Solution {
    public static void swap(int arr[],int i,int j){
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int arr [] = new int[n];
        for(int i=0;i<n;i++){
            arr[i] = scn.nextInt();
        }
        int pivot = scn.nextInt();

        int i =0,j=0;
        while(i<n){
            if(arr[i] > pivot) i++;
            else{
                swap(arr,i,j);
                i++;
                j++;
            }
        }

        for(i =0;i<n;i++){
            System.out.print(arr[i] + " ");
        }

    }
}
```