3 Sum

$n = 6$

arr	-2	0	2	4	-2	-8

equation :-

$$arr[i] + arr[j] + arr[k] == 0$$

answer

-2 , -2 , 4

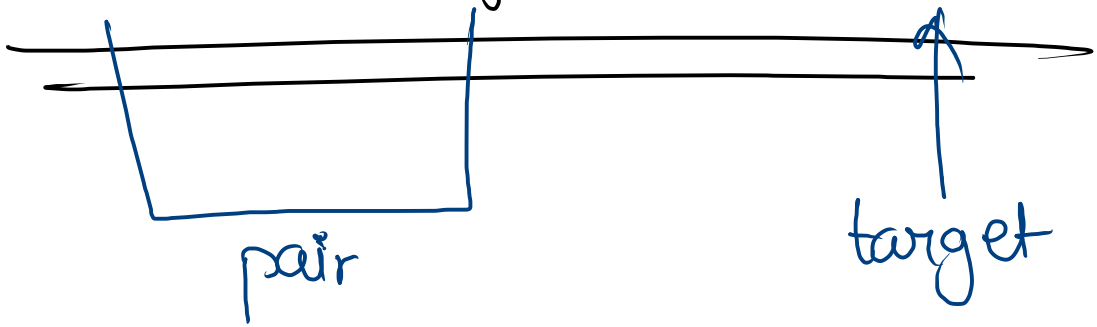
-2 , 0 , 2

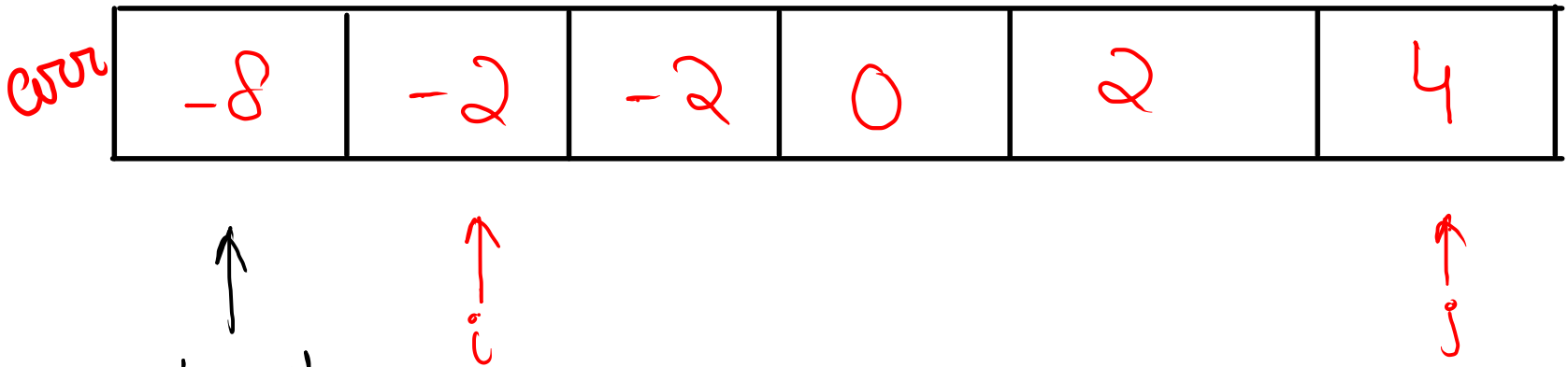
purpose of sorting } → so that we skip all the unessary cases

arr	-8	-2	-2	0	2	4

exp:- arr[i] + arr[j] + arr[k] == 0

gmp [arr[i] + arr[j] == -1 * arr[k]





-2 , -2, 4

-2 , 0, 2

-2, 0, 2

duplicate

psudo code of target sum

- pointer i = 0, j = n-1
- iterate until i < j
- if sum == target
  - print i, j
  - i++, j--
- elseif sum > target ; j--
- else sum < target ; i++

psudo code

- Sort array
- iterate from start to end for target (-1*target)
  - same code for target sum
    - pointer i = 0, j = n-1
    - iterate until i < j
      - if sum == target
        print i, j
        i++, j--
      - else if sum > target ; j--
      - else sum < target ; i++

target = -(-2) = 2

sum = -2 + 4 = 2

sum = 0 + 2 = 2

target = -(-2) = 2

sum = 0 + 4 = 4

sum = 0 + 2 = 2

Code

```
public static void threeSum(int[] arr, int n) {
    Arrays.sort(arr);
    for (int k = 0; k < n; k++) {
        int target = -1 * arr[k];
        int i = k + 1;
        int j = n - 1;
        while ( i < j ) {
            int sum = arr[i] + arr[j];
            if ( sum == target ) {
                System.out.println(arr[k] + " " + arr[i] + " " + arr[j]);
                i++;
                j--;
            } else if (sum < target) {
                i++;
            } else {
                j--;
            }
        }

        while ( k + 1 < n && arr[k] == arr[k + 1] ) {
            k++;
        }
    }
}
```

→ N

→ N

3 sum

target sum

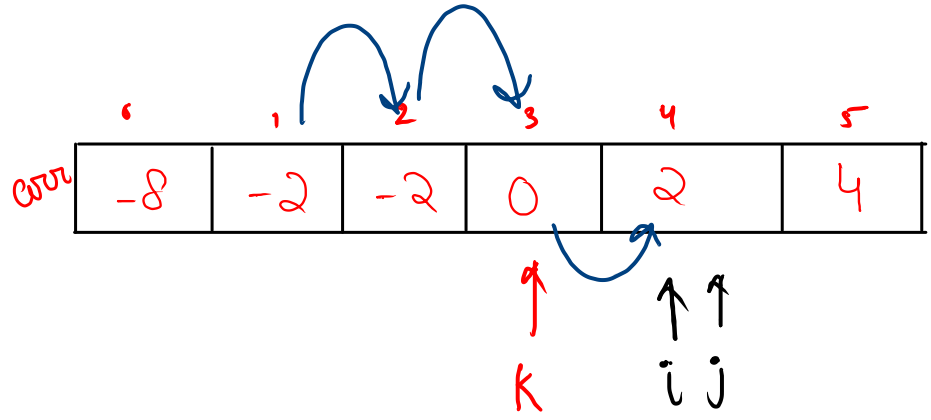
T. C

$O(N \log N + N^2)$

$\simeq O(N^2)$

```java
public static void threeSum(int[] arr, int n) {
    Arrays.sort(arr);
    for (int k = 0; k < n; k++) {
        int target = -1 * arr[k];
        int i = k + 1;
        int j = n - 1;
        while ( i < j ) {
            int sum = arr[i] + arr[j];
            if ( sum == target ) {
                System.out.println(arr[k] + " " + arr[i] + " " + arr[j]);
                i++;
                j--;
            } else if (sum < target) {
                i++;
            } else {
                j--;
            }
        }

        while ( k + 1 < n && arr[k] == arr[k + 1] ) {
            k++;
        }
    }
}
```



target = 8 , sum = -2+4 = 2

sum = -2+4 = 2

sum = 0+4 = 4

sum = 2+4 = 6

target = 2 , sum = -2+4 = 2

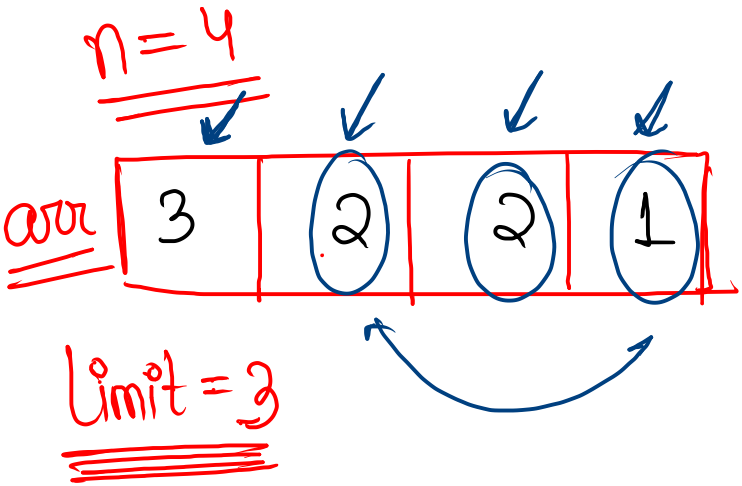
Sum = 0+2 = 2

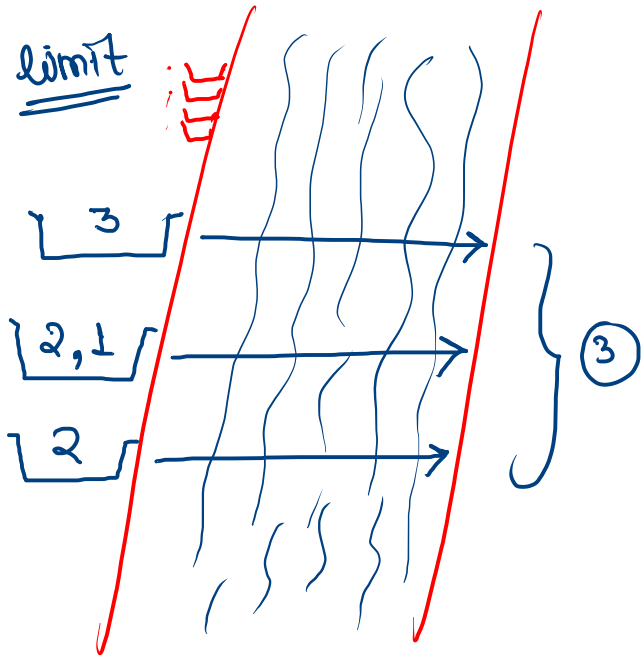
target = 0, sum = 2+4 = 6

Ans -2, -2, 4

-2, 0, 2

Count boat





Condition

↳ weight of persons can't be more than limit

↳ we can carry atmost 2 persons

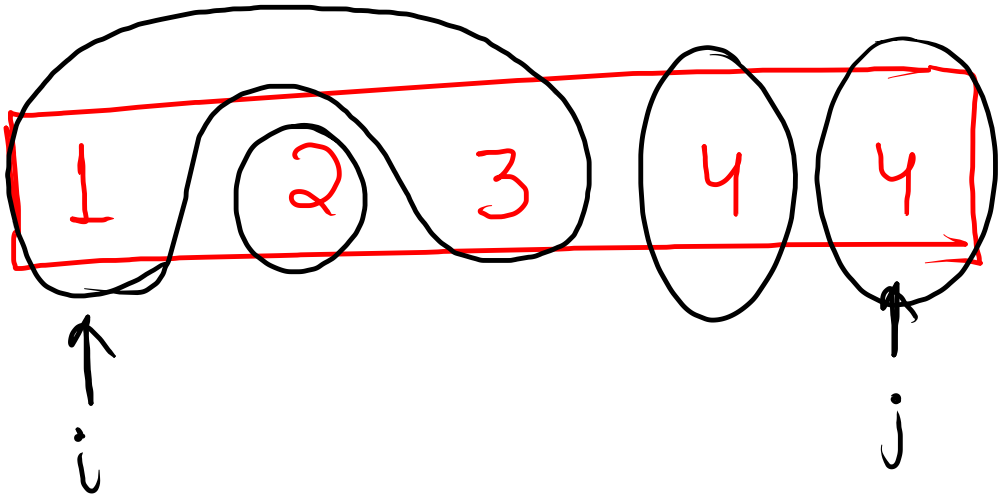
psudo code

count = 0

- Sort
  - i = 0, j = n-1 ;
  - while (i <= j)
    - if sum > limit
      - j--j
    - else
      - i++, j--j
    - Count++j

limit = 4

C = 0 1 2 3 4



2, 2

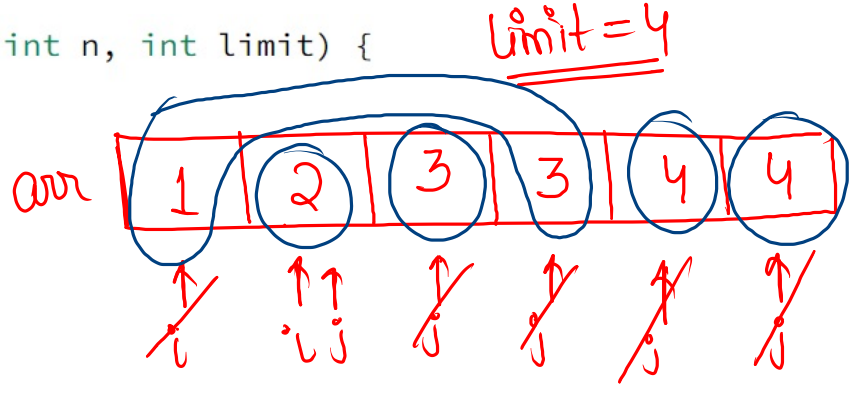
limit 4

Code

```java
public static int countBoats(int[] arr, int n, int limit) {
    Arrays.sort(arr);
    int i = 0;
    int j = n - 1;
    int count = 0;
    while (i <= j) {
        int sum = arr[i] + arr[j];
        if ( sum > limit ) {
            j--;
        } else {
            i++;
            j--;
        }
        count++;
    }
    return count;
}
```

a

b



Sum = 1+4 = 5

Sum = 1+4 = 5

Sum = 1+3 = 4

Sum = 2+3 = 5

Sum = 2+2 = 4

Count = 0 1 2 3 4 5

⇒ Prefix Sum

↳ sum of all the previous elements including itself

arr

1	3	-2	4	0	7

prefix sum array

1	4	2	6	6	13

⇒ Suffix Sum : sum of all the future elements including itself

Ques Print prefix sum & Suffix sum

arr

3	-2	1	0	5	7

prefix sum array — pre

3	1	2	2	7	14

$$pre[i] = arr[i] + pre[i-1]$$

Code

```
public static void main(String[] args) {
    int[] arr = { 3, -2, 1, 0, 5, 7};
    int n = arr.length;
    int[] pre = new int[n];

    pre[0] = arr[0];
    for (int i = 1; i < n; i++) {
        pre[i] = arr[i] + pre[i - 1];
    }

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

Greatest Till Me

n = 8

arr

3	1	5	2	7	1	3	8

prefix max (pre)

3	3	5	5	7	7	7	8

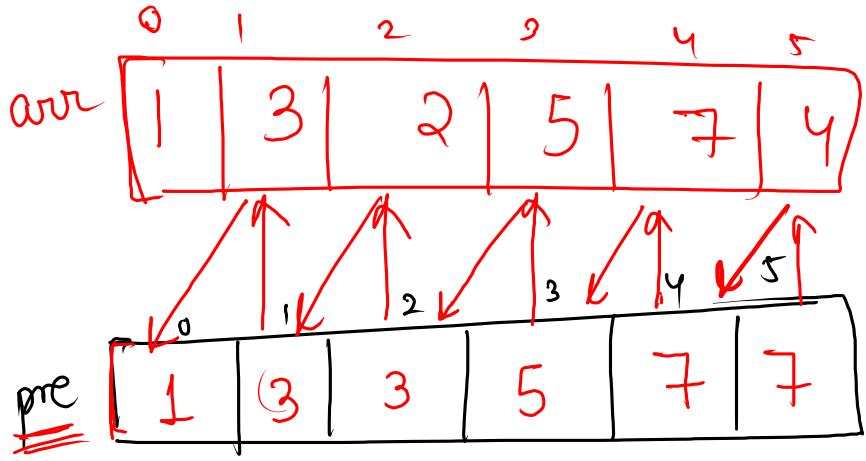
exp:-

pre[i] = Math.max(arr[i], pre[i-1])

```java
public static void main(String[] args) {
    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[] ans = greatestTillMe(arr, n);
    for (int i = 0; i < n; i++) {
        System.out.println(ans[i]);
    }
}

public static int[] greatestTillMe(int[] arr, int n) {
    int[] pre = new int[n];
    pre[0] = arr[0];
    for (int i = 1; i < n; i++) {
        pre[i] = Math.max( arr[i], pre[i - 1] );
    }
    return pre;
}
```



i=1, pre[1] = max(3, 1) = 3

i=2, pre[2] = max(2, 3) = 3

i=3, pre[3] = max(5, 3) = 5

i=4, pre[4] = max(7, 5) = 7

i=5 pre[5] = max(4, 7) = 7