0	1	2	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9	10

Days → 5

(0+n-1)/2

maxwt. = 10 → left

totalwt. = 55 → right

① mid = (10+55)/2 = 32 (capacity)

Day 1 : (1+2+3+4+5+6+7) = 28

Day 2 : (8+9+10) = 27

Update right = mid = 32

② mid = (10+32)/2 = 21

Day 1 : (1+2+3+4+5+6) = 21

Day 2 : (7+8) = 15

Day 3 : (9+10) = 19

③ right = mid = 21

mid = (10+21)/2 = 15

Day 1 : (1+2+3+4+5) = 15

Day 2 : (6+7) = 13

Day 3 : 8 → 8

Day 4 : 9 = 9

Day 5 : 10 = 10

④ mid = (10 + 15) / 2 = 12

Day 1 : (1 + 2 + 3 + 4) = 10

Day 2 : (5 + 6) = 11

Day 3 : 7 = 7

Day 4 : 8 = 8

Day 5 : 9 = 9

Day 6 : 10 = 10 (invalid)

Update left = mid + 1

(13 + 15) / 2 ⇒ 14

(15 + 15) / 2 = 15 Ans

→ getMax ✓

→ getSum ✓

→ days ✓

→ Capacity ✓

```java
public static int findCapacity(int weights[], int D){
    int left = getMax(weights);
    int right = getSum(weights);

    while(left < right){
        int mid = (left + right)/2;
        if(canShipInDays(weights, D, mid)){
            right = mid;
        }
        else{
            left = mid + 1;
        }
    }

    return left;
}
```

(n)

```java
public static int getMax(int weights[]){
    int max = 0;
    for(int i = 0; i < weights.length; i++){
        if(weights[i] > max){
            max = weights[i];
        }
    }
    return max;
}
```

(n)

```java
public static int getSum(int weights[]){
    int sum = 0;
    for(int i = 0; i < weights.length; i++){
        sum += weights[i];
    }
    return sum;
}
```

(n)

```java
public static boolean canShipInDays(int weights[], int D, int capacity){
    int dayNeeded = 1;
    int currentLoad = 0;

    for(int i = 0; i < weights.length; i++){
        if(currentLoad + weights[i] > capacity){
            dayNeeded++;
            currentLoad = 0;
        }
        currentLoad += weights[i];
    }
    return dayNeeded <= D;
}
```

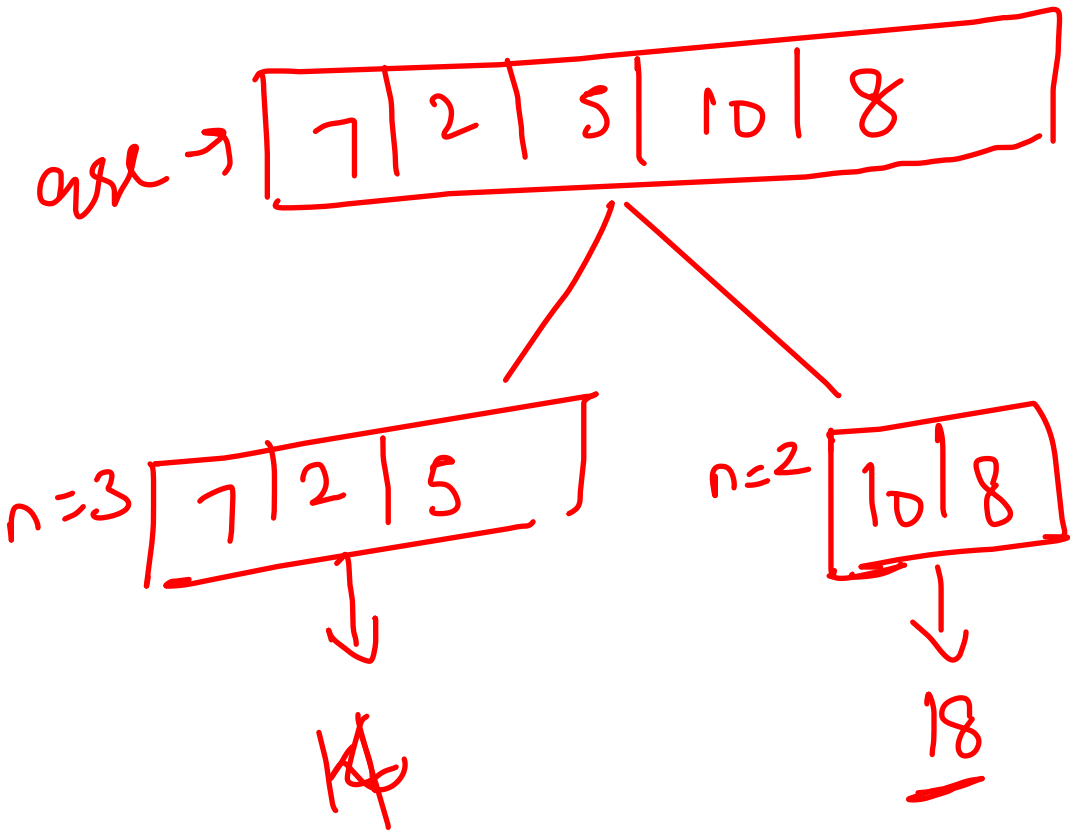
O(n)

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();
int weights[] = new int[n];
for(int i = 0; i < n; i++){
    weights[i] = s.nextInt();
}
int D = s.nextInt();

System.out.println(findCapacity(weights, D));


}
```

O(n) , O(1)



k → 2

min max sum

Subarray → min

Sum → max.

left = max element = 10

right = sum of arr = 32

① mid = (10+32)/2 = 21

currSum = 0

Subarray = 1

Add 7, currSum = 7

Add 2, 9

Add 5, 14

Add 10, 24

Subarray = 2

currentSum = 10+8 = 18

Update right = mid = 21

② mid = (10+21)/2 = 15

CS = 0

Sarr = 1

7+2+5 = 14

Sarr 2

10

Sarr 3

8 (not valid)

left = mid +1 = 16

(3) mid = (16+21)/2 = 18

ES = 0

Sarr = 1

7+2+5 = 14

CS = 0

Sarr = 2

10+8 = 18

right = mid = 18

(4) mid = (16+18)/2 = 17

2 3

Subarr 1 10 8

7+2+5 = 14

left, right = 18

```java
public static int split(int arr[], int k){
    int left = getMax(arr);
    int right = getSum(arr);

    while(left < right){
        int mid = (left + right)/2;
        if(canSplit(arr, k, mid)){
            right = mid;
        }
        else{
            left = mid + 1;
        }
    }

    return left;
}
```

```java
public static boolean canSplit(int arr[], int k, int mid){
    int currSum = 0;
    int subArray = 1;

    for(int i = 0; i < arr.length; i++){
        if(currSum + arr[i] > mid){
            subArray++;
            currSum = arr[i];

            if(subArray > k){
                return false;
            }
        }
        else{
            currSum += arr[i];
        }
    }
        return true;
}
```

```java
public static int getMax(int arr[]){
    int max = 0;
    for(int i = 0; i < arr.length; i++){
        if(arr[i] > max){
            max = arr[i];
        }
    }
    return max;
}

public static int getSum(int arr[]){
    int sum = 0;
    for(int i = 0; i < arr.length; i++){
        sum += arr[i];
    }
    return sum;
}
```

T(→ O(n)

SL → O(1)

HW_target indices

```java
public static int binarySearch(int arr[], int target){
    int left = 0;
    int right = arr.length - 1;
    int result = -1;

    while(left <= right){
        int mid =(left+right)/2;

        if(arr[mid] == target){
            result = mid;
            // continiue search in left half
            right = mid - 1;
        }
        else if(arr[mid] < target){
            left = mid +1;
        }
        else{
            right = mid -1;
        }
    }
    return result;
}
```

TC → O(n)

SC → O(1)

```java
Scanner s = new Scanner(System.in);
int n = s.nextInt();
int arr[] = new int[n];
for(int i=0; i<n; i++){
    arr[i] = s.nextInt();
}

int target = s.nextInt();

Arrays.sort(arr);

int firstIndex = binarySearch(arr, target);
if(firstIndex == -1){
    System.out.println("-1");
}
else{
    for(int i = firstIndex; i < n && arr[i] == target; i++){
        System.out.print(i + " ");
    }
}
}
```