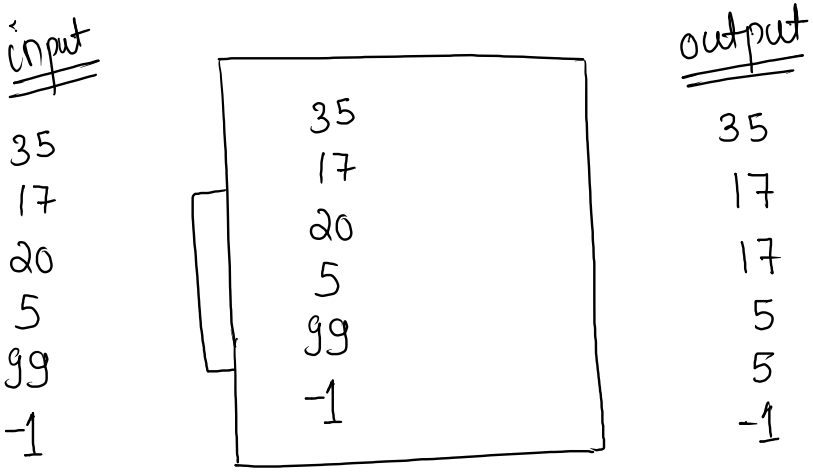
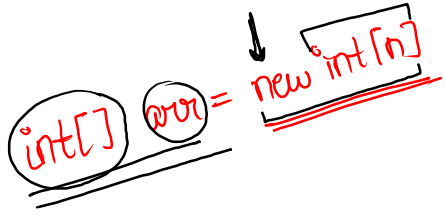
priority queue basics



Code




```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int t = scn.nextInt();
    int[] arr = new int[t];
    for (int i = 0; i < t; i++) {
        arr[i] = scn.nextInt();
    }

    basicPQ(arr);
}
public static void basicPQ(int[] arr) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for (int i = 0; i < arr.length; i++) {
        pq.add( arr[i] );
        System.out.println( pq.peek() );
    }
}
```

minimum digits

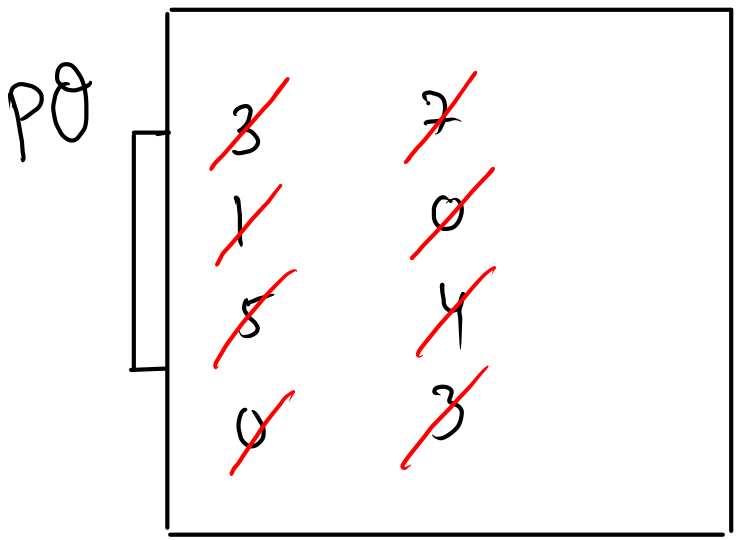
	0	1	2	3	4	5	6	7
arr	3	1	5	0	7	0	4	3

(int)

num1 = 0135

num2 = 0347

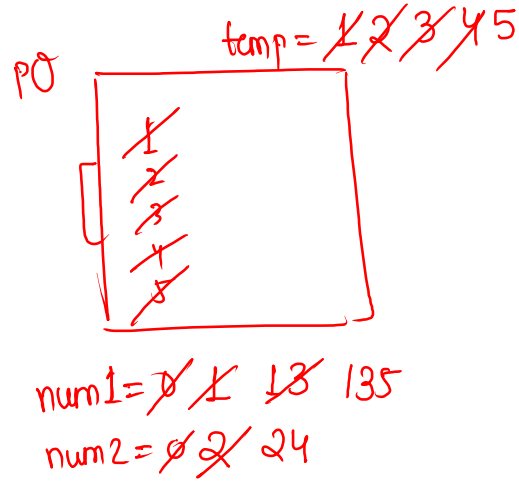
Sum = 482



Code

```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();
    }
    System.out.println(minimumDigits(arr));
}
public static long minimumDigits(int[] arr) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for (int i : arr){
        pq.add(i);
    }
    long num1 = 0;
    long num2 = 0;
    while ( pq.size() > 0 ) {
        int temp = pq.poll();
        if ( pq.size() % 2 == 0 ) {
            num1 = num1 * 10 + temp;
        } else {
            num2 = num2 * 10 + temp;
        }
    }
    return (num1 + num2);
}
```

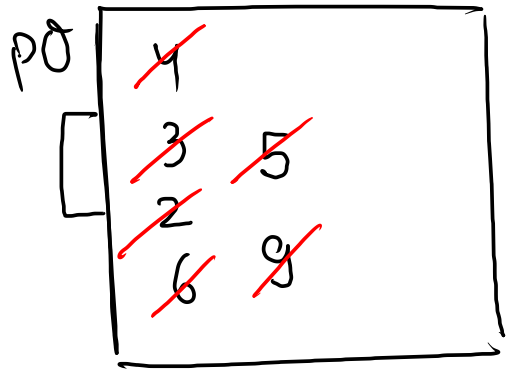
1 2 3 4 5



Minimum Cost of ropes 3

$n = 4$

4	3	2	6



rope1 = ~~2~~ ~~4~~ 6

rope2 = ~~3~~ ~~5~~ 9

length of new rope = ~~5~~ ~~9~~ 15

price = 5 + 9 + 15

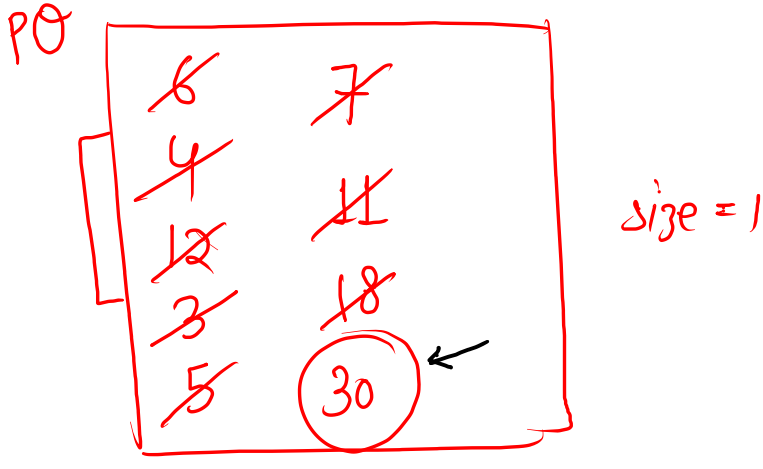
= 29

Code

```java
public static int minimumCost(int[] arr) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for (int i : arr) {
        pq.add(i);
    }

    int price = 0;
    while ( pq.size() > 1 ) {
        int rope1 = pq.poll();
        int rope2 = pq.poll();
        int newRope = rope1 + rope2;
        pq.add( newRope );
        price += rope1 + rope2;
    }
    return price;
}
```

6	4	12	3	5



price = 0 + 7 + 11 + 18 + 30

rope1 = ~~3~~ ~~5~~ ~~7~~ 12

rope2 = ~~4~~ ~~6~~ ~~11~~ 18

newRope = ~~7~~ ~~11~~ ~~18~~ 30

subtract numbers 1

arr [1, 5, 0, 3, 5, 5, 1, 3]

ans = 3

Step1 → choose a minimum non-zero no.

Step2 → remove that num from all +ve value

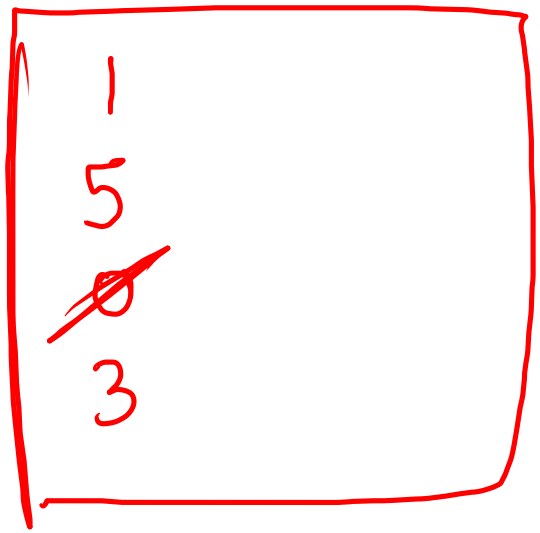
arr [1, 5, 0, 3, 5, 5, 1, 3] → (1)

arr [0, 4, 0, 2, 4, 4, 0, 2] → (2)

arr [0, 2, 0, 0, 2, 2, 0, 0] → (2)

arr [0, 0, 0, 0, 0, 0, 0, 0]

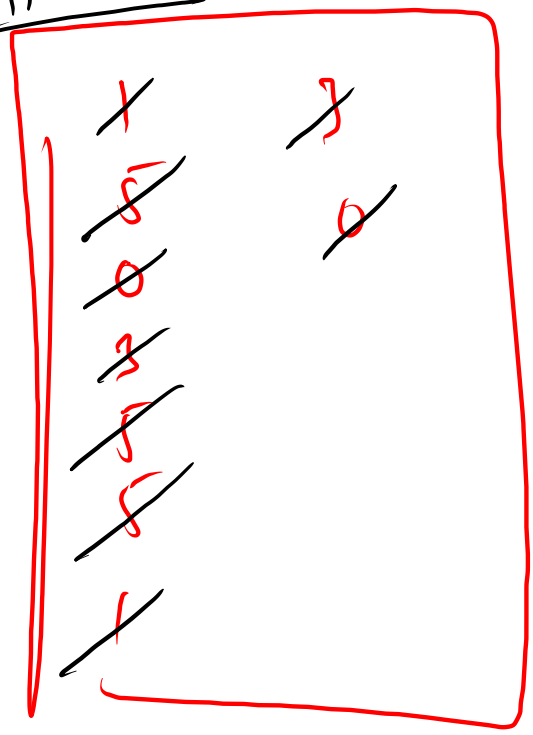
set



set.size()

without zero

approch 1



Count = 3

num = 5

Merge K sorted arrays

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    int k = scn.nextInt();
    for (int i = 0; i < k; i++) {
        int n = scn.nextInt();
        for (int j = 0; j < n; j++) {
            int num = scn.nextInt();
            pq.add( num );
        }
    }

    while ( pq.size() > 0 ) {
        System.out.print(pq.poll() + " ");
    }
}
```