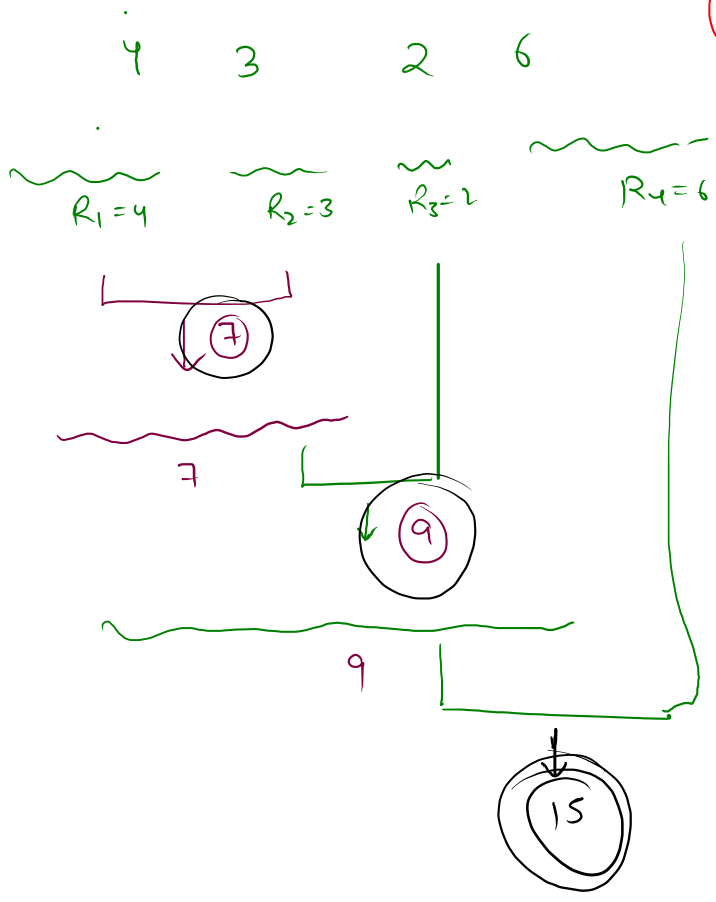
Minimum Cost of ropes 3

Sample Input 0

```
4
4 3 2 6
```

Sample Output 0

```
29
```



$(R_1+(R_2+R_3))+R_4$

5 ⇒ 29

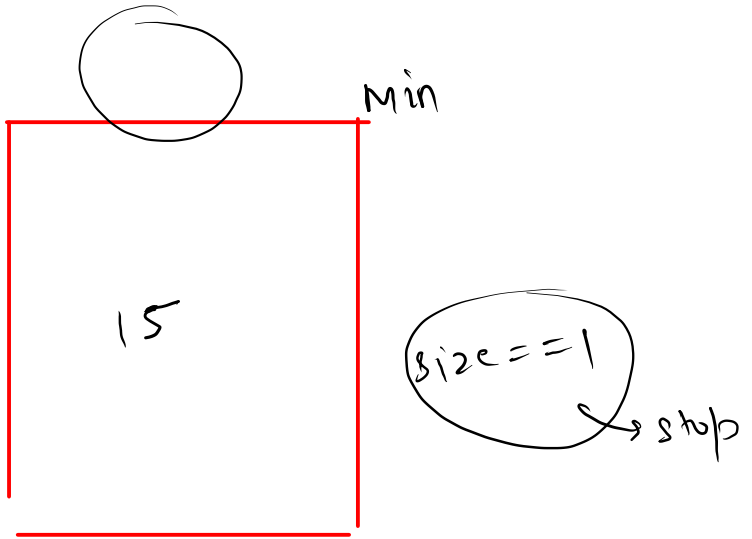
9

9+6 = 15

15
9
7
31

4 2 3 6

cost = 0 + 5 + 9 + 15

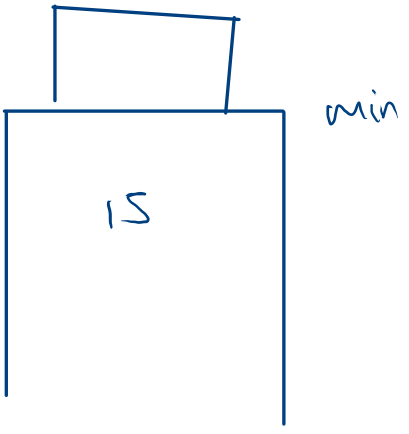


```java
import java.io.*;
import java.util.*;

public class Solution {

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

        int cost = 0;
        while(pq.size() > 1){
            int a = pq.remove();
            int b = pq.remove();
            cost += (a+b);
            pq.add(a+b);
        }
        System.out.println(cost);
    }
}
```

n=4

cost = 5 + 9 + 15

29



minimum digits

Given an array of **digits (values are from 0 to 9)**, find the **minimum possible sum of two numbers** formed from digits of the array. All digits of the given array must be used to form the two numbers.

Any combination of digits may be used to form the two numbers to be summed. Leading zeroes are permitted.

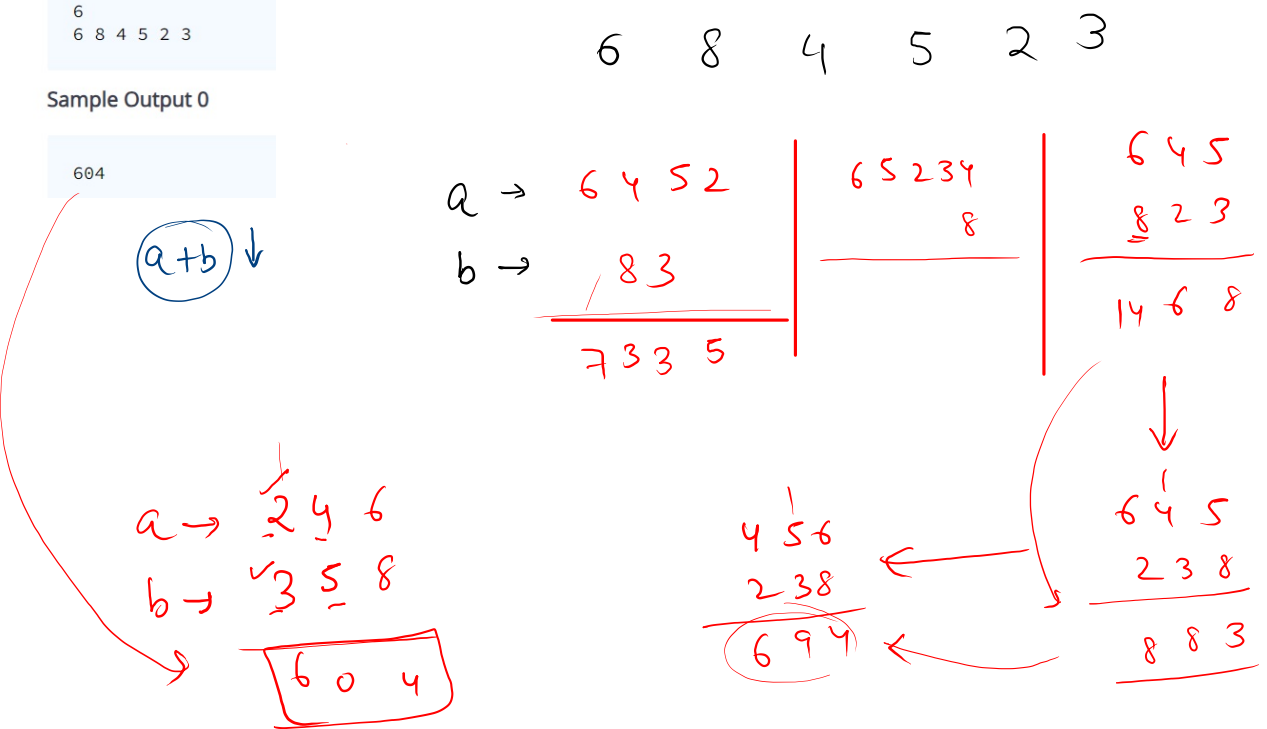
If forming two numbers is **impossible** (i.e. when n==0) then the "sum" is the value of the only number that can be formed.

Sample Input 0

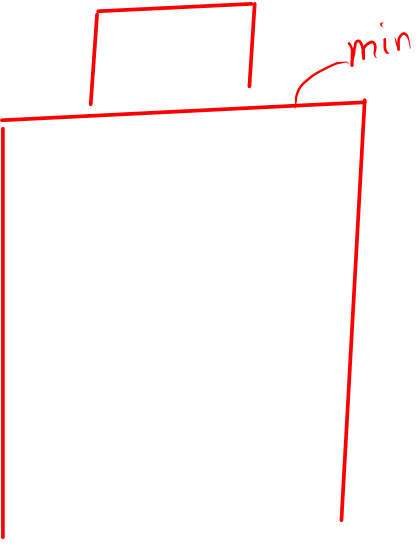
```
6
6 8 4 5 2 3
```

Sample Output 0

```
604
```



6 8 4 5 2 3



a → " 2 4 6 "

b → " 3 5 8 "

→ int

604

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);

        int n = scn.nextInt();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        for(int  i = 0; i <n; i++){
            pq.add(scn.nextInt());
        }

        String a = "";
        String b = "";
        while(pq.size() > 0){
            a += pq.remove();
            if(pq.size() > 0){
                b += pq.remove();
            }
        }
        long v1 = Long.parseLong(a);
        long v2 = Long.parseLong(b);
        System.out.println(v1 + v2);

    }
}
```

6 8 4 5 2 3

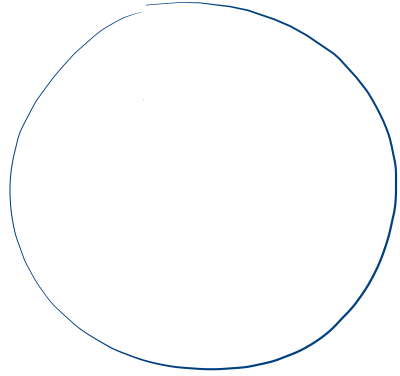
```java
public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);

        int n = scn.nextInt();

        PriorityQueue<Integer> pq = new PriorityQueue<>();
        for(int  i = 0; i <n; i++){
            pq.add(scn.nextInt());
        }


        String a = "";
        String b = "";
        boolean even = true;
        while(pq.size() > 0){
            if(even){
                a += pq.remove();
            }else{
                b += pq.remove();
            }
            even = !even;
        }
        long v1 = Long.parseLong(a);
        long v2 = Long.parseLong(b);
        System.out.println(v1 + v2);

    }
```

6 8 4 5 2 3

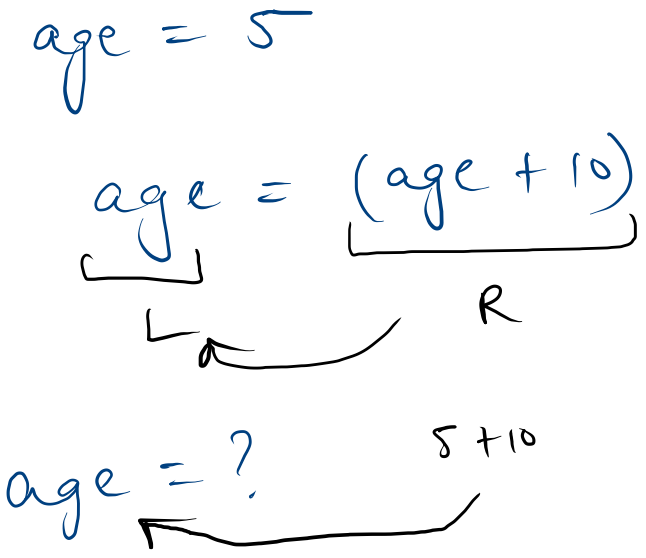
even = true ~~false true false true~~

a → "246"

b → "358"



ans = 604


age = 5
age = (age + 10)
R
age = ?
5 + 10

even = ~~true~~ false

even = (!even)

L ← R

even = !(true)

even = false

even = ~~false~~ true

even = !even

= !(false)

even = true

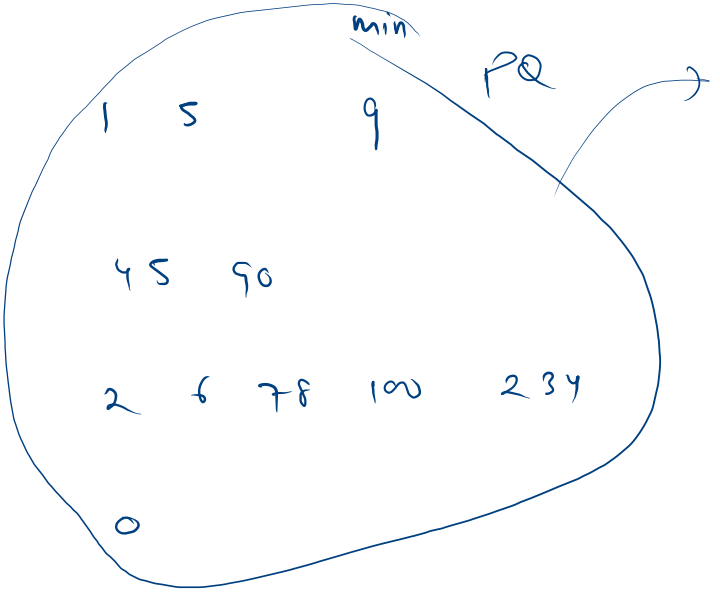
Merge K sorted arrays

Given k different arrays, which are sorted individually (in ascending order). You need to merge all the given arrays such that output array should be sorted (in ascending order).

Hint : Use ~~Heaps~~. PQ

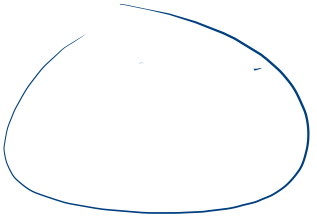
Sample Input 0

```
4
3
1 5 9
2
45 90
5
2 6 78 100 234
1
0
```



Sample Output 0

```
0 1 2 5 6 9 45 78 90 100 234
```



k = 2

s_1 = 4

3 8 13 15

s_2 = 3

2 9 10

2 3 8 9 10 13 15

```java
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        int k = scn.nextInt();
        PriorityQueue<Integer> pq = new PriorityQueue<>();
        while(k-- > 0){
            int s = scn.nextInt();
            while(s-- > 0){
                pq.add(scn.nextInt());
            }
        }
        while(pq.size()  > 0){
            System.out.print(pq.remove() + " ");
        }
    }
}
```