Revision.

1. Count substring with equal 0 & 1.

0 1 1 0 0 0 1 1

1 (2) (3) 2)

1. 0011
   1100
2. equal (0 == 1)

(1, 2) = 1
(2, 3) = 2
(3, 2) = 2.

5.

2. a ~~b~~ c

$\rightarrow 2^n$.

"ac"

"abc"

n → no. of char

check subsequence.

s → a b c

t → a b c

i

j

= "abc"

Repeating and Missing element

Problem | Submissions | Leaderboard | Discussions

$n = 5$

Given an array **arr[]** of size N of positive integers(1 - N). One number 'A' from set **{1, 2, ...N}** is **missing** and one number 'B' **occurs twice** in array. Write a program to print the **repeating element** and the **missing element** in array.

(not-sorted).

$1 \le x \le 5$

$1 \le A[i] \le n$

Sample Input 0

```
4
1 2 4 4
```

Sample Output 0

```
4
3
```

$n = 4$

1	4	4	2

1 2 3 4

[1 ——— n]

Constraints

- 1 ≤ N ≤ 10^4
- 1 ≤ arr[i] ≤ N

---

number = 8

size = 8

$1 \le x \le 8$

{1 2 4 5 6 7 8} — arr

sum = 33

sum

$1 + 2 + 3 + \dots + n \Rightarrow n(n+1)/2$

logic

$ts \Rightarrow n(n+1)/2 = 36$

$36 - sum = 3$

eg. ≤ 5

3 1 4 5

sum = 13

$t = \frac{5(5+1)}{2} = 15$

$t - sum = 2$

n = size

n = 4

		6	

Constraints

- 1 ≤ N ≤ 10^4
- 1 ≤ arr[i] ≤ N

Sample Input 0

```
4
1 2 4 4
```

Sample Output 0

```
4
3
```

sum = 11

n = 4.

'A'

1	2		4

rep = "4"

missing no. = $\frac{n(n+1)}{2}$ − sum of rest value.

rest = sum − rep = 7

missing value.

1 2 4

n = 4.

$t = \frac{n(n+1)}{2} = 10$

s = 7

10 − 7 = 3

1	2	4	4	

freq of '4'

0	1	1	0	2

n+1

freq (n+1)

0	1	1	0	~~2~~
0	1	2	3	4

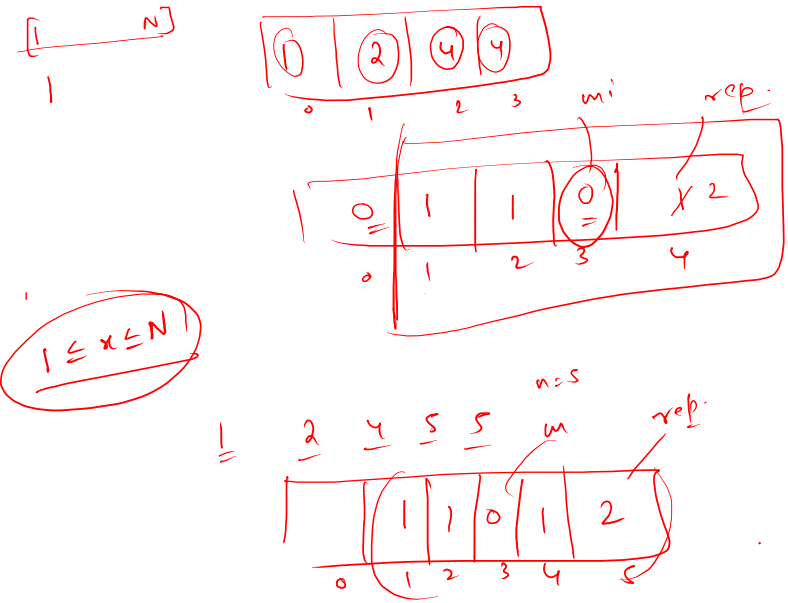
n = 4.

freq (n+1)

n

1 no.

freq[i] == 2.




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

    int [] freq = new int[n+1];
    for(int i = 0; i  < A.length; i++){
        freq[A[i]]++;
    }

    int missingNu = -1;
    int repeatedNu = -1;

     for(int i = 1; i  < freq.length; i++){
        if(freq[i] == 0){
            missingNu = i;
        }
         else if(freq[i] == 2){
             repeatedNu = i;
         }
    }

    System.out.println(repeatedNu);
    System.out.println(missingNu);
```

Subsequence of String

Problem | Submissions | Leaderboard

Given a string, we have to find out all subsequences of it.

Sample Input 0

```
abc
```

Sample Output 0

```
abc ab ac a bc b c
```

abc $n = 3$ . . . $2^n = 8$

[0 ← 0 0 0 — ""
1 ← 0 0 1 → "c"
2 ← 0 1 0 → "b"
3 ← 0 1 1 → bc
4 ← 1 0 0 → a
5 ← 1 0 1 → ac
6 ← 1 1 0 → ab
7 ← 1 1 1 → abc

abcd
$2^4 = 16$.
[15 — 0]

$2^3 = 8$

for ($2^3 - 1$ 7 ; 5 ; 0])

5 → "101"

ac.

'7' → 111

2	7	
2	3	1
2	1	1
	0	1

111

'6' → 110

2	6	
2	3	0
2	1	1
	0	1

last char.

"110"

011

110

7 → 111 → abc . s → "abc" len = 3. total = 8

0 1 2

```java
Scanner scn = new Scanner(System.in);
String s = scn.next();

int total =   (int)Math.pow(2,s.length())  ;
for(int i = total - 1; i >= 0; i--){
    int temp = i;

    String ans = "";
    for(int j = s.length()-1 ; j>=0 ; j--){
        int rem = temp % 2;
         temp /= 2;
        if(rem == 1){
            ans = s.charAt(j) + ans;
        }

    }
    System.out.print(ans + " ");

}
```

i = 7 6

temp = 7 3 1 0

j = 2 1 0

2 ≥ 0

1 ≥ 0

0 ≥ 0 ✓

-1 ≥ 0 ✓

ans = "" → b "c" → cb

r = 1 % 2 == 1 → bc

8

cba

2	7 ✓	
2	3	1
2	1	1
	0	1

n = 3

total of ss → 2^3

[7 6 5 4 3 2 1 0]

111 abc

110 ab

101 ac

...

000 ans = ""

Steps.

$O(n^4)$

Form the largest number

Problem | Submissions | Leaderboard | Discussions

You are given Only **non-negative** integers stored in the array. Arrange the elements in the array such that they form the **largest number**.

i/p

```
4
4 46 8 9
```

Sample Output 0

```
98464
```

desc

0 < k < c ----- 9

| 8 | 4 | 9 | 46 |

46988 4.

eg.

| 2 | 4 |

24

42

| 4 | 2 |

42

eg.

| 3 | 38 |

338

383.

>

| 38 | 3 |

383

eg.

| 3 | 32 |

332

323

| 32 | 3 |

323

Simple sorting in desc. ✗ not work

| 3 | 32 |

a b

332

323

ab ba

332

| 3 | 38 |

338

383

eg.

| 9 | 46 | 8 | 46 | 4 |

| 9 | 8 | 46 | 4 |

98464.

468

446

464

946

| 46 | 8 | 4 | 9 |

0 1 2 3

1-2

1-3

2-3

0→1

0→2

0→3

468

846

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

    for(int i = 0; i < n-1 ; i++){
        for(int j = i+1; j < n; j++){
            String ab = "" + A[i] + A[j];
            String ba = "" + A[j] + A[i];

            int abVal = Integer.parseInt(ab);
            int baVal = Integer.parseInt(ba);

            if(baVal > abVal){
                int tmp = A[i];
                A[i] = A[j];
                A[j] = tmp;
            }
        }
    }

    String ans = "";

    for(int i = 0; i < n; i++){
        ans += A[i];
    }
    System.out.println(ans);
}
```

$O(n^2)$

$O(n^2)$

$O(n)$

$O(n)$
$O(n)$
$O(n)$
$O(n)$

$4\,O(n)$

$O(n^3)$

$O(n^3)$

9			8
468	468	4	9
0	1	2	3

i

$n = 4$

$i = 0$

$j = 1, 2, 3$

$ab = 468$
$ba = 846$

$ab = 84$
$b = 48$

$468 < 846$

89
98

"1 2 3 4" → 1 2 3 4

$ans * 10 + 4$

$ans = 43$