HASHING

Given an array of size N. The number in this array are in range of 1 – N. in this array there is one duplicate element and one missing element.

Find the missing and duplicate element.

A = [2, 1, 2, 3, 4]; len(A) = 5.

Duplicate element is 2
Missing element is 5

In an ideal case, the numbers would be from 1 – n. So, for n = 5, A = [1, 2, 3, 4, 5]

1st Approach: Pattern:

A = [2, 1, 2, 3, 4]

If we sort A, we will get, [1, 2, 2, 3, 4]. We can see that the value at each index is index + 1. For an ideal case, **`value = i + 1`**. So,

A [0] = 1

A [1] = 2

A [2] = 3

A [3] = 4

A [4] = 5

But in **our case**,

A [0] = 1

A [1] = 2

A [2] = 2 ➔ this is the repeated element

A [3] = 3

A [4] = 4

We will now check for 5, as per the ideal array A = [1, 2, 3, 4, 5]

2nd Approach – Brute-Force:

map = {1:2, 2:1, 3:1, 4:1}

Create a map and using a counter, we can increase the counter and check if the value is present in the map for a range of 1 – n.

Implementation:

 # for duplicate element

```
for i in a:
     if i not in d:
          d[i] = 1
     else:
          d[i] += 1
for key, value in d.items():
     if value == 2:
     duplicate_element = key
```

Time Complexity = O (n)

Space Complexity = O(n)

O (n)

 # for missing element

```
count = 1
while count <= len(A):
if count not in d:
     missing = count
else:
     count += 1
```

O (n)

Q) How to optimize it so that space complexity is O (1)?

A = [2, 3, 2, 1, 4]

In an ideal world, A would be [1, 2,3,4,5], the sum of all elements = 15. i.e.,

$$n * (n + 1)/2 = 15$$

OR

$$a + b + c + d + e$$

Since we have a duplicate element, we can write,

$$2a + b + d + e = 15$$

$$2a = 15 - b - d - e$$

Ex: 2

A = [2, 3, 3, 1, 5]; ideal_A = [1, 2, 3, 4, 5] i.e., 1 – 5 since len(A) = 5

a b c d e

We understand that the duplicate value (dup = 3) and the missing element (missing = 4).

A	2	3	3	1	5	=	14
ideal_A	1	2	3	4	5	=	15

Let us substitute the ‘2’ as duplicate (dup) and ‘5’ as missing.

A	~~2~~	~~dup~~	dup	~~1~~	~~5~~	=	14
ideal_A	~~1~~	~~2~~	~~dup~~	mis	~~5~~	=	15
-	-	-	-	-	-		-
XX			dup - mis			=	-1

Square A and ideal_A then subtract square of A from Square of ideal_A.

$\cancel{4}$	$\cancel{dup^2}$	dup^2	$\cancel{1}$	$\cancel{25}$	=	48
$\cancel{1}$	$\cancel{4}$	$\cancel{dup^2}$	mis^2	$\cancel{25}$	=	55
-	-	-	-	-		-
YY		$dup^2 - mis^2$			=	-7

$dup^2 - mis^2 = -7$

➔ (dup – mis) * (dup + mis) = -7

Since, (dup – mis) = -1 → **(1)**, we can substitute it in the above equation.

➔ (dup – mis) * (dup + mis) = -7

➔ (-1) * (dup + mis) = -7

➔ dup + mis = 7 → **(2)**

Eq (1) – Eq (2) ➔

$$
\begin{array}{rcl}
dup - mis & = & -1 \\
dup + mis & = & 7 \\
\hline
2 * dup & = & 6
\end{array}
$$

dup = 6/2 = 3

now, 2 * dup = 6

dup = 6 / 2 = 3

dup = <u>3</u>

Substitute the 'dup' value in Eq **(2)**

dup + mis = 7

mis = 7 – dup = 7 – 3

mis = <u>4</u>

So, the duplicate element (dup) is 3 and the missing element is 4.

CODE:

```
def solve(A):
    XX = sum(A) - n*(n+1)/2      # y - x

    sum1 = 0                     # sum of square of n natural numbers
    for i in range(1, len(A)+1):
        sum1 += i * i

    sum2 = 0                     # sum of squares of array element.
    for i in A:
        sum2 += i*i

    YY = sum2 - sum1             # (Y**2) - (X**2)

    ZZ = YY/XX                   # Y + Z

    _Y = (ZZ + XX)/2
    _X = ZZ - _Y

    return _X, _Y


if __name__ == "__main__":
```

```python
    A = [1, 2, 2, 3, 4]
    n = len(A)

    print(solve(A))
```

NOTE - regarding Math:

1. Sum of the First ‘n’ Natural Numbers = **n = n(n**+1) / 2
2. Formula for the Sum of the First N Squares = (N * (N + 1) * (2N + 1)) / 6
3. Basic Math (addition, subtraction, multiplication, division).