(Course) (Abdul Bari)

udemy

* Topic :- Array (Student challenge)

① Finding Missing element

(a) Single missing element :- (Sorted)

- If index starts from 1

A	1	2	3	4	5	6	8	9	10	11	12
	0	1	2	3	4	5	6	7	8	9	10

Sum of n natural no = $\frac{n(n+1)}{2}$

```
{
    for (i=0; i<11; i++)
          sum = sum + A[i]

    s = n * (n+1)/2
        ↓
        12

    return s - sum;
}
```

- If index doesn't start from 1 {using A[i] - i trick}

A	6	7	8	9	10	11	13	14	15	16	17
	0	1	2	3	4	5	6	7	8	9	10

l = 6
h = 17
n = 11

A[i] - i trick

6 - 0 = 6   7 - 1 = 6   8 - 2 = 6

```
diff = l - 0;
for (i=0 ; i<n ; i++)
{
    if (A[i] - i != diff)
    {
        return i + diff;
    }
    return -1;
}
```

O(n)

(b) <u>Multiple</u> <u>Missing</u> <u>element</u>

```
diff = l - 0;
for (i=0 ; i<n ; i++)
{
    if (A[i] - i != diff)
    {
        while ( diff < A[i] - i)
        {
            cout << i + diff;
            diff ++;
        }
    }
```

(C) MME (Method - 2 Hash table)

A	3	7	4	9	12	6	1	11	2	10
	0	1	2	3	4	5	6	7	8	9

l = 1
h = 12
n = 10

		1	1	1	1		1	1		1	1	1	1
H	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	0	~~0~~	~~0~~	0	~~0~~	~~0~~	~~0~~	0
	0	1	2	3	4	5	6	7	8	9	10	11	12

```
{ Hash table
     Bitset }
{
    for (i=0; i<n; i++)
    {
        H[A[i]]++;
    }

    for (i=l; i<=h; i++)
    {
        if (H[i] == 0)
            cout << i << " ";
    }
}
```

2) Finding + Counting duplicated in sorted Array

(a)

3	6	8	8	10	12	15	15	15	20

n = 10

```
last duplicate = 0;
for (i=0; i<n-1; i++)
{
    if (A [i] == A [i+1] && A [i] != last duplicate)
    {
        printf ("%d \n", A [i]);
        last duplicate = A [i];
    }
}
```

Counting :-

```
for (i=0; i<n-1; i++)
{
    if (A [i] == [A [i+1])
    {
        j = i+1;
        while (A [i] == A [j])
            j++;
        printf ("%d", j-i);
        i = j-1;
    }
}
```

$\boxed{O(n)}$

(b) using hashing

```
for (i=0; i<n; i++)
{
    H [A [i]] ++;
}
```

```
for (i=0; i<=max; i++)
    if (H[i]>1)
        printf ("%d %d", i, H[i]);
```

O(n)

(d) In Unsorted Array

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

n = 10

O(n²)

```
for (i=0; i<n-1; i++)
{
        count = 1;
        if (A[i] != -1]
      { for (j = i+1; j<n; j++)
        {
            if (A[i] == A[j])
            {
                count++;
                A[j] = -1;
            }
        }
      }
      if (count>1) printf (A[i],
                             count);
}
```

(e.) Using Hashing

{ Same as Sorted Array }

↓

Just find the max element

③ Finding pair of elements with sum k

a+b=k

(a)

```
for (i=0; i<n-1; i++)
{
    for (j=i+1; j<n; j++)
    {
        if (A[i]+A[j]==k)
            printf (A[i] A[j]);
    }
}
```

$O(n^2)$

(b) Using hashing

```
for (i=0; i<n; i++)
{
    if (H[k-A[i]] != 0)
        printf ("%d + %d = %d" A[i]
                k-A[i], k);

    H[A[i]]++;
}
```

$O(n)$

(c) For Sorted array

A										
	1	3	4	5	6	8	9	10	12	14
	0	1	2	3	4	5	6	7	8	9

$a+b=10$

```
i=0, j=n-1

while (i<j)
{
    if (A[i] + A[j] == k)
    {
        printf (—);
        i++;
        j--;
    }
    elce if (A[i]+A[j] <k)
            i++;
    else
            j--;
}
```

$\boxed{O(n)}$

1). Finding max and min in a single scan

$\boxed{n-1}$

Best case
O(n)

```
min = A[0];
max = A[0];

for (i=1; i<n; i++)
{
    if (A[i] < min)
        min = A[i];
    else if (A[i] > max)
        max = A[i];
}
```

worst case
$\boxed{2(n-1)}$
O(n)

$\boxed{O(n)}$