3) Difference (A - B)

* Unsorted

A

3	5	10	4	6
0	1	2	3	4

m

B

12	4	7	2	5
0	1	2	3	4

n

C

3	10	6							
0	1	2	3	4	5	6	7	8	9

$m * n$

$= n * n$

$= n^2$

Time : $O(n^2)$

* Sorted

A

3	4	5	6	10
0	1	2	3	4

m

B

2	4	5	7	12
0	1	2	3	4

n

C

3	6	10							
0	1	2	3	4	5	6	7	8	9

$\theta(m+n)$

time : $\theta(n)$

121. Finding Single missing Element in a Sorted Array

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

$\frac{n(n+1)}{2} = \frac{12(13)}{2} = 78$

```
* sum = 0;
for(i=0; i<11; i++) {          // Using Sum of first n Natural no.
    sum = sum + A[i];
}
s = n*(n+1)/2;
s - sum;
printf("Missing no is %d", s-sum);
```

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

6+6 = 12

6-0 = 6
7-1 = 6
8-2 = 6
9-3 = 6
:
11-5 = 6
13-6 = 7

l = 6
h = 17
n = 11

```
diff = l - 0;
for (i=0; i<n; i++) {
    if (A[i] - i != diff) {
        printf("missing el %d \n", i+diff);
        break;
    }
}
```

Time - O(n)

122. Finding Multiple Missing Elements in an array

A	6	7	8	9	11	12	15	16	17	18	19
	0	1	2	3	4	5	6	7	8	9	10
	6	6	6	6	7	7	9	9	9	9	9

6+4 = 10
7+6 = 13
8+6 = 14

l = 6
h = 19
n = 11

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

Time - O(n)

123. Finding Missing Element in an Unsorted Array Method 2 using hashing

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 $O(n^2)$
Time - O(n)

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

Hashtable / BitSet

```
for(i=0; i<n; i++){          — n
    H[A[i]]++;
}
for(i=l; i<=h; i++){         — n
    if(H[i]==0)
        printf("%d\n", i);   — 2n
}
```

Scanning

Time O(n)

124. Finding Duplicates in a Sorted array

A	3	6	8	8	10	12	15	15	15	20
	0	1	2	3	4	5	6	7	8	9
	i		i				i			

n=10

* lastDuplicate = 0; // for Printing
for(i=0; i<n-1; i++){
if(A[i] == A[i+1] && A[i] != lastDuplicate)
{
printf("%d\n", A[i]);
lastDuplicate = A[i];
}
}
Time - O(n)

* for(i=0; i<n-1; i++){ // for Counting
if(A[i] == A[i+1]){
j = i+1;
while(A[j] == A[i])
j++;
printf("%d is appearing %d times",
A[i], j-i);
i = j-1;
}
}
Time - O(n)

125. Finding Duplicates in Sorted Array using Hashing

A	3	6	8	8	10	12	15	15	15	20
	0	1	2	3	4	5	6	7	8	9

$+n$
$2n$

Time - $O(n)$

H	0	0	0	~~0~~ 3	0	0	~~0~~ 1	0	~~0~~ 2	0	~~0~~ 1	0	~~0~~ 1	0	0	~~0~~ 3	0	0	0	0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

~~0~~ 1	}
20	

```
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]);
}
```

126. Finding Duplicates in an Unsorted Array

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

$n = 10$

$\frac{n(n-1)}{2} = n-1 + n-2 + \dots + 3+2+1$

$\frac{n^2-n}{2} =$ Time - $O(n^2)$

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

Time - $O(n^2)$

* Using Hashing

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

$n = 10$

H	0	0	ϕ_1	ϕ_2	ϕ_1	ϕ_1	ϕ_3	ϕ_1	ϕ_2
	0	1	2	3	4	5	6	7	8

n

$2n$

Time - $O(n)$

↑

Linear

127. Finding a Pair of Elements with Sum k
(a+b = k)

A	6	3	8	10	16	7	5	2	9	14
	0	1	2	3	4	5	6	7	8	9

a+b=10
n=10

i j

$n-1 + \dots + 3 + 2 + 1 = \frac{n(n-1)}{2}$

$O(n^2)$

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

n
n+n

Time - $O(n^2)$ = Quadratic

* Using Hashing

A	6	3	8	10	16	7	5	2	9	14
	0	1	2	3	4	5	6	7	8	9

a+b=10

H	0	0	1	1	0	1	1	1	1	1	1	0	0	0	1	0	0
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

```
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]]++;
}
```

Time - O(n)

128 Finding a Pair of Elements with sum k in Sorted Array

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

a+b = 10

n = 10

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

Time - O(n)

129. Finding Max and Min in a single Scan

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

min = ~~5 8 3~~ -1

max = ~~9 8 9~~ 10

```
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];
}
```

Compare → n-1 → 10, 9, 8, 7, 2, 1

Compare → 2(n-1) → 1, 2, 3, 5, 8, 9, 10

Best	n-1	O(n)
Worst	2(n-1)	O(n)

130 Solutions for Quiz 3

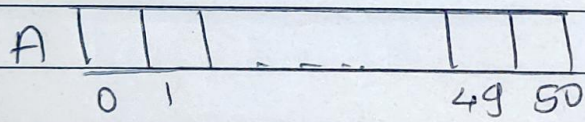
1) A[1...n]

reverse (1, k)
reverse (k+1, n)
reverse (1, n)

A	2	3	4	5	6	7	8
	4	3	2	5	6	7	8
	4	3	2	8	7	6	5
	5	6	7	8	2	3	4
	1	2	3 (k)	4	5	6	7 (n)

rotates left by k position

2) 500 elements
Range 0-100
frequency of elements > 50



Array of 50 members

3) a

2	4	8	9	13	16	19	23	25
0	1	2	3	4	5	6	7	8

i, j

S = 10

```
if (a[j] - a[i] < s) j++;
```

4) A

8	7	5	12	9	3	4
0	1	2	3	4	5	6

O(1)

5) A

7	12	15	6	4	13	9	5
0	1	2	3	4	5	6	7

n+n = 2n

O(n)