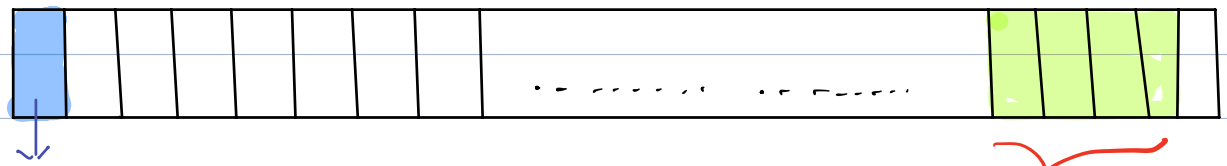
Array

Array is a collection of same type data stored at contiguous memory locations



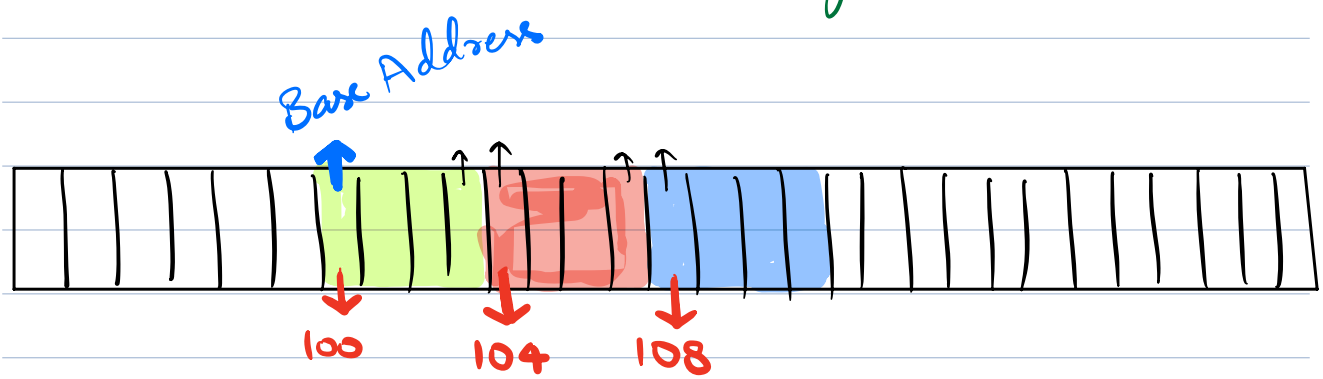
1 Byte = 8 Bits.

Integer.

Integer = 4 Bytes

Store an integer array of size 3.

4 × 3 = 12 Bytes



Declare ⇒ int arr[3];

arr ⇒ Base Address

Initialise :

```
                0  1  2  3  4
int arr[5] = { 0, 1, 2, 3, 4 };
```

$N \Rightarrow [0, x] \Rightarrow [0, N-1]$

$x - 0 + 1 = N$

$x = N - 1$

$i = 0 \Rightarrow 100$ (Base Address)

$i = 1 \Rightarrow 100 + 1 \times 4 = 104$

$i = 2 \Rightarrow 100 + 2 \times 4 = 108$

$i = 3 \Rightarrow 100 + 3 \times 4 = 112$

$\vdots$

$i \Rightarrow 100 + i \times 4$

$\Downarrow$

Base Address + i x (size of one element)

Access the element on the index i = arr[i]

O(1)

arr[-1] > Out of Bound Exception.
arr[N]

No DS apart from array is allowed

Given an integer array of size N. Print all the elements.

0	1	2	3	4	5
5	6	2	3	1	9

A: [5, 6, 2, 3, 1, 9]

Soln Code

```
for (i=0; i<N; i++) {
    print (A[i]);
}
```

T.C. = O(N)

Given an integer array of size N.

Count the no. of elements having atleast one element greater than itself.

1) A: [3, -2, 6, 8, 4, 8, 5]

Ans = 5

2) A: [2, 3, 10, 7, 3, 2, 10, 8]

Ans = 6

Soln > Brute force (Check Everything)

∀ Elements of the array, check if a greater element is present.

Code

```
count = 0;
for (i=0; i<N; i++) {
    // A[i].
    for (j=0; j<N; j++) {
        if (A[j] > A[i]) {
            count++;
            break;
        }
    }
}
```

T.C. = $O(N^2)$

(2) Optimised Solution.

Which element won't have any element greater than itself present in the array

⇒ Max Element.

Ans = # Elements in Array − 1

1 ⇓ Count of max element.

Steps

1) Find the max element → O(N)
2) Count of max element → O(N)
3) Ans = N − count of max → O(1)

Code

```
1) int max = A[0];
   for (i=1; i<N; i++) {
       if (A[i] > max) {
           max = A[i];
       }
   }
```

O(N)

2) int count = 0;

```
for (i = 0; i < N; i++) {
    if (A[i] == max) {
        count ++;
    }
}
```

O(N)

3) return N - count;

T.C. = O(N + N) = O(N)

H.W: find the max of an array & freq of max in 1 loop.

Given an integer array of size N.

Check if there exists a pair (i, j) s.t. A[i] + A[j] = K (i != j)

K ↓ Input

A = { 3, 5, 2, 7, 3 } ⇒ Yes

0 1 2 3 4

K = 6

11

6

A[0] + A[0] = 6 ✗
(i, j)

A[0] + A[4] = 6 ✓

Sol^n 1) Brute force (Check all poss.)

i ⇒ [0, N-1] (N)

j ⇒ [0, N-1] (N)

> i != j

```
for (i = 0; i < N; i++) {
    for (j = 0; j < N; j++) {
        if (i != j) {
            if (A[i] + A[j] == K)
                return True;
        }
    }
}
```

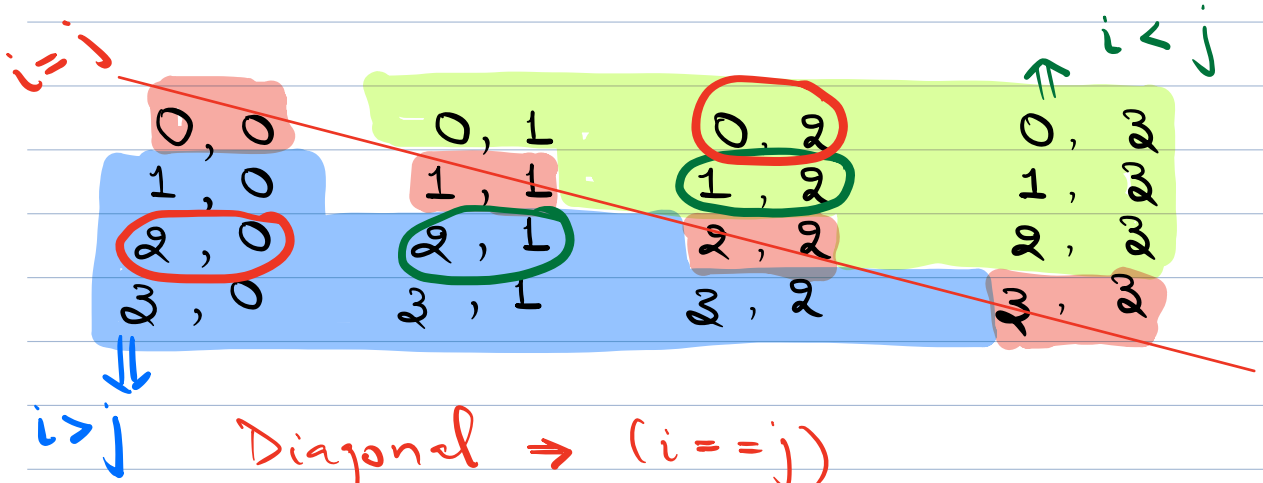
}

return false;

T.C. = $O(N^2)$

2) Optimise

N = 4 [3, 2, 1, 5]



Diagonal → (i == j)

(i = 1, j = 2) ⇒ A[1] + A[2]
(i = 2, j = 1) ⇒ A[2] + A[1]

(i = 2, j = 0) ⇒ A[2] + A[0]
(i = 0, j = 2) ⇒ A[0] + A[2]

⇒ Instead of iterating over all pairs, choose either upper or lower quadrant.

```
for (i = 0;  i < N;  i++) {
    for (j = i+1;  j < N;  j++) {
        if (A[i] + A[j] == K) {
            return True;
        }
    }
}
return false;
```

$i = N-1$, $j = [i+1, N-1]$

$\Downarrow$

N

T.C. = $O(N^2)$

Given an integer array of size N. Reverse the array w/o using extra space.

⇒ S.C. = $O(1)$

A = [-1, 4, 7, 6]

⇒ [6, 7, 4, -1]

A : [8, 3, 2, 5, 1]

⇒ [1, 5, 2, 3, 8]

Soln ⇒ Take new array & copy elements in reverse order

A : [8, 3, 2, 5, 1] ← (0, 1, 2, 3, 4)

B = [1, 5, 2, 3, 8] → S.C. = O(N)

Override A = [1, 5, 2, 3, 8]

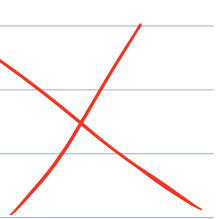
Not accepted.

2) Try to modify original array.

	0	1	2	3	4
A :	8	3	2	5	1

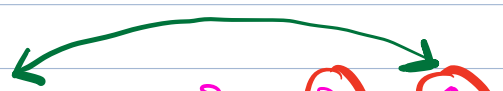
$0 \longrightarrow N-1$
$1 \longrightarrow N-2$
$2 \longrightarrow N-3$
$\vdots$
$i \longrightarrow N-i-1$

```
for (i=0; i<N; i++) {
    A[i] = A[N-i-1];
}
```

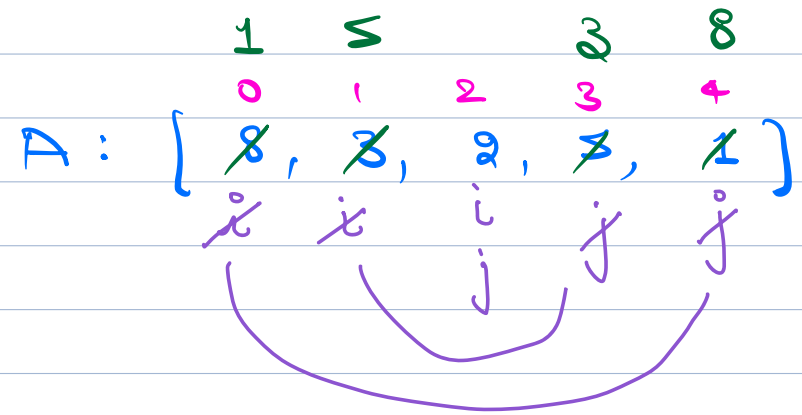


	i	i	i	i	i
	0	1	2	3	4
A :	~~8~~	~~3~~	~~2~~	~~5~~	~~1~~
⇒	1	5	2	5	1

3) Correct sol^n (Swapping)



	0	1	2	3	4
A :	~~8~~	~~3~~	~~2~~	~~5~~	~~1~~
	1	5	2	3	8

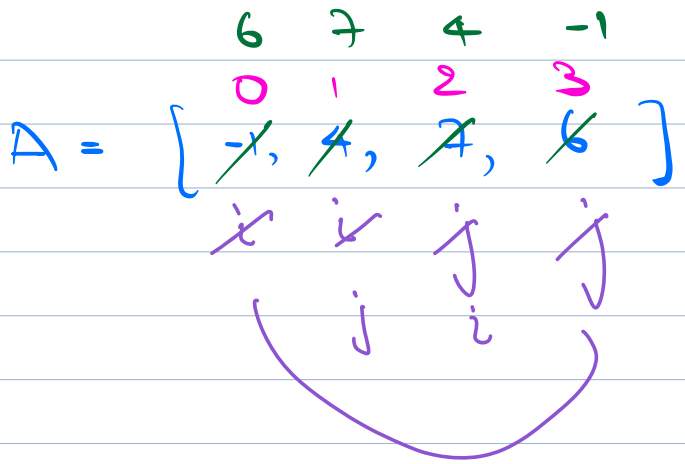


Code

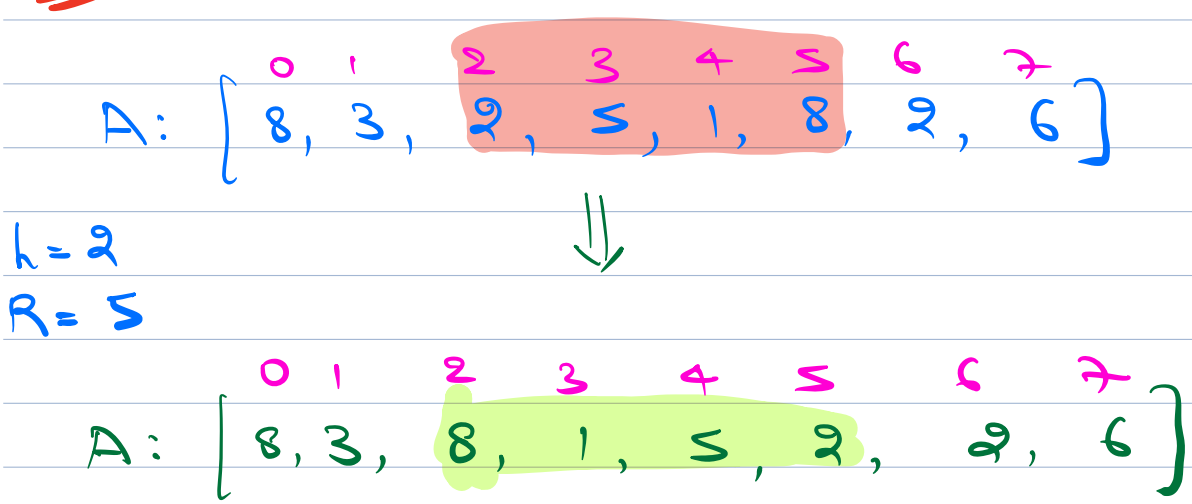
```
void reverse (int[] A){
    int N = A.length();
    i = 0;    j = N-1;
    while ( i < j ) {
        temp = A[i];
        A[i] = A[j];
        A[j] = temp;

        i++;
        j--;
    }
}
```

T.C. = O(N)

S.C. = O(1)



Reverse array from L to R



L = 2
R = 5

Code

```
void reverse (A, l, r) {
    i = l;   j = r;
    while ( i < j ) {
        temp = A[i];
        A[i] = A[j];
        A[j] = temp;

        i++;
        j--;
    }
}
```

T.C. = O(N)
S.C. = O(1)