Space Complexity

Integer ⇒ 4 Bytes = 32 bits
Long ⇒ 8 Bytes = 64 bits

Eg

```
func (int N) {
        int x = 5;        // 4 Bytes
        int y = 4         // 4 Bytes

        int z = x + y;    // 4 Bytes
        print (z);
}
```

Total space = 12 Bytes.

⇒ Space complexity does not include the Input & Output.

Space Complexity is also written in Big O notations.

Q1

```
func( int N) {
    int arr[10];            // 4×10 = 40 Bytes
    int x                   // 4 Bytes
    int y                   // 4 Bytes
    long z                  // 8 Bytes

    int [] arr1= new int[N]; //4N Bytes
                      ↓
                     4 × N
}
```

Total memory = 40+4+4+8 +4N

$= \cancel{56} + \cancel{4}N$

S.C. = O(N)

Q2

```
func (int N) {
    int arr[10];            // 4×10 = 40 Bytes
    int x                   // 4 Bytes
    int y                   // 4 Bytes
    long z                  // 8 Bytes

    int [] arr1= new int[N]; //4N Bytes
    long [][] l = new long[N][N]; //8N^2 Bytes
}
```

$$\text{Total space} = \cancel{56} + \cancel{4N} + \cancel{8}N^2$$

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

Code

```
                     I/P          I/P
int getMax ( int arr[],   int N) {
      int ans = A[0];            // 4
      for (i= 1; i <N; i++) {
            ans = max (ans, arr[i]);
      }
      return ans;
}
```

Iterating over an array

$$A = \begin{array}{ccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 4, & 1, & \underline{3}, & 7, & \underline{6}, & \underline{9}, & \underline{2} \end{array}$$

$$ans = \cancel{4}\ \cancel{7}\ 9$$

→ the above code is used to find the max of the array

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

Array

Array is the collection of same data types.

Datatype ⇒ int, float, char, ... etc

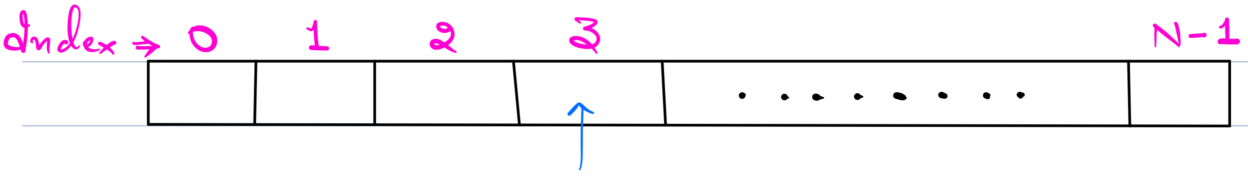
Declare

Name

int arr[N] ;

Datatype

Size



arr[i]

⇓

Access the element on the index i

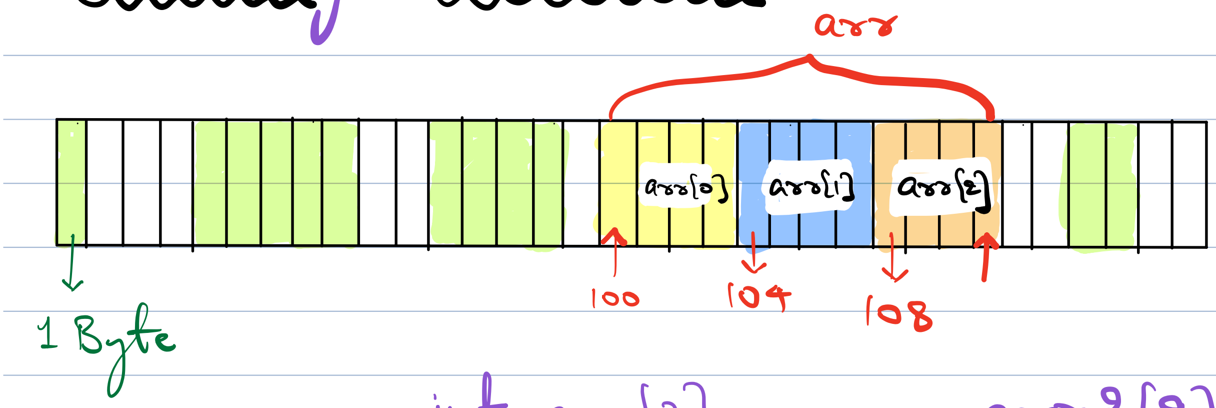
T.C. = O(1)
(Random access)

Types of Arrays

1) Static Array

2) Dynamic Array. (flexible)

Memory Allocation



int arr[2] → Stores the mem location of the first block of arr

↓

3×4 = 12 Bytes

↓

12 continuous blocks

arr2[2]

arr [0] = 100

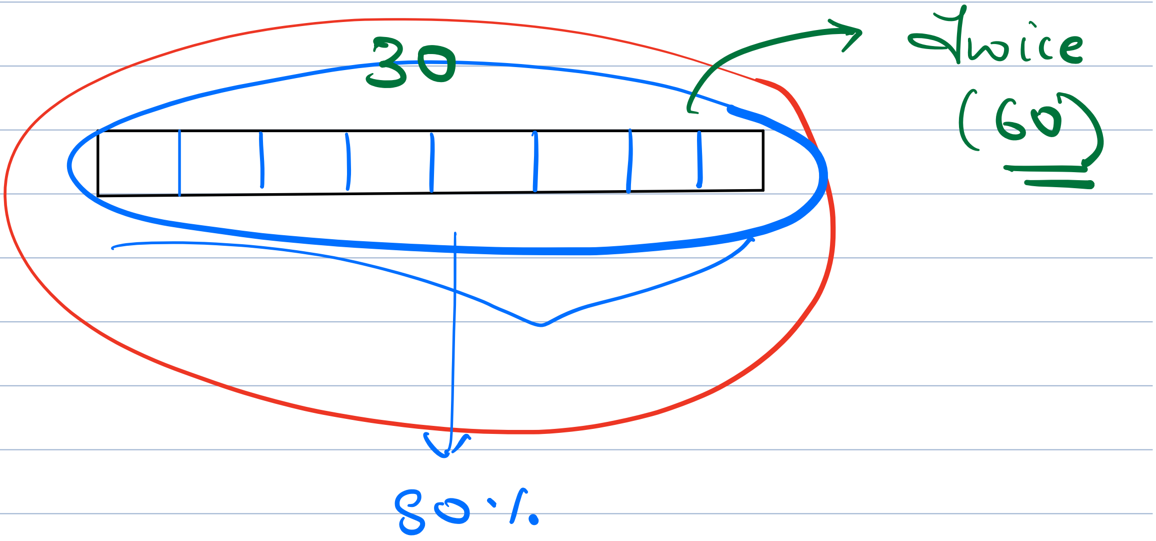
arr [1] = 104 = 100+4×1

$arr[2] = 102 = 100 + 4 \times 2$

$\vdots$

$arr[i] = \text{Base Address} + i \times \left(\text{Mem used by the datatype}\right)$

Dynamic



Q Print all elements of the array.

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

Q// Given an integer array of size N. Reverse the entire array.

Eg : $N = 5$

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & \\ arr = \{ & 1, & 2, & 3, & 4, & 5 & \} \end{array}$$

$$\begin{array}{lcccccc} & 0 & 1 & 2 & 3 & 4 & \\ arr = \{ & 5, & 4, & 3, & 2, & 1 & \} \end{array}$$

$0 \rightarrow 4 \ (N-1)$
$1 \rightarrow 3 \ (N-2)$
$2 \rightarrow 2 \ (N-3)$
$\vdots$
$i \rightarrow (N-1-i)$

Sol^n

1) Create a new array

```
int arrrev[N];

for (i=0; i<N; i++ ){

            arrrev[N-1-i] = arr[i];
}
return arrrev;
```

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

$\Downarrow$

S.C. = $O(1)$??

$$arr = \{ \overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{4}, \overset{4}{5} \}$$

$$arr = \{ \underset{0}{5}, \underset{1}{4}, \underset{2}{3}, \underset{3}{2}, \underset{4}{1} \}$$

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

$$arr = \{ \overset{0}{1}, \overset{1}{2}, \overset{2}{2}, \overset{3}{2}, \overset{\overset{i}{4}}{1} \}$$

$1 \downarrow$ arr[4] ?? ✗

$\underbrace{2, 1}$ → store original elements somewhere

~~arr2[N/2] ??~~ ⇒ S.C. = O(N)

0 → 4 , 4 → 0

$$arr = \{ \overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{4}, \overset{4}{5} \}$$

swap

a = ~~5~~ 6
b = ~~6~~ 6

temp = a;
a = b
b = a

```
for (i=0; i < N/2 ; i++ ) {

    temp = arr [N-1-i];
    arr[N-1-i] = arr[i];
    arr [i] = temp;

}
```

T.C. = $O(N)$

S.C. = $O(1)$

$$arr = \{\underbrace{\overset{0}{1}, \overset{1}{2}, \overset{\leftarrow \; 2}{3}, \overset{3}{4}, \overset{\overset{i}{4}}{5}}_{\text{reversed twice}}\}$$

$$arr = \{\overset{0}{1}, \overset{1}{2}, \overset{2}{3}, \overset{3}{4}\}$$

$4/2 = 2$

$$A: [\overset{0}{5}, \overset{1}{4}, \overset{\overset{j}{\overset{i}{2}}}{3}, \overset{3}{2}, \overset{4}{1}] \qquad A = [\overset{0}{4}, \overset{\overset{j}{1}}{3}, \overset{\overset{i}{2}}{2}, \overset{3}{1}]$$

Till when do we swap → $(i<j)$

Code

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

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

Q. Given an integer array of size N. & two integers s & e.

Reverse the part of the array from index s to the index e.

Eg: A = {1, 2, 3, 4, 5, 6, 7}

	0	1	2	3	4	5	6
A =	1	2	3	4	5	6	7

s = 1, e = 5

	0	1	2	3	4	5	6
A =	1	6	5	4	3	2	7

A = {1, 6, 5, 4, 3, 2, 7}

Code

```
void  reverse (int [] arr, int N, int s, int e) {
    i = s ;
    j = e   ;

    while ( i < j ) {

        temp = arr [i];
        arr [i] = arr [j];
        arr [j] = temp;

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

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

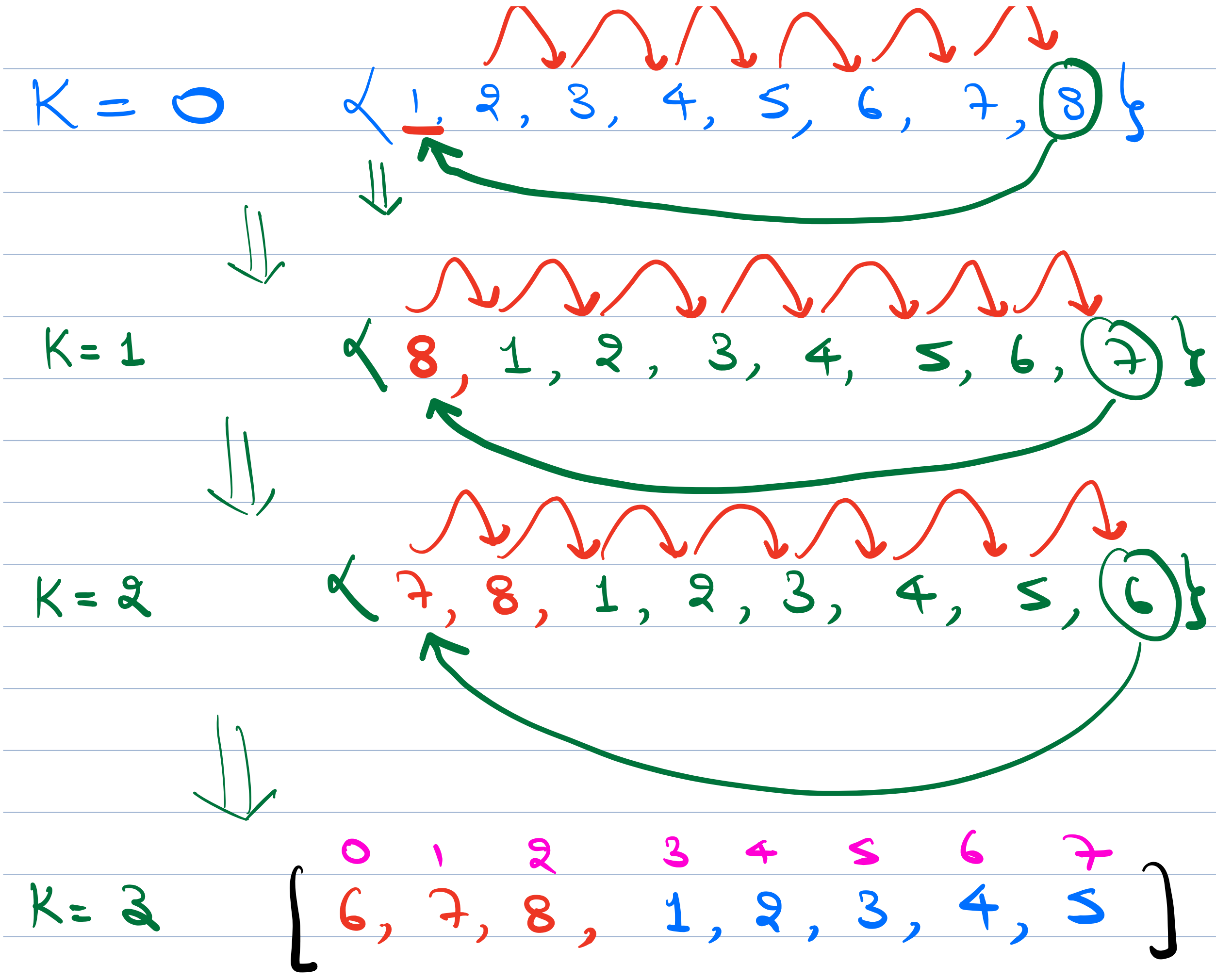
Q Given an array of size N.
Rotate the array right to left K times.
(If K = 1, last element comes to 1st position)

$$\begin{array}{lcccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \text{Eg: } A = \{ & 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8 \} \end{array}$$

K = 3

K = 8 (N) ⇒ Original array   S.C. = O(1)

K = 9 (N+1) = K = 1 | K = 10 (N+2) = K = 2



Code

```
K = K % N ;
for (j=0; j<K; j++) {                // O(K)
        temp = A[N-1];
        for (i= N-2; i ≥ 0; i--) {
                A[i+1] = A[i];
        }                                    } O(N)
        A[0] = temp;
}
```

$$T.C. = O(N \times K)$$

$$\Downarrow$$

Optimise ??

H.W. Hint Reverse

$$A = \left\{ \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1, & 2, & 3, & 4, & 5, & 6, & 7, & 8 \end{array} \right\}$$

$$K = 3 \quad \left\{ \begin{array}{cccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 6, & 7, & 8, & 1, & 2, & 3, & 4, & 5 \end{array} \right\}$$

$$\uparrow$$

$$Rev : \{ 8, 7, 6, 5, 4, 3, 2, 1 \}$$

Exp $T.C. = O(N)$

$S.C. = O(1)$

H.W. Read about Dynamic array syntax in your language.

Next ⇒ Range Queries = Prefix Sum.