TODAY:

→ Space complexity
→ Intro to array
→ Solve problems.

→ Start → all classes ka revision material.
↳ written notes + github link.
↳ revision videos.
↳ Every Sunday / Saturday 11am revision class for that week.

all people who will attending
↳ Linkedin optimisation
↳ hunt job using linkedin
↳ Strategies
↳ AI tools to use for job applicⁿ
↳ recruiter's perspective.

people who attend all
Contents →
pip above > 80%.
Nice attend 70%.
↓ ↓ ↓
One of my project
(Internship) kivio
CTO
↓
prospective global.com

Space Complexity

4 byte

```
func( int N ){          TC → O(1)
    int x;      ⟶      x  [ ]   4 bytes.
    int y;      ⟶      y  [ ]   4 byte
    long z;     ⟶      z  [  ]  8 byte.
}
```

16 byte

Note: Do not consider input size

Space complexity is O(1) 2 → if input changes space taken is always constant.

Ques 1

```
func( int N ){ ↑
    int x;      ⟶  4 byte
    int y;      ⟶  4 byte
    long z;     ⟶  8 byte.
    int[] A = new int[N]; ↑   N x 4 byte.
}
```

16 + 4N ⟶ O(N)

Ques 2

```
func( int N ){ ↑
    int x;      ⟶  4 byte
    int y;      ⟶  4 byte
    long z;     ⟶  8 byte.
    int[] A = new int[N]; ↑   N x 4 byte.
    int[][] L = new int[N][N];
}
```

16B + 4N + 4N²

N + N² = O(N²)

N x N = N² ⟶ N

Given a program find the Space complexity

```
int maxArr(int arr[], int N) {
    int ans = arr[0];
    for(i from 1 to N-1) {
        ans = max(ans, arr[i]);
    }
    return ans;
}
```

arr* — 2*

N — 2* — I/P space so ignore

ans — 4 byte

i — 4 byte

8 byte = O(1)

ON = 

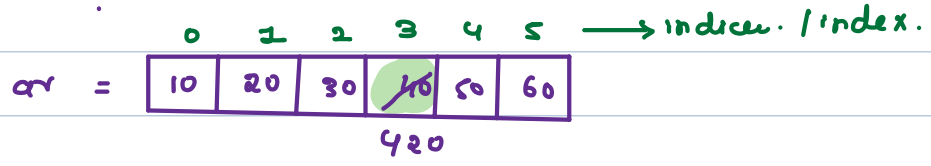
i = 0, 1 .. n

600 problems → T.C

→ S.C

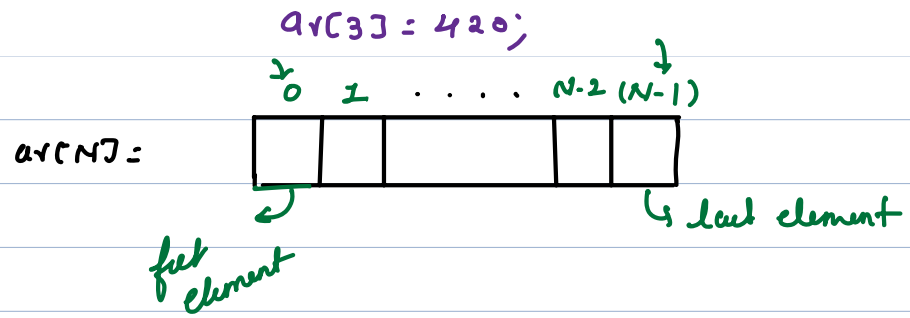
(Recursion material python people)

Introduction to array



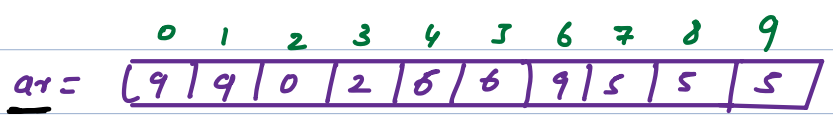
Note: array starts from zero

```
print(ar[3]);
```



0, N-1

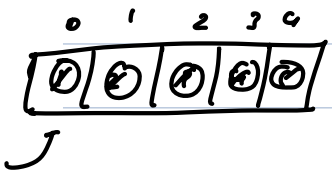
print all the elements in an array



print(ar) → o/p : #add.

address of array.

n = ar.sizeof()



pseudocode :-

```
for i → 0 to n-1 {
    print(ar[i]);
}
```

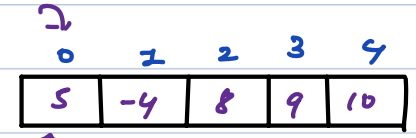
Query

```
i = 2;
print(ar[i]);
```

am : O(1)

Quizes

int arr[5] = {5,-4,8,9,10}
Print sum of 1st and 5th element
Choose the correct answer



add 2nd element & add 10th element.

↓

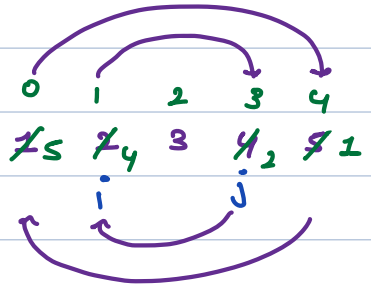
arr[1] + arr[9].

print(arr[0] + arr[4]).

→ Given an array of size N. Reverse the entire array

N = 5

arr: { 1, 2, 3, 4, 5 } → o/p => arr: { 5, 4, 3, 2, 1 }



i=0, j=n-1

i, j	swap(arr[i], arr[j]).	
i<j	0, 4	swap(1,5) i++, j--
	1, 3	swap(2,4) i++, j--
	2, 2	

↳ stop here i<j

```
function reverse ( arr , N ) {
    i = 0    j = N-1;        4byte + 4bytes
    while ( i < j ) {  ⟶  N/2  ⟶  T.C : O(N)
        temp = arr[i];      4byte  12B  ⟶  S.C. : O(1)
        arr[i] = arr[j];
        arr[j] = temp
        i++ , j--
    }
}
```

→ Given an array of size N, and you are given 'l' and 'r' Reverse the array form l to r.

Ex: arr[10]; { 2 4 [-1 3 6 8 11] 13 -5 10 }

0	1	2	3	4	5	6	7	8	9
2	4	-1	3	6	8	11	13	-5	10

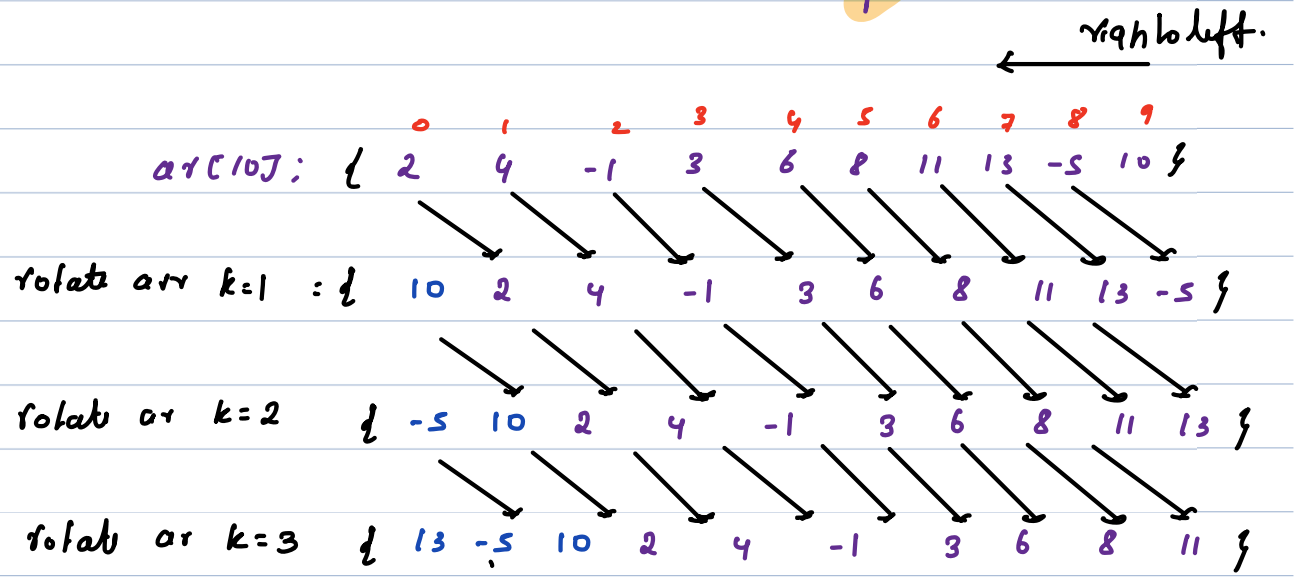
l = 2
r = 6

→ part of an array
→ Continuous. (subarray)

```
function reverseSubArr ( arr , N , l , r ) {
    i = l    j = r;        4byte + 4bytes
    while ( i < j ) {  ⟶  N/2  ⟶  T.C : O(N)
        temp = arr[i];      4byte  12B  ⟶  S.C. : O(1)
        arr[i] = arr[j];
        arr[j] = temp
        i++ , j--
    }
}
```

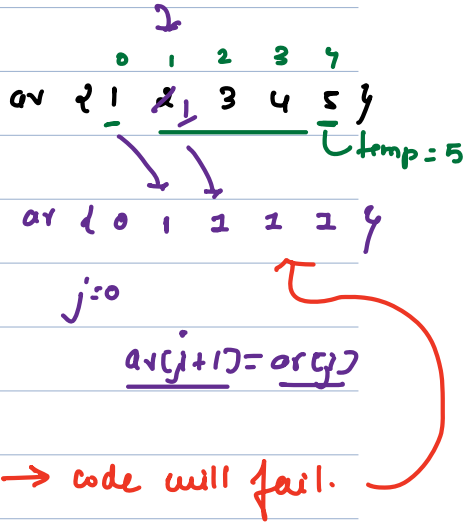
Given an arr of size N. Rotate the array from right to left k times.

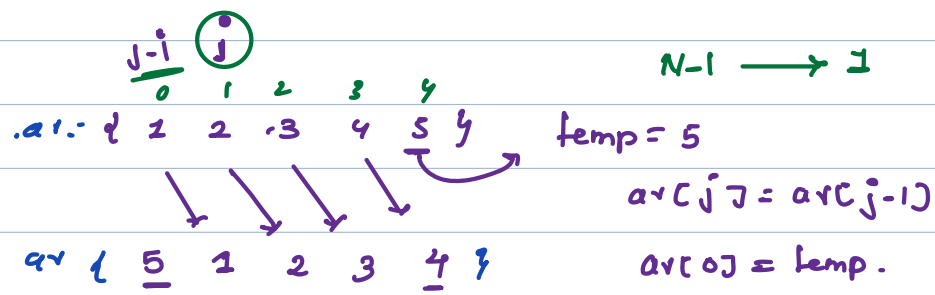
Note k=1 last element will come at first position.



Brute force

```
for i → 0 to k-1 {
    temp = arr[N-1];
    for j → 0 to N-2
        arr[j+1] = arr[j];
    arr[0] = temp;
}
```





pseudocode

4 byte

```
for i → 0 to k-1 {          ⟶ k
    4B   temp = arr[N-1];
    for j → N-1 to 1        ⟶ N
4byte        arr[j] = arr[j-1];
    }
    arr[0] = temp;
}
```

T.C : O(N*k)

S.C : O(1).

Break: 10:38 pm ↓ 10:48 pm

Observation :- k = 3 ←

arr : { 1 2 3 4 5 6 7 }

k=1

7 1 2 3 4 5 6

k=2

6 7 1 2 3 4 5

k=3

{ 5 6 7 1 2 3 4 }

reverse original array → { 7 6 5 4 3 2 1 }

↓ reverse first k element (0 – k-1)

↓ reverse k ⟶ N-1

Some

arr ⟶ { 5 6 7 1 2 3 4 }

Pseudocode

```
function rotateK(arr[], k){
    n = arr.size();
    if(k==0){
        return;
    }
    reverse(arr, n);
    reverseSubAr(arr, n, 0, k-1);
    reverseSubAr(arr, n, k, n-1);
}
```

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

reverse → O(N) · 3N

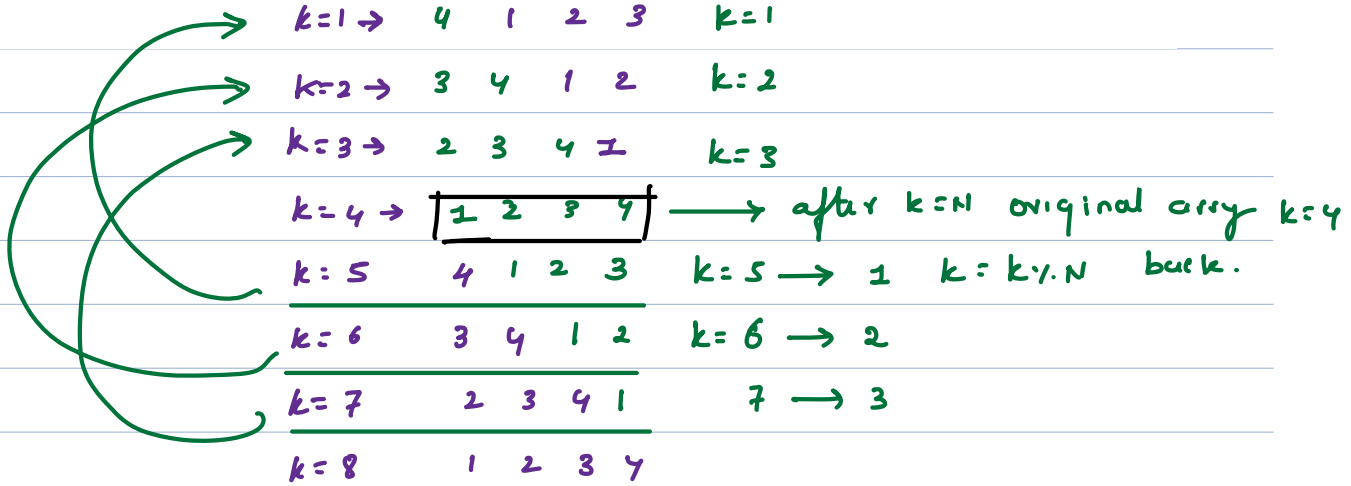
reverseSubAr → O(N)

reverseSubAr → O(N)

error

k > N code will break

↳ code is incomplete.



k = k % N

= 5 % 4 = 1

= 6 % 4 = 2

= 7 % 4 = 3

8 % 4 = 0.

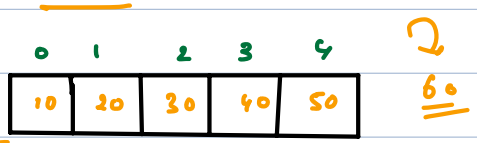
```
function rotatek ( ar[], k ) {                    T.C : O(N)   S.C : O(1)
    n = ar.size();          |  if (k==0) {
    k = k % n;              |      return;
    reverse ( ar, n);                          → O(N)
    reverseSubAr (ar, n, 0, k-1);              → O(N)
    reverse SubAr ( ar, n, k, n-1);            → O(N)
}                                                 k
```

Arrays →

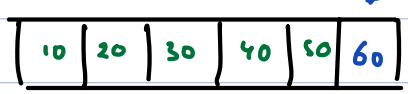
Drawback → Array size is fixed

Strength : Can access element in O(1)

```
int ar[] = int[5];
```



copy all element → Add one element



→ Dynamic arrays

Java : Arraylist

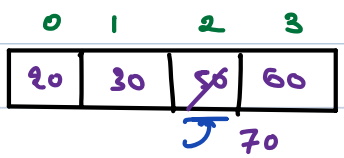
```
ArrayList<DataType> al = new ArrayList<>();

ex: ArrayList<Integer> al = new ArrayList<>()
    al.add(1);
    al.add(2);
```

Python

```
thisList = [] // creating a list
this.append(1);
the.append(2);
```

Print all element is array list

Sopln(al); → all elements.



access element at i^{th} index

i = 2

al.get(i);

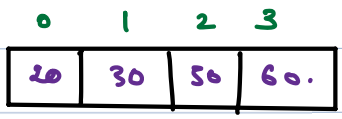
Update the value at i^{th} index

al.set(index, value)

al.set(i, 70);

Print all element in list

print(thisList);



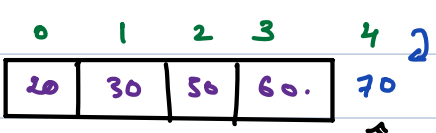
access element at i^{th} index

i = 2.

thisList[i];

Update the value at i^{th} index

thisList[i] = 70



added / appended at last

al.add(70) thisList.append(70)

thisList(5);

al.get(5); → error (Index out of bound)

Next class ↘ ↗ ↗ ↓

↳ prefix sum → vvvvvvvvImp