## Time Complexity

- Brg 0, Iterations
- Sc
- Ap / Gp / log ..

## Arrays

→ int ar[5] = { 3, 2, 8, 9, 7 }

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|

→ int ar[N] = → 1st Element index: 0
→ Last Element index: N-1

index:

→ i = 0; i < N; i++) {
  print(ar[i])
}

Note: To access arr[i] it is O(1)

→ TC: O(N)

10) Given N array elements, count no: of Elements having atleat 1 element, greater than itself [No External Libraries]

ar[7]: { -3, -2, 6, 8, 4, 8, 5 }

5 Element ✓ ✓ ✓ X ✓ X ✓

7-2=5 ✓

ar[8] = { 2, 3, 10, 7, 3, 2, 10, 8 }

6 Elements ✓ ✓ X ✓ ✓ ✓ X ✓

8-2 = 6 ✓

ar[] = { 2, 5, 1, 4, 8, 0, 8, 1, 3, 8 }

10-3 = 7 ✓

Observations:

1) For Max element we won't have any element greater than that

2) Get count of max elements = cnt

3) Final ans: N-cnt

Pseudo Code

1) Iterate & get Max ✓→

2) Iterate & get cnt of Max ✓

3) N-cnt ✓

```
int max = 0/ar[0]/INT_MIN/
for(i=0; i<N; i++){
    if(max < ar[i]){
        max = ar[i]
    }
}
```

Ex: ar[3] : -2 -8 -3 max

max = 0 : 0 0 0 } 0

```
int cnt = 0
for(i=0; i<N; i++){
    if (ar[i] == max){
        cnt++
    }
}
return N - cnt
```

TODO : HW

1) Calculate in a single iteration

Total Iterations = 2N

TC : O(N) SC : O(1)

2)

// Return type bool

Given N array elements, check if there exists a pair i, j such that ar[i] + ar[j] = k and i != j

i and j: Index Value
k: Given Sum.

i != j

i + j = k

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| ar[] : | 3 | -2 | 1 | 4 | 3 | 6 | 8 |

k = 10 : i = 3, j = 5 ar[3] + ar[5] = 10 ✓ return True

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| ar[] : | 2 | 4 | -3 | 7 |

k = 5 : return False

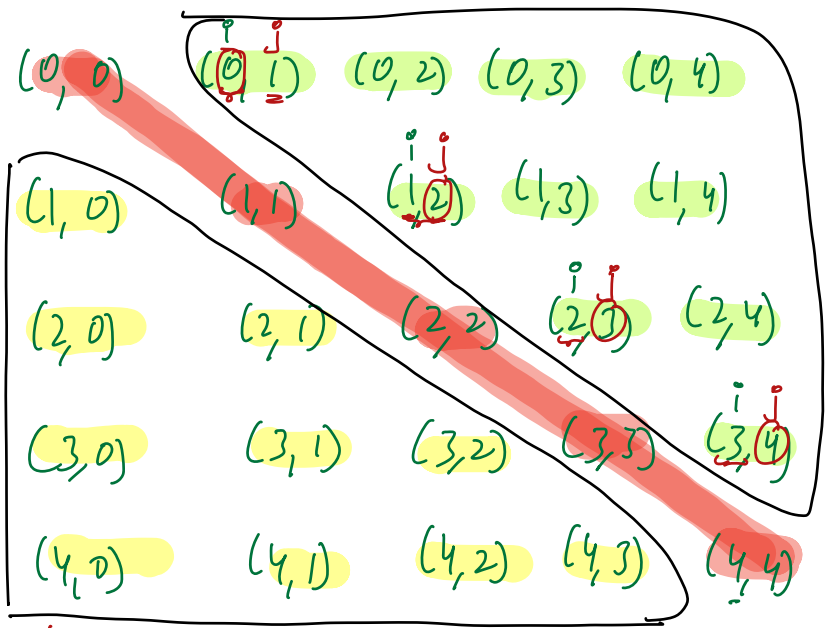
| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| ar[] : | 2 | 4 | -3 | 7 |

k = 8 : i = 1, j = 1, ar[1] + ar[1] = 8 X [i != j]

Idea:

1) For every pair check if sum == k

N = 4 : 0, 1, 2, 3, 4




TODO Lower:

Obs:

Either iterate in upper triangular a lower.

// iterate in upper.

```
i = 0 ; i < N; i++) {      i = N-1
    j = i+1 ; j < N; j++) {
        if (ar[i] + ar[j] == k)
            return True
    }
}
return False
```

```
i = 0; i < N; i++) {
    j = 0; j < N; j++) {
        if( i == j)
            continue } skip loop
        if (ar[i] + ar[j] == k) {
            return True
    }
}
```

Iterations: $O(N^2)$ SC: $O(1)$

// Iterate in upper.

```
for(i=0; i<N; i++) {
    for(j=i+1; j<N; j++) {
        if (ar[i]+ar[j]==k)
            return true
    }
}
```

SC: O(1)

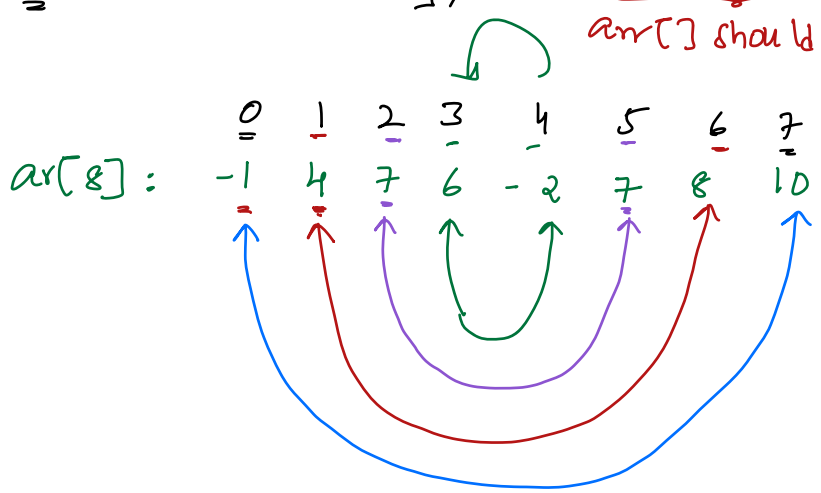
| i | j : [i+1, N-1] | Iterations |
|---|---|---|
| 0 | [1, N-1] | N-1 |
| 1 | [2, N-1] | N-2 |
| 2 | [3, N-1] | N-3 |
| ⋮ | ⋮ | ⋮ |
| N-1 | [N, N-1] | 0 |

Sum of N-1 Natural Numbers → $\frac{(N)(N-1)}{2}$

TC: $O(N^2)$

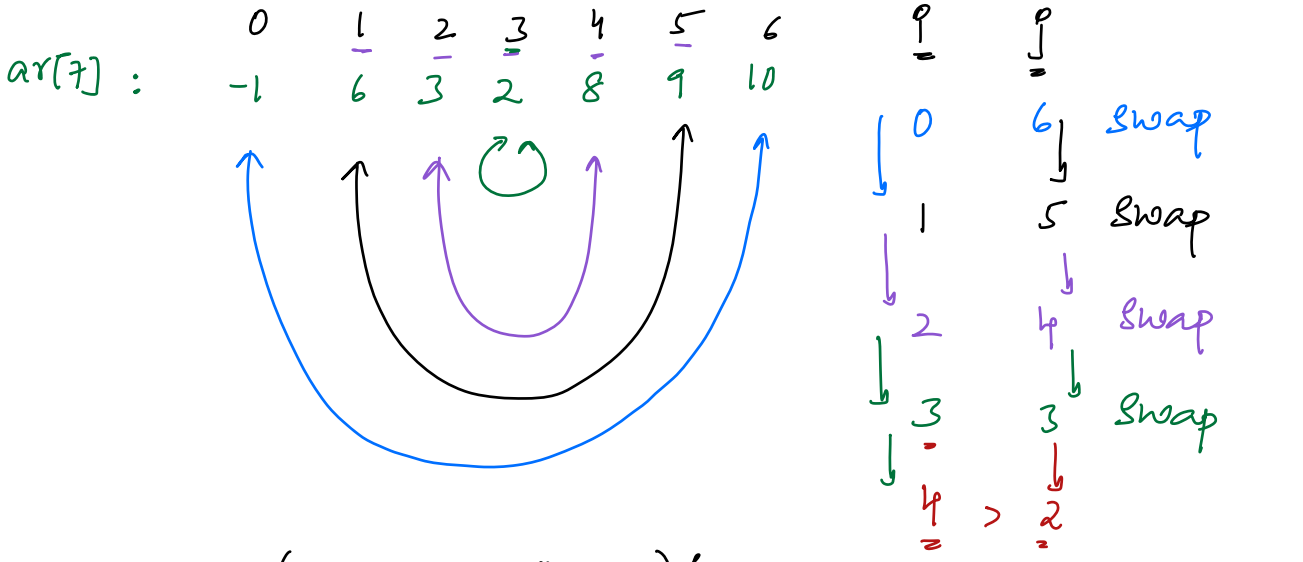
3Q) Given an array, Reverse entire arr[] → No Extra Space → Expected SC: O(1)

arr[] should change




| i | j : (N-1) | |
|---|---|---|
| 0 | 7 | Swap |
| 1 | 6 | Swap |
| 2 | 5 | Swap |
| 3 | 4 | Swap |
| 4 | 3 | Break |

When i > j break




```
_ reverse (int ar[], int N){
    i = 0, j = N-1        → ar i < j
    while ( i < = j ) {   → when i > j, while loop breaks
        {swap ar[i] && ar[j]}   TODO swap with temp
        i++, j--                swap without extra
                                variable.
    }
}
```

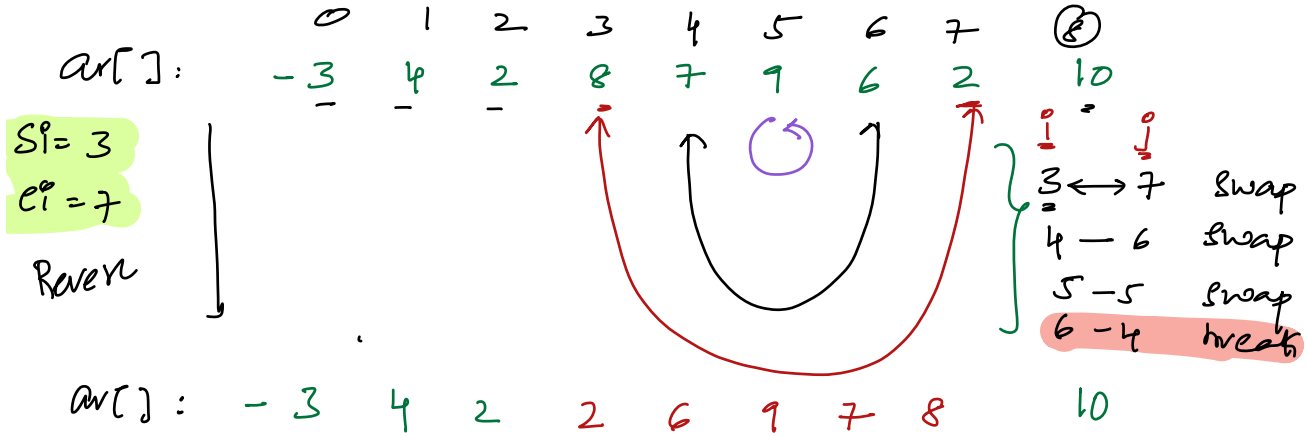
10:40pm break

Iterations: $\{N/2\}$

TC: O(N)

SC: O(1)

48) Given N array elements & si, ei reverse entire array from from [si – ei], si <= ei




```
_reven (int ar[], int si, ei){
    i = si, j = ei
    while ( i <= j ) {          → When i > j,
        {swap ar[i] && ar[j]}
        i++, j--
    }
}
```

// How many elements
[s - e] = {e - s + 1}

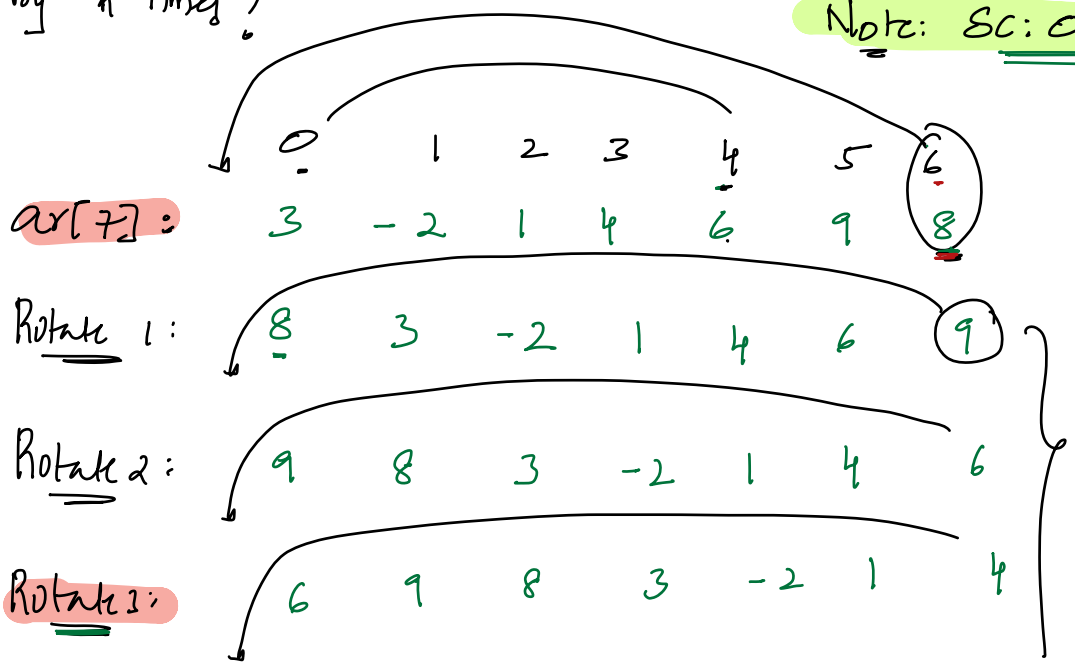
// Iterations =
$\frac{e-s+1}{2}$

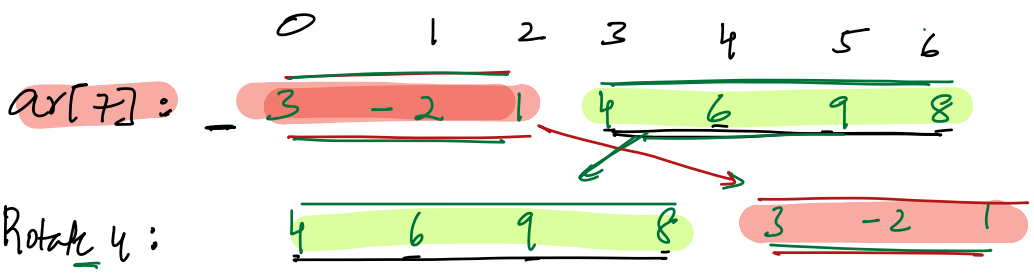
TC: O(N)

50)

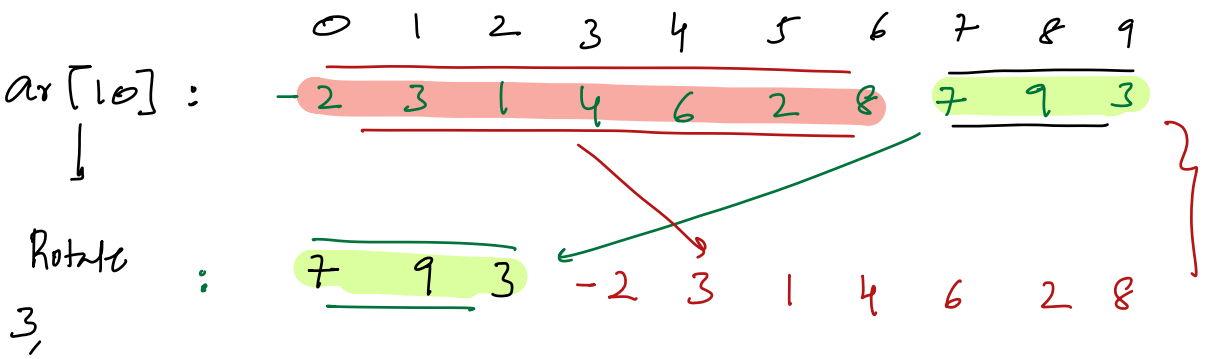
Given an array of N Elements, Rotate array in clockwise from last - first by k times?

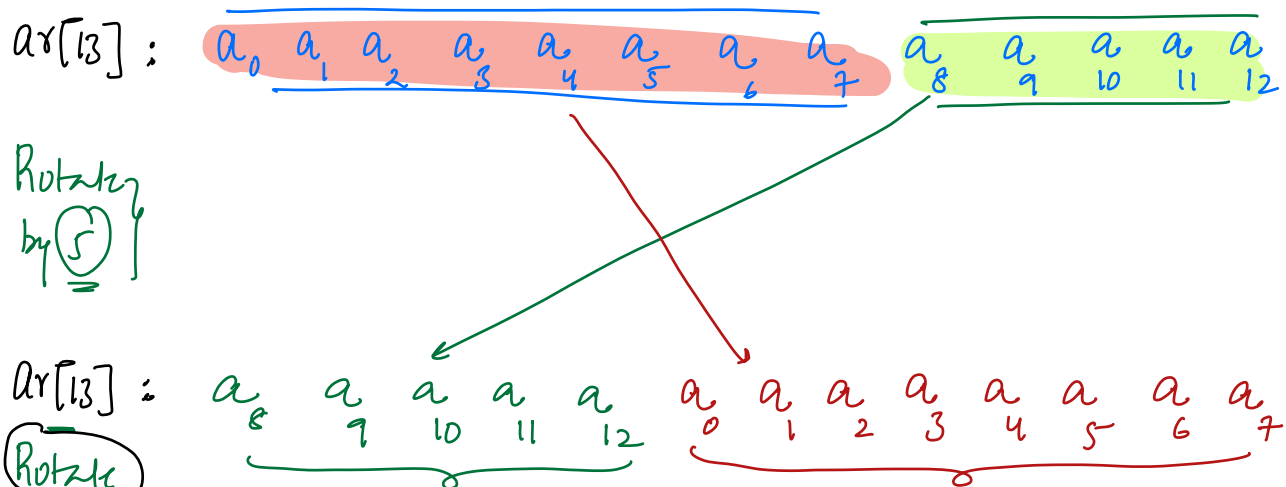
Note: SC: O(1)




↳ last 4 Elements They are start from 3 Element They are last




k = 5 2) Reverse first 5 Elements 3) Reverse last 8 Elements

Rotate $a_8\ a_9\ a_{10}\ a_{11}\ a_{12}\ a_0\ a_1\ a_2\ a_3\ a_4\ a_5\ a_6\ a_7$

Pseudocode

Q) Rotate array by k times

- reverse(ar, 0, N-1) ✓ ⇒ N/2
- reverse first k elements
  - reverse(ar, 0, k-1) ✓ ⇒ k/2
- reverse last N-k elements
  - reverse(ar, k, N-1) ✓ ⇒ (N-k)/2

Total Iterations ⇒ N

TC: O(N)

SC: O(1)

Note: k ≥ N :

- reverse(ar, 0, N-1) ✓
- reverse(ar, 0, k-1) ✗ Array out of bounds

ar[6] : $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$

Rotate 0 : $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$

Rotate 1 : $a_5$ $a_0$ $a_1$ $a_2$ $a_3$ $a_4$

Rotate 2 : $a_4$ $a_5$ $a_0$ $a_1$ $a_2$ $a_3$

Rotate 3 : $a_3$ $a_4$ $a_5$ $a_0$ $a_1$ $a_2$

Rotate 4 : $a_2$ $a_3$ $a_4$ $a_5$ $a_0$ $a_1$

Rotate 5 : $a_1$ $a_2$ $a_3$ $a_4$ $a_5$ $a_0$

Rotate 6 : $a_0$ $a_1$ $a_2$ $a_3$ $a_4$ $a_5$

| Rotate | Rotate | Rotate |
|---|---|---|
| 6 | 12 | 18 |
| 7 | 13 | 19 |
| 8 | 14 | 20 |
| 9 | 15 | 21 = 21%6 = 3 |
| 10 | 16 | |
| 11 | 17 | |

24 = 28 = 28%6

If k > N: k = k % N

- → reverse(ar, 0, N-1)
- → reverse(ar, 0, k-1)
- → reverse(ar, k, N-1)

int ar[5] } store 5 elements

int ar[N] } store N Elements

Dynamic arrays } → array size not fixed

| C++ | Python | Java | C# | Javascript | C → C++ |
|---|---|---|---|---|---|
| Vector | list | Array list | Array list | Array | Change your language |

list → Dynamic array

list<int> a : → Dynamic array of type int

↳ a.size() = 0

- a.insert(20)
- a.insert(30)
- a.insert(40)
- a.insert(50)
- a.insert(60)

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 20 | 30 | 40 | 50 | 60 |

insert() : Insert element at back.

a.size() : 5 → {0, 1, 2, 3, 4}

TC : O(1)

// a.insert() a.size() }

```
——— fun (int arr[], int N){
|   // insert array elements in llist
|   & return
|
|   list<int> l;                          TC: O(N)
|   for(i=0; i< N; i++){
|   |   l.insert(arr[i])
|   }
|   return l
}
```

```
// Given a list iterate & print
——— fun( list<int> l ){
|
|   int N = l.size();
|   for(i=0; i< N; i++){                  TC: O(N)
|   |   print(l[i])
|   }
}
```

Doubts:

1) 
```
for(i=0; i<N; i++){
    a = arr[i]
    b = k-a
    Search for b, in array of n/2 }
```

TC: $O(N^2)$

2)
```
for(i=N; i>0; i=i/2){
    for(j=0; j<i; j++){
        {print(C)}
    }
}
```

| i | j : [0, i-1] | Total iterations i = i/2 |
|---|---|---|
| N | [0, N-1] | N |
| N/2 | [0, N/2-1] | N/2 |
| N/4 | [0, N/4-1] | N/4 |
| N/8 | [0, N/8-1] | N/8 |
| ⋮ | | ⋮ |
| 1 | [0, 0] | 1 |

$2^{\log_2 N} = N$

Total Iterations =

$$N/2^0 + N/2^1 + N/4 + N/8 + \cdots + N/2^{\log_2 N} \} \rightarrow \log_2 N + 1$$

Total Terms

→ Sum of GP: $a = N,\ r = 1/2,\ t = \log_2 N + 1$

→ $r < 1$ :

$$\frac{(a)(1 - r^t)}{1 - r}$$

$$= \frac{(N)\left(1 - (1/2)^{\log_2 N + 1}\right)}{1/2}$$

$$= (2)(N)\left(1 - (1/2)^{\log_2 N + 1}\right)$$

$$= (N)(2)\left(1 - \underbrace{(1/2)^{\log_2 N + 1}}_{\approx 0}\right)$$

$$= (N)(2)(1)$$

$$= O(N)$$

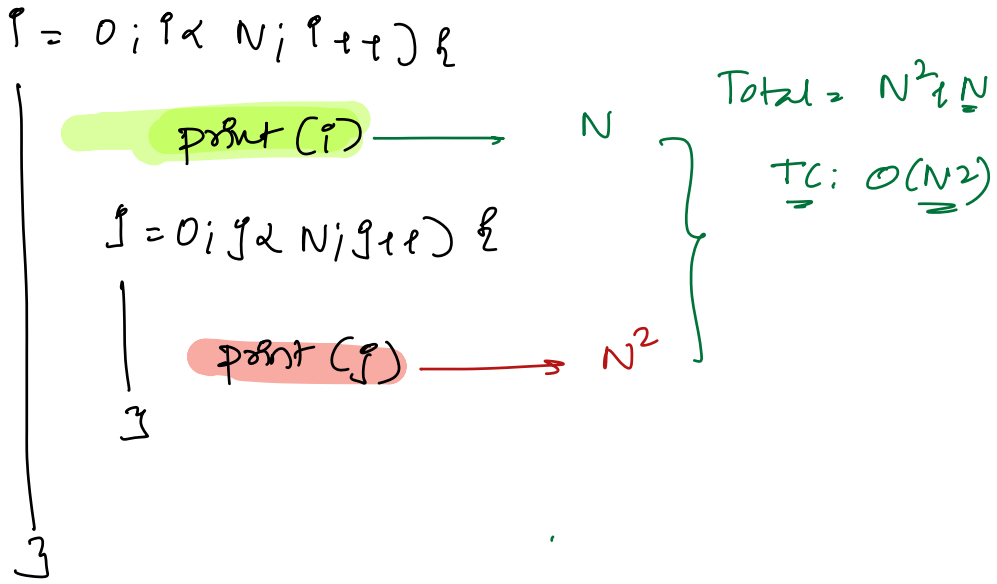
$(1/2)^1 = 1/2 = 0.5$

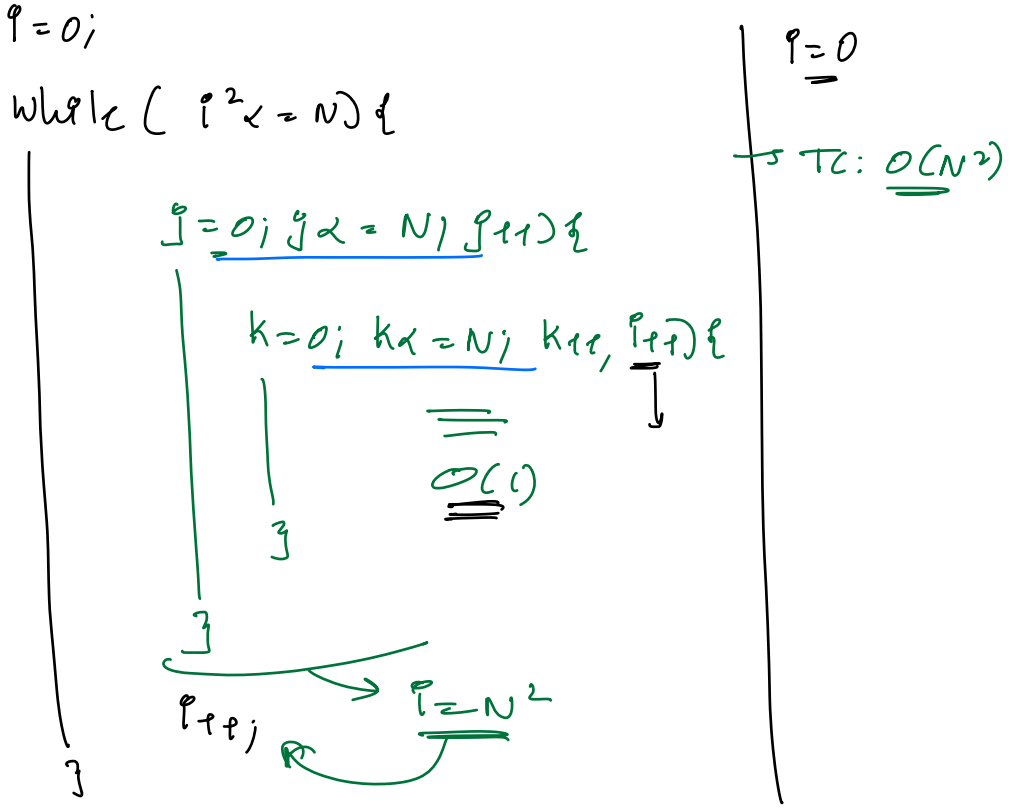
$(1/2)^2 = 1/4 = 0.25$

$(1/2)^3 = 1/8 = 0.125$

$(1/2)^4 = 1/16 = 0.0625$

$(1/2)^n \rightarrow$ inc $\} \approx 0$

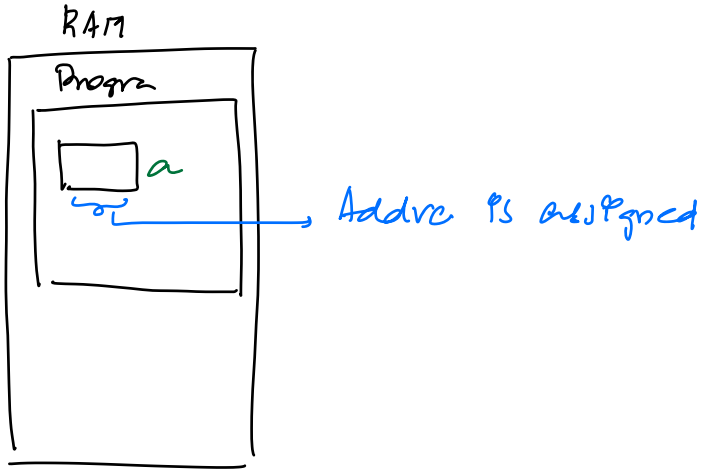

for(i=0; i<N; i++){
print(i)
N
for(j=0; j<N; j++){
print(j)
N²
}
}
Total = N² + N
TC: O(N²)


i=0;
while(i²<N){
for(j=0; j<N; j++){
k=0; k<N; k++, i++){
O(1)
}
}
i++;
i=N²
}
i=0
TC: O(N²)

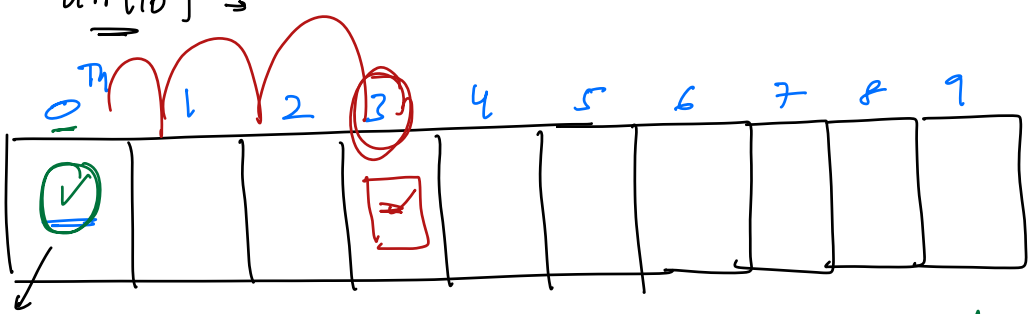
Why TC → O(1)

→ int a : 4B

→ print (&a)




→ int arr[10] →




Addrs :

print(&(arr[0]))

print (arr)

arr →

arr[3] → *(arr + 3)

pointer arithmetics

addrs of 0th index

addition to address

print (arr[3]) → O(1)

Q2) f(N), g(N) → f(N) = N g(N) = N²

→ O(f(N) + g(N)) → O(N + N²) → O(N²)

→

i = 0, j = N-1 : [j-i = N-1]

↓ ↓ ↓ → diff reduced by 2

i = 1, j = N-2 j-i = N-3

↓ ↓ | → reduce by 2

i = 2, j = N-4 j-i = N-5

⋮

After N/2 iteration diff == 0

j-i >= (0)

Eg:

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| ar[]: | 3 | 2 | -9 | 6 | 4 | -1 | 8 | 2 | 10 |

// Code:

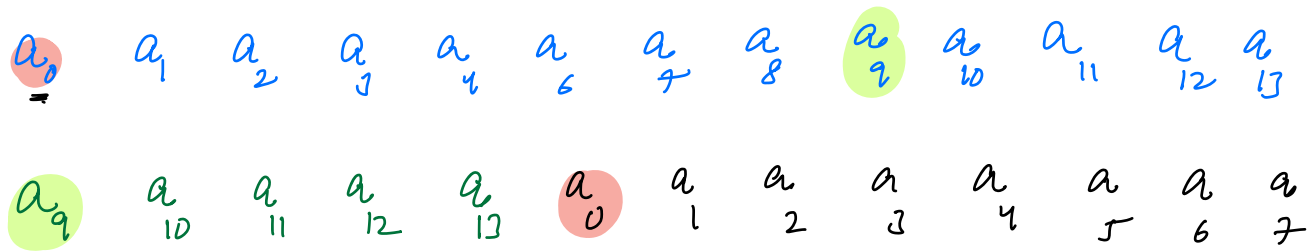
```
for(i = 0; i < N || i++){
    a = ar[i]
    b = k - a
    if ( b in ar[] )
}
```

→ TC → O(N²)

SC:

b in ar[] → O(N)

ar[ ] : (k)




Swap: a[9] → a[0]
ar[0] → temp

ε N = 20 k = 4

→ gcd + Modular } Advanced Session

// Rotate array by swap with gcd technique //