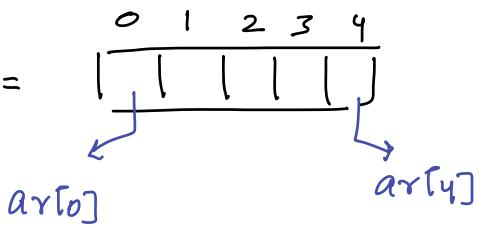
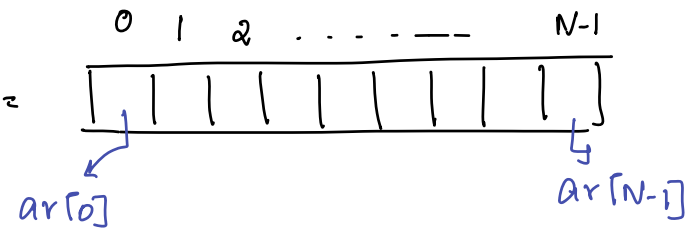
Today's Content

- → Arrays
- → Doubt → public chat
- → Answer → private chat

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




```
int ar[N] =
```



ar[-1] or ar[N] or ar[N+1] ... Error

Input Space

We are passing array as parameter, in function

```
void printAr(int ar[]) {
    int n = ar.length
    for(int i=0; i<n; i++){
        print(ar[i])
    }
}
```

TC: O(N)

SC: O(1)

1Q) Given N array elements, count no: of elements, having atleast 1 element greater than itself

ar[7] = { -3 -2 6 8 4 8 5 } # ans = 5

	0	1	2	3	4	5	6
	-3	-2	6	8	4	8	5
c=0	+1	+1	+1		+1		+1

ar[8] = { 2 3 10 7 3 2 10 8 } # ans = 6

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

ar[10] = { 2 5 1 4 8 0 8 1 3 8 } # ans = 7

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

ar[5] = { 8 8 8 8 8 } # ans = 0

	0	1	2	3	4
	8	8	8	8	8

obs1: For man element we cannot have number greater than itself

obs2: Iterate & get no: of man elements = c

Final ans : (Total number of elements) - (c)

Pseudocode:

```
int countgreater(int ar[]) {
    int am = ar[0];
    int N = ar.length
    for(int i=1; i< N; i++){
        if(am < ar[i]) { am = ar[i] }
    }
    int c=0;
    for(int i= 0; i< N; i++){
        if(am == ar[i]) { c=c+1; }
    }
    return N-c
}
```

→ N → N

TC → O(N)

SC → O(1)

TODO: Try it solve above question by iterating on array only 1 time

↳ Next Session, doubt's session part we will discuss

28) Given N array elements, check if there exists a pair (i, j) such that $ar[i] + ar[j] == k$ && $i != j$

→ boolean

→ { Indices cannot be same but elements can be same }

Note: i & j are index value, k is given sum

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

k = 10 :

i	j
3	5

$\Rightarrow ar[3] + ar[5] = 10$: return True

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

k = 5 : No pair, return false

All ideas

1) All pairs → O(N²)
2) optimized all pairs → O(N²)
3) Hashmap → O(N) → future
4) Sorting + BS → O(NlogN)
5) Sorting + 2 Pointer → O(NlogN)

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

k = 8 :

i	j
1	1

→ $ar[1] + ar[1] = 8$, wrong $i != j$

return false

	0	1	2	3	4
ar[ ] :	3	5	2	7	3

k = 6

i	j	→ $ar[i] + ar[j]$
0	4	→ $ar[0] + ar[4]$ → 6 == Return True

Ideal: → TC: O(N²) SC: O(1)

```
bool Sum(int ar[], int k){
    int N = ar.length;
    for(i=0; i<N; i++){
        for(j=0; j<N; j++){
            if(i!=j &&
               ar[i]+ar[j]==k){
                return True
            }
        }
    }
    return False
}
```

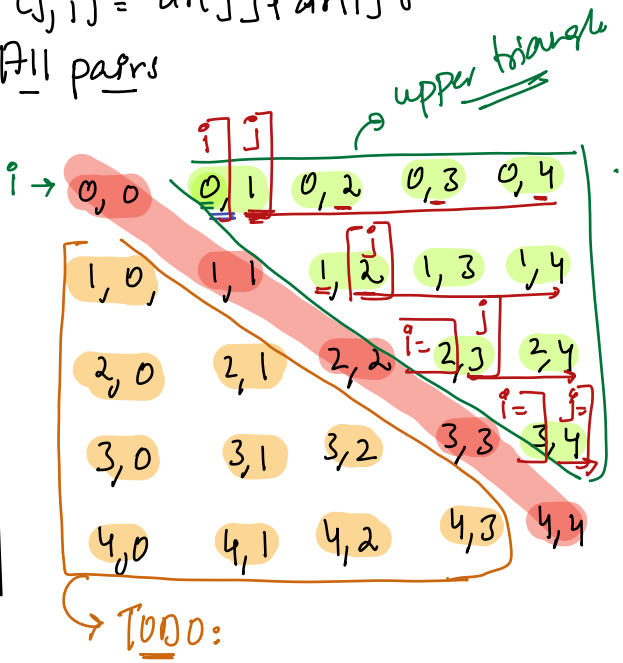
Idea: For every pair get sum == k

N=5 → Index: {0, 1, 2, 3, 4}

(i, j) = ar[i]+ar[j] } same

(j, i) = ar[j]+ar[i] }

All pairs



Idea2: → TC: O(N²) SC: O(1) → 2nd approach optimized approach!

```
bool checkSum(int ar[], k){
    int N = ar.length
    for(i=0; i<N; i++){
        // last i = N-1, j=N, it wont go inside j loop
        for(j=i+1; j<n; j++){
            if(ar[i]+ar[j]==k){
                return True
            }
        }
    }
    return False
}
```

Calculate Iterations

i	j : [i+1, N-1]	Total Iterations
0	[1, N-1]	N-1
1	[2, N-1]	N-2
2	[3, N-1]	N-3
3	[4, N-1]	N-4
⋮		
N-1	[N, N-1]	0

$$S_N = \frac{(N)(N+1)}{2}$$

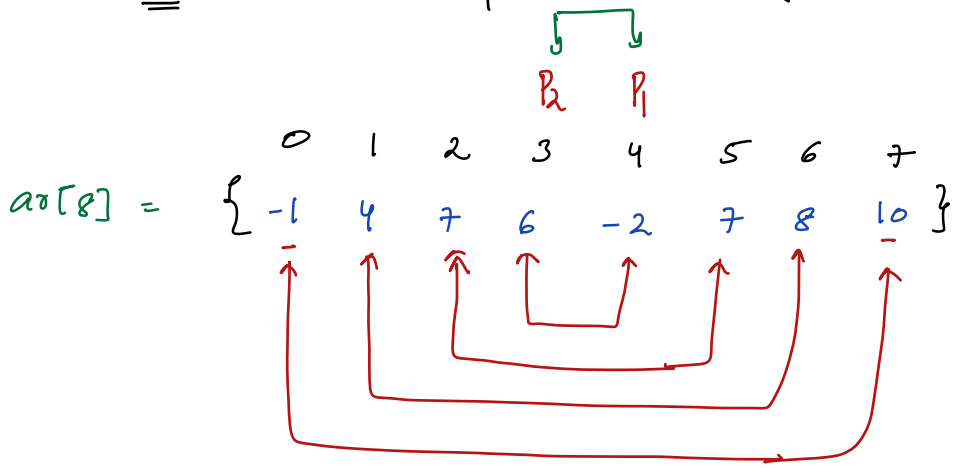
$$S_{N-1} = \frac{(N-1)(N)}{2}$$

Total Iter = Sum of N-1 Natural numbers

(N-1) + (N-2) + (N-3) + ... 2 + 1

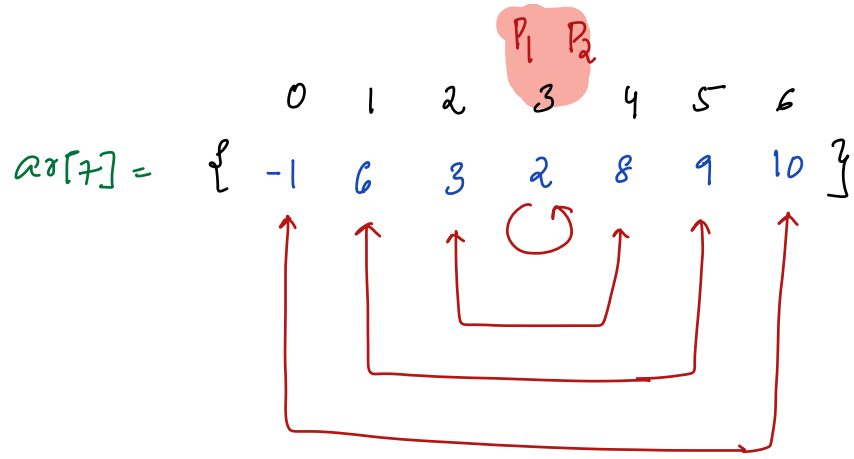
3(8) Given an array, Reverse entire arr[] } Expected SC: O(1)

Note: arr[] itself should change



P_1	P_2	
0	7	Swap
1	6	Swap
2	5	Swap
3	4	Swap
4	3	break

$P_1 > P_2$: break



P_1	P_2	
0	6	Swap
1	5	Swap
2	4	Swap
3	3	break

$P_1 == P_2$ break

Breaking Condition

→ $P_1 > P_2$ break

→ $P_1 == P_2$ break

} $P_1 >= P_2$ break ⟷ $P_1 < P_2$ continue the swapping process

```
void reverse(int ar[]){
    int p1 = 0, p2 = ar.length - 1
    while(p1 < p2){
        Swap ar[p1], ar[p2]
        p1++; p2--
    }
}
```

Iterations: $N/2 \rightarrow O(N) \rightarrow$ TL

SC: $\rightarrow O(1) \rightarrow$ Independent of input size

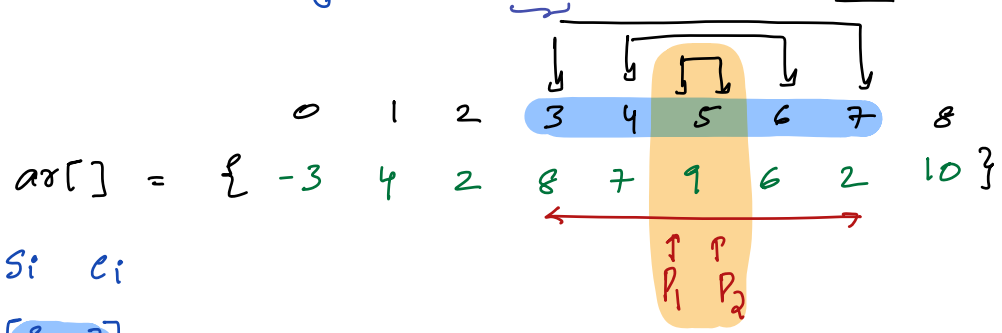
↳ Constant Space

```
int temp = ar[p1]
ar[p1] = ar[p2]
ar[p2] = temp
```

TODO

408) Given N array element & $[s_i \& e_i]$ → start index, → end index

reverse array from $[s_i\ e_i]$ Note $s_i <= e_i$ && index will range



s_i e_i

[3 7]

ar[] = { -3 4 2 2 6 9 7 8 10 }

start index

```
void reversepart(int ar[], int s, int e) {
```

end index

```
    int P1 = s, P2 = e
    while(P1 < P2) {
        swap(ar[P1], ar[P2])
        P1++, P2--
    }
}
```

→ TC: O(N)

SC: O(1)

TODO: using for loop

{ How to do it using for loop, Stay back & discuss with me }

58) Given N array elements, Rotate array from last to first by k times SC: O(1), k ≥ 0 → {google/amazon}

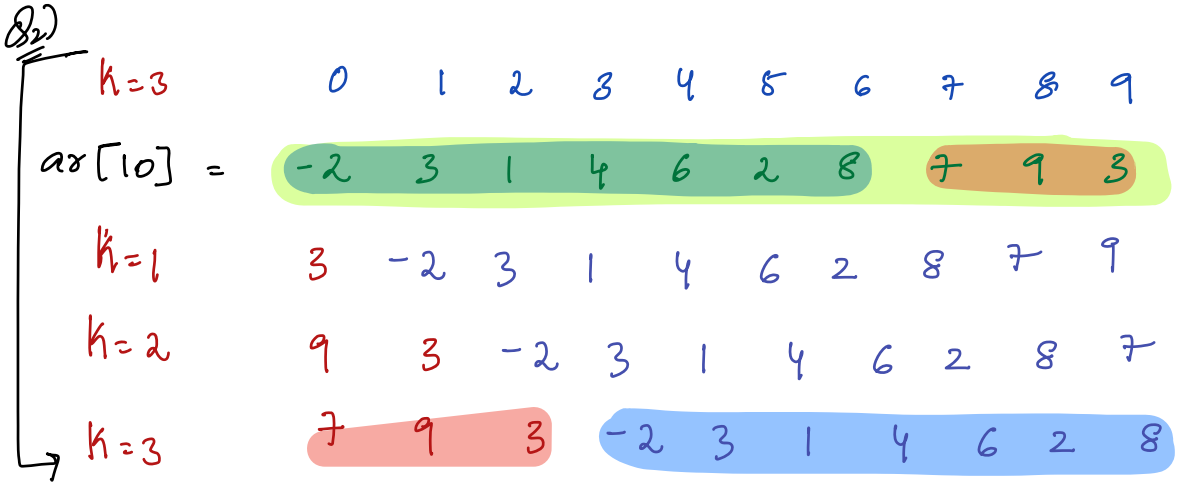
k = 3

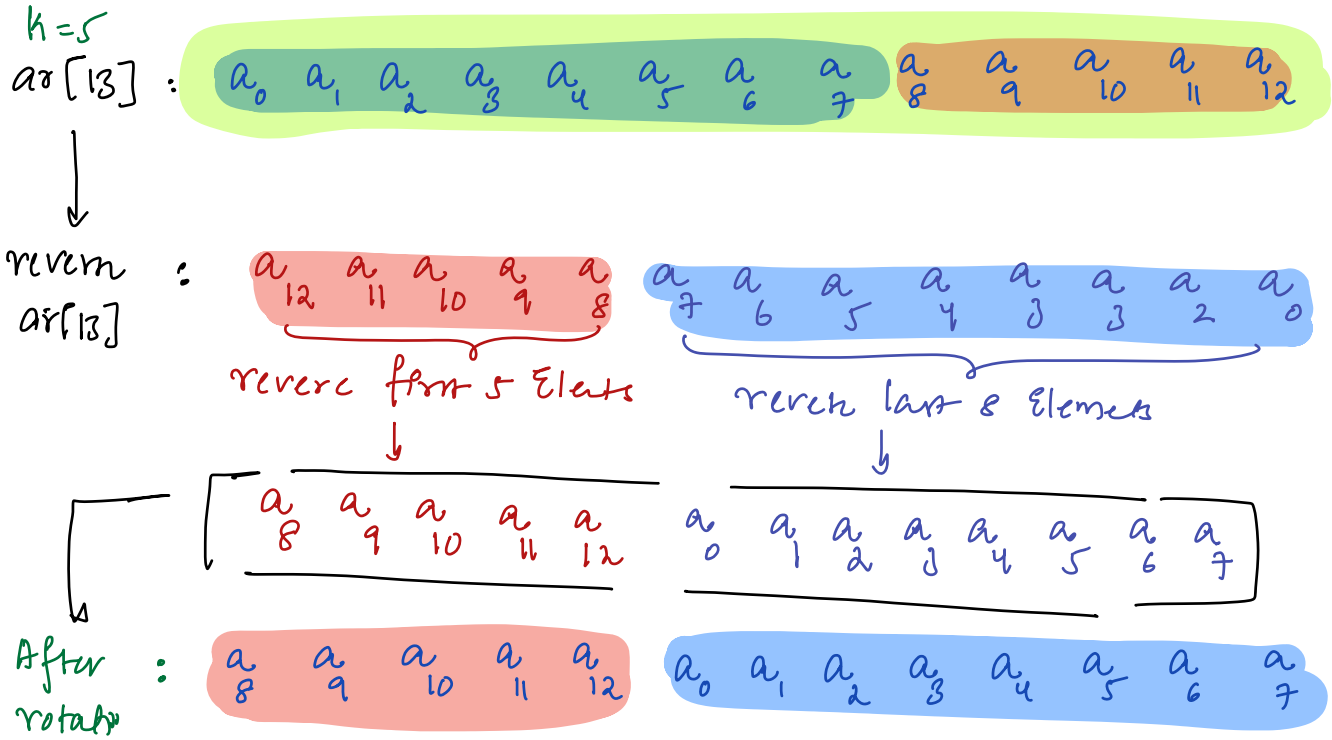
	0	1	2	3	4	5	6
ar[7] =	3	-2	1	4	6	9	8
k = 1 :	8	3	-2	1	4	6	9
k = 2 :	9	8	3	-2	1	4	6
k = 3 :	6	9	8	3	-2	1	4

k = 4

	0	1	2	3	4	5	6	7	8
ar[9] =	4	1	6	9	2	14	7	8	3
k = 1	3	4	1	6	9	2	14	7	8
k = 2	8	3	4	1	6	9	2	14	7
k = 3	7	8	3	4	1	6	9	2	14
k = 4	14	7	8	3	4	1	6	9	2

Q2)





Idea:

Step1: Revern entire array → reverse part (ar, 0, N-1)

Step2: Reven first k Elements → reverse part (ar, 0, k-1)

Step3: Reven last N-k Elements → reven part (ar, k, N-1)

→ Code worst work

```
void Rotate times ( int ar[], int k ) {
    int N = ar.length;
    // Reverse entire array
    reverse part (ar, 0, N-1)
    // Reverse first k elements
    reverse part (ar, 0, k-1)
    // Reverse last N-k Elements
    reverse part (ar, k, N-1)
}
```

	Iterations
reverse part (ar, 0, N-1) →	N/2
reverse part (ar, 0, k-1) →	k/2
reverse part (ar, k, N-1) →	(N-k)/2

Total Iterations

$N/2 + k/2 + \frac{N-k}{2}$

$\Rightarrow \frac{N}{2} + \frac{k}{2} + \frac{N}{2} - \frac{k}{2}$

$\Rightarrow$ N Iterations

TC → O(N)

SC → O(1)

Idea: → rotate end to start by 8 times

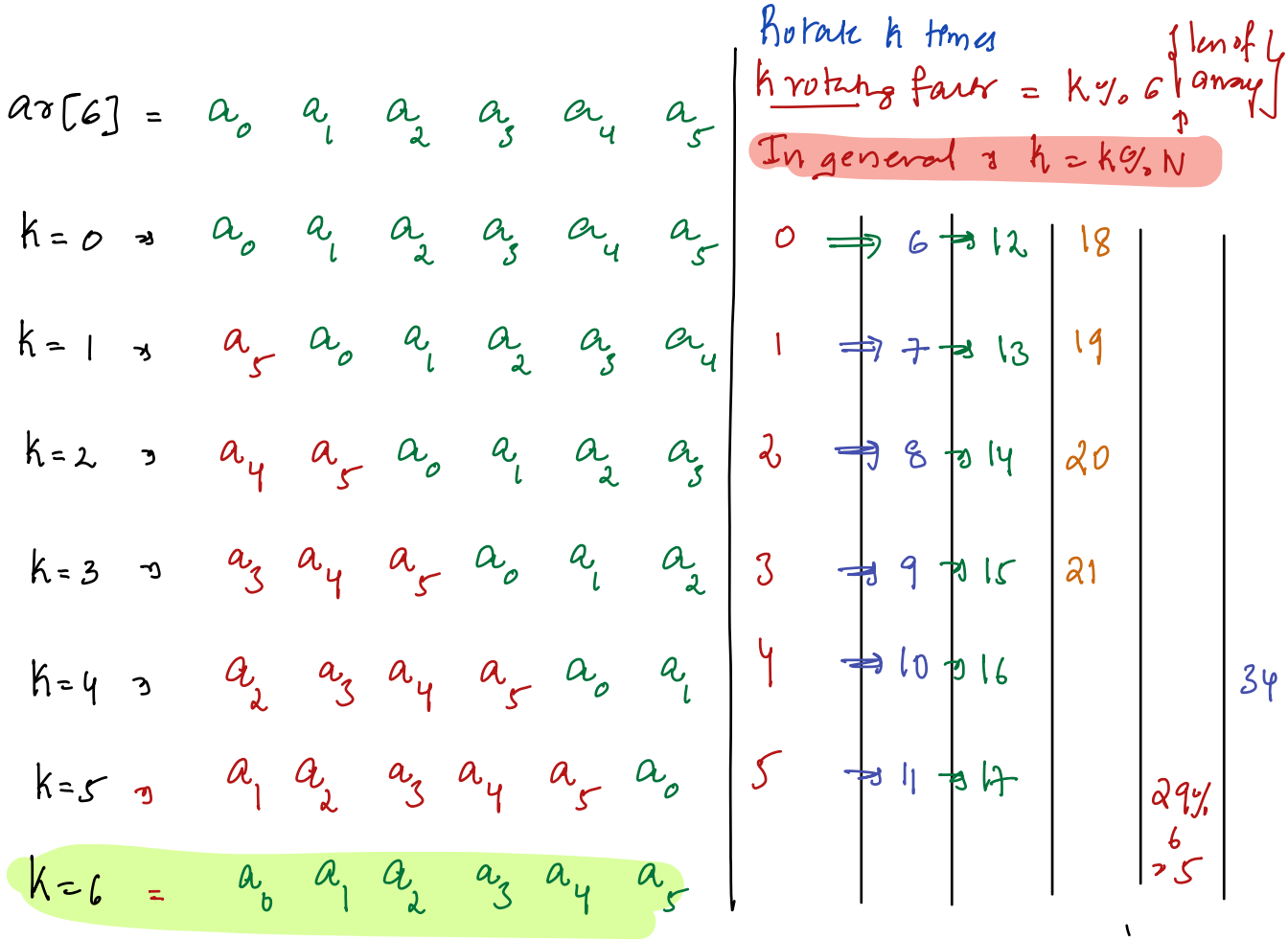
N = 6, k = 8

reverse part (ar, 0, N-1) : reverse part (ar, 0, 5)

reverse part (ar, 0, k-1) : reverse part (ar, 0, 7)

reverse part (ar, k, N-1)

↳ Error: Index out of bounds




```
void Rotate times ( int ar[], int k) {   → Correct code
    int N = ar.length;
    k = k%N
    reverse part (ar, 0, N-1)
    reverse part (ar, 0, k-1)
    reverse part (ar, k, N-1)
}
```

% → remainder

/ → Quotient

2/6 ⇒ 0, 2%6 ⇒ 2

4%6 ⇒ 4

4%10 ⇒ 4

Doubts:

```
void reverse1(int ar[]){
    int P1 = 0, P2 = ar.length-1
    while(P1 < P2){
        Swap ar[P1], ar[P2]
        P1++; P2--
    }
}
```

```
void reverse2(int ar[]){
    int P1 = 0, P2 = ar.length-1
    for(  ; P1 < P2; P1++, P2--){
        Swap ar[P1], ar[P2]
    }
}
```

```
void reverse3(int ar[]){
    int N = ar.length
    for(i = 0; i < N; i++){
        for(j = N; j > 0; j--){
            ≡≡
        }
    }
}
```

ar[6] = a_0 a_1 a_2 a_3 a_4 a_5 | k = k%N

k = 0 → a_0 a_1 a_2 a_3 a_4 a_5 | 0 | 6 = 0 | 12 = 0 | 18 = 0

k = 1 → a_5 a_0 a_1 a_2 a_3 a_4 | 1 | 7 = 1 | 13 = 1 | 19 = 1

k = 2 → a_4 a_5 a_0 a_1 a_2 a_3 | 2 | 8 = 2 | 14 = 2 | 20 = 2

k = 3 → a_3 a_4 a_5 a_0 a_1 a_2 | 3 | 9 = 3 | 15 = 3 | 21 = 3

k = 4 → a_2 a_3 a_4 a_5 a_0 a_1 | 4 | 10 = 4 | 16 = 4 | 22 = 4

k = 5 → a_1 a_2 a_3 a_4 a_5 a_0 | 5 | 11 = 5 | 17 = 5 | 23 = 5

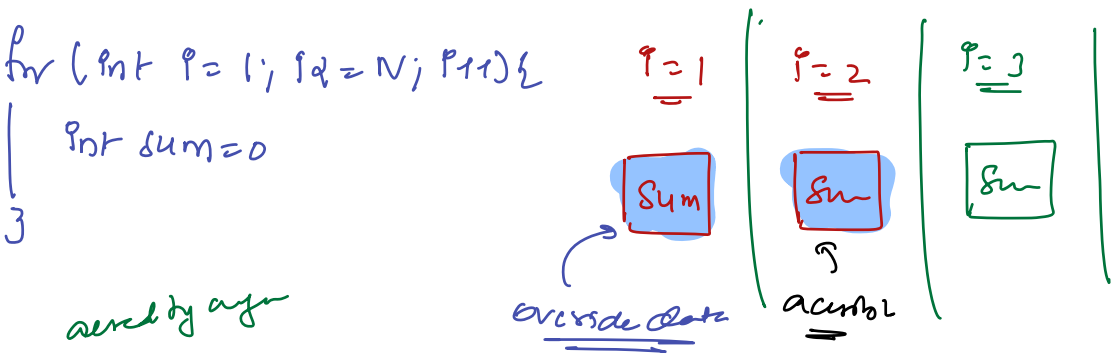
k = 6 = a_0 a_1 a_2 a_3 a_4 a

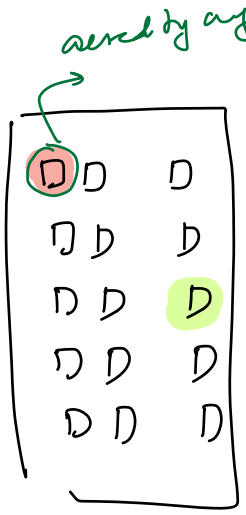
Befor solving

1) Read Node
2) Get all conceptual doubts clear
3) Solve assignment
4) Solve homework → 40

- → TA ⇒ for asking for hints
- → ask batch mates → for hints
- → & In Doubt session → hints

```
//
for (int i = 1; i <= N; i++){
    int sum = 0
}
```





$i = 3;\ i <= N;\ i = i + 3 \rightarrow N/3$

$i = 1;\ i <= N;\ i = i + 3 \rightarrow N/3$

$i = 2;\ i <= N;\ i = i * 2 \rightarrow N/2$

$i = N;\ i >= 1;\ i = i/2 \rightarrow \log_2 N$

$i = N;\ i >= 1;\ i = i/3 \rightarrow \log_3 N$