* Space complexity or Auxilary Space :-

1) Auxilary space :- It is the extra space or temporary space taken by an algorithm.

.: Space complexity = input spake + Auxilary space.

o Suppose :-
Take an i/p of array ~~siz~~ size N & do somthin with it.
So the ~~sap~~ Space complexity will be the input you're taking from the size N + extra space the Algo is using
This is known as Space complexity.

- So, In the binary search the space complexity was constant. So, that means auxilary space was constant it was not taking any extra space
- Taking 3 variable i.e Start, End & Mid.
- If the array is of size 100 or more than that. Every single time its only going to take ~~3~~ that variable i.e start, end & mid.

Hence, constant

Q

```
for( i = 1 ; i ≤ N ; ' ) {
    for( j = 1 ; j ≤ k ; j++) {
            // some operations that
               takes time at t
    }
    i = i+k
}
```

Here,

1) inner loop is running k times & for every time it's running it's taking t amout of time.

∴ O(kt) time.

2) If inner loop is running ~~onec~~ once, so it's taking t amount of time. So, here it's actully running k times. Hence kt.

3) Ans :- O(kt * times outer loop is running)

conditions :- ~~wex~~ i will starts from 1, & the loop will break when i is ≤ N & i is incrementing with k

4) So let's say, i=1, 1+k, 1+2k, 1+3k ..... 1+xk

So, if the last value is "xk that means is & satisfy the condtions. Hence x should be ≤ N

1 + xk is the value & x is the no. of times its running

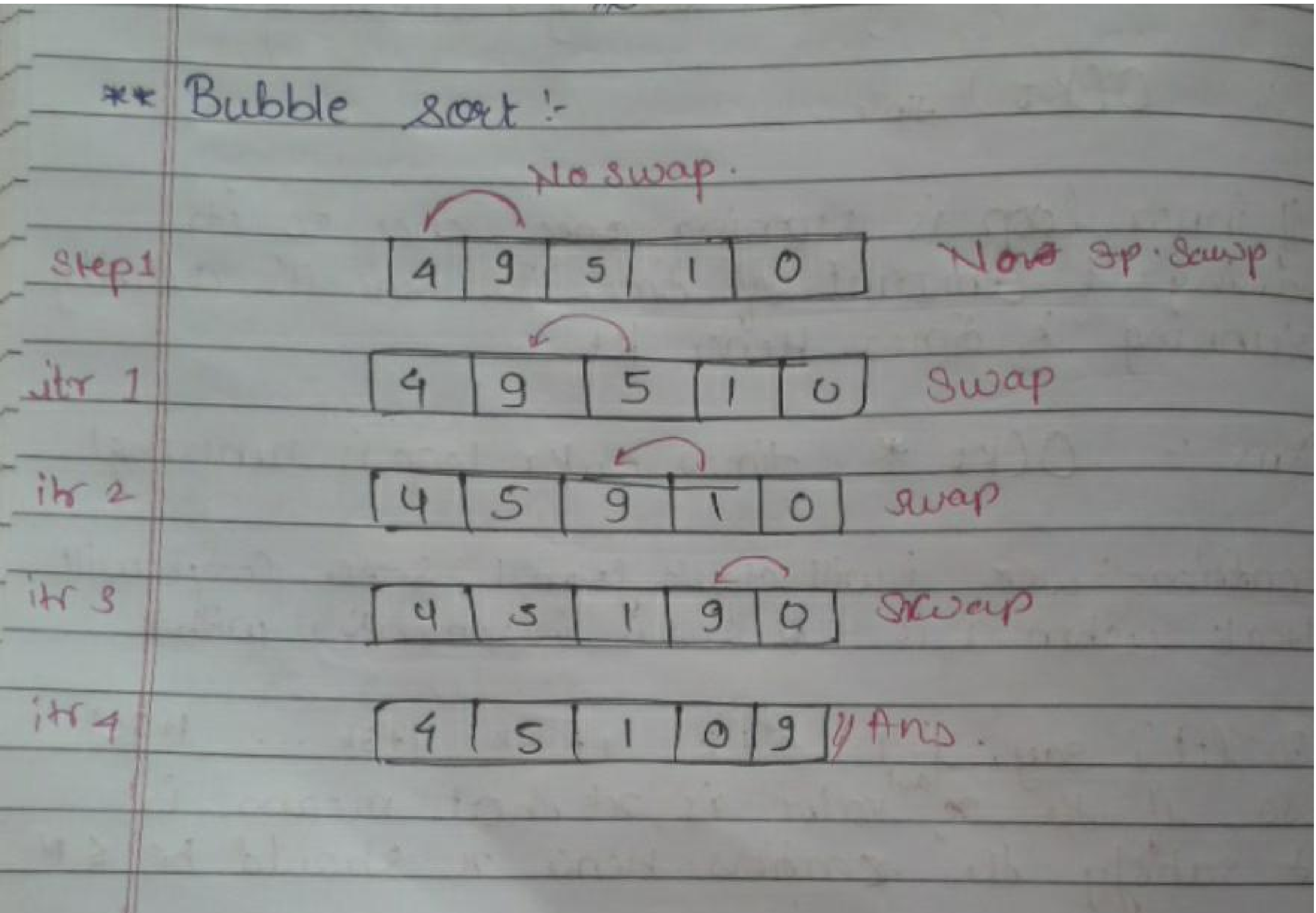
$\therefore \quad 1 + xk \leq N$

$xk \leq N-1$

$\therefore \quad x = \frac{N-k}{k}.$ - { x = no of times the outer loop is running }

Complexity :- $O\left(\cancel{k}t * \frac{(N-1)}{\cancel{k}}\right)$ - { const are removed }

$\therefore = O(N * t)$



** Bubble sort :-

No swap.

step1	4	9	5	1	0	Now 3p. Swap
itr 1	4	9	5	1	0	Swap
itr 2	4	5	9	1	0	swap
itr 3	4	5	1	9	0	swap
itr 4	4	5	1	0	9	// Ans.

* worst & Average case time complexity :-
O(n * n)
Worst case occurs when array is reverse sorted.

* Best case time complexity :-
O(n)
Best case occurs when array is already sorted

* Auxilary space
O(1)

* Boundary cases :-
Bubble sort takes minimum time (order of n) when elements are already sorted.

* Sorting in place :- Yes

* Stable :- Yes

** Selection Sort :-

Worst complexity : n^2

Average complexity :- n^2

Best complexity :- n^2

Space complexity :- 1

Method :- selection

stable : No

Note:- The good thing about selection sort is it never makes more than O(n) swaps & can be useful when memory write is a costly operation.

Swap

Eg:- (max) 4 , 5 , 1 , 2 , 3

swap

4 , 3 , 1 , 2 , 5

swap

2 , 3 , 1 , 4 , 5

Swap

2 , 1 , 3 , 4 , 5

1 , 2 , 3 , 4 , 5.

** Insertion sort :-

Time complexity :- O(n * 2)

Auxiliary Space :- O(1)

Boundary Cases :- Insertion sort takes a maximum time to sort if element are sorted in reverse order. And it takes minimum time (Order of n) when elements are already sorted.

Sorting in Place :- Yes.

Eg :-

$\overset{\text{swap}}{\underbrace{5 \;,\; 3}} \;,\; 4 \;,\; 1 \;,\; 2$

$\overset{\text{sort \& swap}}{\underbrace{3 \;,\; 5 \;,\; 4}} \;,\; 1 \;,\; 2$

$\overset{\text{sort}}{\underbrace{3 \;,\; 4 \;,\; 5 \;,\; 1}} \;,\; 2$

$\overset{\text{sort}}{\underbrace{1 \;,\; 3 \;,\; 4 \;,\; 5 \;,\; 2}}$

1 , 2 , 3 , 4 , 5 //