Time Complexity & Array

Time complexity →

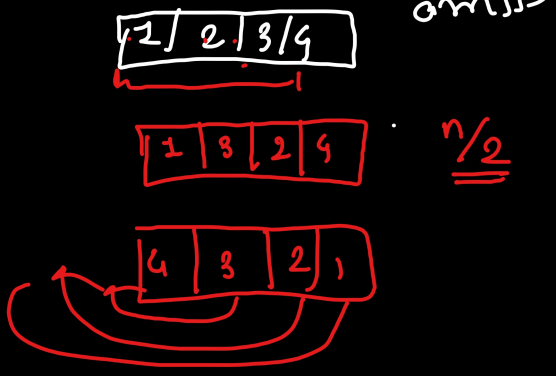
ignore { Best case - O(n)
Average case - O(n*n/2) O(n²)
Worst case - O(n²)

O(n*n) - O(n)²

insertion sort.

```
for(i=1; i<n; i++){
    (i-1 → 0){ break → } 1→
}
```

arr[j] > arr[j]



n = 5, 10, 16, 2000, ⟷

O(n)

O(1) ← Best case
O(logn) ←
O(√n) ←
O(n) ←
O(nlogn) ←
O(n√n) ←

} Loops → time

n=5 a=0

O(n) for(i=0; i<=n; i++){ 1)
→ a+=i
}

O(1) + O(1) + O(1) + O(1) + O(1) = 5
= 4 O(n)

5 { i=0, i=1, i=2, i=3, i=4, i=5

a=0, a=1, a=3, a=6, a=10, a=15

- i ⟶ n
O(1)+O(1)+O(1)
O(n)

2)
```
for(i=0; i<=n; i++){ ⟶ n        n=3
   for(j=0; j<=m; j++){ }⟶ m    m=4
   }
}
```
n*n

(n*m) O(n*n) = O(n²)

12

```
* * * *
* * * *
* * * *
```

3)
```
for(i=0; i<n; i++){ - O(n)
   for(j=0; j<=m; j++){ O(m)
      for(k=0; k<m; k++){ - O(m)
      }
   }
}
```
→ O(n)*O(m)*O(m)

O(n*m²) O(3)

4)
```
main() {
   for(i=0; i<n; i++){ - O(n) →
   }
   for(i=0; i<n; i++){ - O(n) →
   }
}
```
O(n) + O(n)

↳ O(n)

5)
```
{ for(i=0; i<n; i++){ }  20
    for(j=0; j<m; j++){ }  10 - O(m*n)
  }
  for(j=0; j<20; j++){ } 10 → O(20)
}
```
m=10, n=20

20*10 = 200 + 10

210

+ O(m*n) + O(20)

O(m*n) → overall T.C

210

199 + (1)

200

for(i=0; i<sqrt(n); i++) (√n)

ArrayList ⇒

→ same as array

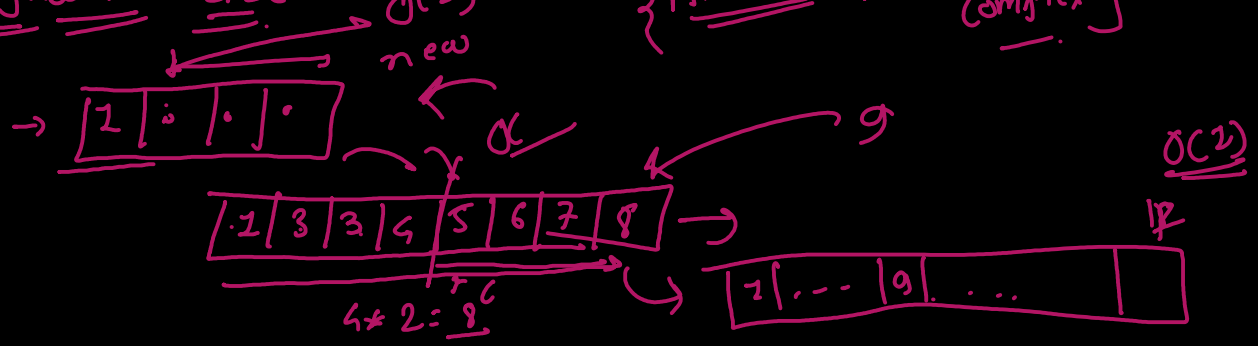
→ Dynamic.

→ slow as compare to arrays

→ n → arr[] int[] -

dynamic size → O(1)

{Amortized Time Complexity O(n)



1) Bubble Sort
2) Selectn ~
3) Insertn ~