# Bubble sort

i th round → i th largest element at right place

| i = 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 10 | 1 | 7 | 6 | 14 | 9 |

## Round 1

1st largest element → sort → placed at end

1 10 7 6 14 9

1 7 10 6 14 9

1 7 6 10 14 9

1 7 6 10 14 9

1 7 6 10 9 14

5 comparison

14 right place

## Round 2

2nd largest element → sort

1 7 6 10 9 14

1 6 7 10 9 14

1 6 7 10 9 14

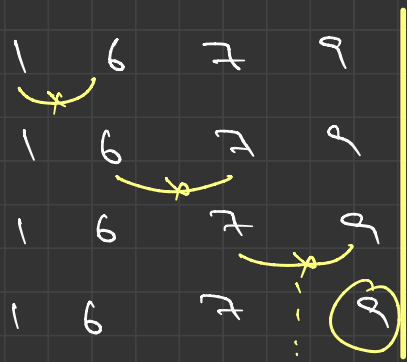
1 6 7 9 10 14

4 comparison

## Round 3

3rd largest element → sort

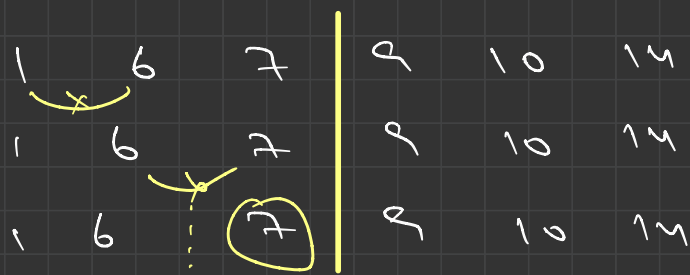
3 comparison




## Round 4

4th largest element → sort

2 comparison




## Round 5

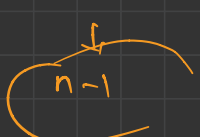
5th largest element → sort

1 comparison


## Round 6

→ single element left

↳ no need → already sorted in 5 rounds




Total comparison = $5+4+3+2+1 = \frac{n(n-1)}{2}$

$O(n^2)$

Entire array is sorted

Condition

if $a < b$ ✗

$a > b$ (swap)

Rounds = $n-1$

---

worst

0 → $< (n-1)$

0 → $< (n-2)$

0 → $< (n-3)$

⋮

best

1 → $< (n-1)$

⋮

Sorted

$O(n)$

Time complexity ⇒ $O(n^2)$

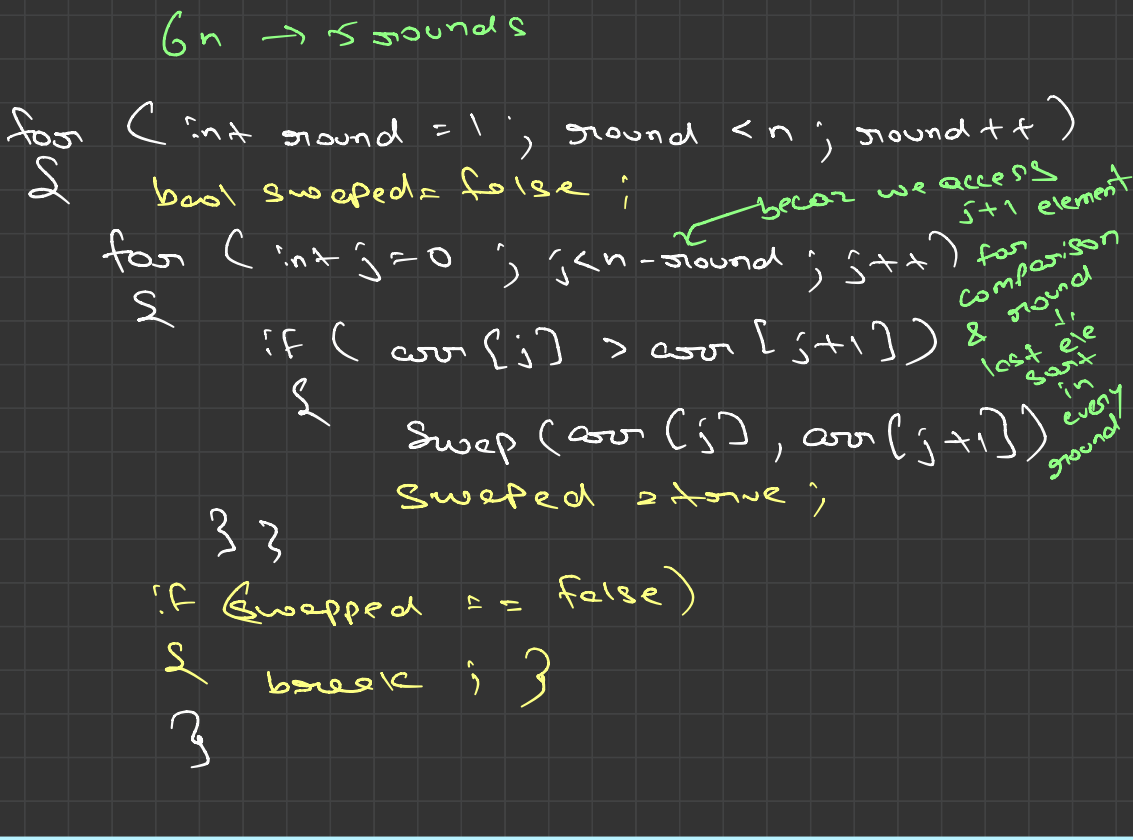
Space complexity ⇒ $O(1)$

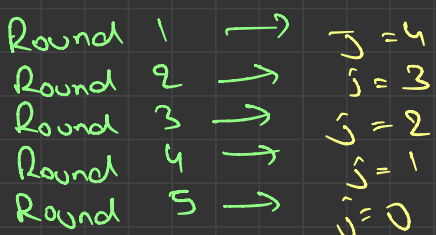
---

optimisation

a b c d e ⇒ no swap

$a < b < c < d < e$ → sorted




```
void bubbleSort(int arr[], int n)
    {
        for(int i = 0; i < n-1; i++){
            for(int j = i+1; j < n; j++){
                if(arr[j] < arr[i]){
                    swap(arr[i],arr[j]);
                }
            }
        }
    }
```