

## ② Bubble Sort

↳ sorting algorithm

consider a list of elements:

[38 | 9 | 29 | 7 | 2 | 15 | 28]

→ If we want to sort this list,

① Start with comparing 1<sup>st</sup> two numbers  
38, 9 as  $38 > 9$  we will swap  
the numbers.

↳ [9 | 38 | 29 | 7 | 2 | 15 | 28]

② Repeat same process for 2<sup>nd</sup> and 3<sup>rd</sup> element.  $38 > 29 \Rightarrow$  swap

[9 | 29 | 38 | 7 | 2 | 15 | 28]

↓

Similarly

[9 | 29 | 7 | 38 | 2 | 15 | 28]

→ comparing 3<sup>rd</sup> & 4<sup>th</sup>

↔

↓

[9 | 29 | 7 | 2 | 38 | 15 | 28]

↔

↓

[9 | 29 | 7 | 2 | 15 | 38 | 28]

↔

↓

[9 | 29 | 7 | 2 | 15 | 28 | 38]

↔

so highest  
number  
goes to  
end.

That's why it is known  
as bubble sort as  
bubble pops up from  
bottom.



Now we have 38 in right position  
we will repeat same process.

9	29	7	2	15	28	38
---	----	---	---	----	----	----

→ No swap as  
9 is at good  
position.



9	7	29	2	15	28	38
---	---	----	---	----	----	----

→ swap 7, 29



9	7	2	29	15	28	38
---	---	---	----	----	----	----

→ swap 2, 29



9	7	2	15	29	28	38
---	---	---	----	----	----	----



9	7	2	15	28	29	38
---	---	---	----	----	----	----



No more swap as  
29 has reach  
at its position.

⇒ We will keep on doing this for every  
element of list until we get.

2	7	9	15	28	29	38
---	---	---	----	----	----	----

⇒ So we will use 2 for loops → one for  
getting each number to end and  
one for comparing that number with  
every other number -

↳ so Time Complexity:  $O(n^2)$

Space Complexity:  $O(1)$

↳ Because we  
are not using  
extra space  
we are just  
swapping in  
same array.