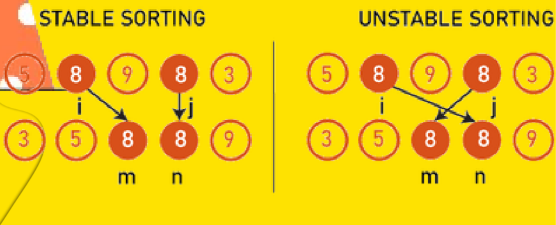
Sorting algorithm

Understanding Bubble Sort-2



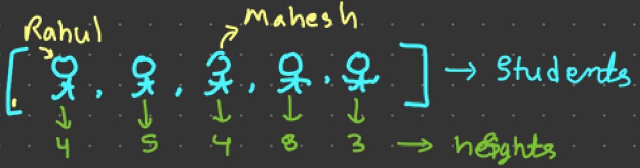
@KRISHAN KUMAR

|| follow ||
for more update

Stable Sort & Unstable Sort

Stable Sort → if you have two elements with equal keys, and one comes before the other in the input, a stable sort ensures that the element that was initially before the other will be before it in the sorted output.

ex →

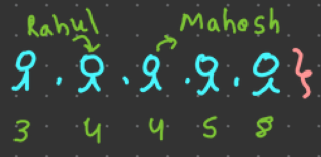


Problem

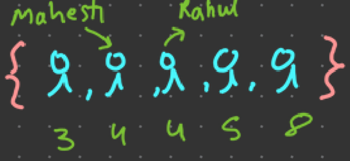
The class teacher said range all the students based on their height

↳ for this problem their could be two posibilities :-

① ⇒



② ⇒



Both are Sorted

⇓

Stable Sorting

before sorting →

Rahul stands before mahesh

After sorting →

Still Rahul Standing before mahesh

⇓

This called as → Stable Sorting

⇓

UnStable Sorting

Before sorting →

Rahul Stands before mahesh

After sorting →

Rahul Standing after mahesh

This change called as unstable Sorting

Real world use case of stable & unstable sorting

Fear mongering → Spreading fake news.

Whatsapp → Spreading fake news that Rohit Sharma Retired

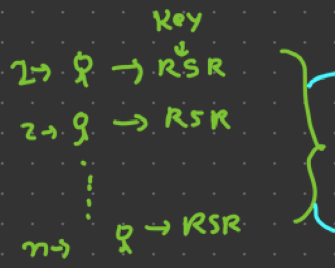
↓

Now people fowarding news.

↓

In all massages → Same keys → Rohit Sharma Retred

↓



→ if i sort in → Unstable sorting {order will be Rearrange}

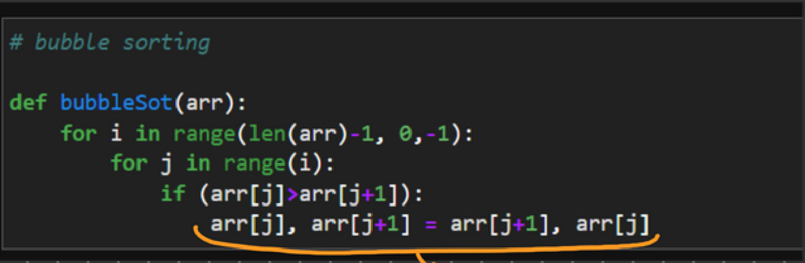
Not a Good aproach

→ if i sort → Stable sorting {order will be the Same}

Right aproach

By Using Stable sorting and making data in arranging we will find the person who actually start this news.

Q This sorting algorithm follow which one



```
# bubble sorting

def bubbleSot(arr):
    for i in range(len(arr)-1, 0,-1):
        for j in range(i):
            if (arr[j]>arr[j+1]):
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

swapping

Ⓐ Stable Sorting

Ⓑ Unstable Sorting

Ans → swapping happens $\xrightarrow{\text{only when}}$ arr(J) > arr[J+1]

So,

if two value are same then we don't swapping here

So → **this algorithm follow → ~~> Stable Sorting.**

Lets understand with an example:- arr → [3, 1', 2, 4, 1]

①

[3, 1', 2, 4, 1]

↳ [1', 3, 2, 4, 1]

↳ [1', 2, 3, 4, 1]

↳ [1', 2, 3, 4, 1]

↳ [1', 2, 3, 1, 4]

②

[1', 2, 3, 1, 4]

↳ [1', 2, 3, 1, 4]

↳ [1', 2, 3, 1, 4]

↳ [1', 2, 1, 3, 4]

③

[1', 2, 1, 3, 4]

↳ [1', 2, 1, 3, 4]

↳ [1', 1, 2, 3, 4]

Before sorting arr → [3, 1', 2, 4, 1]

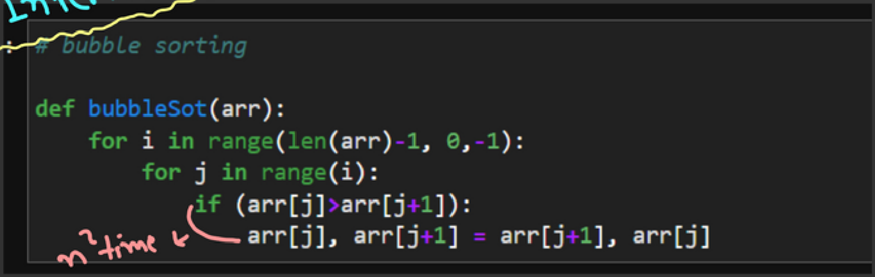
after sorting arr → [1', 1, 2, 3, 4]

↳ as we can see that before sorting [1'] is before [1] and after the sorting [1'] still before [1].

↳ So this is Stable sorting.

{ Becase order doesnt change here. }

Interview Q. 2



```python
# bubble sorting

def bubbleSot(arr):
    for i in range(len(arr)-1, 0,-1):
        for j in range(i):
            if (arr[j]>arr[j+1]):
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

n^2 time

Q Apply bubble sorting on this code where $T(C) \neq O(n^2)$ array should be already sorted.

Ans in this code loop Runs → n^2 time

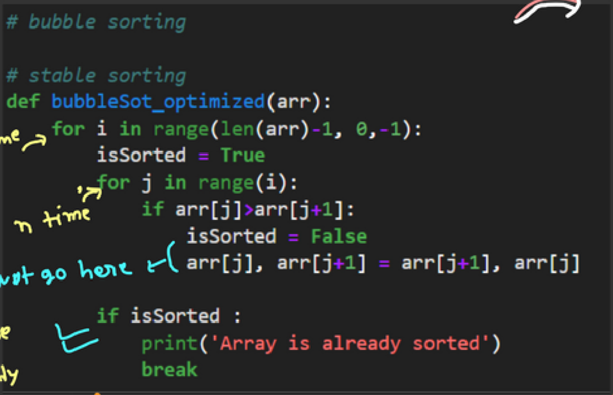
So the $T(C) \Rightarrow n^2$

Because bubble sort not that much smart who catch given array is sorted OR not → He run Both loops,

So,

we need to make some changes in the code in this way where if give sorted array to the code it detect himself the is it sorted OR not. and React differently when got sorted array.

Now the optimized code :



```python
# bubble sorting

# stable sorting
def bubbleSot_optimized(arr):
    for i in range(len(arr)-1, 0,-1):
        isSorted = True
        for j in range(i):
            if arr[j]>arr[j+1]:
                isSorted = False
                arr[j], arr[j+1] = arr[j+1], arr[j]

        if isSorted :
            print('Array is already sorted')
            break
```

① time

n time

not go here

Because already sorted array

Let → [1, 2, 3, 4, 5] j i

isSorted = True

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]

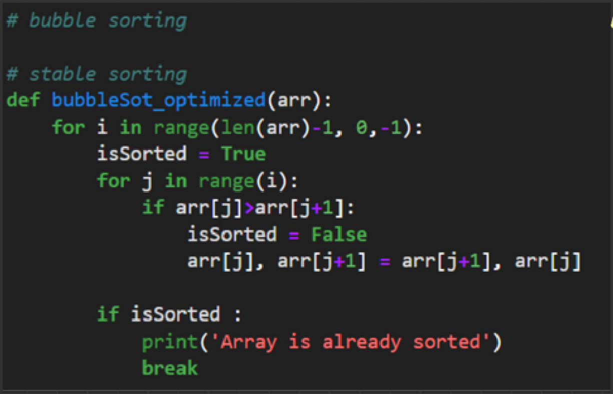
[1, 2, 3, 4, 5]

Now in this case it will not go out from the outer For loop again ~~and~~ print the statement and Terminate there.

in this way I don't have to run till n^2

So the $T(C) \neq O(n^2)$ where $T(C) = O(n \times 1) \Rightarrow \underline{O(n)}$

Q What if we are use nearly sorted array ?



```
# bubble sorting

# stable sorting
def bubbleSot_optimized(arr):
    for i in range(len(arr)-1, 0,-1):
        isSorted = True
        for j in range(i):
            if arr[j]>arr[j+1]:
                isSorted = False
                arr[j], arr[j+1] = arr[j+1], arr[j]

        if isSorted :
            print('Array is already sorted')
            break
```

Let ⟶ [1,2,3,5,4]

Ans ⟶

①

j i

[1,2,3,5,4]

is sorted = True

↪ [1,2,3,5,4]

↪ [1,2,3,5,4]

↪ [1,2,3,5,4]

is sorted = False

[1,2,3,4,5]

then it came to outer loop

②

[1,2,3,4,5]

is sorted = True

↪ [1,2,3,4,5]

↪ [1,2,3,4,5]

↪ [1,2,3,4,5]

↪ [1,2,3,4,5]

⇓

Sorted Fully

Now then loop Break

loop Runs only for } ⟹ 2 times