Is Selection Sort Stable ??

n^2

NO file

↳ The major benefit of Selection sort is very less swaps

→ comparison

→ swaps

temp

sorted →

2, 4, 5, 7', 6, 7'', 10

0 1 2 3 4 5 6

↑ i

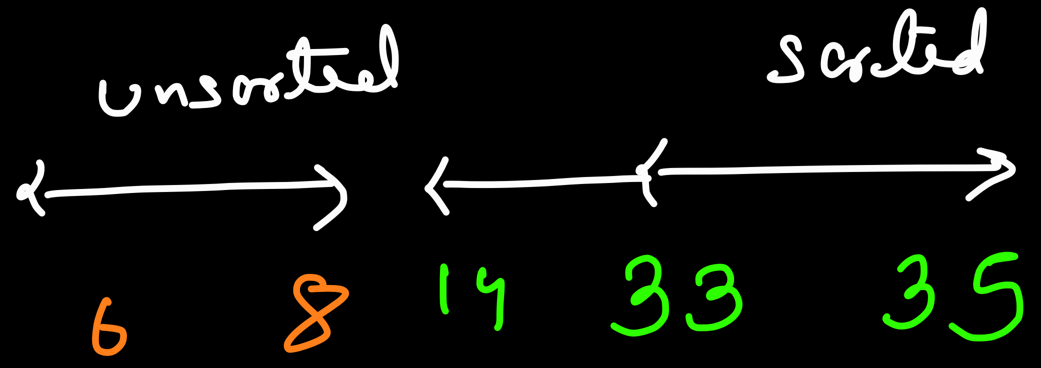
↑ j

min_index = ~~2~~ ~~3~~ 5

Can we make it Stable ??

Usecase→ We can use selection sort when swap operation are heavy , eg → heavy. files

Bubble Sort



1) the date is divided into two parts where 2nd part is Sorted & the first part is unsorted.

2) the max value from unsorted region is lesser than or equal to the least element of sorted region.

3) We cannot immediatily shift the element but insted we have to bubble up.

→ swap

In every iteration bubble up the max element to it's correct position

🟧 unsorted
🟩 sorted

10 , 14 , 27 , 33 , 35

↑j

0	1	2	3	4
2	1	3	4	5

i ↑ j

$i = \cancel{1}\ 2$

n-1 comparisons

n-1 swaps

n-2 compare

n-2 swaps

n-3 comparison

n-3 swaps

Stop → $j < n - i - 1$

$n-1 + n-2 + n-3 \cdots\cdots + 2 + 1$ comp

$n-1 + n-2 + n-3 - - - - - - + 2 + 1$ swap

$$\frac{n(n-1)}{2} + \frac{n(n-1)}{2}$$

$$n(n-1) \approx O(n^2)$$

No. of swaps are high in bubble sort.

i = 0

1 2 3 4 5

↑ j

n-1 comparison

0 swaps

swapped = false

if this variable remains false after one iteration, that means array has been sorted.

Best Case → $\Omega(n)$ → if array is already sorted then bubble sort is very efficient.

Worst Case → $O(n^2)$

Avg → $\underline{O(n^2)}$

→ Inplace ?? → Yes

Stable ?? → Yes

Bad → n^2 swaps

$6 \quad 7' \quad 7'' \quad 4 \quad 8$

$\uparrow$
j

adjacent swap

$a[j] <= a[j+1]$ → don't swap

else → swap

1 2 3

Q → Given an array of integers, return the k^th largest element.

k^th order statistic

$[4, 1, 16, 3, 2, 9]_1$ $k = 2$

$O(n)$

↳ quickselect

ans → 9

$n \le 10^6$

$k \le 10^2$

↳ Sort the array → return k^th element from last

$O(nk)$ 1, 7, 2, 4, 9, 16

↪

We can do K iterations of bubble sort

$\underbrace{\text{iterations of bubble sort}}$

↓

Pushing the biggest element of unsorted region at the last.

↪

Q:- Implement bubble sort without any loops using recursion.

$f(arr, k)$ =

```
for(i=0; i< k ; i++){
    if (arr[i] > arr[i+1])
        swap(i, i+1);
}
```

This function pushes the k^{th} largest element to it's correct sorted pos.

$arr \rightarrow \underline{\underline{[0, k]}}$

$f(arr, n-1)$

$f(arr, n-2)$

$f(arr, n-3)$

$\vdots$

```
function f (arr, k) {
        if(k==0) return;

        for(i=0; i<k ; i++){
                if(arr[i] >arr[i+1])
                        swap (arr, i, i+1);
        }

        f(arr, k-1);

}
```

```
function   f(arr, k, i)  {
        if (k==0)  return;
         if (i==k)  {
              f(arr, k-1, 0);
              return;
        }
        if (arr[i] > arr[i+1]) {
                swap(arr, i, i+1);
         }
        f(arr, k, i+1)

}
```