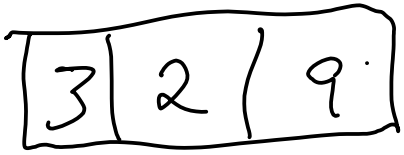
Bubble Sort

```
Function BubbleSort(A):
    For j in 0 to length(A)-2:        # inclusive
        For i in 0 to length(A)-2-j:
            if A[i] > A[i+1]:
                swap(A[i], A[i+1])
```

The swap is $O(1)$; the inner loop is $O(n)$; the outer loop is $O(n)$; the whole function is $O(n^2)$.

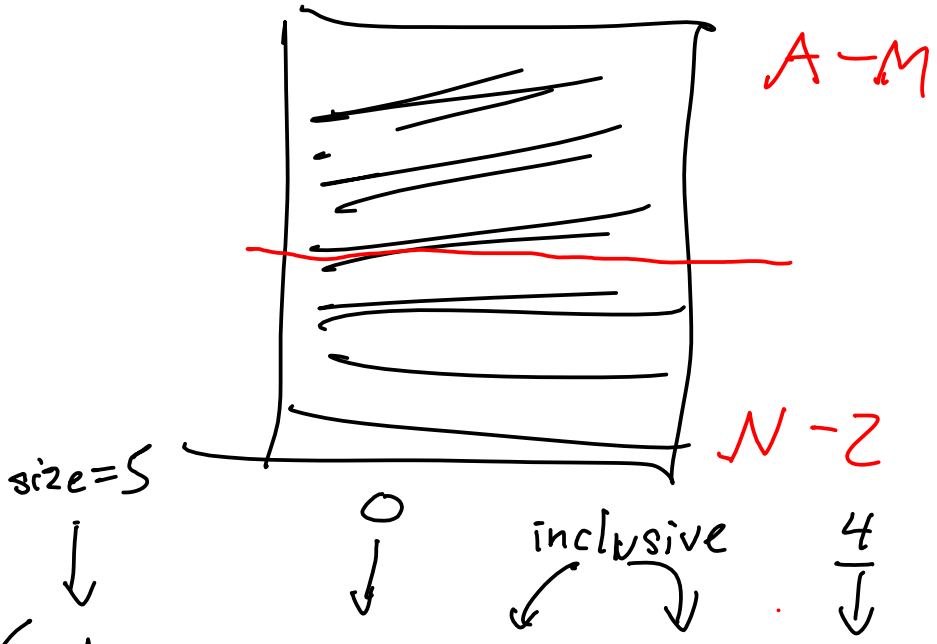
n^2



QuickSort

in place

recursive




```
Function QuickSort (A, startIndex, endIndex)
    If startIndex < endIndex:
        splitIndex = Partition(A, startIndex, endIndex)
        QuickSort (A, startIndex, splitIndex - 1)
        QuickSort (A, splitIndex + 1, endIndex)
```

C.A.R. Hoare

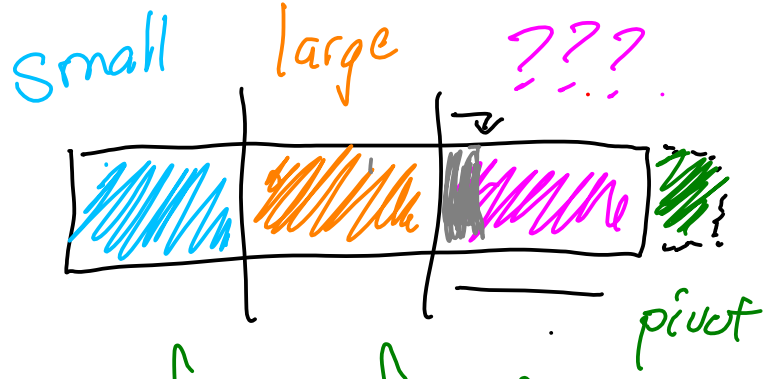
Lomuto

O(n)

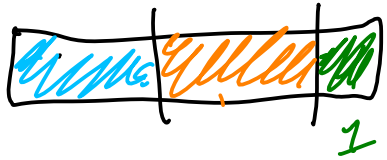
inclusive

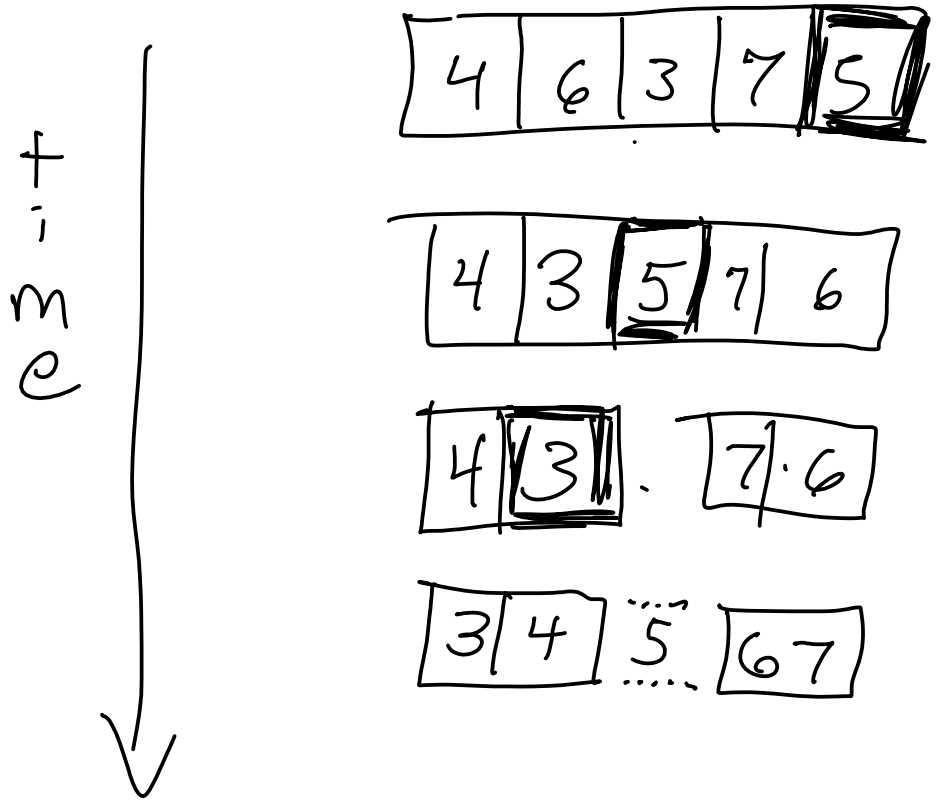
```
Function Partition(A, startIndex, endIndex):
    frontierIndex ← startIndex
    fenceIndex ← startIndex
    pivot ← A[endIndex]
    while frontierIndex < endIndex:
        if A[frontierIndex] < pivot:
            swap(A[frontierIndex], A[fenceIndex])
            fenceIndex ← fenceIndex + 1
        frontierIndex ← frontierIndex + 1
    swap(A[endIndex], A[fenceIndex])
    return fenceIndex
```

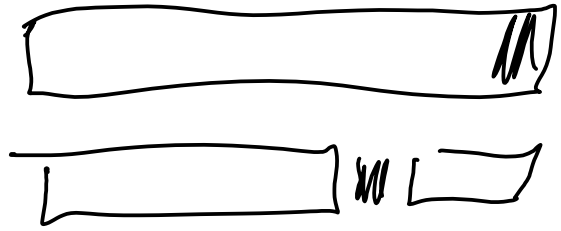
O(n)

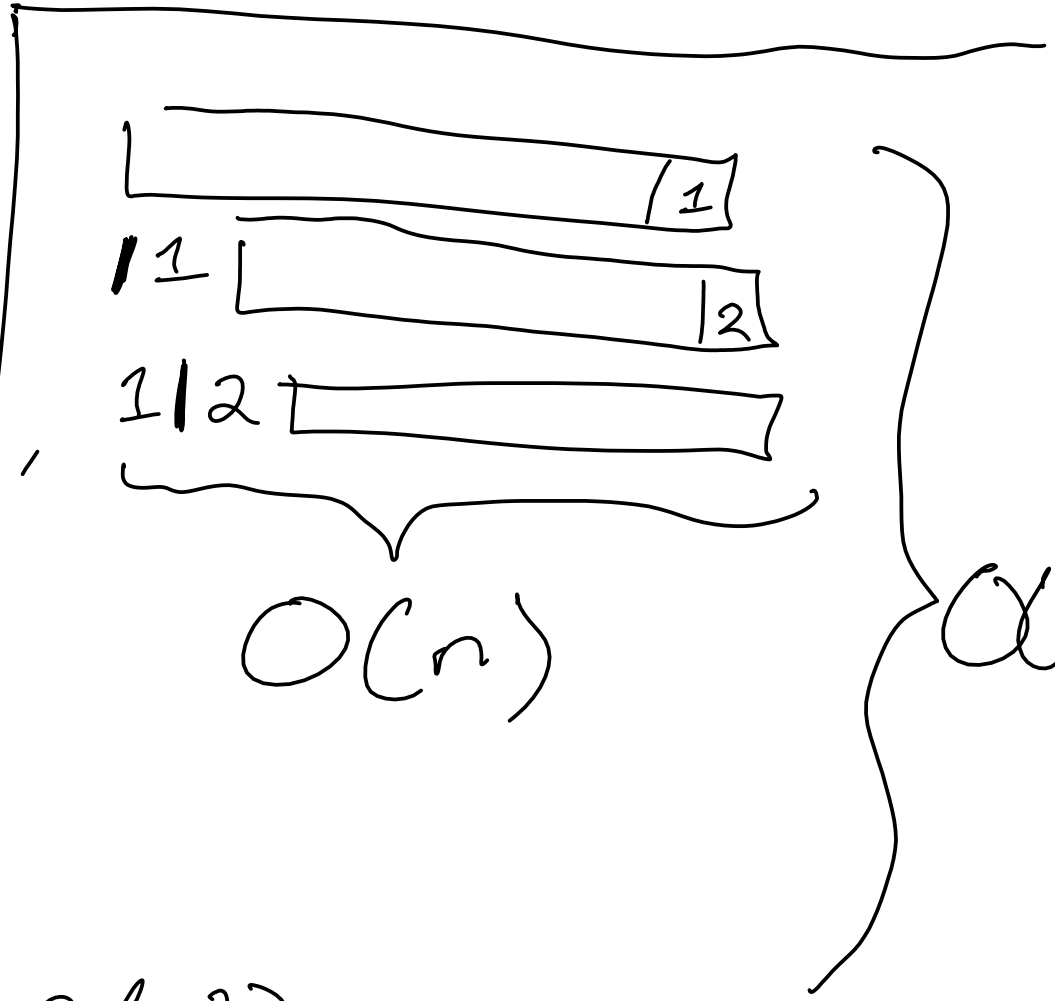


invariant









Wast case: QS is $O(n^2)$

MagicSort(A) $O(1)$

GamblerSort(A):

Flip a coin

While coin is tails:

Flip again

MagicSort(A)

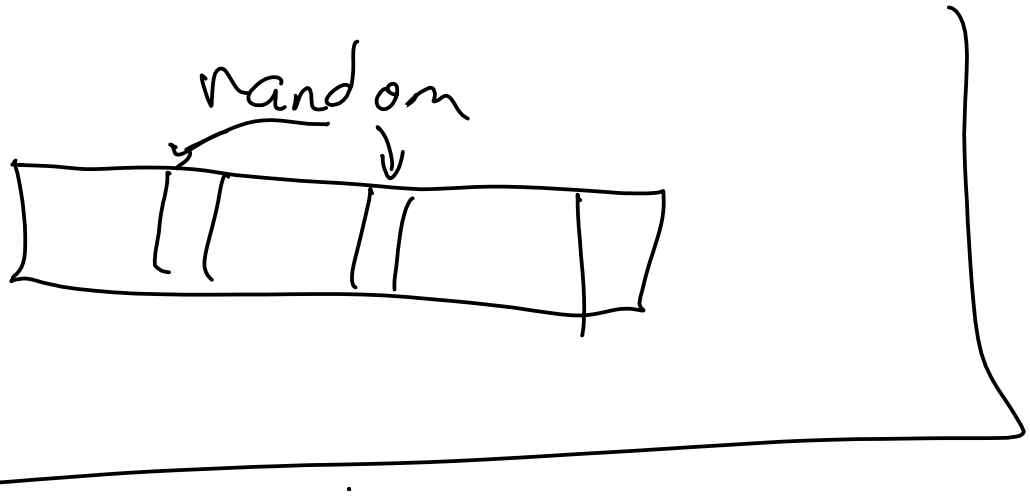
Worst Case: never!

Expected: $O(1)$

Flip	1	2	3	4	...
prob	$1/2$	$1/4$	$1/8$	$1/16$	...

$$1/2 + 2/4 + 3/8 + 4/16 + \dots$$

$$= 2$$



Worst case → everyone is out to get me!
bad data
bad luck

Expected worst case
bad data
normal luck

QS is $O(n \log n)$