Algorithm 1 Partition(A,p,q)

```
x ← A[p]
i ← p
for j ← p + 1 to q do
    if A[j] ≤ x then
        i ← i + 1
        exch A[i] ⟷ A[j]
    end if
end for
exch A[p] ⟷ A[i]
return i
```

Algorithm 2 QuickSort(A,p,q)

```
if p < r then                                   ▷ Boundary Case
    r ← Partition(A, p, q)
    QuickSort(A, p, r − 1)
    QuickSort(A, r + 1, q)
end if
```

The Initial Call: $QuickSort(A, 1, n)$
The Boundary Case:

- if there are zero or one elements, there is nothing to do because the array is sorted either because it is an empty array or it only has one element

Worst Case Running Time:

- Input sorted or reverse sorted
- One side of each partition has no elements

$T(n) = T(0) + T(n-1) + \mathcal{O}(n) =$
$= T(0) + T(n-1) + \mathcal{O}(n)$ ▷ T(0) can be absorbed into O(n)
$= \mathcal{O}(n^2)$ ▷ Arithmetic Series

Best Case Runnig Time:

- If we are lucky, partition splits the array into $\frac{n}{2}, \frac{n}{2}$ every time

$T(n) = T(\frac{n}{2}) + \mathcal{O}(n) = \mathcal{O}(n \log(n))$

Average Case Running Time: $T(n) = \mathcal{O}(n \log(n))$