Quick Sort

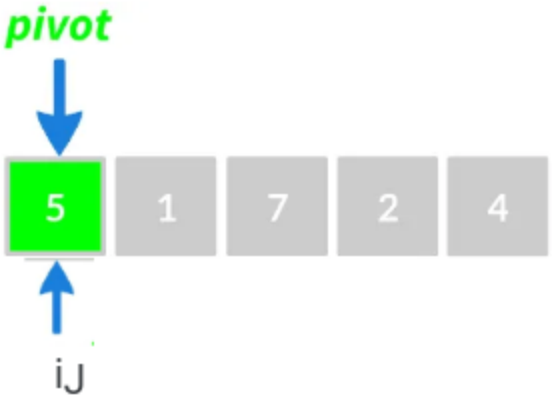
- The `Quick` : Capable of sorting a list of data elements significantly faster than any sorting algorithm
- Completely based on `splitting` of an array `[ PARTITION ]` into smaller ones.
- `Most Efficient` | `Not Stable` | `In-place sorting`
- `Randomised Quick Sort`: Random generator that generates random pivot element among the elements in the list.

Partition Algorithm

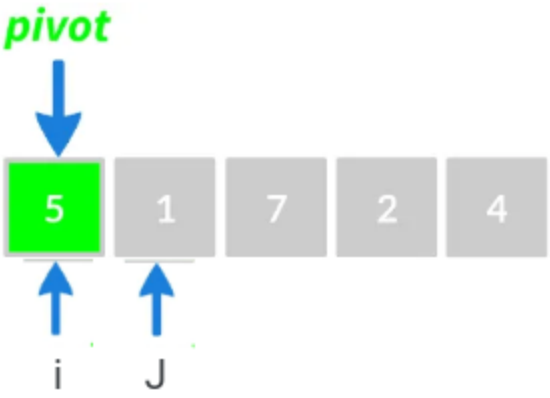
Divide and conquer Methodology

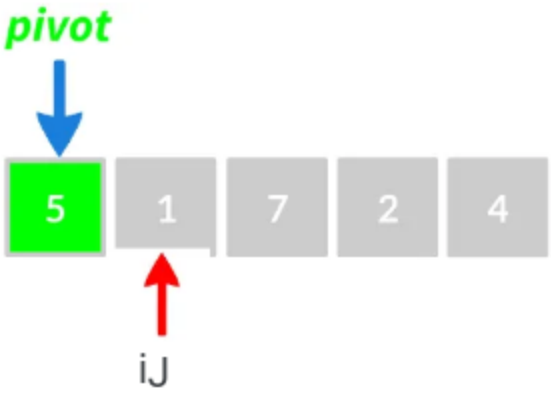
- Objective: Partition `L` into `lower` and `upper` segments with respect to the pivot.
- Choose a `pivot` element.

- Responsibility: If `True` Keep moving
  - `i` : Check for element < Pivot
  - `j` : Check for element > Pivot

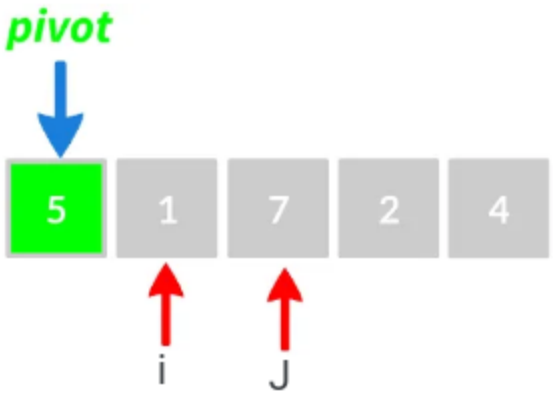


- Initial Position:

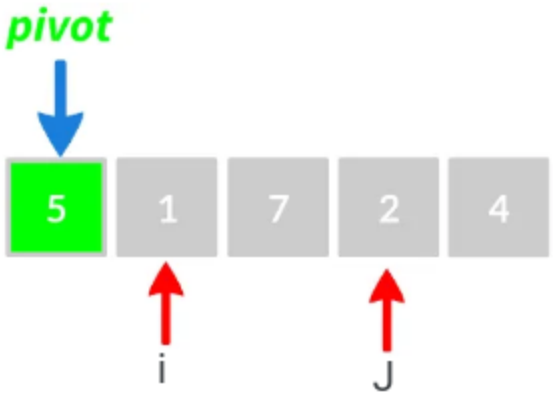




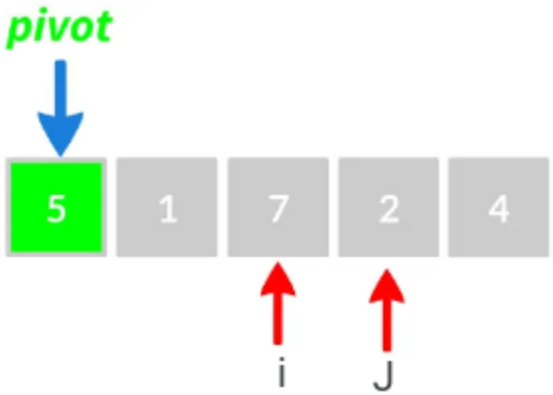
- Move J



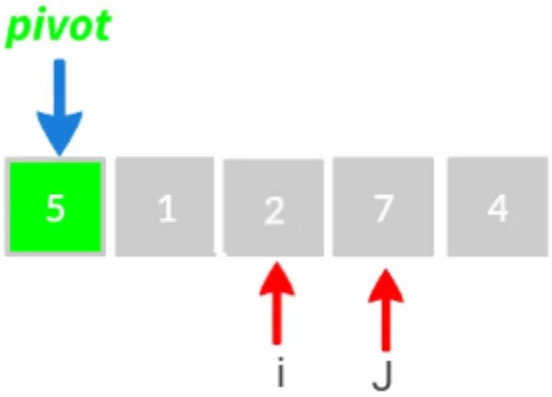
- Move J



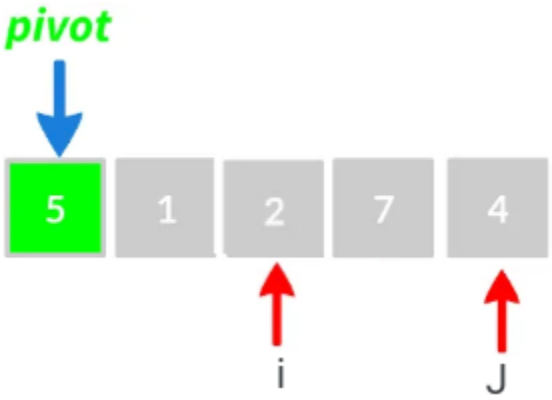
- If `j < pivot` Then `move i`

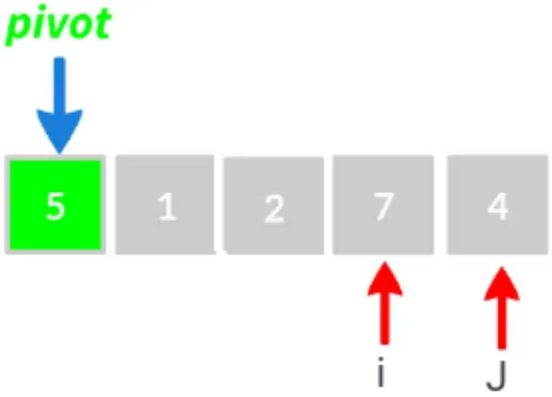


- `Swap i ~ j`

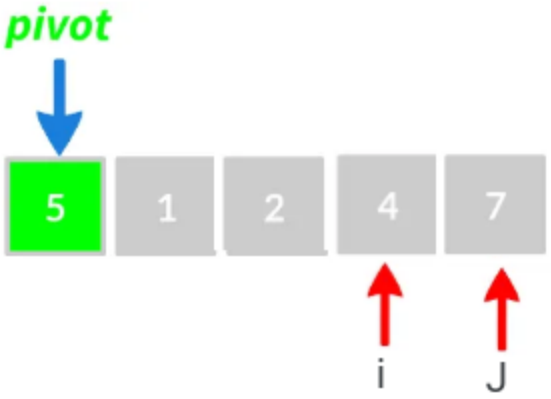


- Again Check j

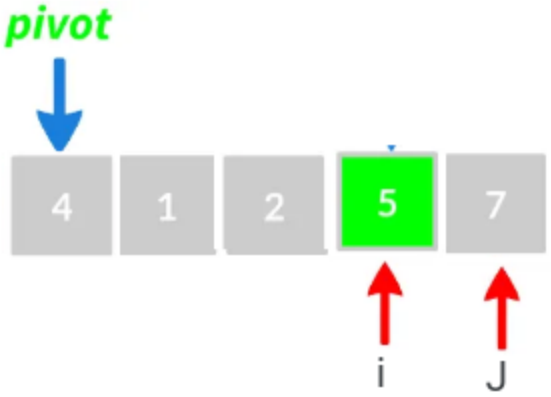




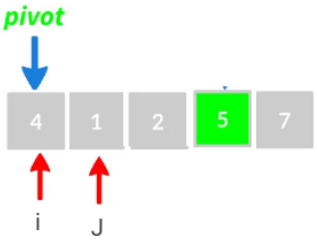
- Swap

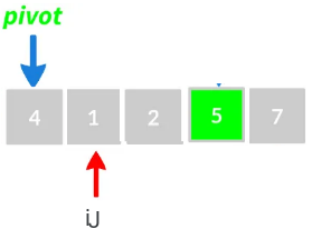


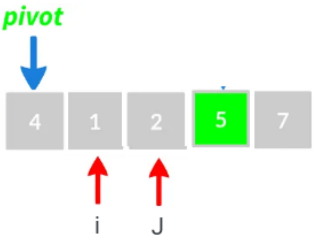
- Swap pivot with i

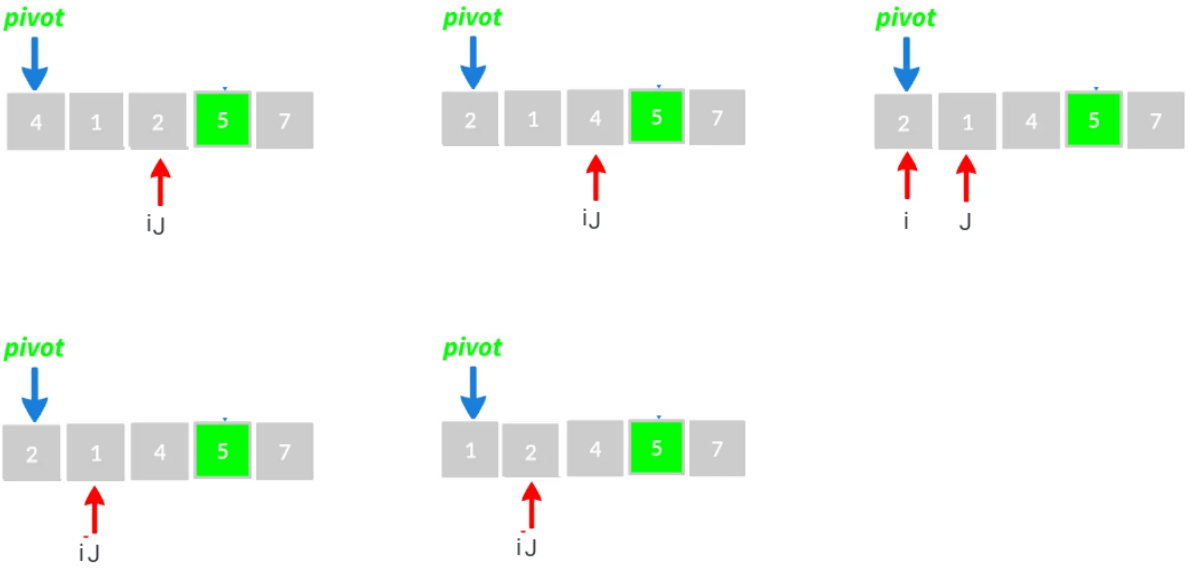


- `Recursively` sort the two partitions. [4 is pivot







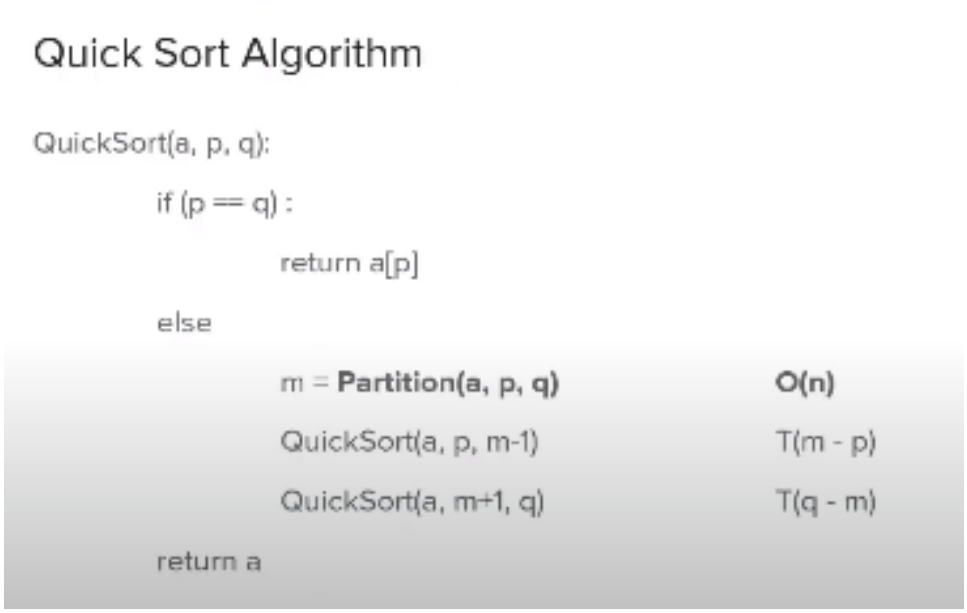


PSEUDO-CODE:

Partition:

```
Def partition(a=list,p= low,q= high):
    x = a[p]  # pivot element
    i = p
    For j in range(i+1,q):

      IF `J > Pivot Element` :

        J + 1

      IF `J ≤ Pivot Element` :

        i + 1

        swap (a[i] and a[J])

    swap(a[p], a[i])

    return i` # returns the index
```

Quick Sort:



m = returns index based on Partition algorithm

why is it unstable?

- Consider three cases:

Ascending	Descending	Similar
1 , 2 , —, n	n,(n-1),—, 1	n,n,n,n,n