Quicksort – Τυχαιοκρατική

✓ – recursive

✓ – pivot : one of the items in the array that meets the following 3) conditions, after we sorted it. :

① Correct position (in final, sorted array)
② Items to the left are smaller.
③ Items to the right are larger.

✓ ΠΧ

2	6	5	3	8	7	1	0

Now let's choose a pivot

2	6	5	3	8	7	1	0

pivot

(ορίζω το μέσο στοιχείο

✓ – Move the pivot to the end of our array :

2	6	5	0	8	7	1	3

(Swap last element with pivot)

✓ – Now look for 2 items :

① itemFromLeft that is larger than pivot.
First item starting from the left

② itemFromRight that is smaller than pivot.
first item starting from the right

✓ ① Starting at 2, we can see that 6 is the first item such that : item > pivot

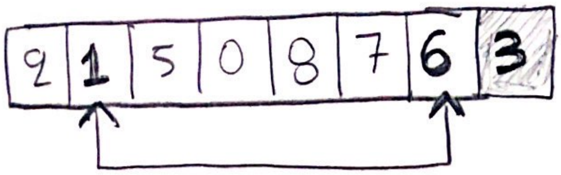
✓ ② Starting at 1, we can see that 1 is the first item such that : item < pivot

2	6	5	0	8	7	1	3

itemFromLeft

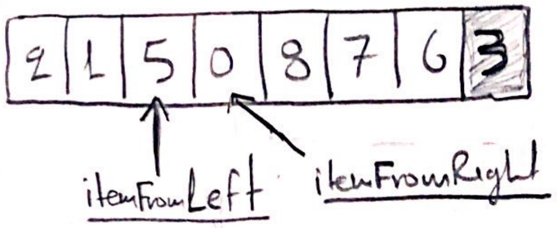
itemFromRight

✓ – Lets swap itemFromLeft, itemFromRight:

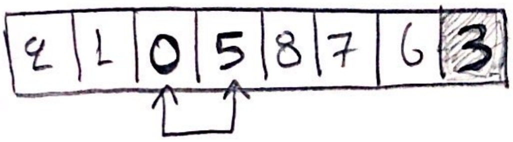


– Repeat the process.

✓ –

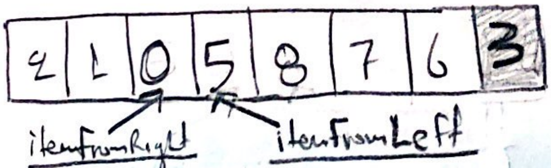


✓ – Let's swap itemFromLeft, itemFromRight:



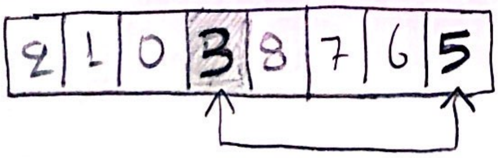
– Repeat the process.

✓ –



✓ – (index itemFromRight) < (index itemFromLeft) ⇒ Stop ****

✓ – Swap itemFromLeft with pivot:



****

Pivot is now in the correct spot:

① Correct position in final sorted array. ✓

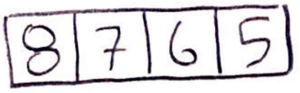
② Items to the left are smaller ✓

③ Items to the right are larger. ✓

1st loop ended

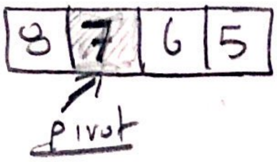
1- Quicksort is recursive.

Choose a larger partition & run through the same process once more



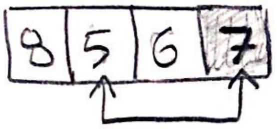
8	7	6	5

} Now let's choose a pivot.



8	7	6	5

✓- Move the pivot to the end of our array:

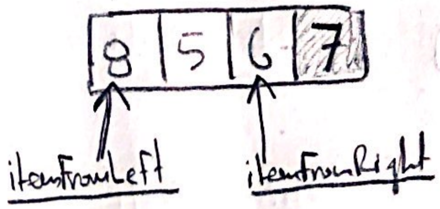


8	5	6	7

✓- Now look for 2 items:

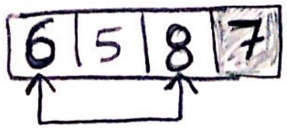
(1) itemFromLeft that is larger than pivot.
First item starting from the left

(2) itemFromRight that is smaller than pivot.
first item starting from the right



8	5	6	7

✓- Let's swap itemFromLeft, itemFromRight:



6	5	8	7

- Repeat the process.

✓ -

6	5	8	7

itemFromRight itemFromLeft

✓ - (index itemFromRight) < (index itemFromLeft) ⟹ Stop

- Swap itemFromLeft with pivot :

6	5	7	8

****

Pivot now in the correct spot :

① Correct position in final sorted array ✓

② Items to the left are smaller ✓

③ Items to the right are larger ✓

2	1	0	3	6	5	7	8

} Now we have 3 7 in the correct position.

⇓

⋮

⇓

- How to choose a pivot ???
  - We want to chose one that "divides the array in half".
  - method : median of 3

•

2	6	5	3	8	7	1	0	4

} Look at the 1st, midle, last element of our array

•

2	6	5	3	4	7	1	0	8

} We sort them properly and chose the midle element as our pivot

```
Quicksort(A, low, high){
    if(low < high){
        pivot_location = Partition(A, low, high)
        Quicksort(A, low, pivot_location)
        Quicksort(A, pivot_location+1, high)
    }
}
```

```
Partition(A, low, high){
    pivot = A[low]
    leftwall = low

    for(i = low+1 to high){
        if(A[i] < pivot){
            swap(A[i], A[leftwall])
            leftwall = leftwall + 1
        }
    }

    swap(pivot, A[leftwall])

    return(leftwall)
}
```

Worst case : $\boxed{O(n^2)}$

Average case : $\boxed{O(n.\log n)}$