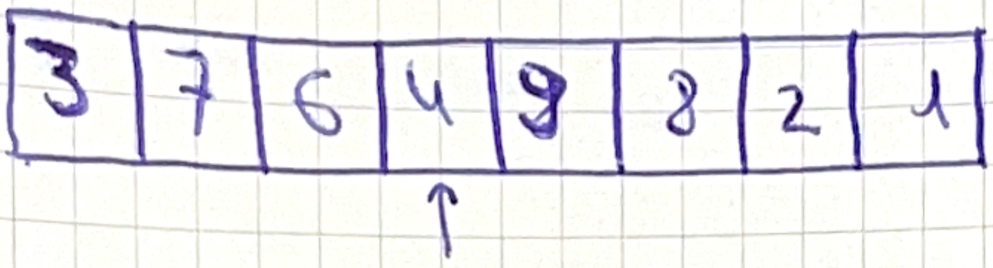
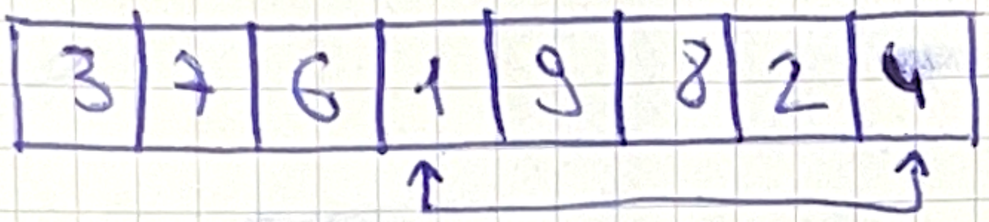
Report:

Quicksort:



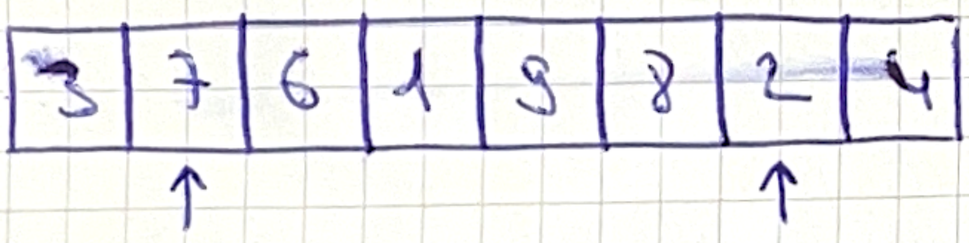
choose pivot

① move pivot to the end of the array

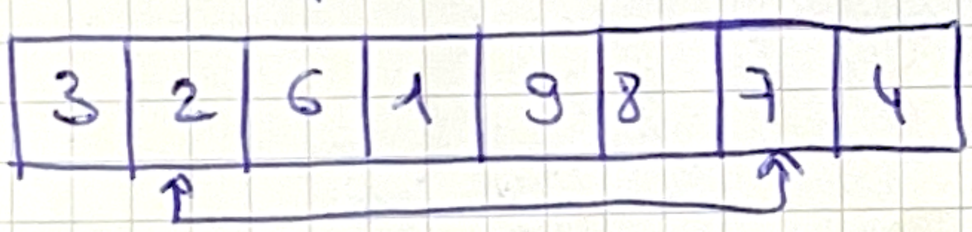


② search for: the item from the left that's larger than pivot

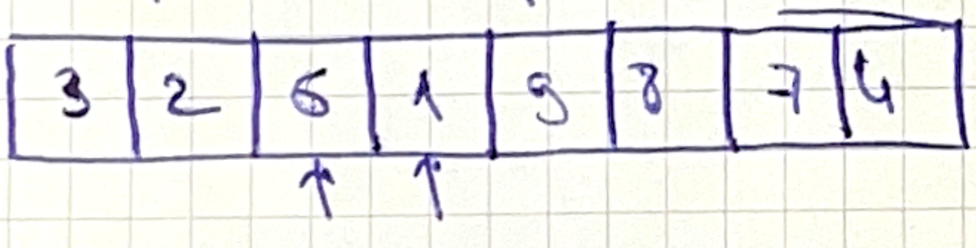
and item from the right that's smaller than the pivot

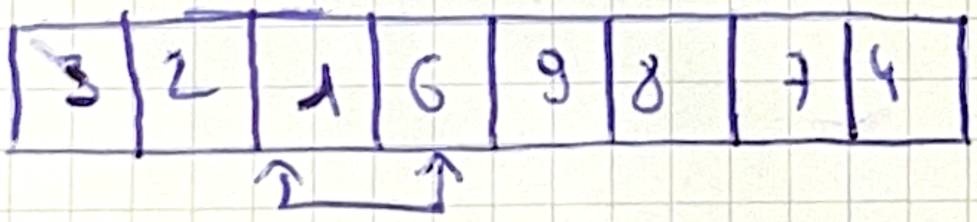


③ swap the 2 items:



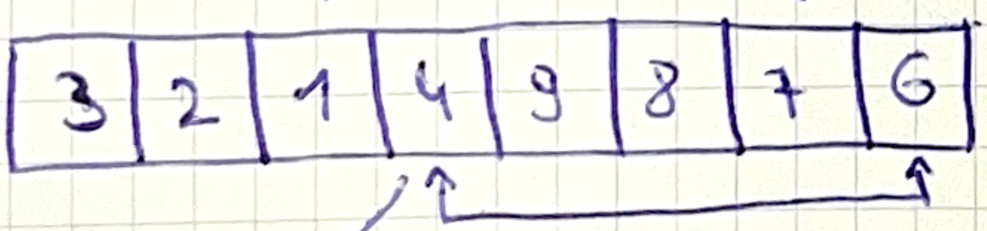
④ repeat the process:





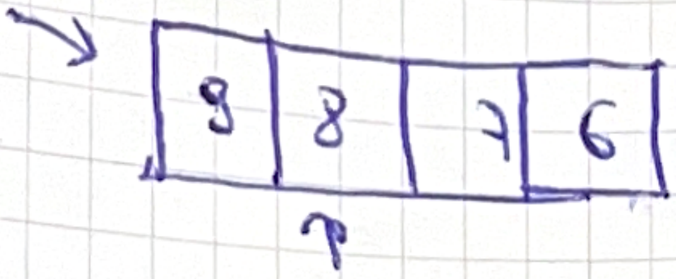
→ now the item from left has a greater index than the item from right

⑤ swap the item from left and the pivot:

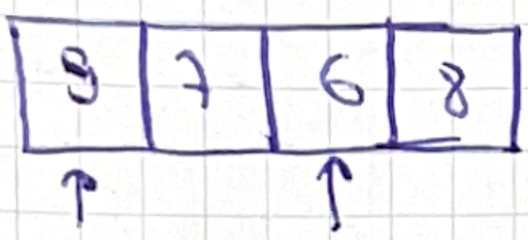


now all items to left are smaller and all items to the right are larger

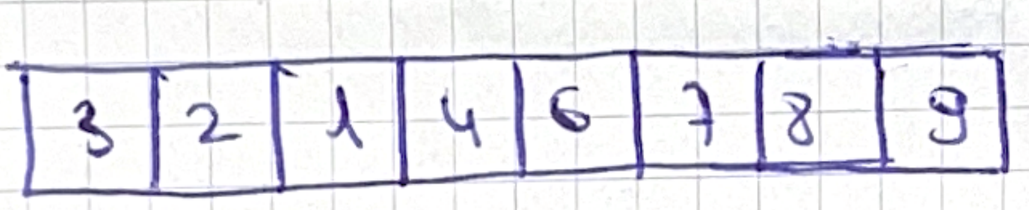
⑥



⑦ same process again:



swap item from left to the end (with the pivot).

etc... 

→ choose a pivot that divides numbers in half, or is close to it

Time complexities:

① complexity of bubble_sort_recursive: $O(n^2)$
→ because the function has to go over the numbers until it reaches one to swap, so when coming to the end, the function has to check more and more pairs (of the list).
each time.

② time complexity of linear search: $O(n)$
→ the function has to go through the whole list to find the target value.

③ time complexity binary search: $O(\log_2 n)$
→ because with each iteration the list is halfed

④ time complexity of quicksort: $O(n \log n)$:
→ because the time decreases, but it also depends on which pivot you choose.

Reflection:

I was mainly struggling with knowing when to use loops, because the ~~rest~~ recursion already makes the function loop so to say.
I was also struggling with not accidentally getting into a forever loop sometimes.
Otherwise it was okay.