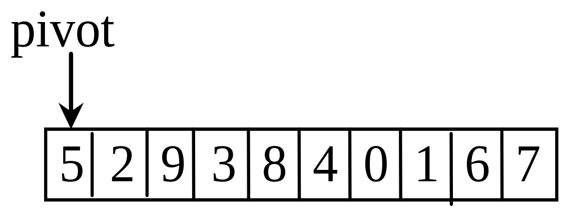
Quicksort

Quicksort

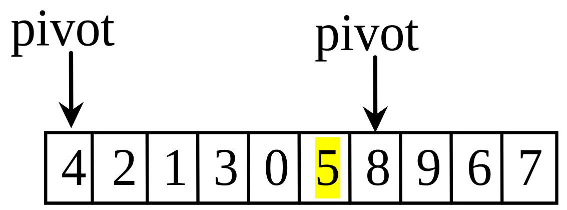
- Quicksort, developed by C. A. R. Hoare (1962), works as follows:

- The algorithm selects an element, called the *pivot*, in the array. Divide the array into two parts such that all the elements in the first part are less than or equal to the pivot and all the elements in the second part are greater than the pivot. Recursively apply the quicksort algorithm to the first part and then the second part.
- It creates two empty arrays to hold elements less than the pivot value and elements more significant than the pivot value, and then recursively sort the sub-arrays.
- Quicksort is a sorting algorithm, which is commonly used in computer science. Quicksort is a divide and conquers algorithm.

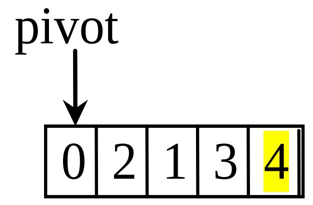
Quicksort



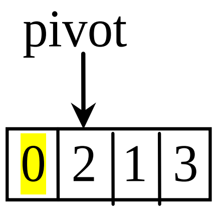
(a) The original array



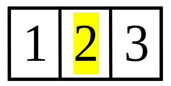
(b)The original array is partitioned



(c) The partial array (4 2 1 3 0) is partitioned

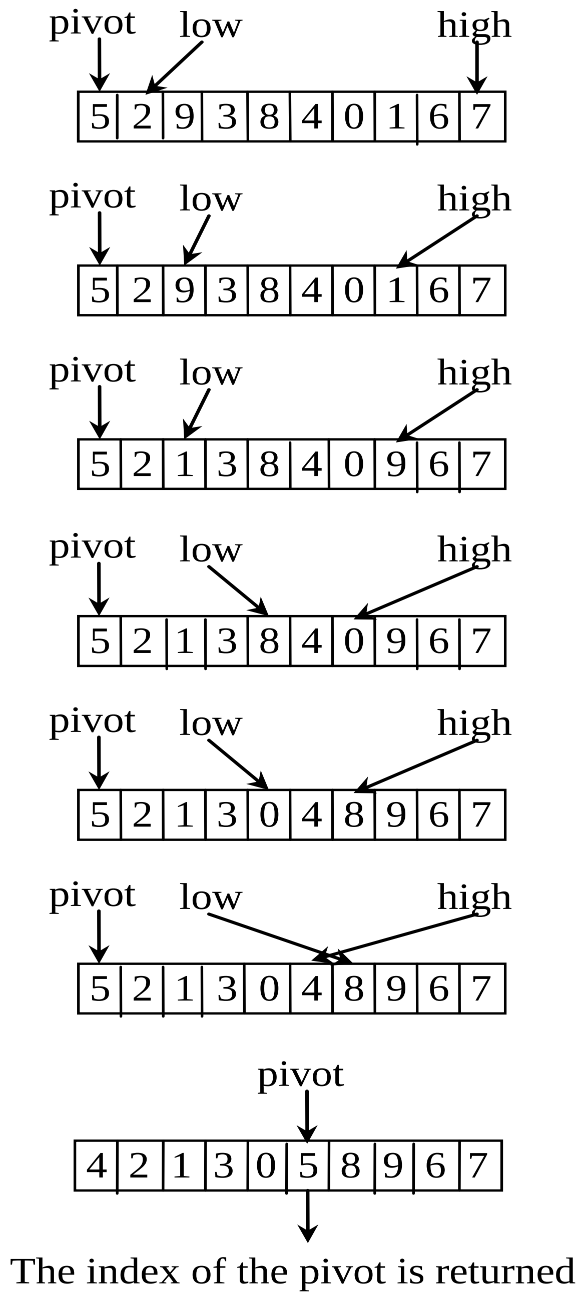


(d) The partial array (0 2 1 3) is partitioned



(e) The partial array (2 1 3) is partitioned

Partition position



(a) Initialize pivot, low, and high

(b) Search forward and backward

(c) 9 is swapped with 1

(d) Continue search

(e) 8 is swapped with 0

(f) when high < low, search is over

(g) pivot is in the right place

Quick Sort Time

To partition an array of *n* elements, it takes *n* comparisons and *n* moves in the worst case. So, the time required for partition is *O(n)*.

In the worst case, each time the pivot divides the array into one big subarray with the other empty. The size of the big subarray is one less than the one before divided. The algorithm requires time:

$$(n-1)+(n-2)+\ldots+2+1=O(n^2)$$

Worst-Case Time

In the worst case, each time the pivot divides the array into one big subarray with the other empty. The size of the big subarray is one less than the one before divided. The algorithm requires:

$$(n-1)+(n-2)+\ldots+2+1=O(n^2)$$

Best-Case Time

In the best case, each time the pivot divides the array into two parts of about the same size. Let *T(n)* denote the time required for sorting an array of elements using quick sort. So,

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

Average-Case Time

On the average, each time the pivot will not divide the array into two parts of the same size nor one empty part. Statistically, the sizes of the two parts are very close. So the average time is *O(logn)*. The exact average-case analysis is beyond the scope of this book.

Pseudo code for quicksort

1. Find a “pivot” element in an array. The pivot item is the basis for comparison for a single round.
2. Start the pointer (the left pointer) at the first item in the array.
3. Start the pointer (the right pointer) at the last item in the array.
4. While the value at the left pointer in the array is less than the pivot value, move the left pointer to the right (add 1). Continue until the value at the left pointer is greater than or equal to the pivot value.
5. While the value at the right pointer in the array is greater than the pivot value, move the right pointer to the left (subtract 1). Continue until the value at the right pointer is less than or equal to the pivot value.
6. If the left pointer is less than or equal to the right pointer, then swap the values at these locations in an array.
7. Move the left pointer to the right by one and the right pointer to the left by one.
8. If the left pointer and right pointer don’t meet, go to step 1.

The End!