DATA STRUCTURES & ALGORITHMS

GAILY REY APRIL M. GUZON

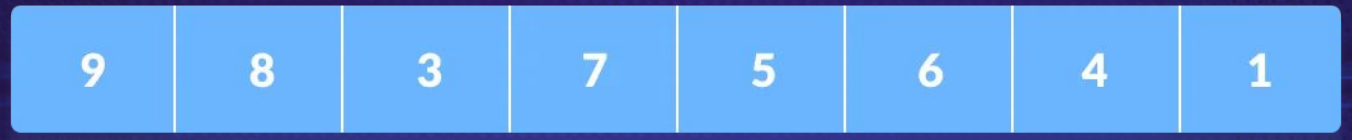
Searching and Sorting Algorithms

Shell Sort Algorithm

1. Shell sort is an algorithm that first sorts the elements far apart from each other and successively reduces the interval between the elements to be sorted. It is a generalized version of insertion sort.
2. In shell sort, elements at a specific interval are sorted. The interval between the elements is gradually decreased based on the sequence used. The performance of the shell sort depends on the type of sequence used for a given input array.

How Shell Sort Works?

1. Suppose, we need to sort the following array.



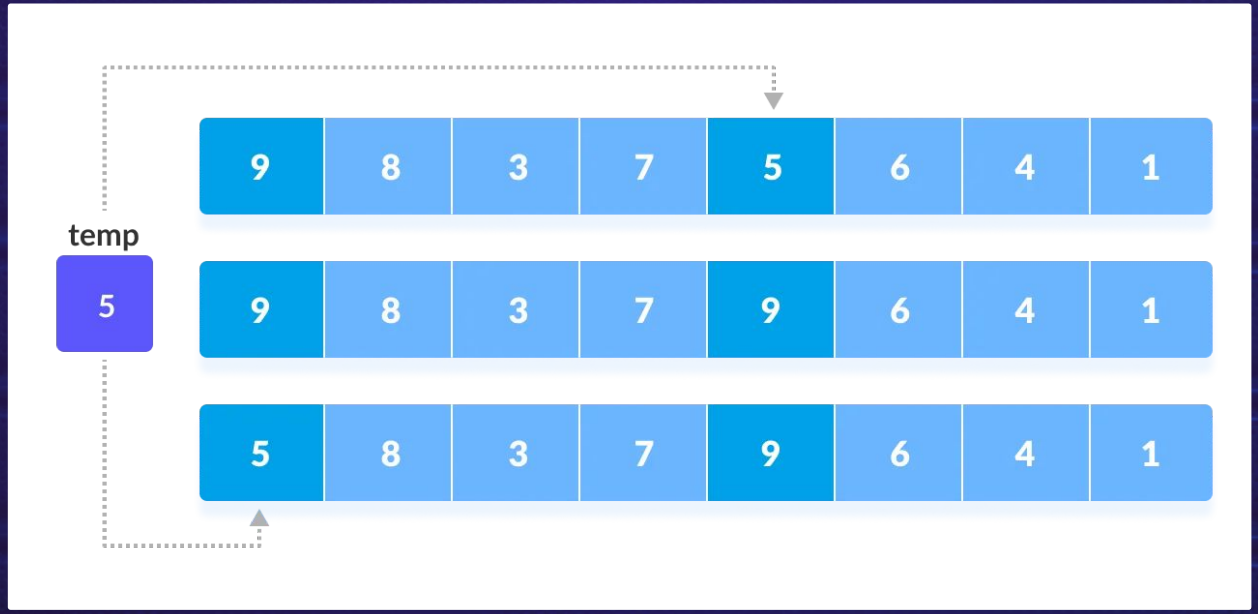
2. We are using the shell's original sequence (N/2, N/4, ...1) as intervals in our algorithm.

How Shell Sort Works?

1.In the first loop, if the array size is N=8, the elements lying at the interval of N/2=4 are compared and swapped if they are not in order.

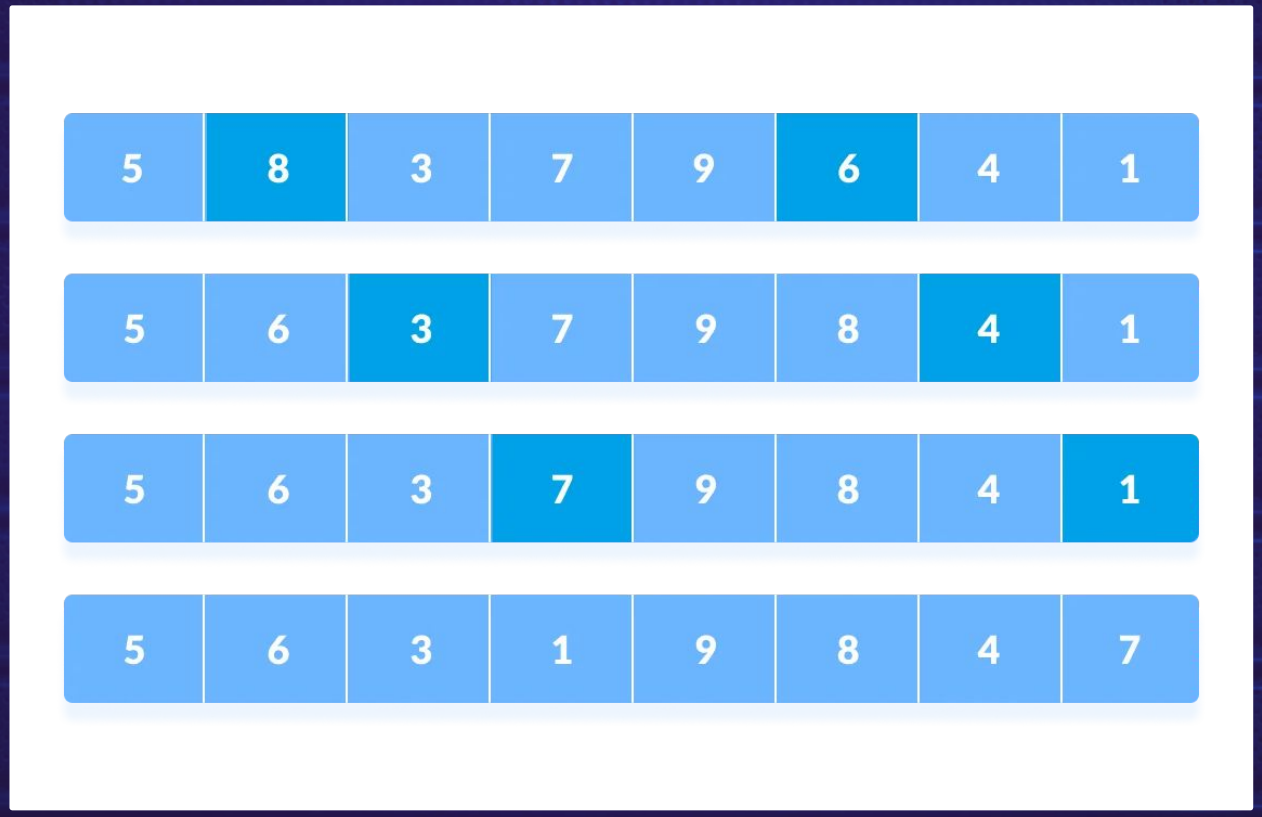
2.The 0th element is compared with the 4th element.

Suppose the 0th element is greater than the 4th element. In that case, the 4th element is first stored in in temp variable and the 0th element (ie. greater element) is stored in the 4th position and the element stored in temp is stored in the 0th position.



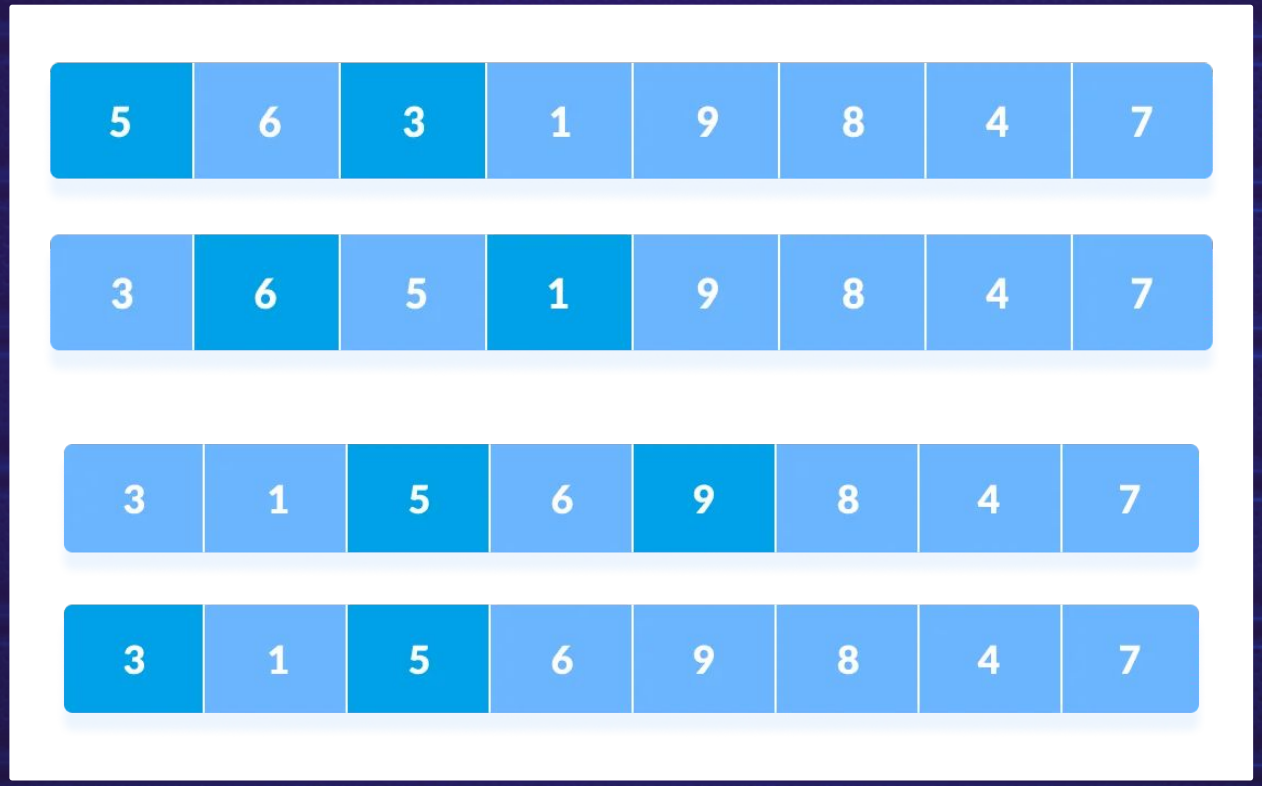
How Shell Sort Works?

1.This process goes on for all the remaining elements.



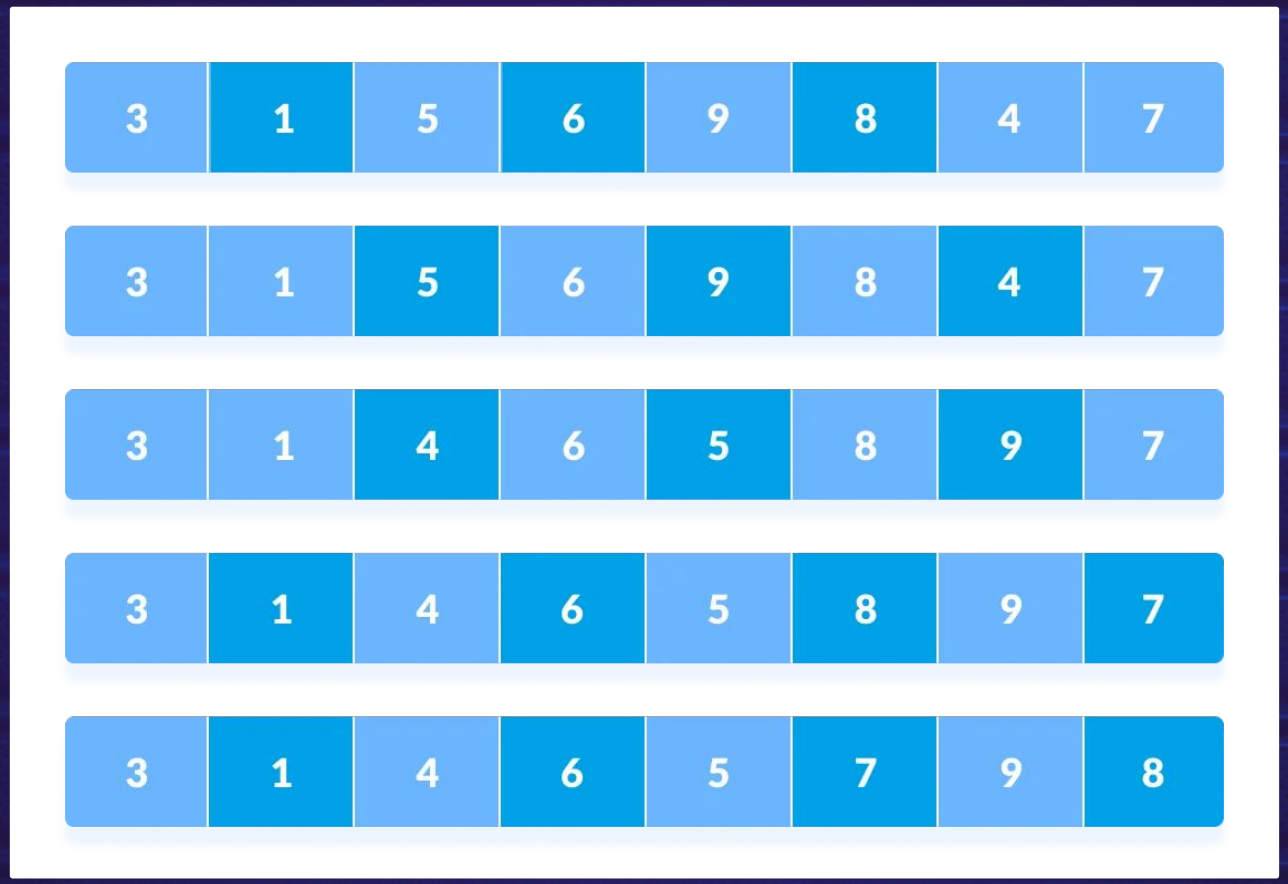
How Shell Sort Works?

3. An interval of N/4= 8/4=2 is taken in the second loop, and the elements lying at these intervals are sorted.



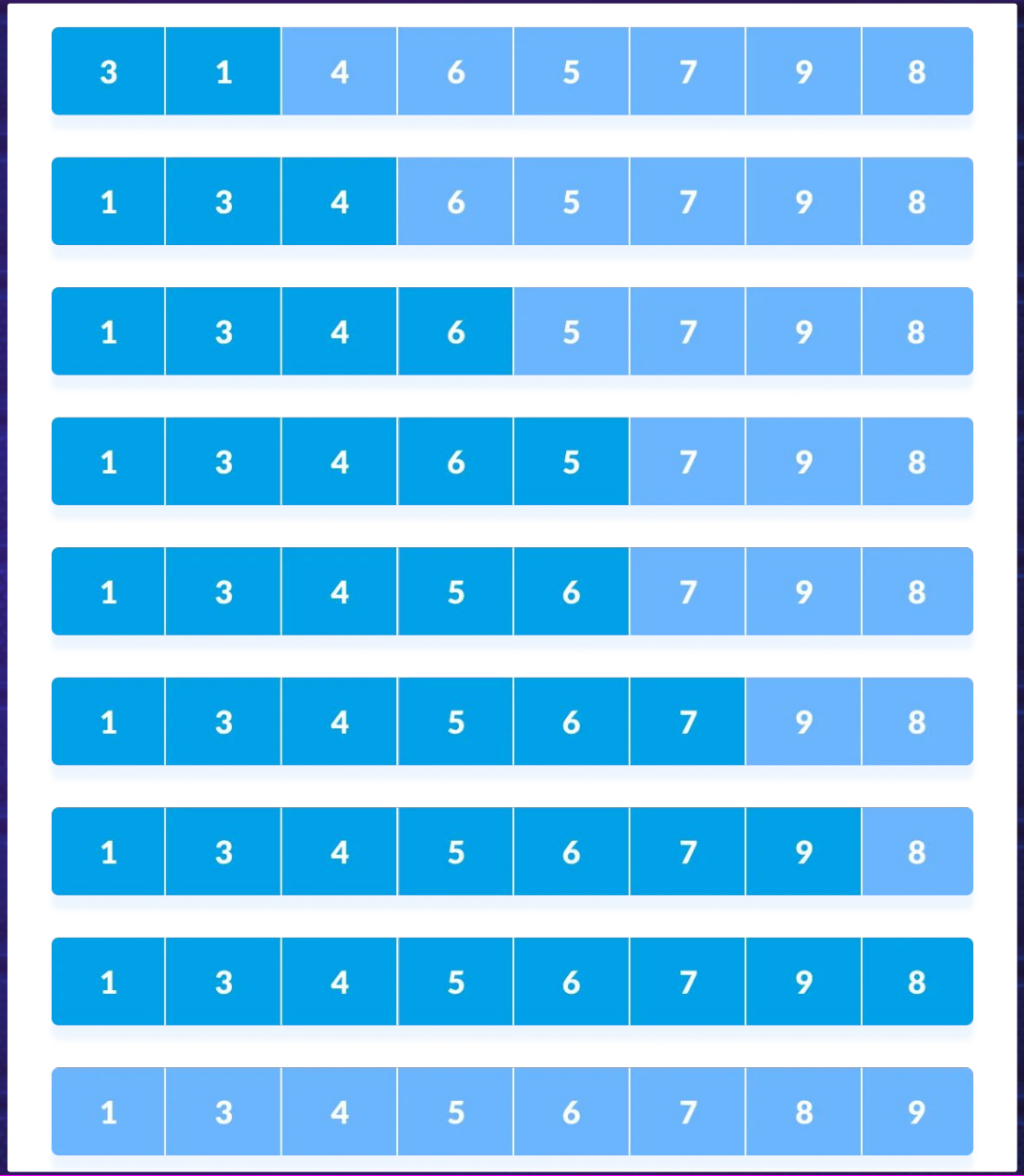
How Shell Sort Works?

4. The same process goes on for remaining elements.



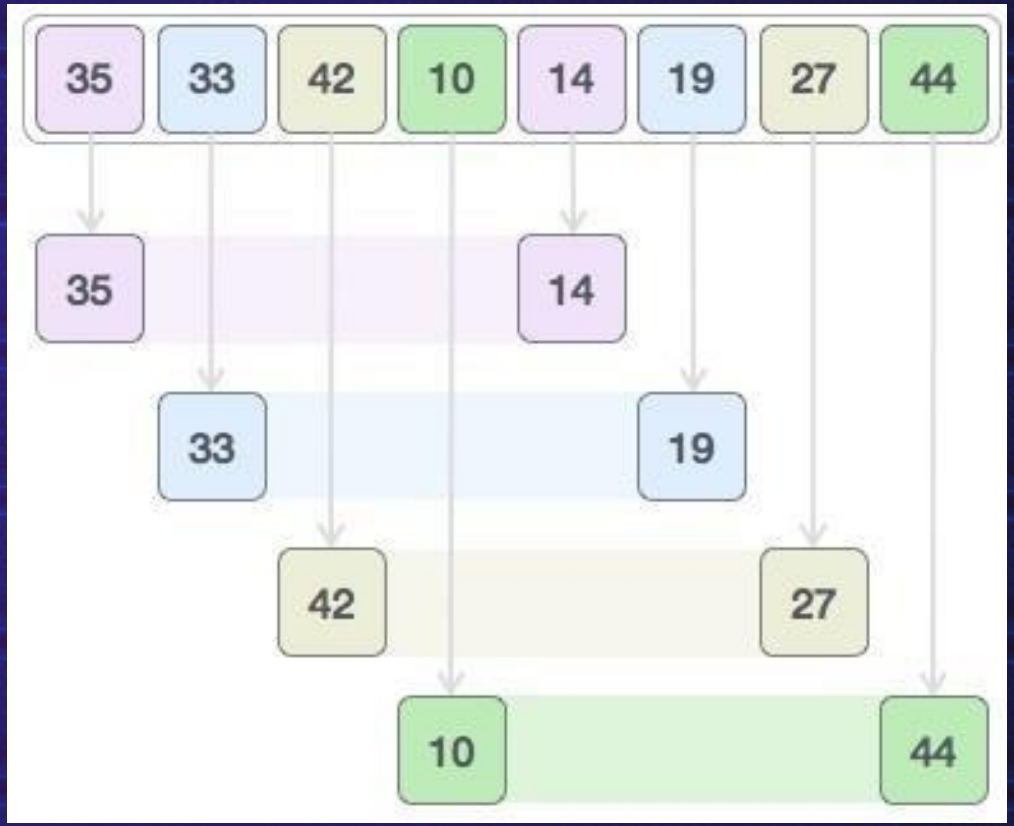
How Shell Sort Works?

5. Finally, when the interval is N/8 = 8/8 = 1, the array elements lying at the interval of 1 are sorted. The array is now completely sorted.



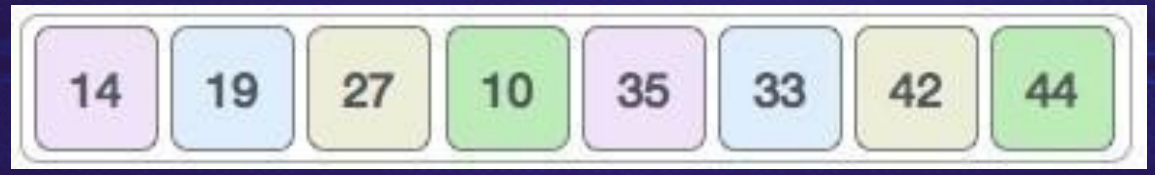
Another Example

For our example we have an unsorted list containing elements [35, 33, 42, 10, 14, 19, 27, 44]. For an easier understanding we take the interval of 4. Make a virtual sub-list of all values located at the interval of 4 positions. Here these values are {35, 14}, {33, 19}, {42, 27} and {10, 44}

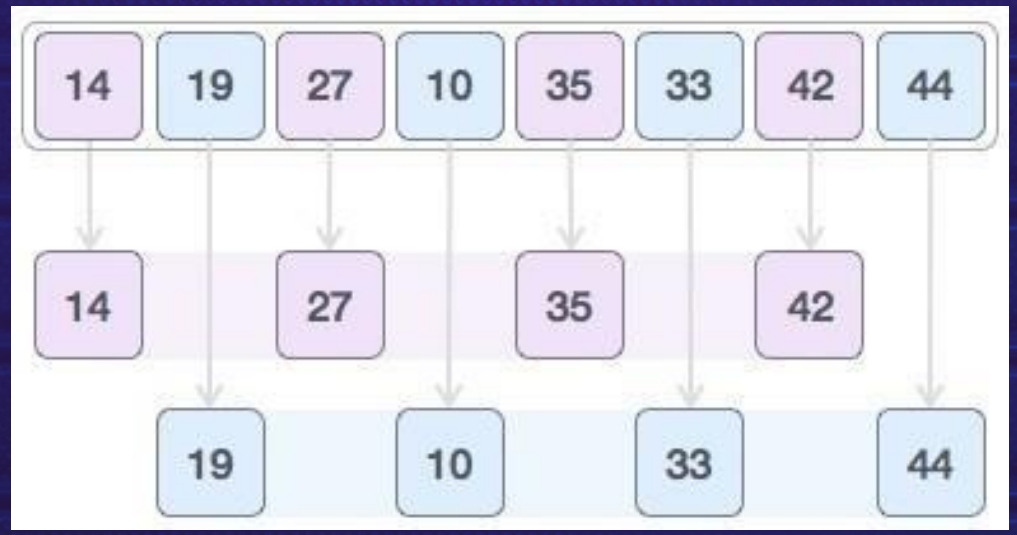


Another Example

We compare values in each sub-list and swap them (if necessary) in the original array. After this step, the new array should look like this –

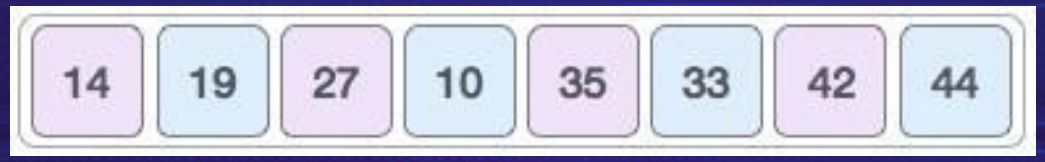


Then, we take interval of 1 and this gap generates two sub-lists - {14, 27, 35, 42}, {19, 10, 33, 44}



Another Example

We compare and swap the values, if required, in the original array. After this step, the array should look like this –



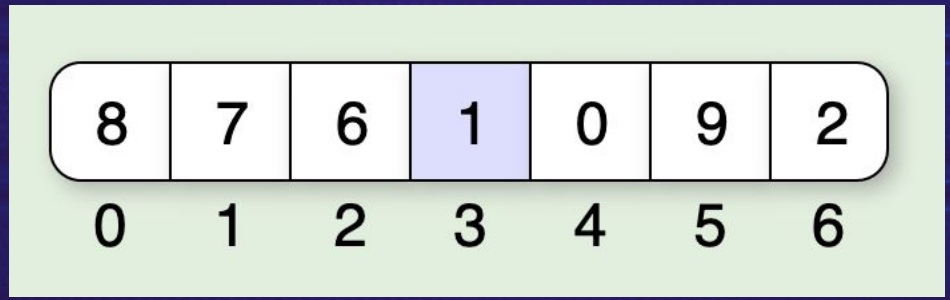
Another Example

Finally, we sort the rest of the array using interval of value 1. Shell sort uses insertion sort to sort the array. Following is the step-by-step depiction –

Working of Quicksort Algorithm

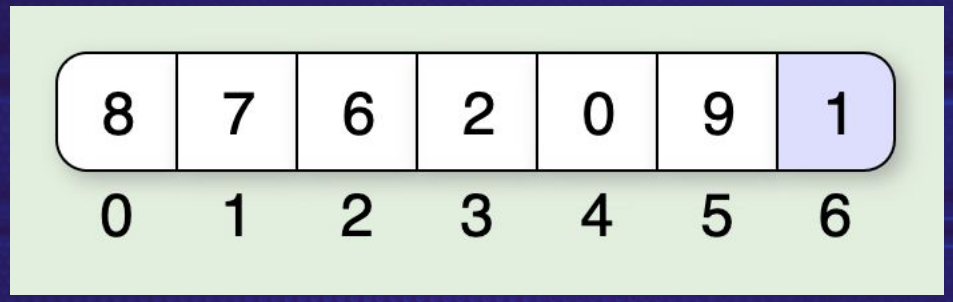
1. Select the Pivot Element

There are different variations of quicksort where the pivot element is selected from different positions.

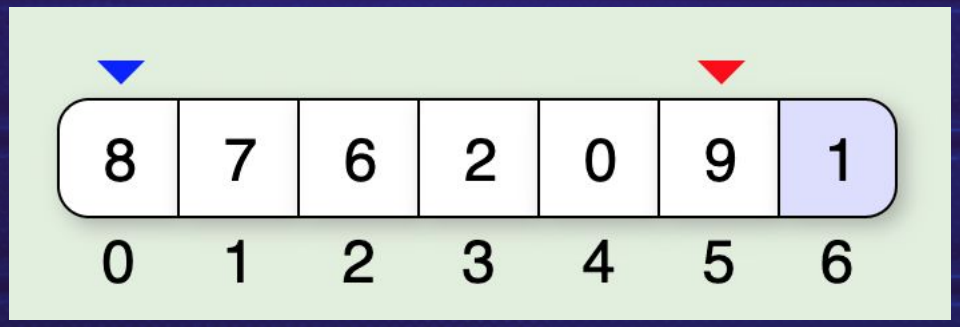


Working of Quicksort Algorithm

Move the pivot to the end

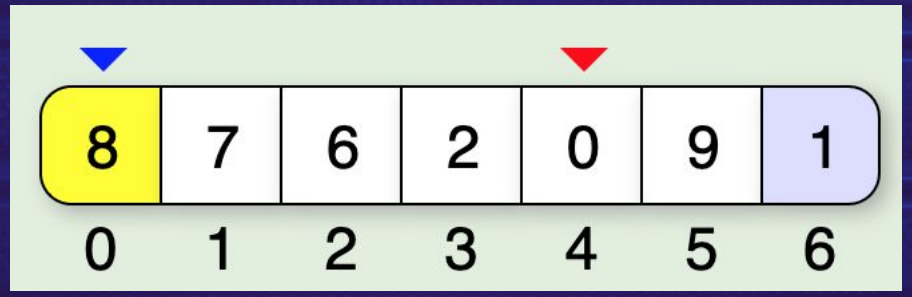


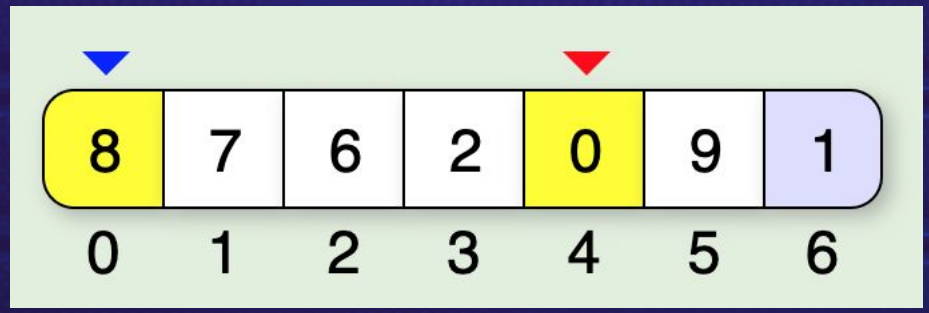
Partition the subarray



Working of Quicksort Algorithm

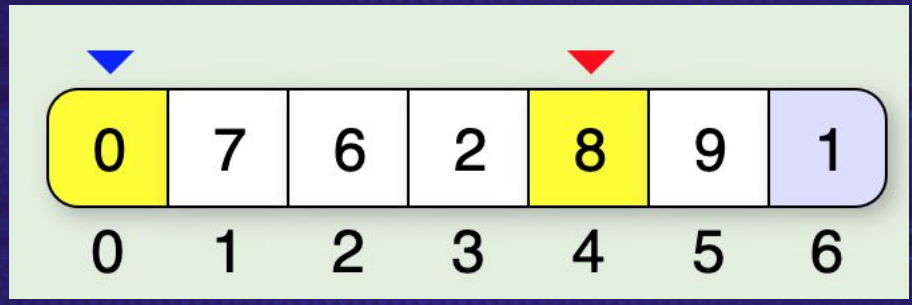
Move the right bound to the left until it crosses the left bound or finds a value less than the pivot.



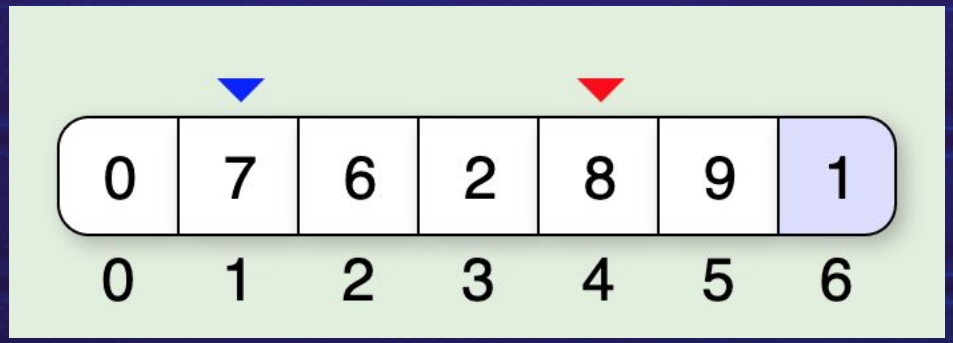


Working of Quicksort Algorithm

Swap the selected values.

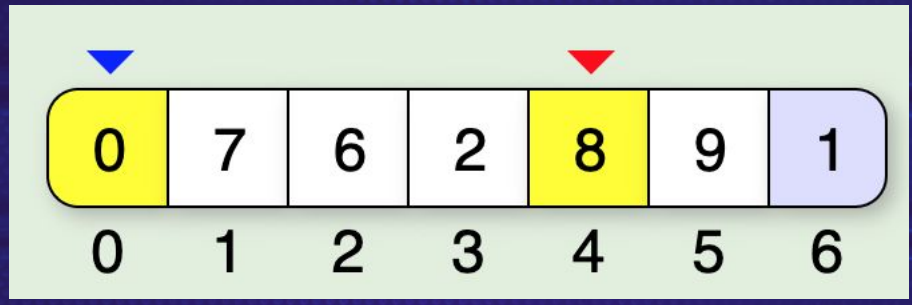


Move the left bound to the right until it reaches a value greater than or equal to the pivot.

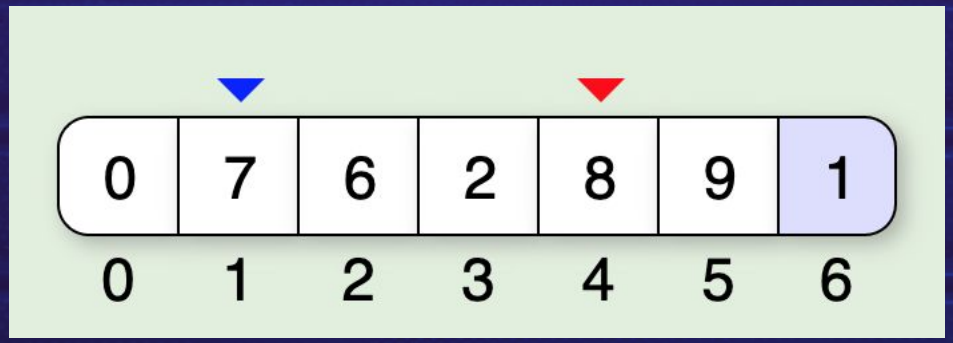


Working of Quicksort Algorithm

Swap the selected values.

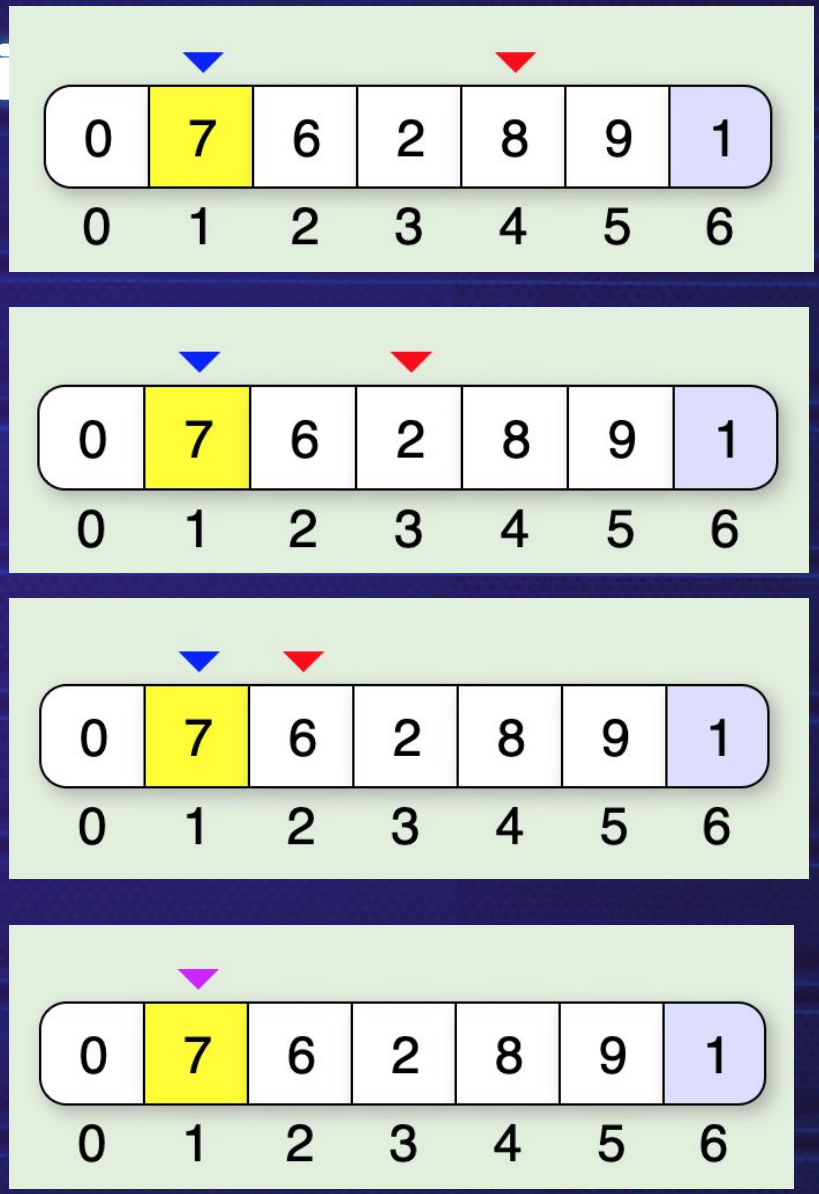


Move the left bound to the right until it reaches a value greater than or equal to the pivot.



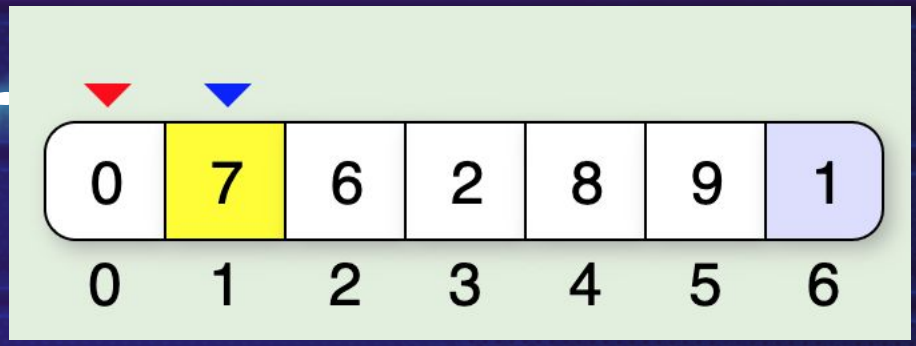
Working of Quicksort Algori

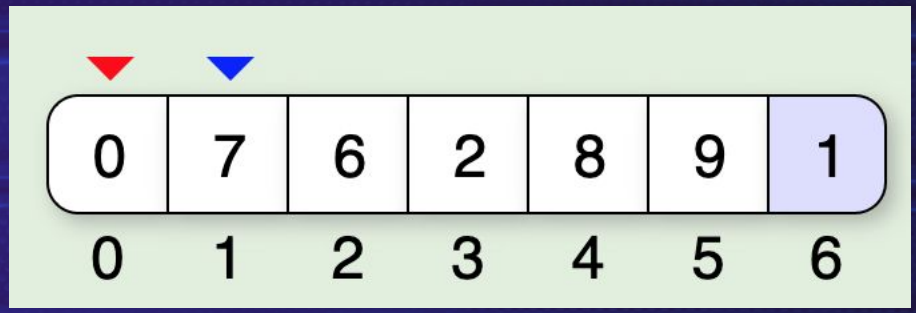
Move the right bound to the left until it crosses the left bound or finds a value less than the pivot.

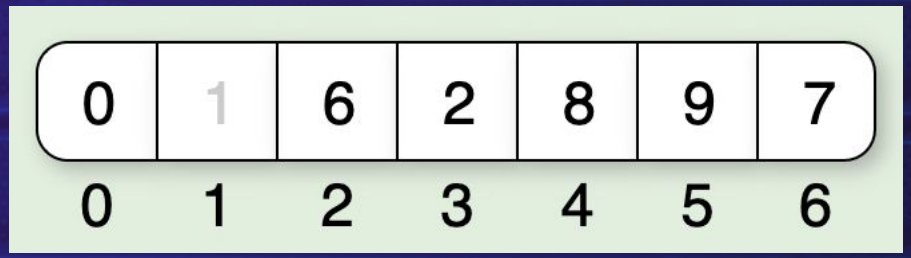


Working of Quicksort Algor

- Bounds have crossed
- When the right bound crosses the left bound, all elements to the left of the left bound are less than the pivot and all elements to the right are greater than or equal to the pivot.

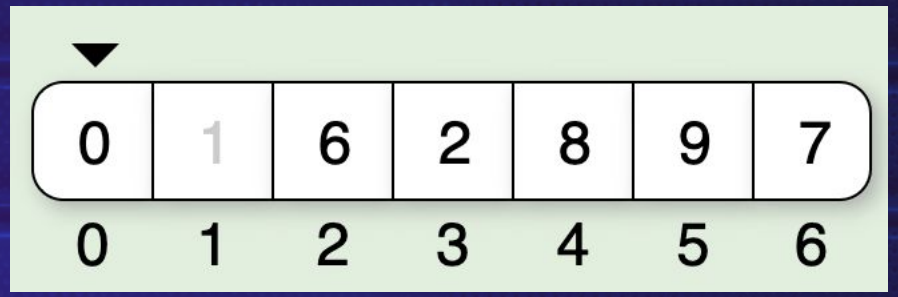


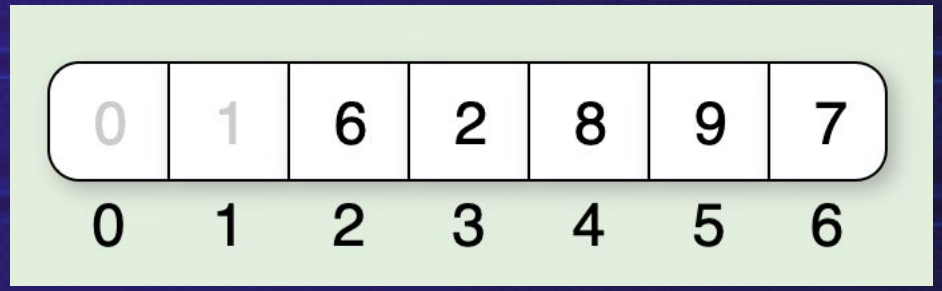




Working of Quicksort Algorithm

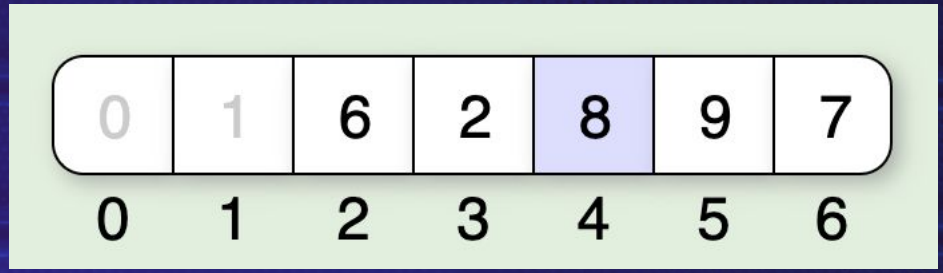
- Left sublist contains a single element which means it is sorted.
- Call quicksort on the right sublist.



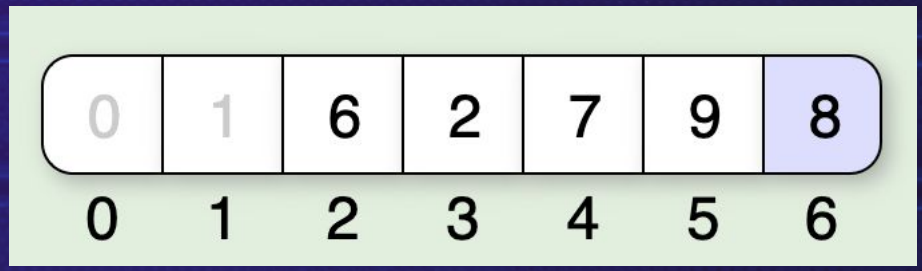


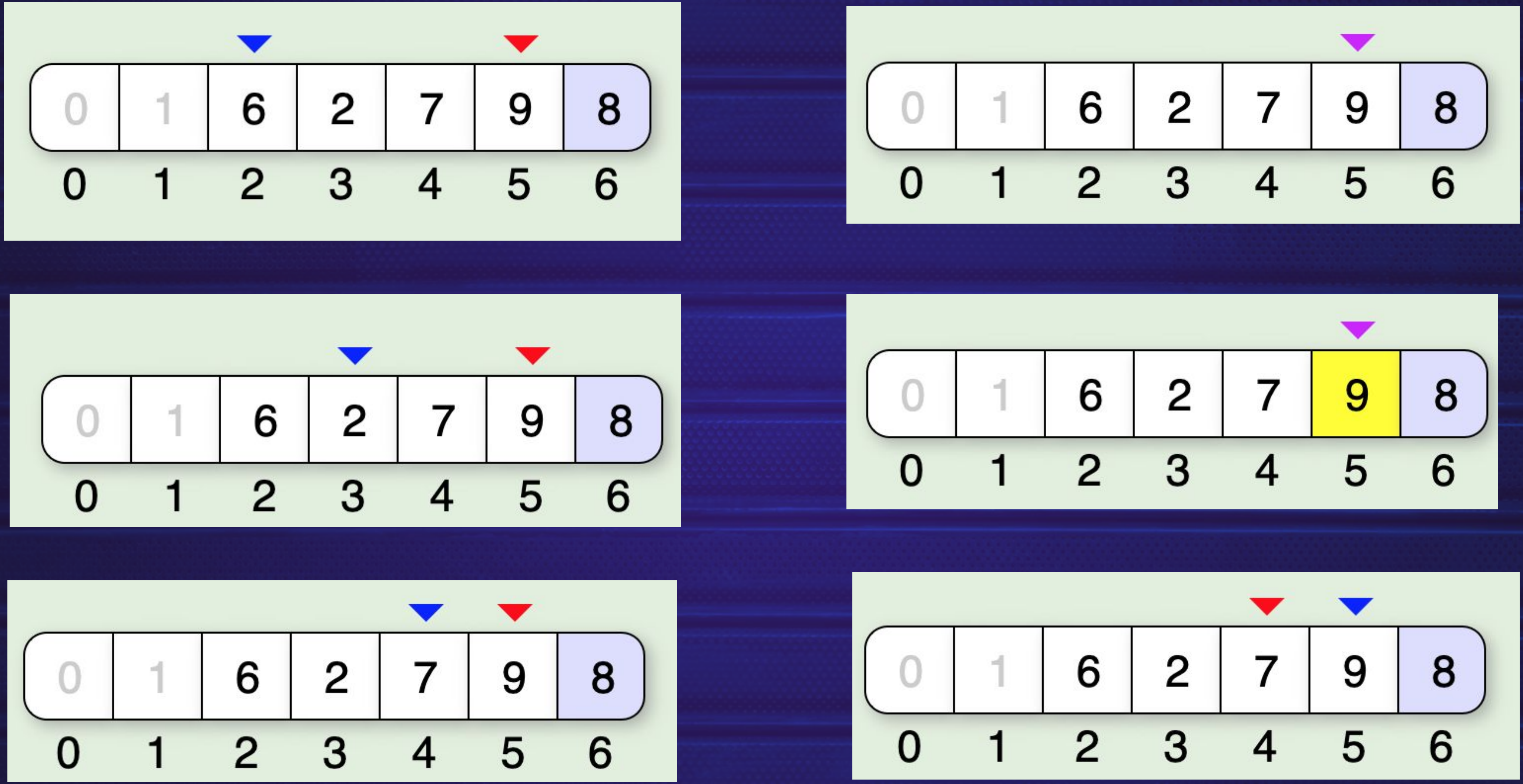
Working of Quicksort Algorithm

- Select the pivot.

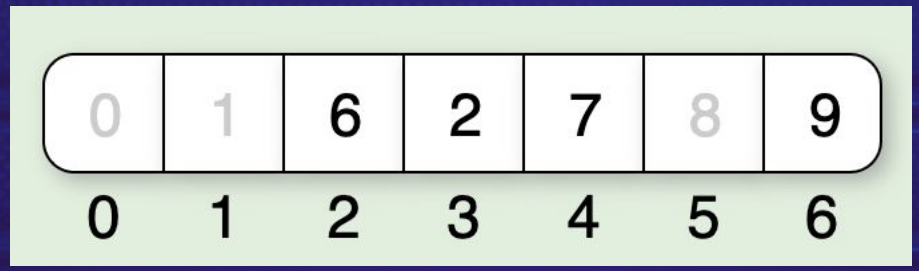


- **Repeat the Process**
- Move the pivot to the end.



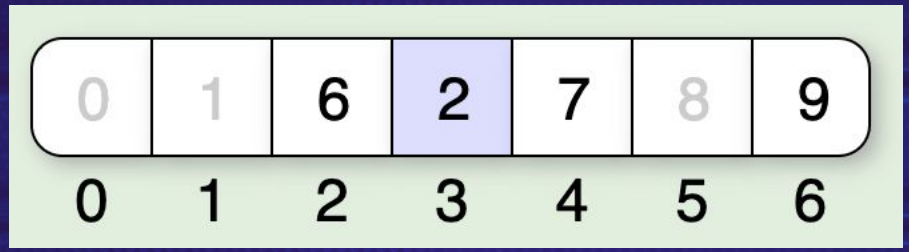

0 1 6 2 7 9 8
0 1 2 3 4 5 6
0 1 6 2 7 9 8
0 1 2 3 4 5 6
0 1 6 2 7 9 8
0 1 2 3 4 5 6
0 1 6 2 7 9 8
0 1 2 3 4 5 6
0 1 6 2 7 9 8
0 1 2 3 4 5 6
0 1 6 2 7 9 8
0 1 2 3 4 5 6

When the right bound crosses the left bound, all elements to the left of the left bound are less than the pivot and all elements to the right are greater than or equal to the pivot.

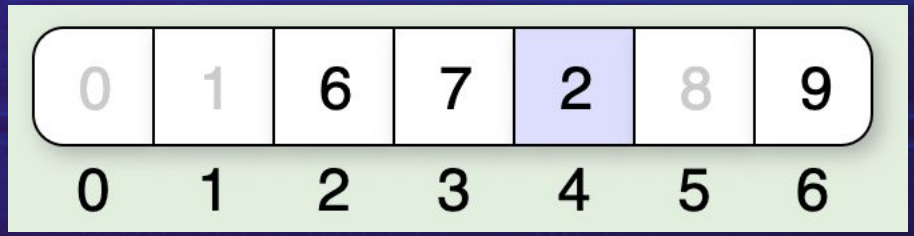


Call quicksort on the right sublist.

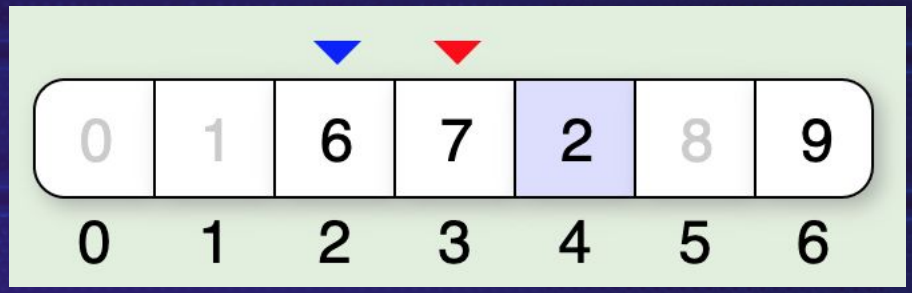
Select pivot



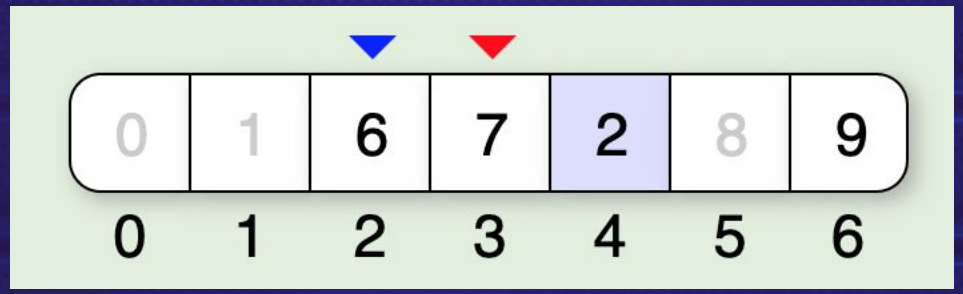
Move pivot to the end

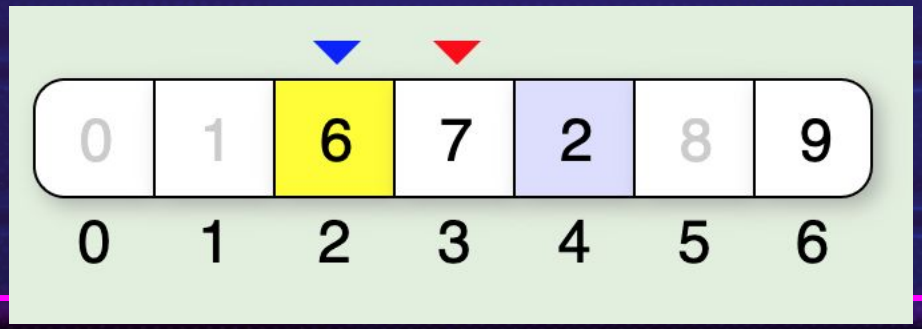


Partition the subarray.

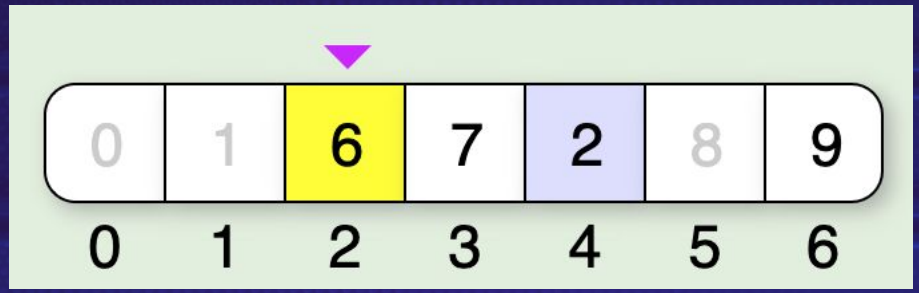


Move the left bound to the right until it reaches a value greater than or equal to the pivot.

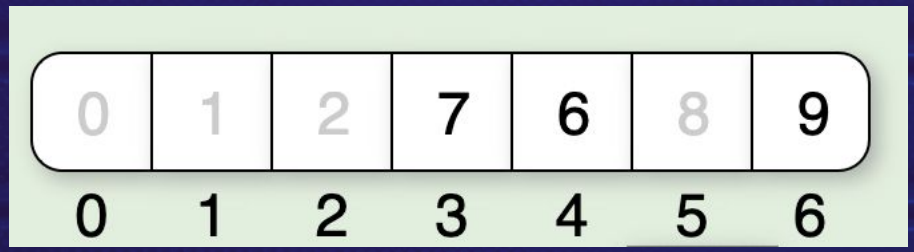




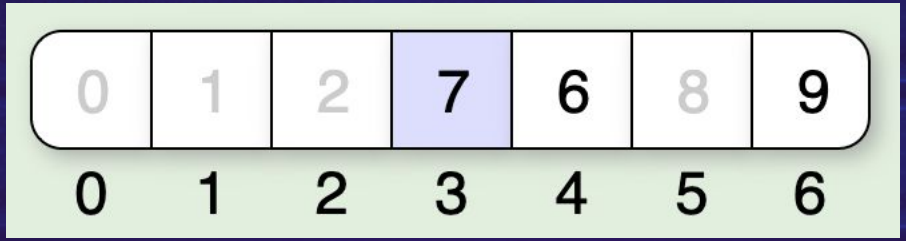
Move the right bound to the left until it crosses the left bound or finds a value less than the pivot.



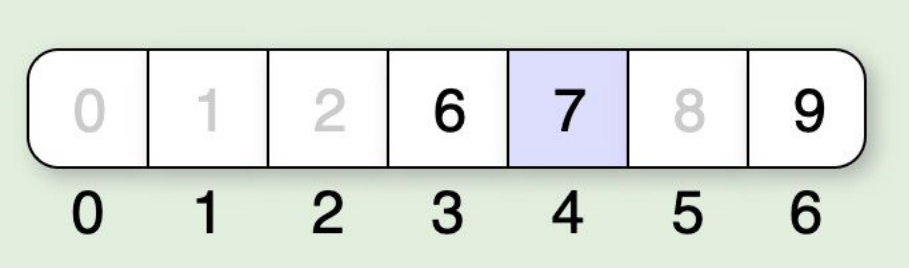
When the right bound crosses the left bound, all elements to the left of the left bound are less than the pivot and all elements to the right are greater than or equal to the pivot.

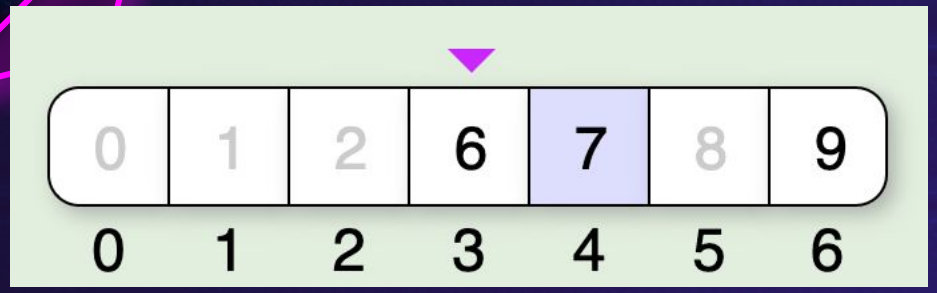


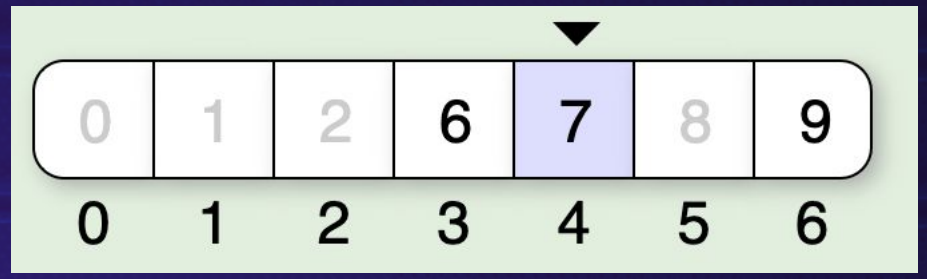
Call quicksort on the right sublist.
Select a pivot

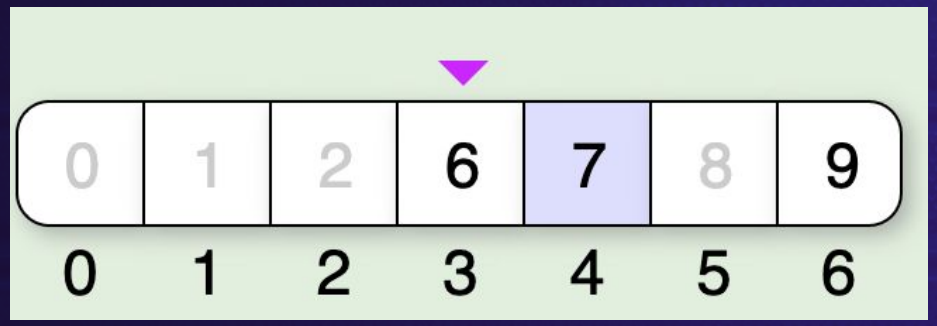


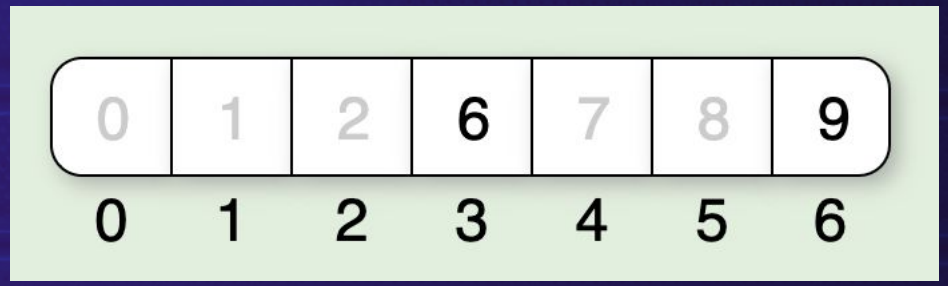
Move pivot to the end

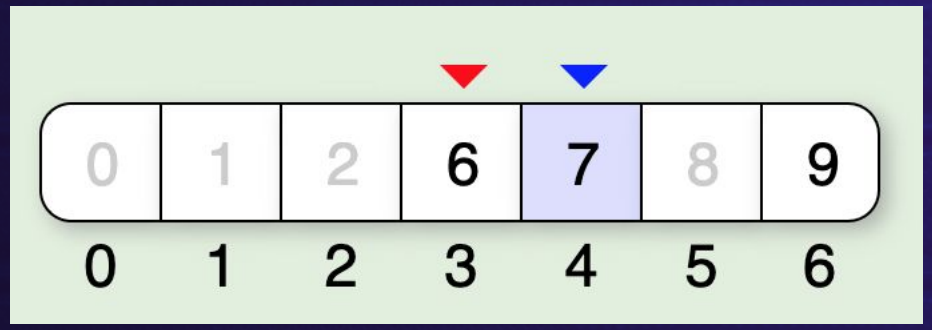



0 1 2 6 7 8 9
0 1 2 3 4 5 6

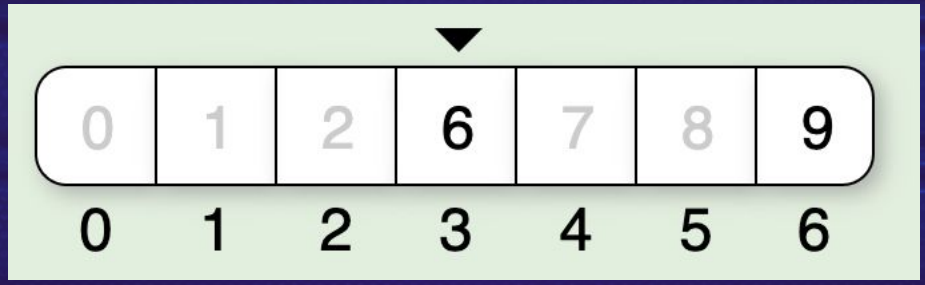

0 1 2 6 7 8 9
0 1 2 3 4 5 6


0 1 2 6 7 8 9
0 1 2 3 4 5 6


0 1 2 6 7 8 9
0 1 2 3 4 5 6


0 1 2 6 7 8 9
0 1 2 3 4 5 6

Left sublist contains a single element which means it is sorted.



Right sublist contains a single element which means it is sorted.