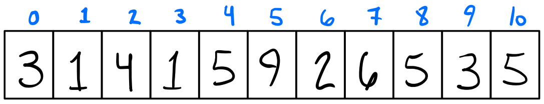
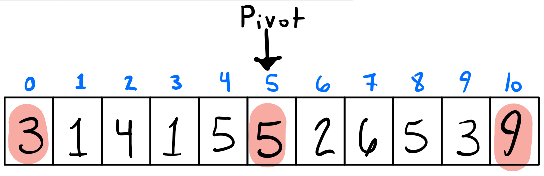


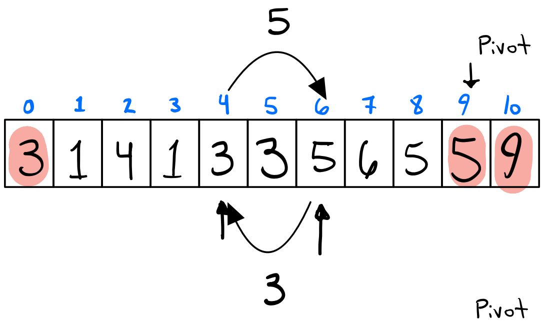
0	1	2	3	4	5	6	7	8	9	10
3	1	4	1	5	9	2	6	5	3	5

Sort median of 3.



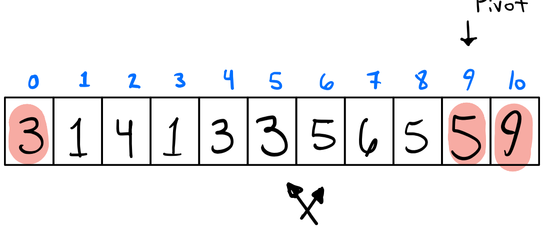
0	1	2	3	4	5	6	7	8	9	10
3	1	4	1	5	5	2	6	5	3	9

Move pivot to right-1.
Move pointers until
items can be swapped.



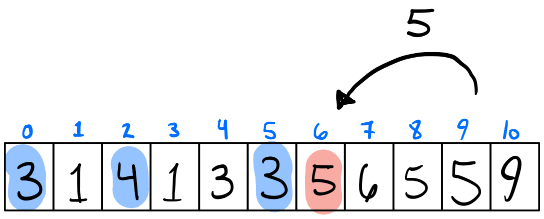
0	1	2	3	4	5	6	7	8	9	10
3	1	4	1	3	3	5	6	5	5	9

Pointers have crossed.
End of partition #1.



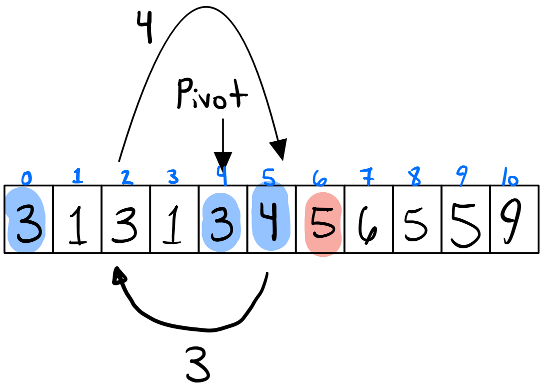
0	1	2	3	4	5	6	7	8	9	10
3	1	4	1	3	3	5	6	5	5	9

Pivot is moved back to center. Start partition #2 by median of 3 and sort median of 3.



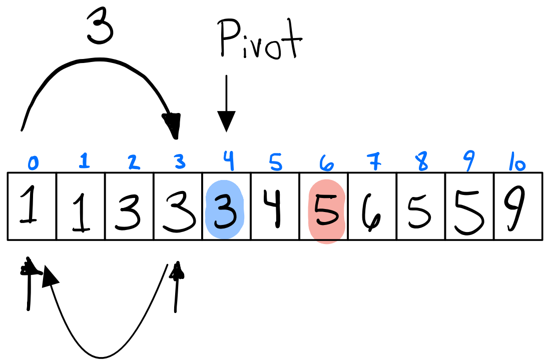
0	1	2	3	4	5	6	7	8	9	10
3	1	4	1	3	3	5	6	5	5	9

Sort median of 3. Move pivot to right-1.



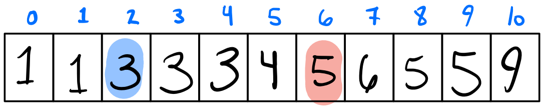
0	1	2	3	4	5	6	7	8	9	10
3	1	3	1	3	4	5	6	5	5	9

Move pointers until items can be swapped.



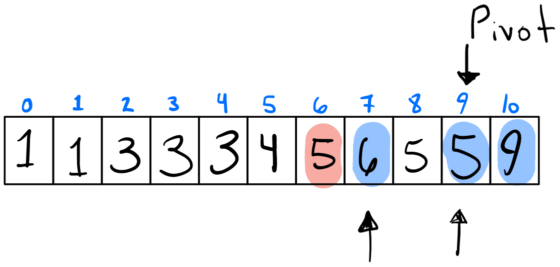
0	1	2	3	4	5	6	7	8	9	10
1	1	3	3	3	4	5	6	5	5	9

End of partition. Cutoff point of 3 reached for [0,1] and [3,5]. Use insertion sort.



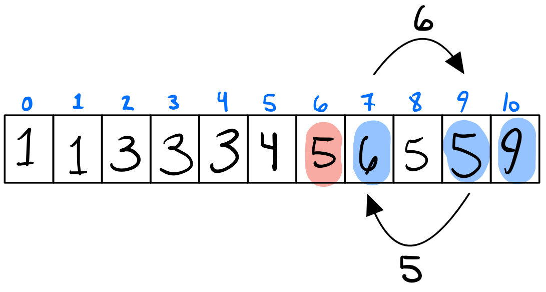
0	1	2	3	4	5	6	7	8	9	10
1	1	3	3	3	4	5	6	5	5	9

Sort median of 3 move pivot to right - 1



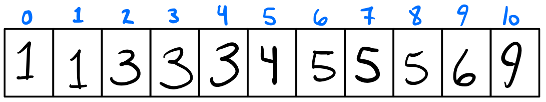
0	1	2	3	4	5	6	7	8	9	10
1	1	3	3	3	4	5	6	5	5	9

Swap



0	1	2	3	4	5	6	7	8	9	10
1	1	3	3	3	4	5	6	5	5	9

Cut off point of 3 is reached. Insertion sort is implemented. The whole array is now sorted using quick sort.



0	1	2	3	4	5	6	7	8	9	10
1	1	3	3	3	4	5	5	5	6	9