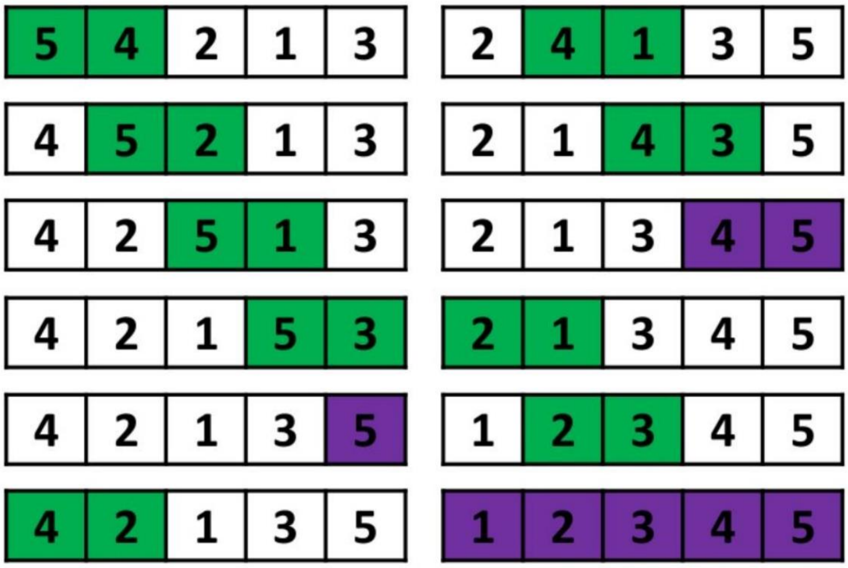
\~~~~~~

Bubble Sort

- ❑ **Example 2:**



Bubble Sort

- ❑ **Time Complexity:** $O(n^2)$ as there are two nested loops

- ❑ Example of worst case

5	4	3	2	1

~~~~~~
~~~~~~

Selection Sort

- Example 2:

12	10	16	11	9	7

12	10	16	11	9	7
7	10	16	11	9	12
7	9	16	11	10	12
7	9	10	11	16	12
7	9	10	11	16	12
7	9	10	11	12	16

Selection Sort

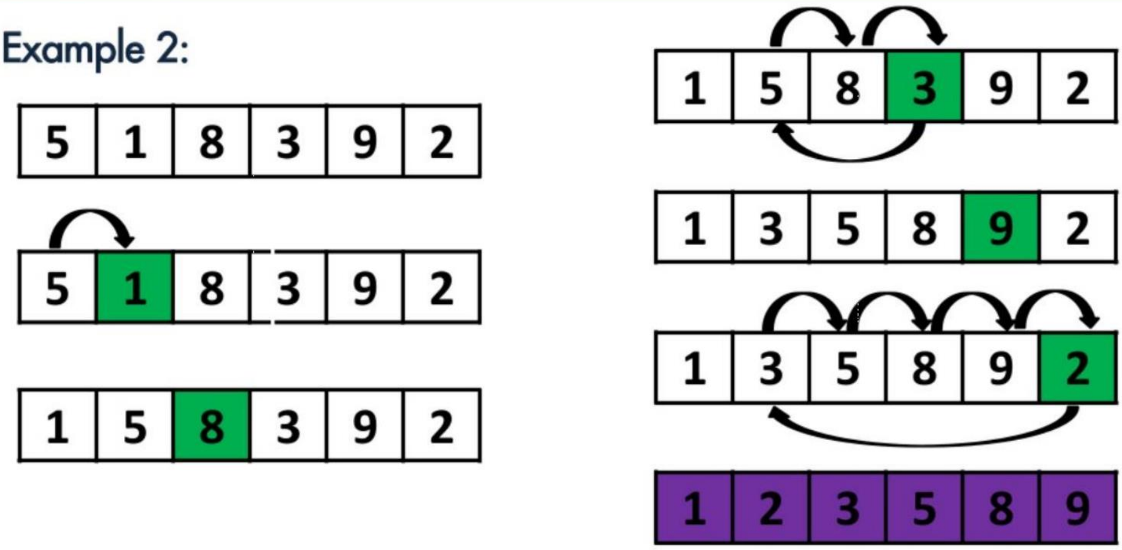
- **Time Complexity:** $O(n^2)$ as there are two nested loops

- Example of worst case

2	**3**	**4**	**5**	**1**

Insertion Sort

- ❑ Example 2:



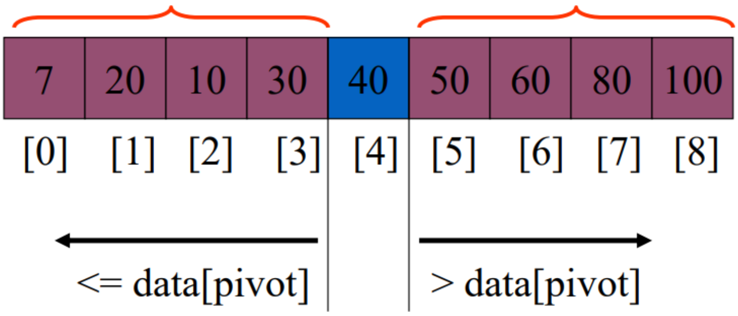
Insertion Sort

- ❑ **Time Complexity:** $O(n^2)$

- ❑ Example of worst case

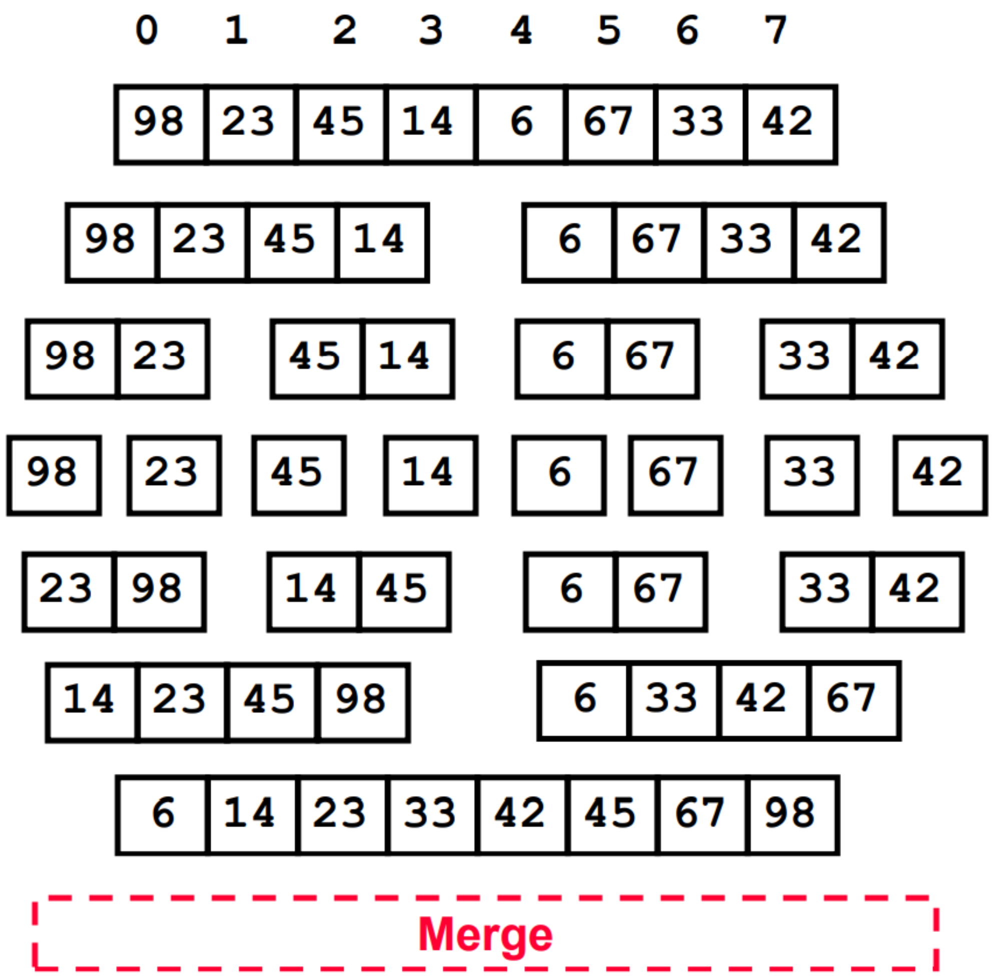
5	4	3	2	1

Recursion: Quicksort Sub-arrays



Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: O(n logn)
- Worst case running time: O(n^2)!!!



running time: O(n logn)

- **Bubble sort and Insertion sort –**
  Average and worst case time complexity: n^2
  Best case time complexity: n when array is already sorted.
  Worst case: when the array is reverse sorted.
- **Selection sort –**
  Best, average and worst case time complexity: n^2 which is independent of distribution of data.
- **Merge sort –**
  Best, average and worst case time complexity: nlogn which is independent of distribution of data.
- **Heap sort –**
  Best, average and worst case time complexity: nlogn which is independent of distribution of data.
- **Quick sort –**
  It is a divide and conquer approach with recurrence relation:

```
T(n) = T(k) + T(n-k-1) + cn
```

Worst case: when the array is sorted or reverse sorted, the partition algorithm divides the array in two subarrays with 0 and n-1 elements. Therefore,

```
T(n) = T(0) + T(n-1) + cn
Solving this we get, T(n) = O(n^2)
```

Best case and Average case: On an average, the partition algorithm divides the array in two subarrays with equal size. Therefore,

```
T(n) = 2T(n/2) + cn
Solving this we get, T(n) = O(nlogn)
```