Data Structures

Sorting Technique – Quick Sort

Team Emertxe

Introduction

Introduction

Quick Sort

- It is also called as Partition Exchange Sort
- It works on Divide and Conquer Algorithm

Introduction

Quick Sort

- It is also called as Partition Exchange Sort
- It works on Divide and Conquer Algorithm

Introduction

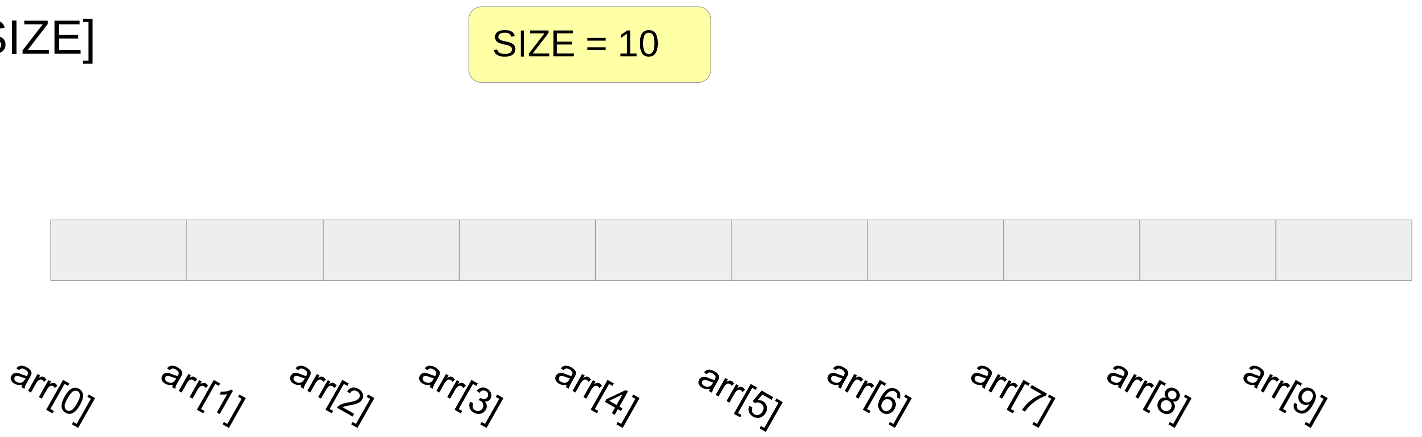
Quick Sort

- It uses recursive calls for sorting the elements.
- It picks an element as pivot and partitions the given array around picked pivot
- All elements in the first sublist are arranged to be smaller than the pivot
- All elements in the second sublist are arranged to be larger than the pivot
- The same partitioning and arranging process is performed repeatedly on the resulting sublists until the whole list of items are sorted.

Data Structure – Sorting Techniques

Concept

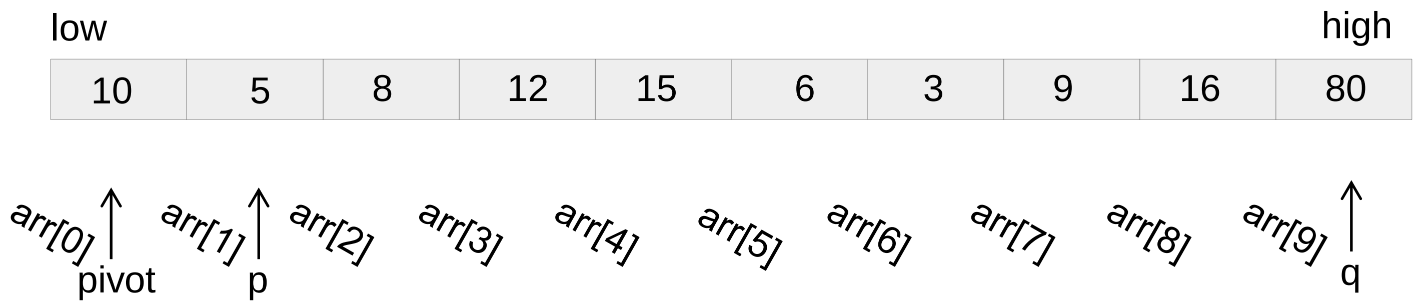
- arr[SIZE]



Concept

- arr[SIZE]

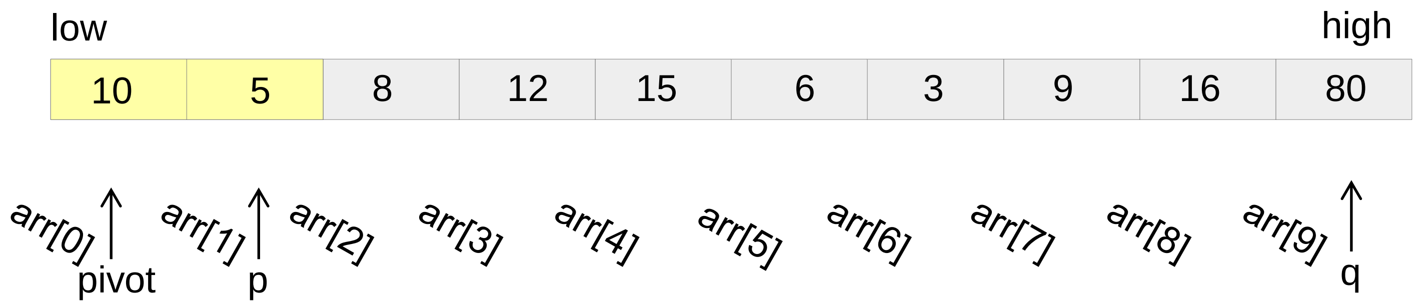
SIZE = 10



Concept

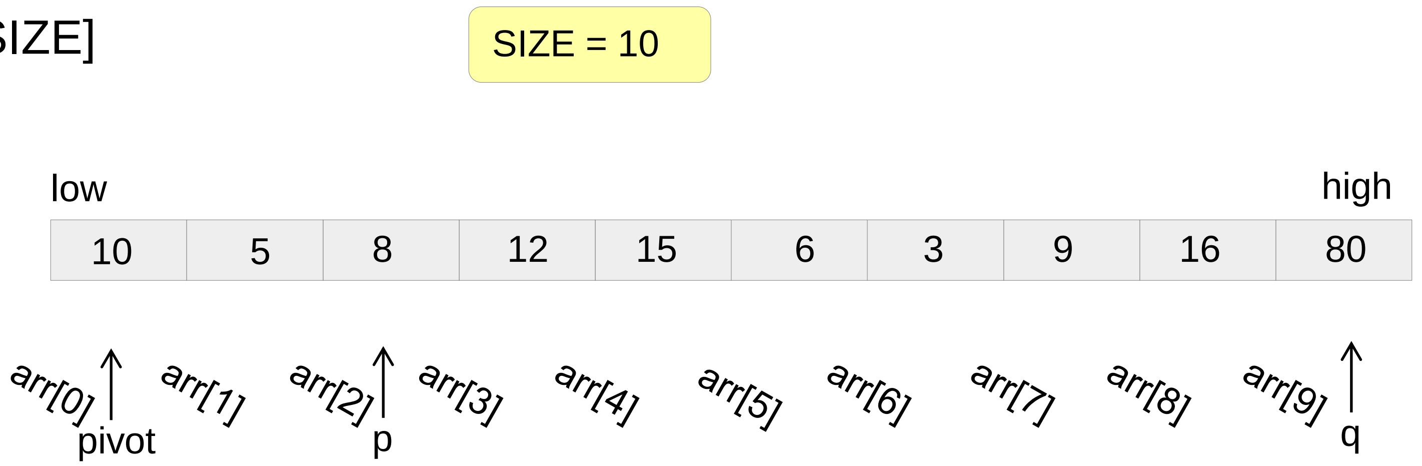
- arr[SIZE]

SIZE = 10



Concept

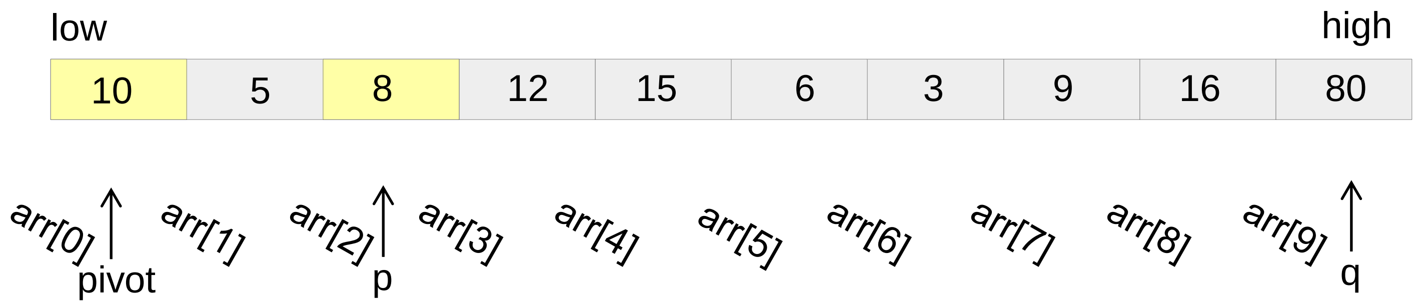
- arr[SIZE]



Concept

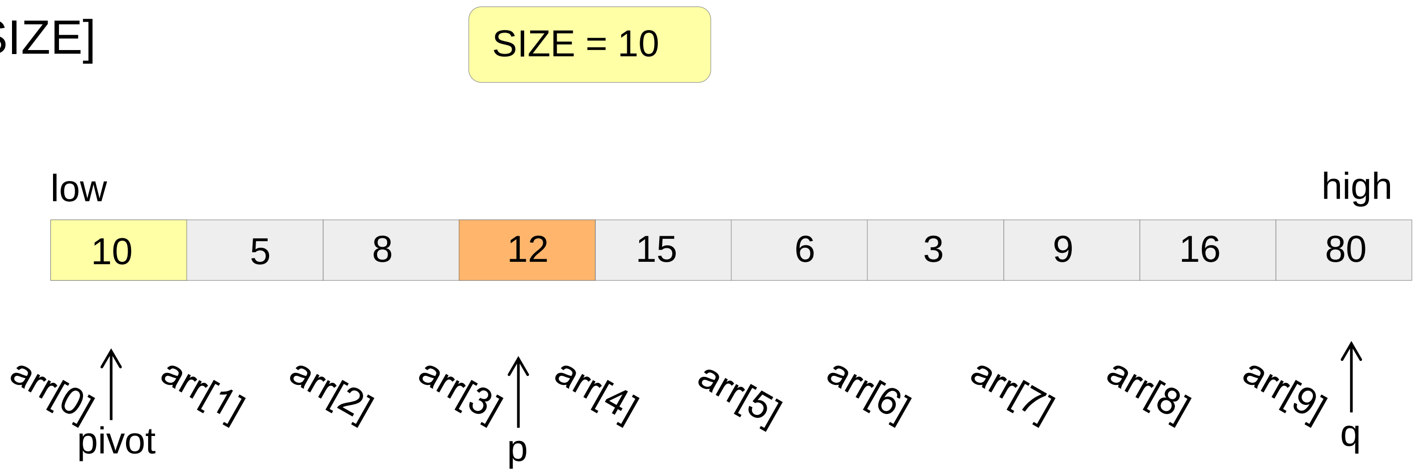
- arr[SIZE]

SIZE = 10



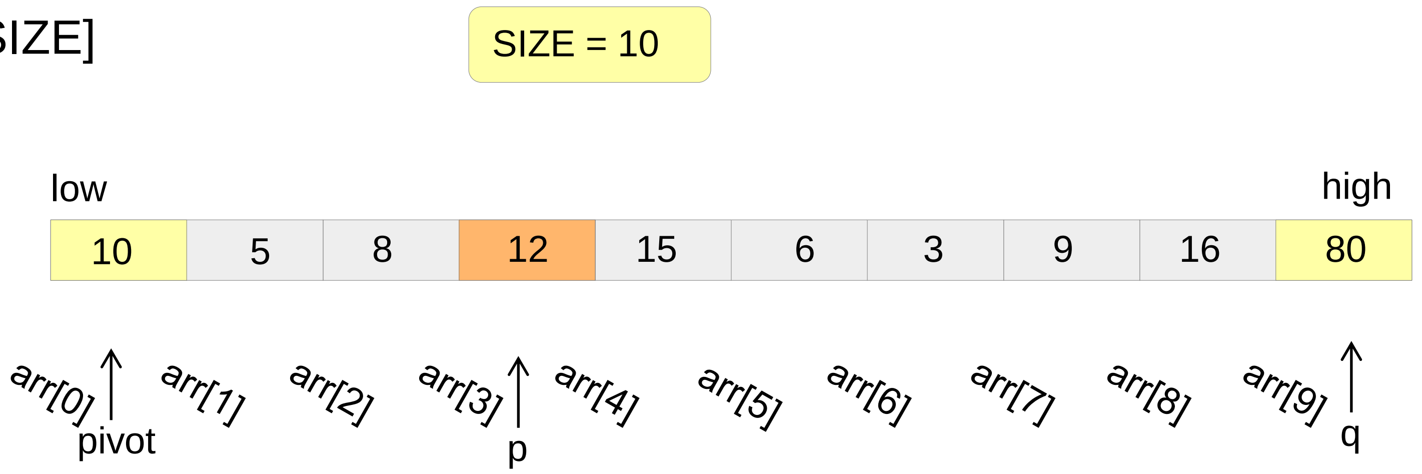
Concept

- arr[SIZE]



Concept

- arr[SIZE]

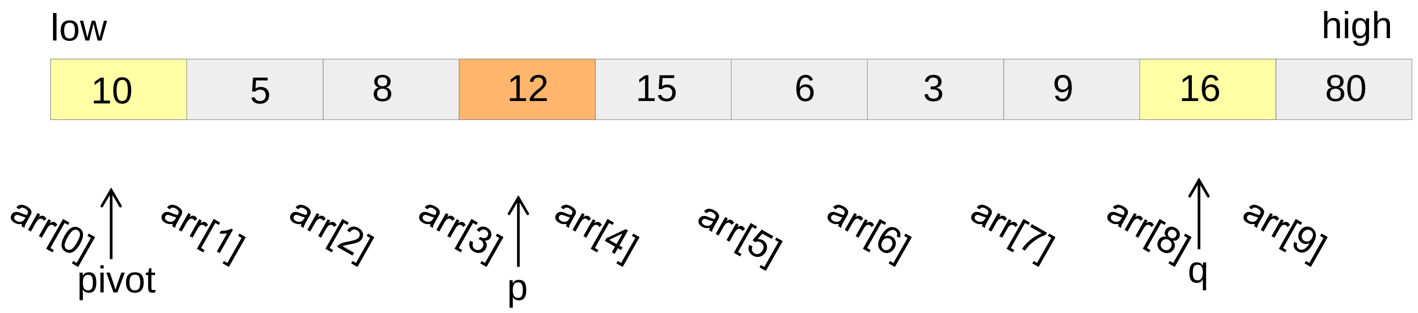


Data Structure – Sorting Techniques

Concept

- arr[SIZE]

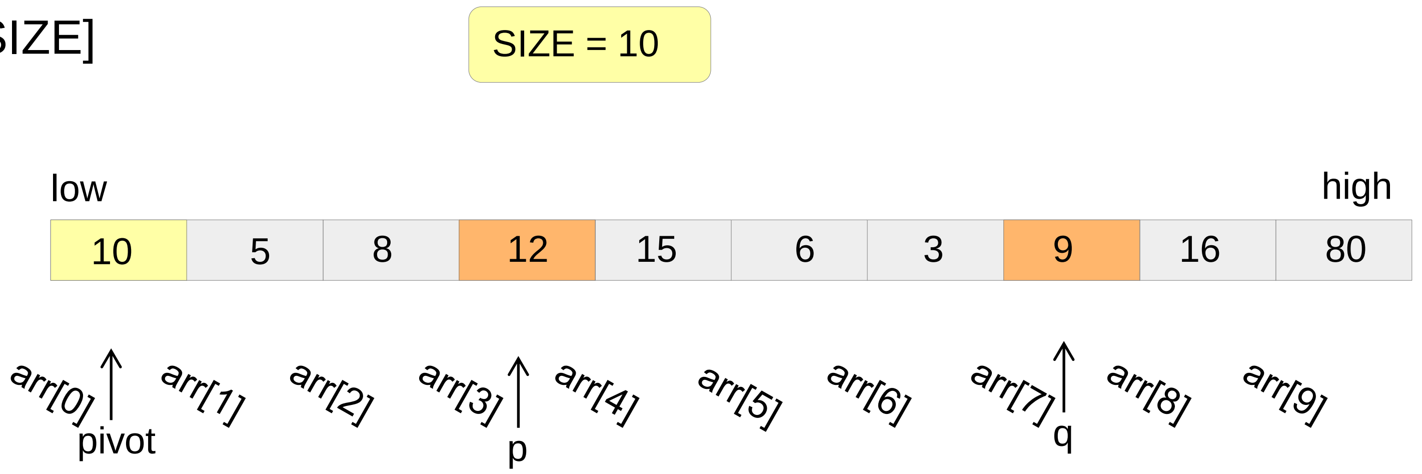
SIZE = 10



Data Structure – Sorting Techniques

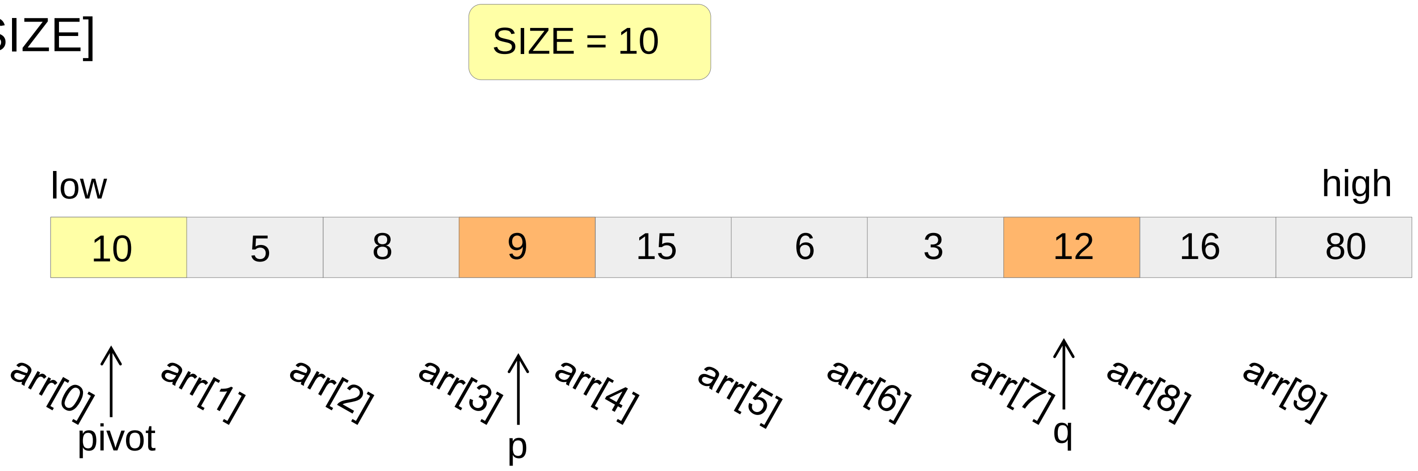
Concept

- arr[SIZE]



Concept

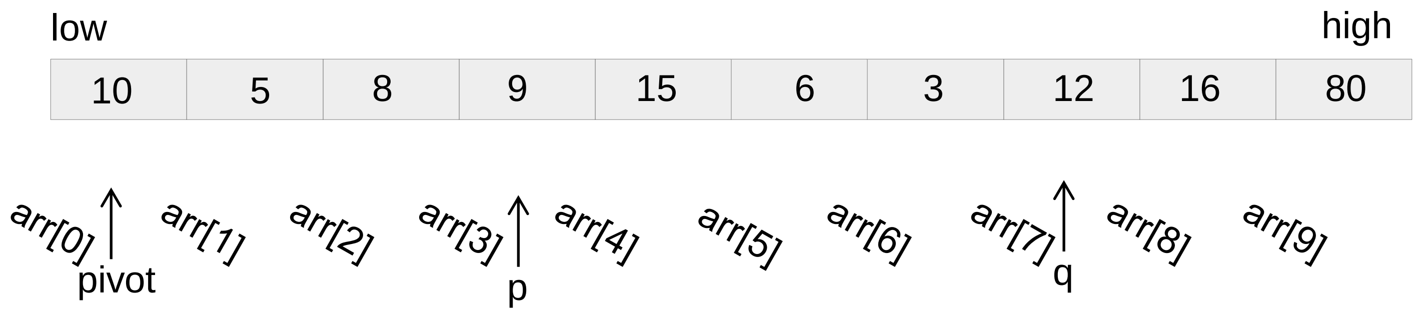
- arr[SIZE]



Concept

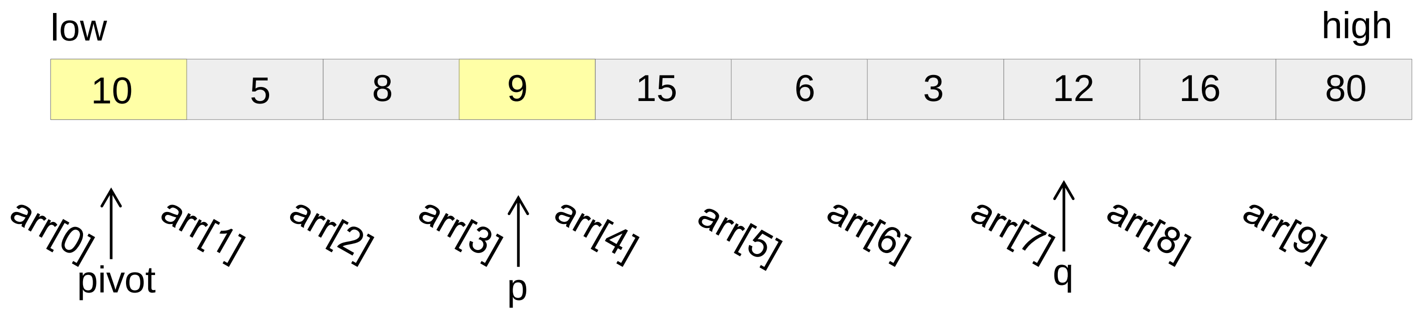
- arr[SIZE]

SIZE = 10



Concept

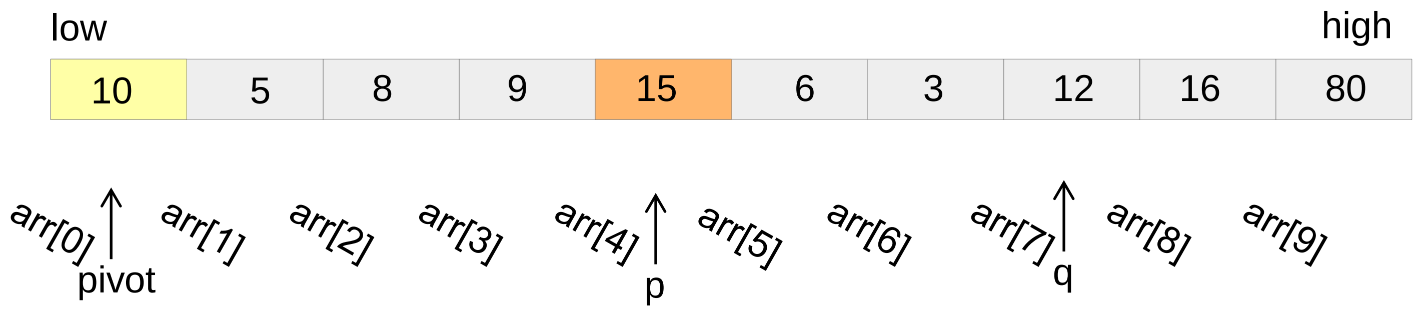
- arr[SIZE]



Concept

- arr[SIZE]

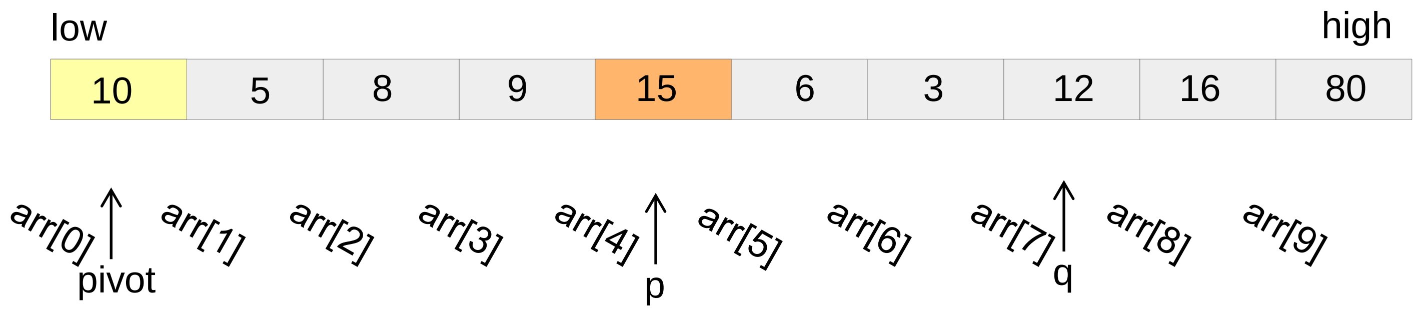
SIZE = 10



Concept

- arr[SIZE]

SIZE = 10

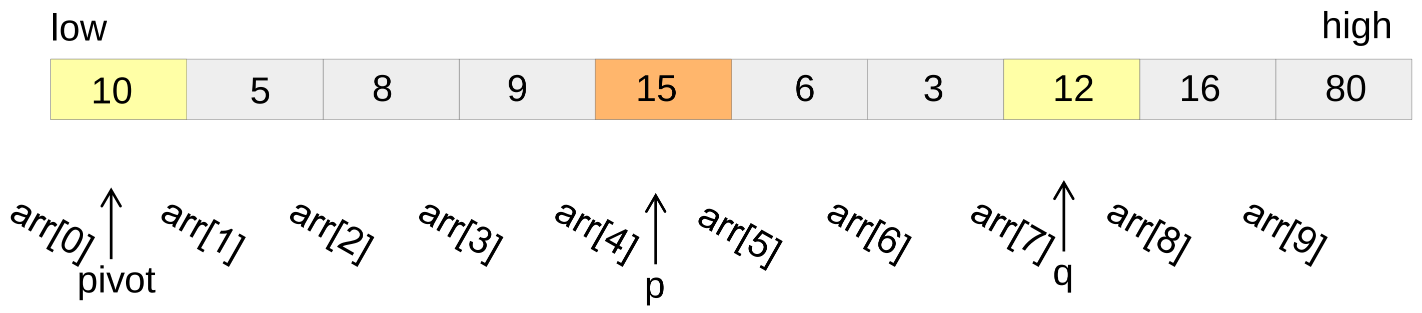


Data Structure – Sorting Techniques

Concept

- arr[SIZE]

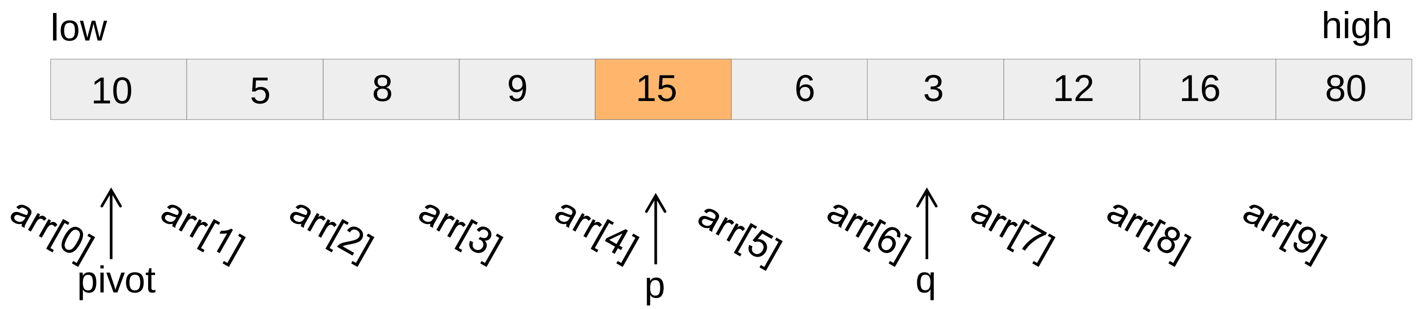
SIZE = 10



Concept

- arr[SIZE]

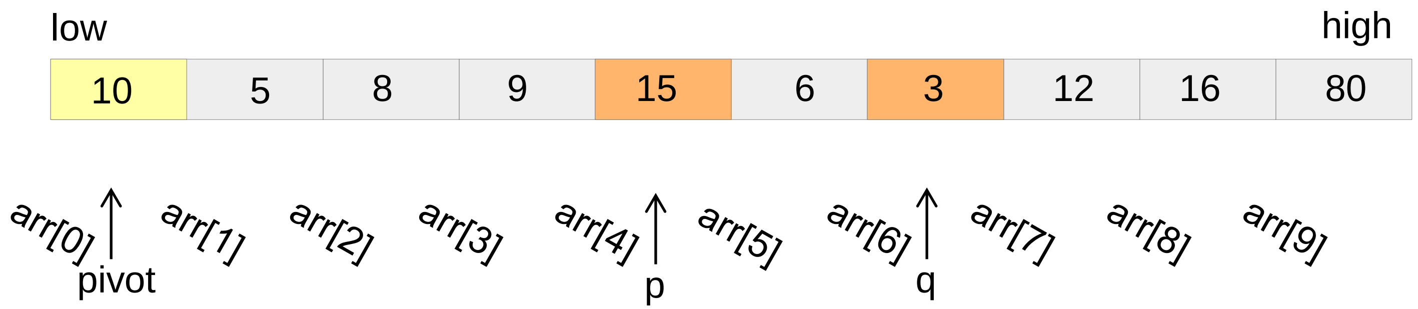
SIZE = 10



Concept

- arr[SIZE]

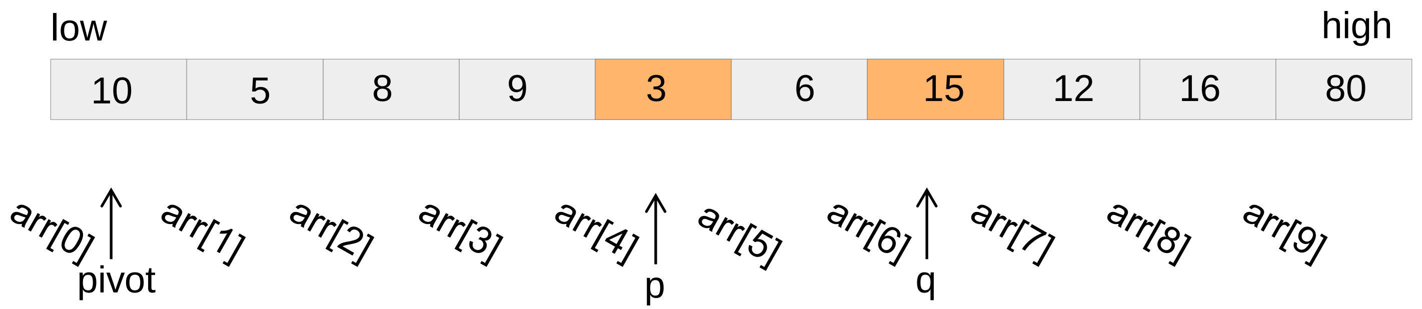
SIZE = 10



Concept

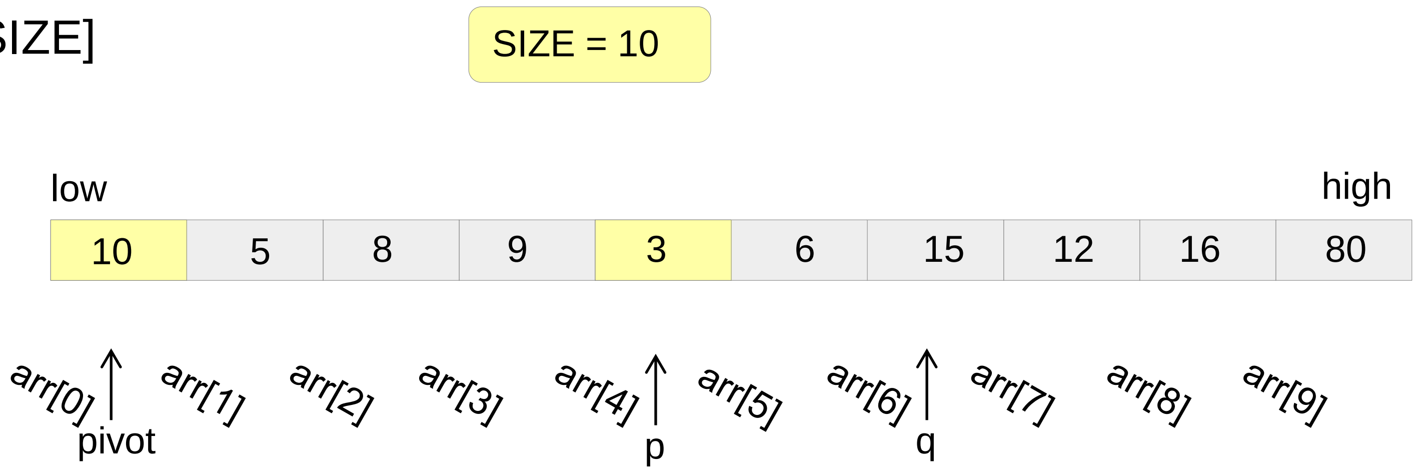
- arr[SIZE]

SIZE = 10



Concept

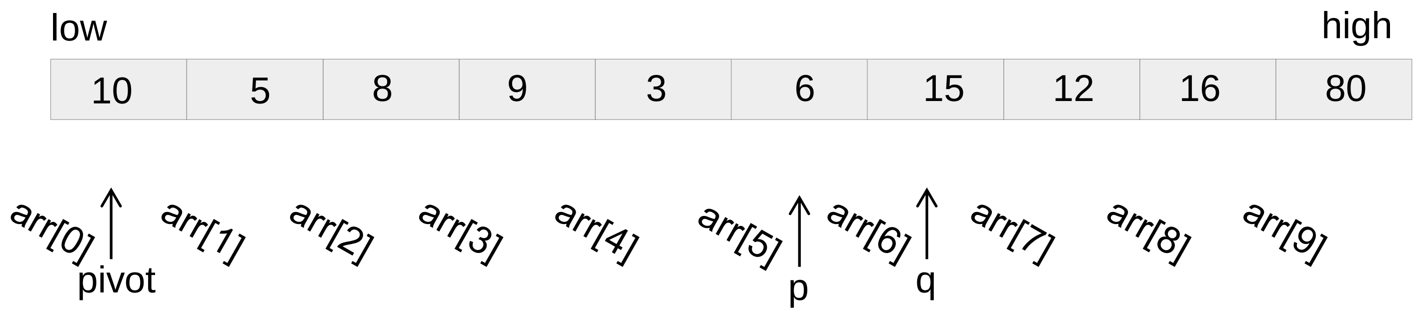
- arr[SIZE]



Concept

- arr[SIZE]

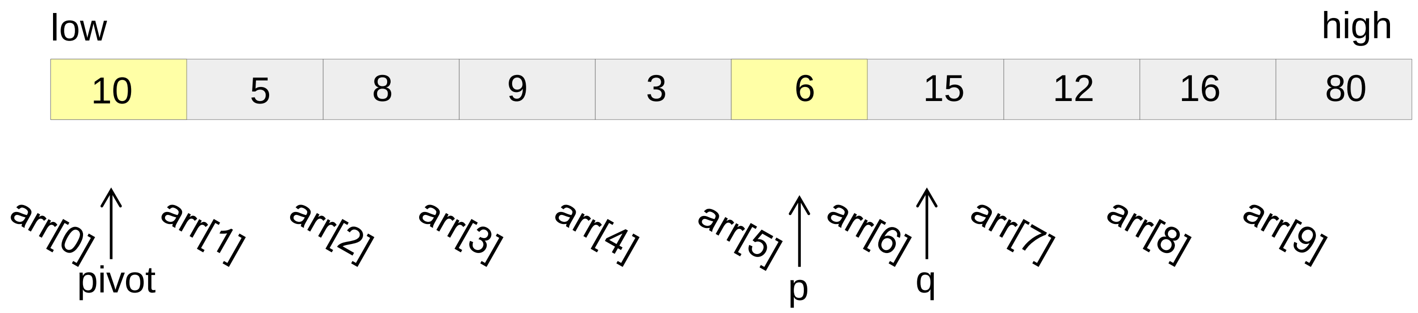
SIZE = 10



Concept

- arr[SIZE]

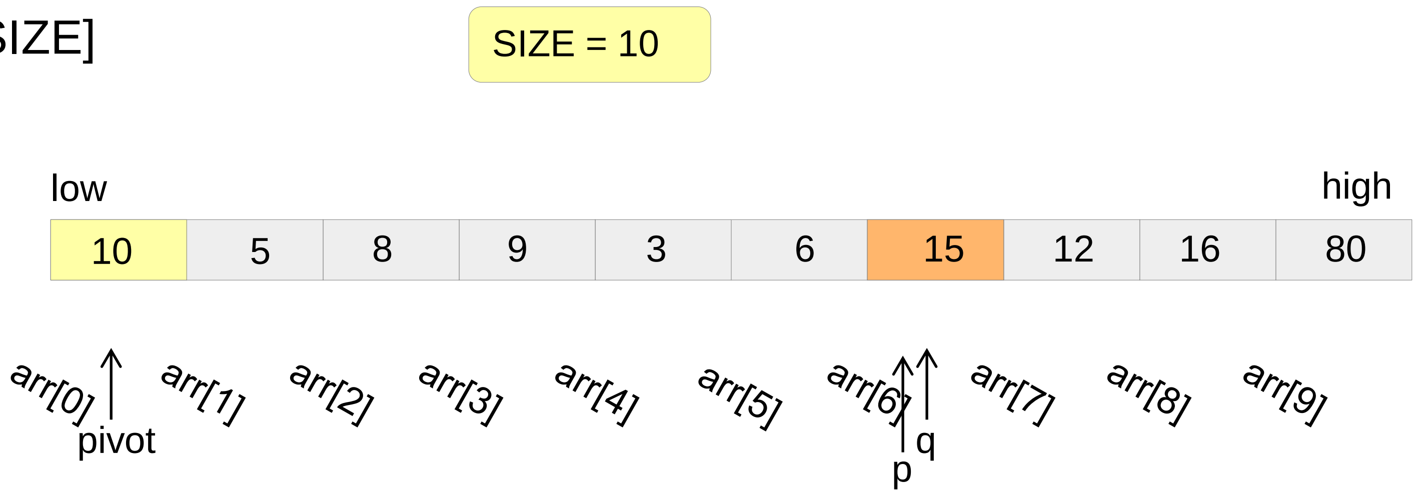
SIZE = 10



Data Structure – Sorting Techniques

Concept

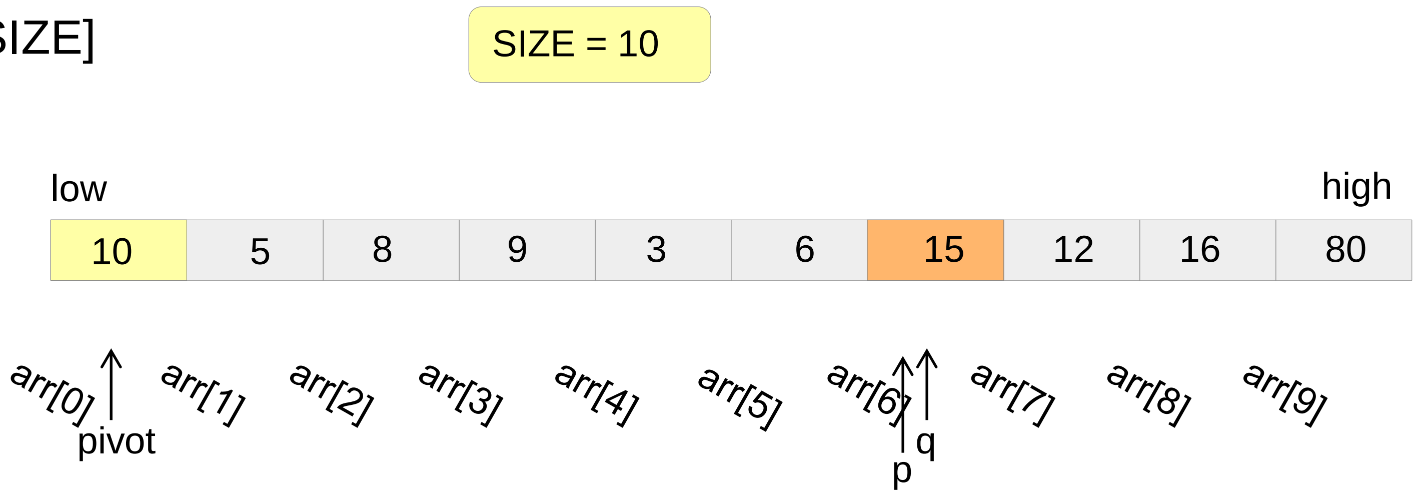
- arr[SIZE]



Data Structure – Sorting Techniques

Concept

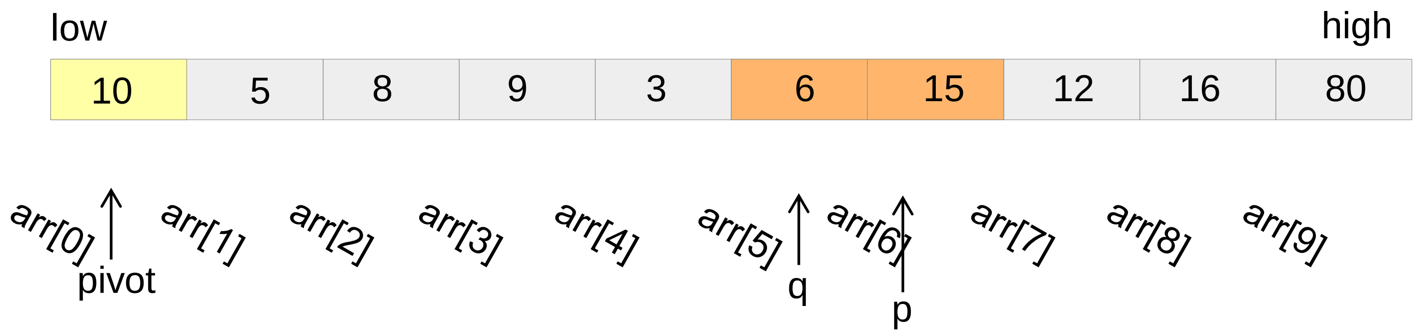
- arr[SIZE]



Concept

- arr[SIZE]

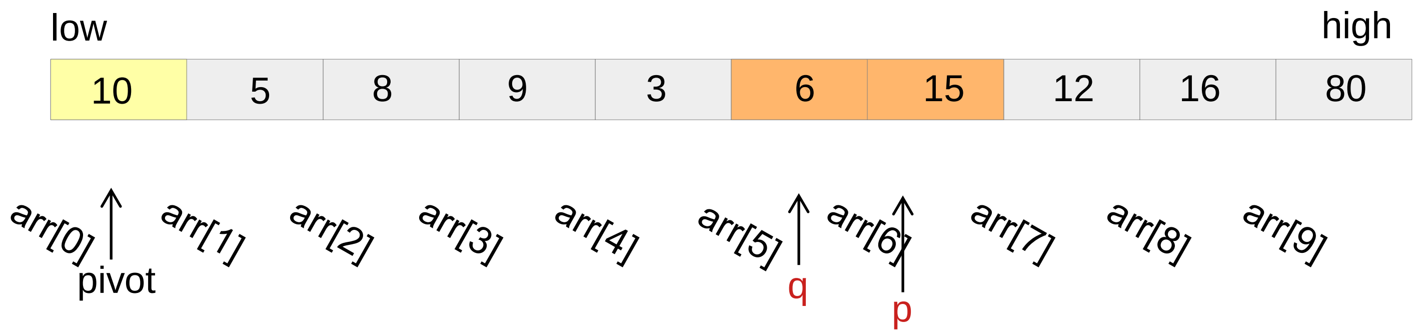
SIZE = 10



Concept

- arr[SIZE]

SIZE = 10

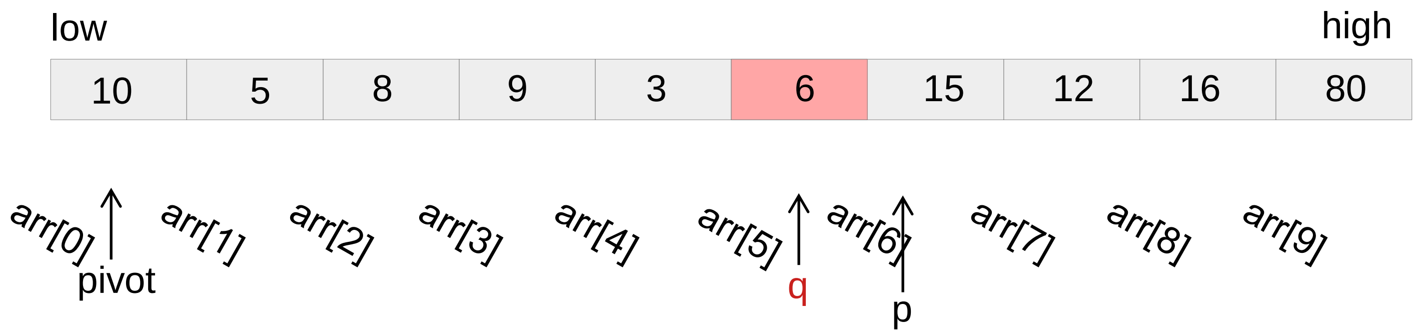


Data Structure – Sorting Techniques

Concept

- arr[SIZE]

SIZE = 10

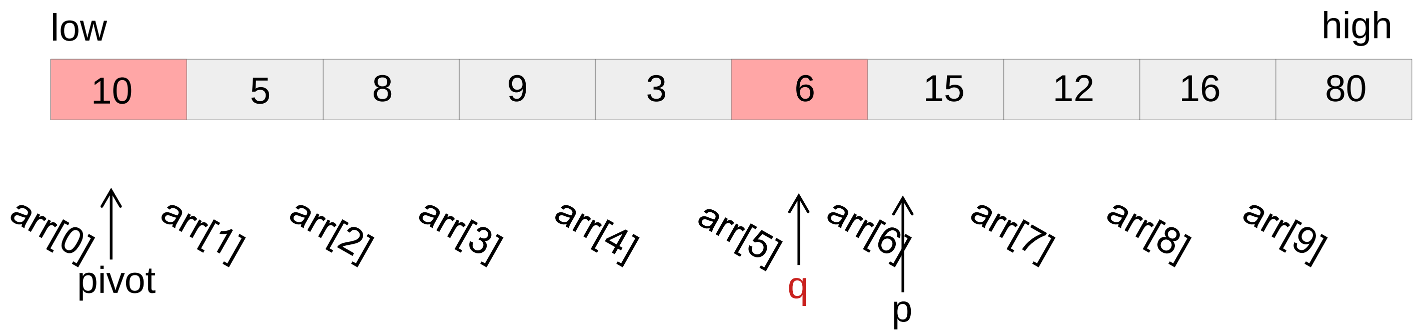


Data Structure – Sorting Techniques

Concept

- arr[SIZE]

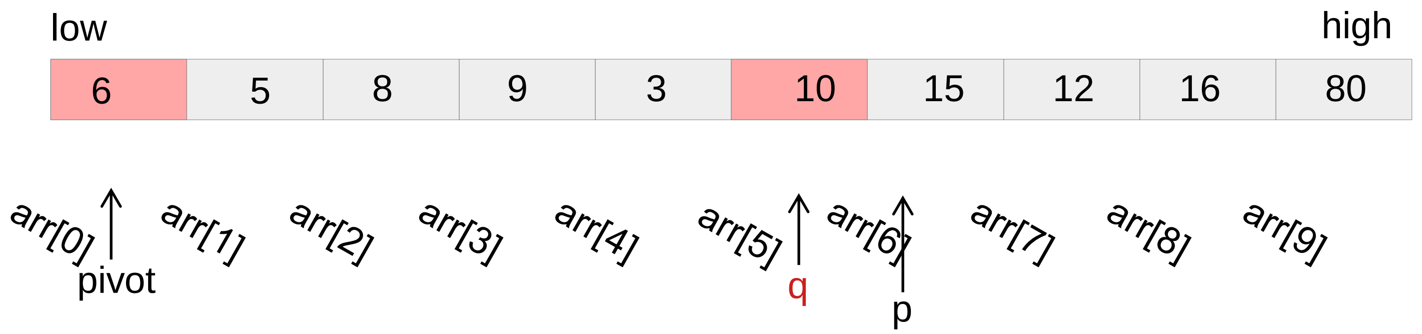
SIZE = 10



Concept

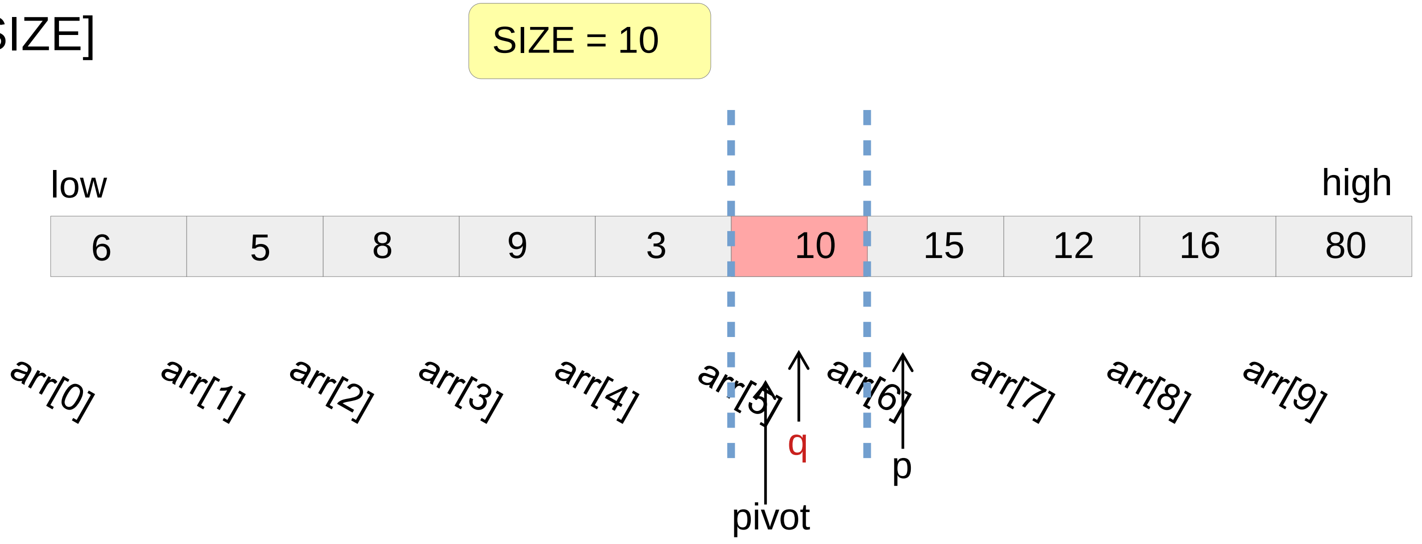
- arr[SIZE]

SIZE = 10



Concept

- arr[SIZE]

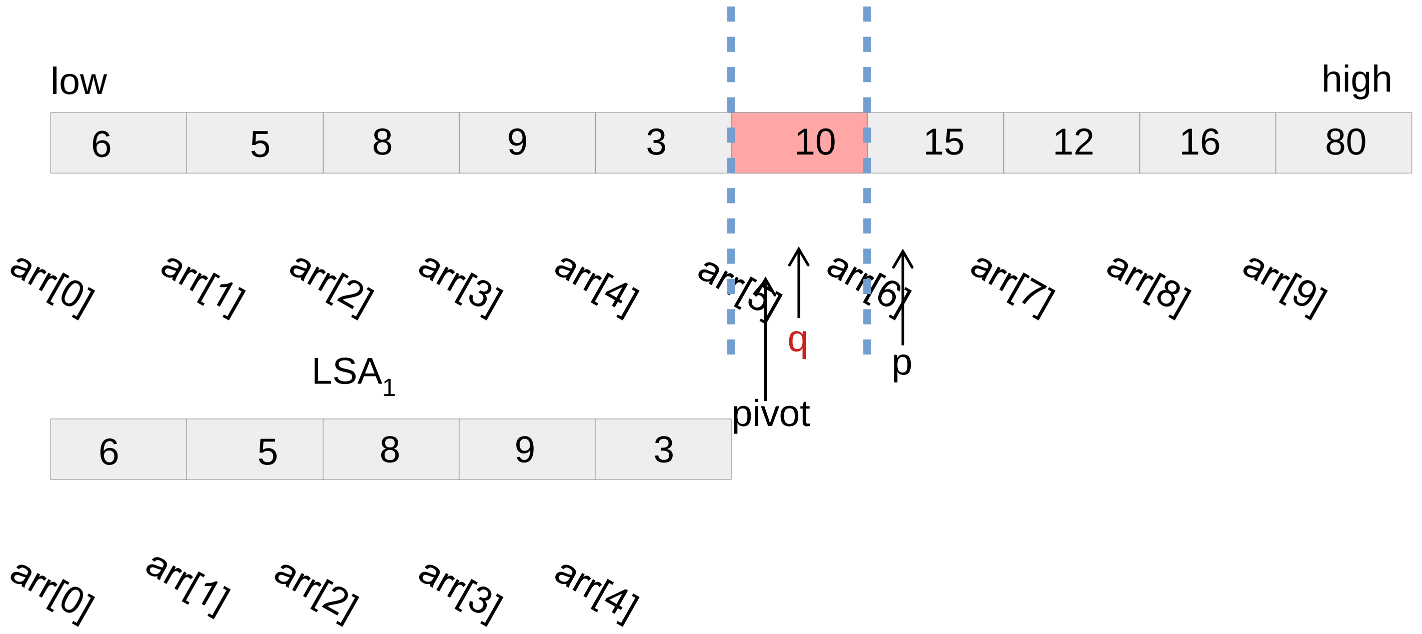


Data Structure – Sorting Techniques

Concept

- arr[SIZE]

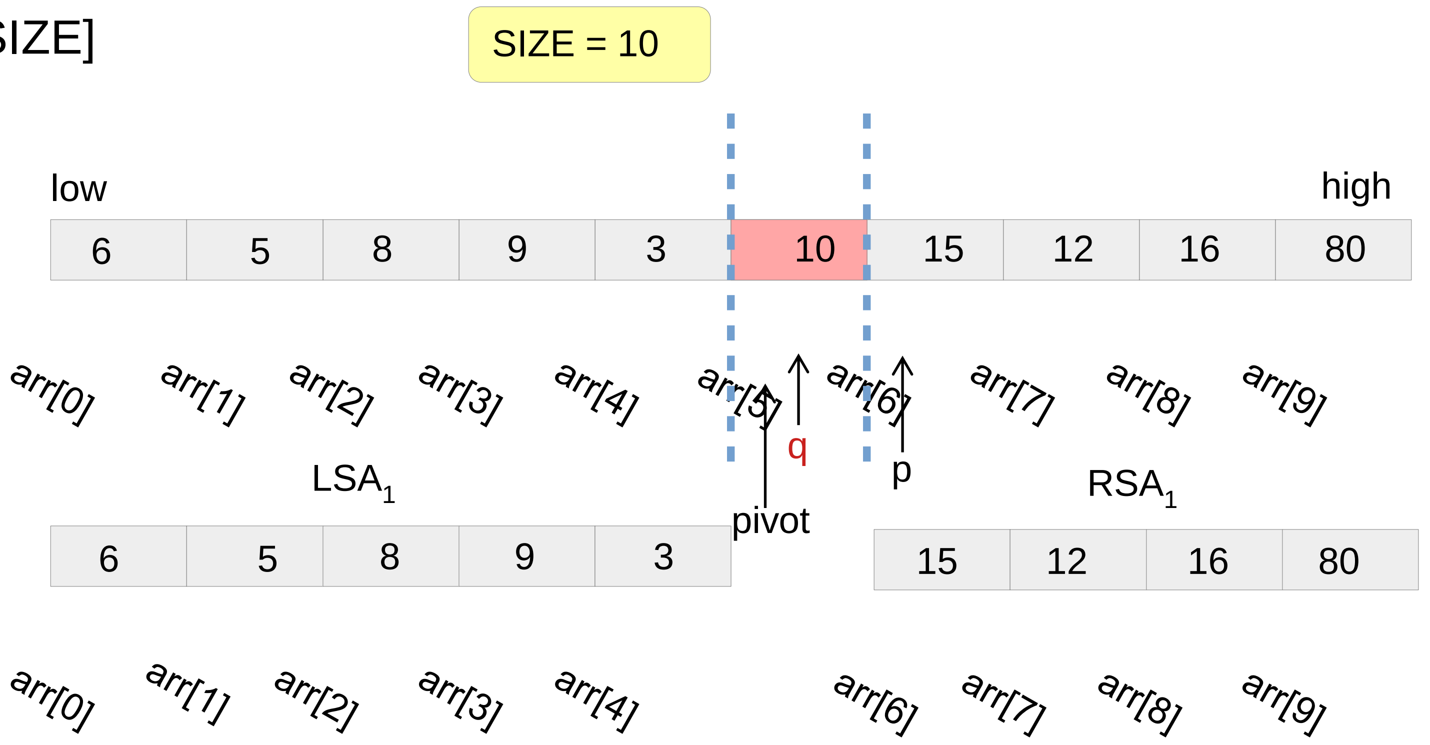
SIZE = 10



Data Structure – Sorting Techniques

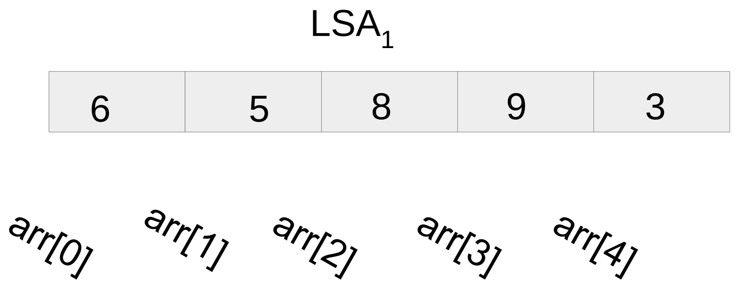
Concept

- arr[SIZE]

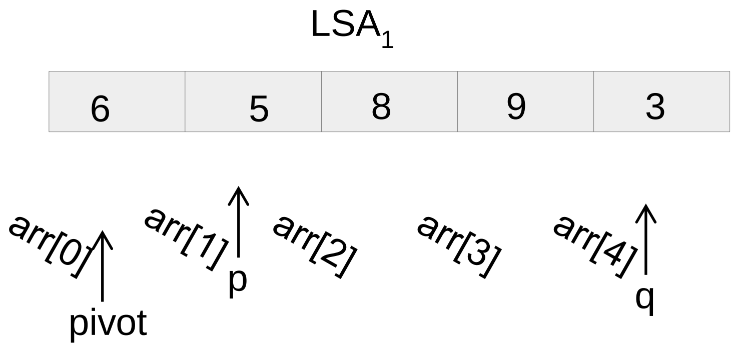


Data Structure – Sorting Techniques

Concept

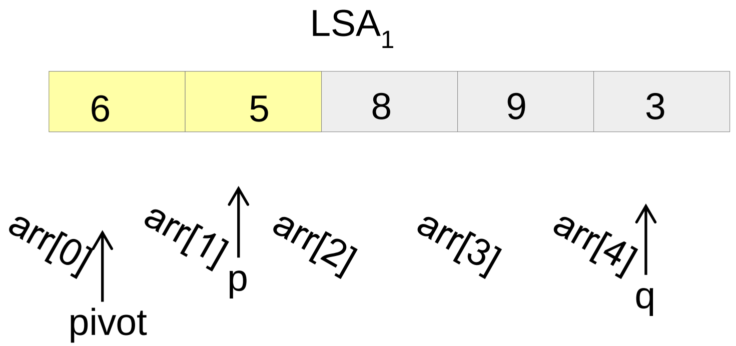


Concept



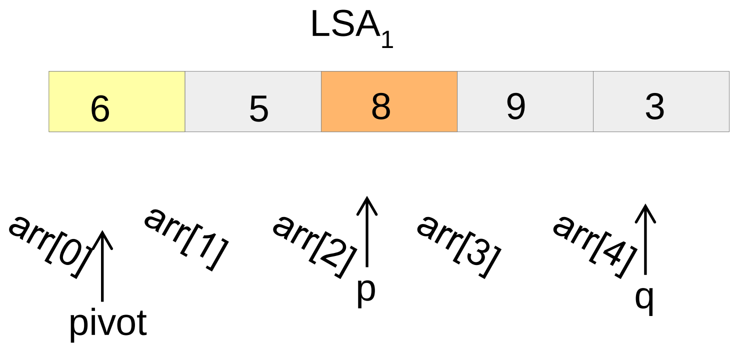
Data Structure – Sorting Techniques

Concept



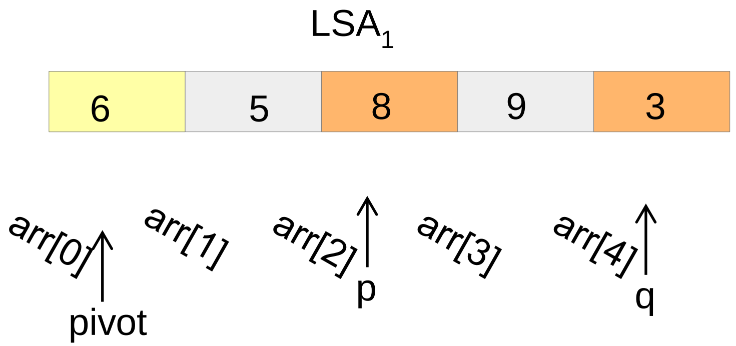
Data Structure – Sorting Techniques

Concept

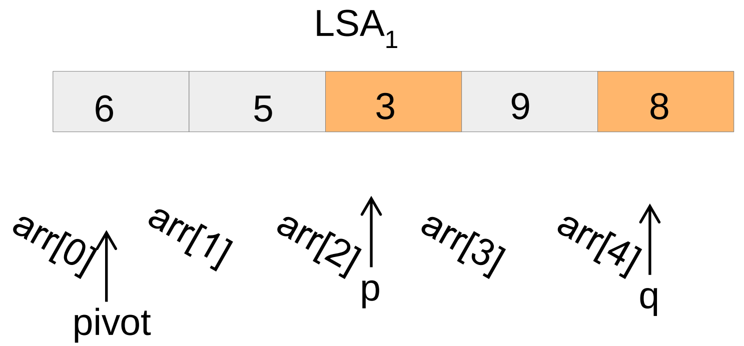


Data Structure – Sorting Techniques

Concept

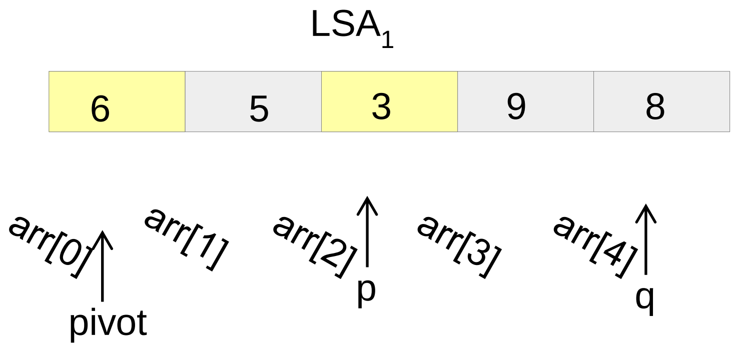


Concept

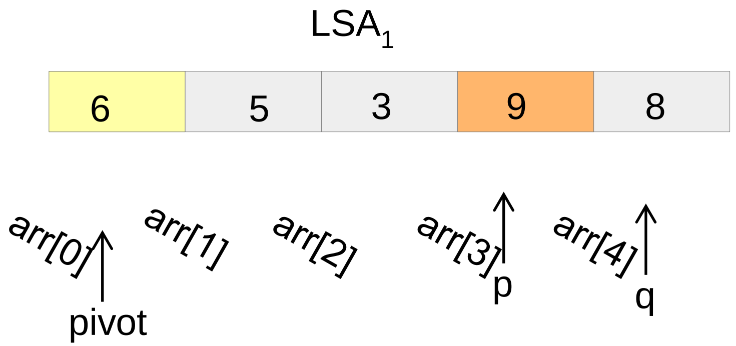


Data Structure – Sorting Techniques

Concept

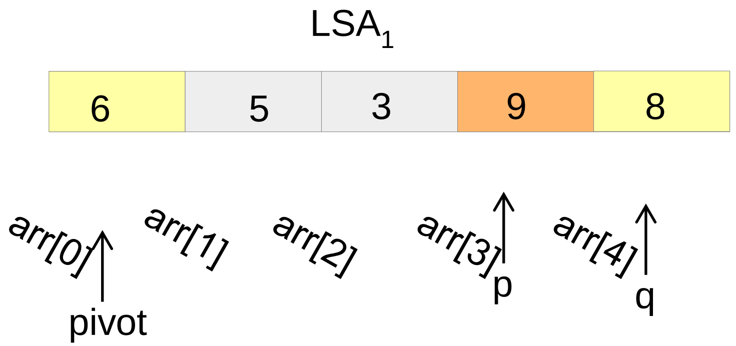


Concept

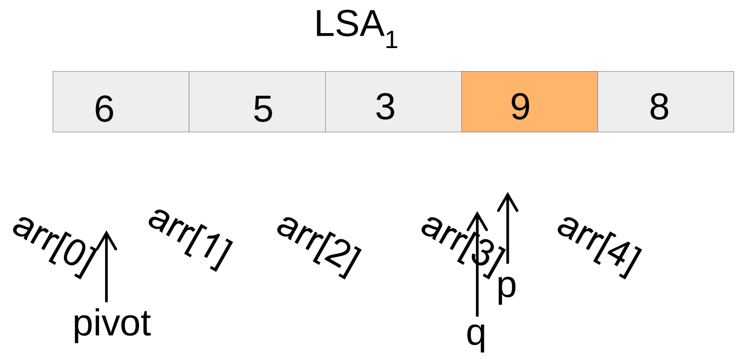


Data Structure – Sorting Techniques

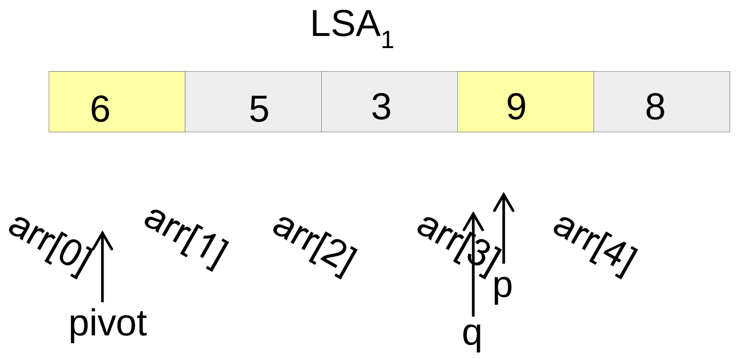
Concept



Concept

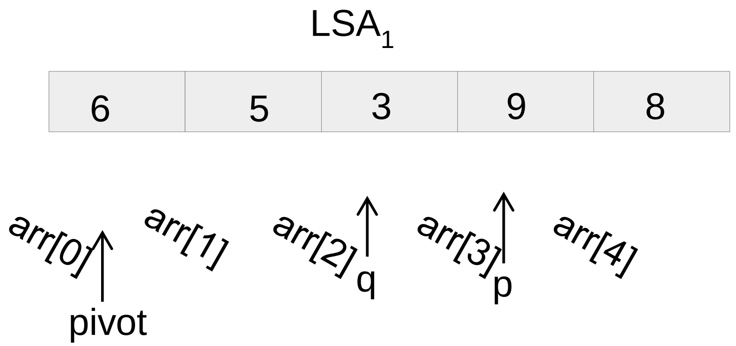


Concept



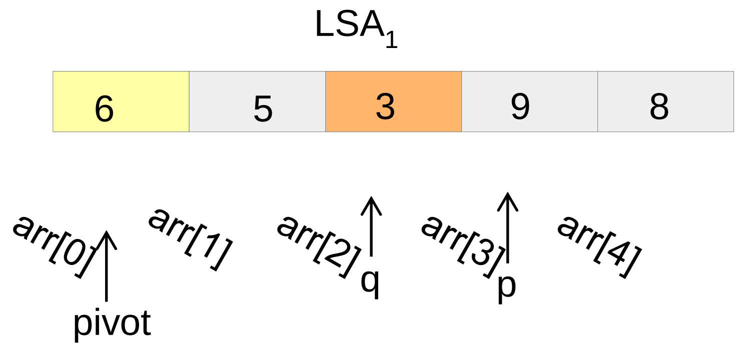
Data Structure – Sorting Techniques

Concept



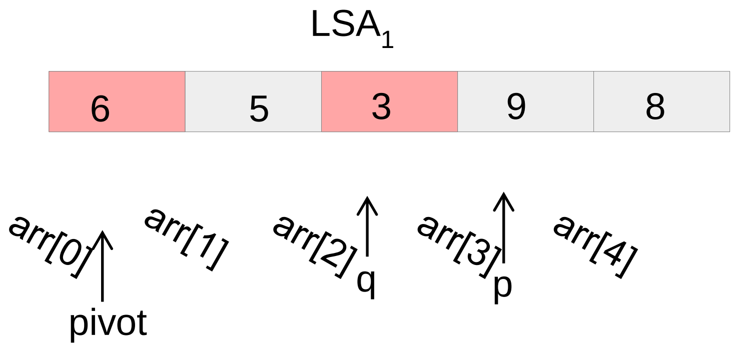
Data Structure – Sorting Techniques

Concept

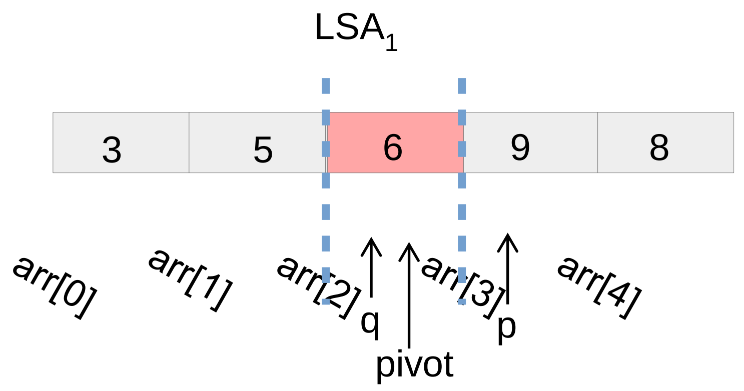


Data Structure – Sorting Techniques

Concept

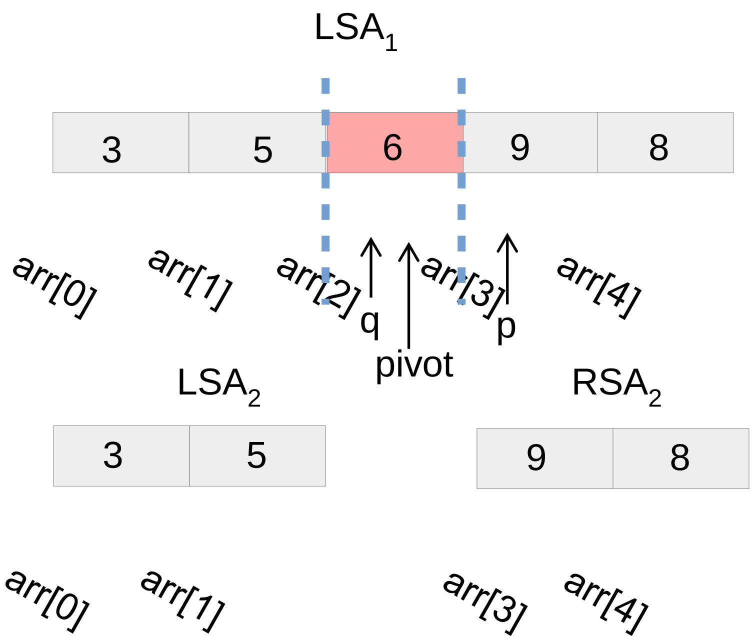


Concept

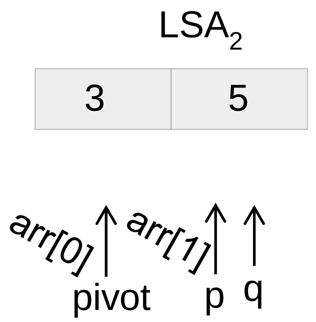


Data Structure – Sorting Techniques

Concept

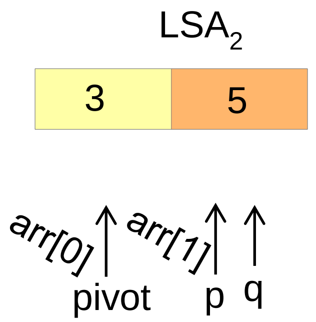


Concept

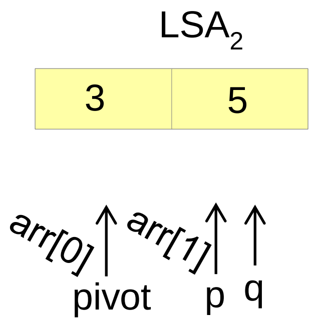


Data Structure – Sorting Techniques

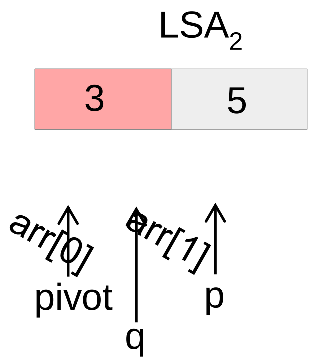
Concept



Concept

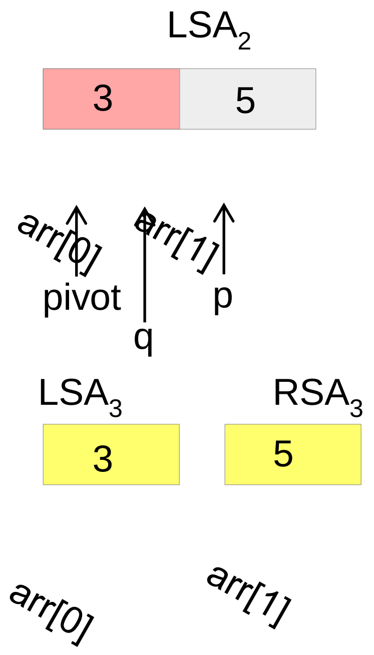


Concept



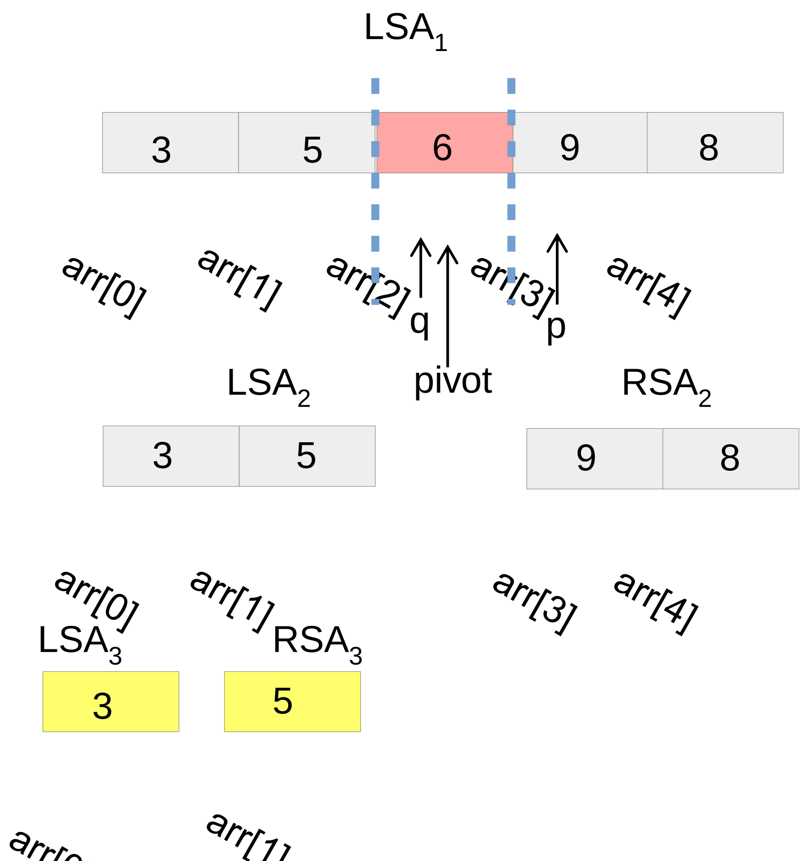
Data Structure – Sorting Techniques

Concept

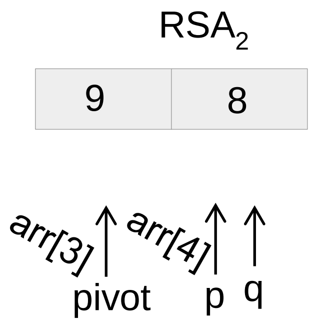


Data Structure – Sorting Techniques

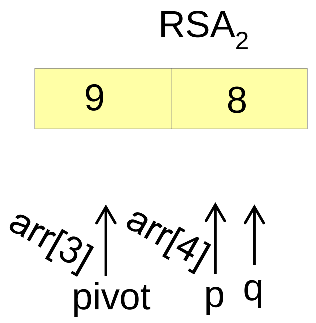
Concept



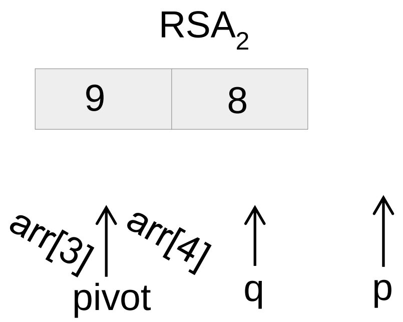
Concept



Concept



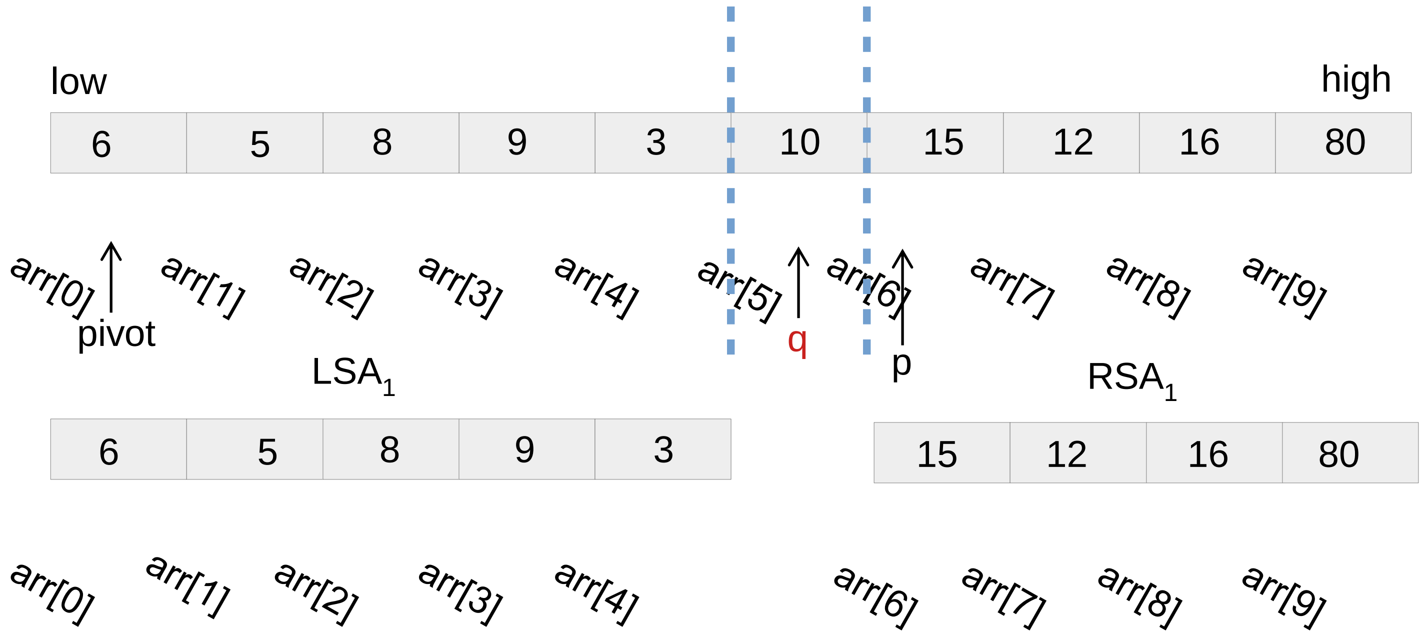
Concept



Concept

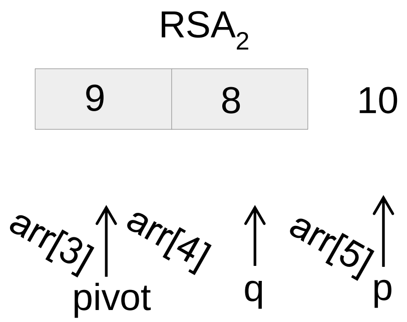
- arr[SIZE]

SIZE = 10



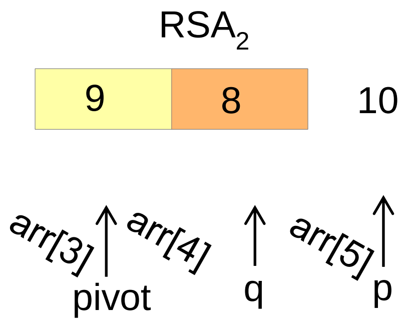
Data Structure – Sorting Techniques

Concept

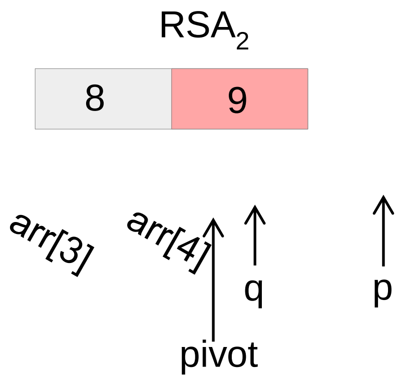


Data Structure – Sorting Techniques

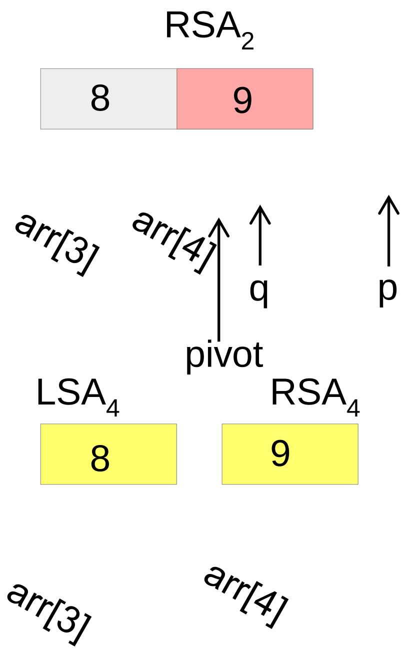
Concept



Concept

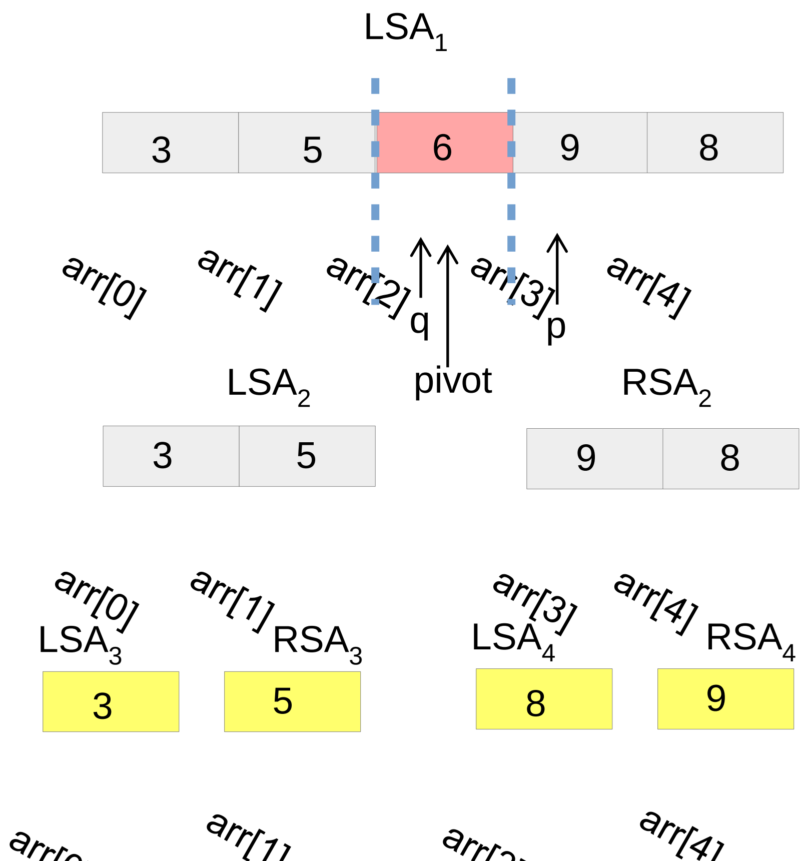


Concept

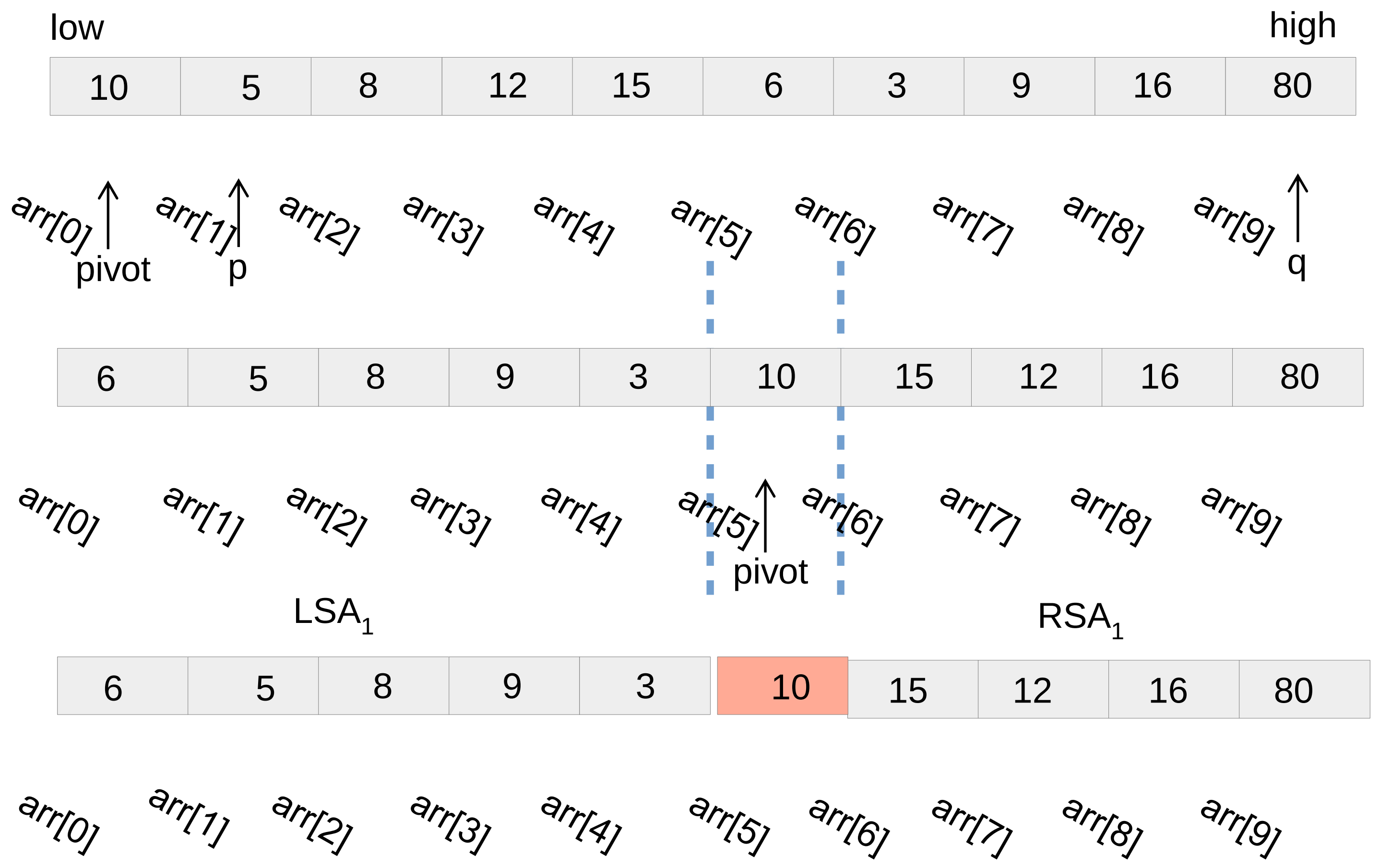


Data Structure – Sorting Techniques

Concept

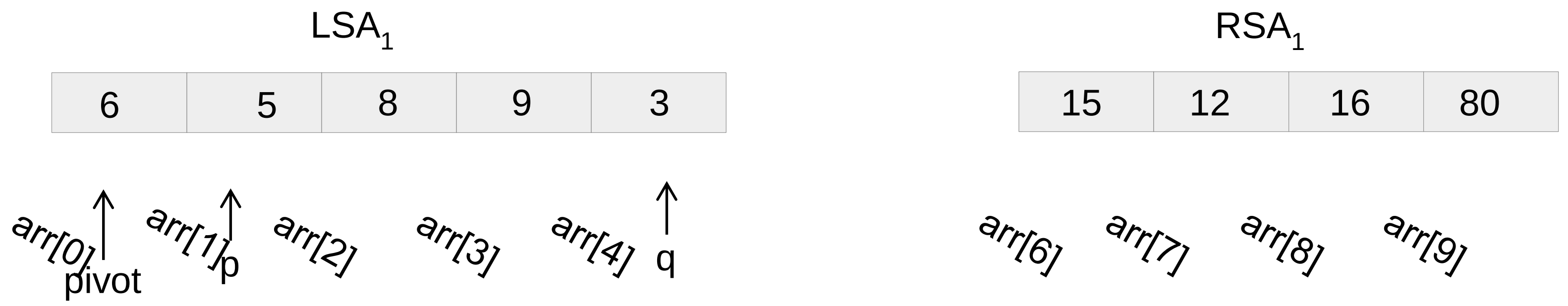


Concept

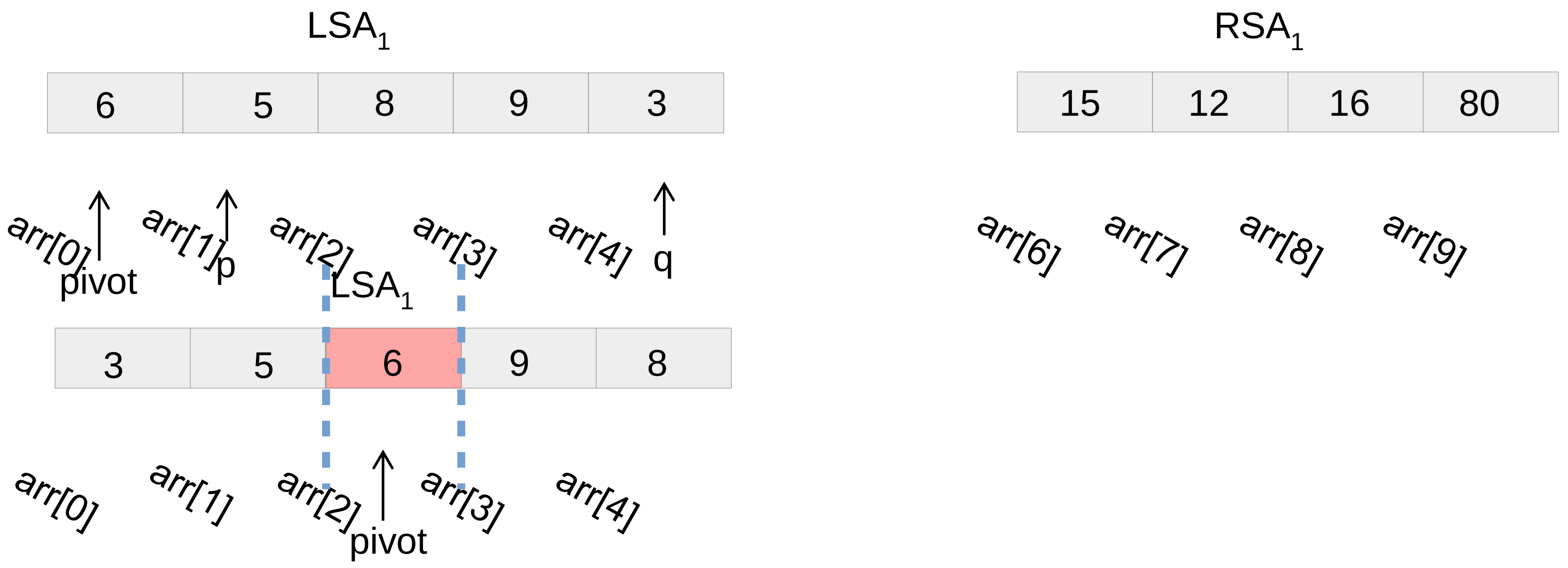

low
high
10 5 8 12 15 6 3 9 16 80
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7] arr[8] arr[9]
pivot
p
q
6 5 8 9 3 10 15 12 16 80
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7] arr[8] arr[9]
pivot
LSA1
RSA1
6 5 8 9 3 10 15 12 16 80
arr[0] arr[1] arr[2] arr[3] arr[4] arr[5] arr[6] arr[7] arr[8] arr[9]

Data Structure – Sorting Techniques

Concept

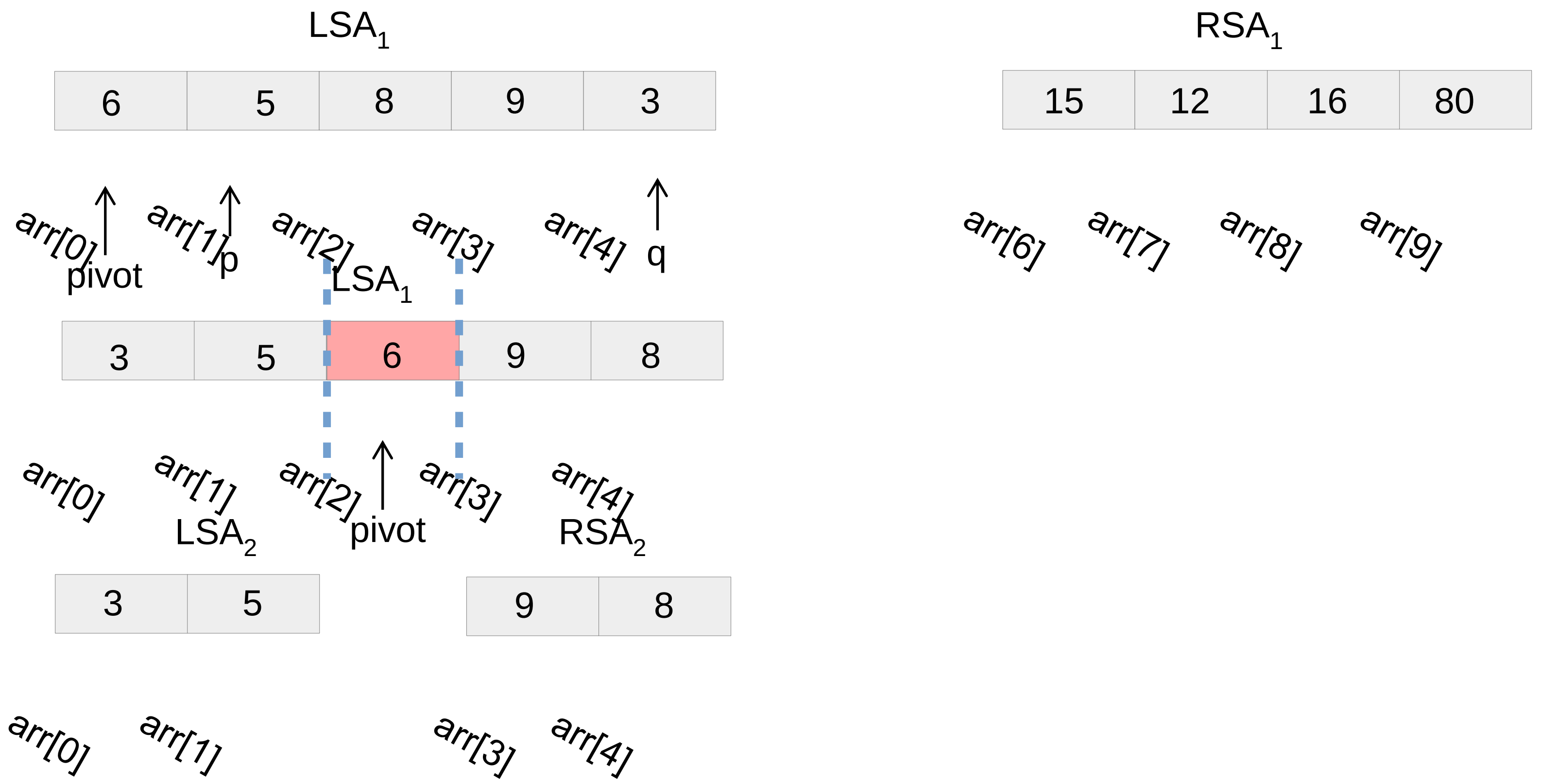


Concept


LSA1
6
5
8
9
3
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
pivot
p
q
RSA1
15
12
16
80
arr[6]
arr[7]
arr[8]
arr[9]
LSA1
3
5
6
9
8
arr[0]
arr[1]
arr[2]
arr[3]
arr[4]
pivot

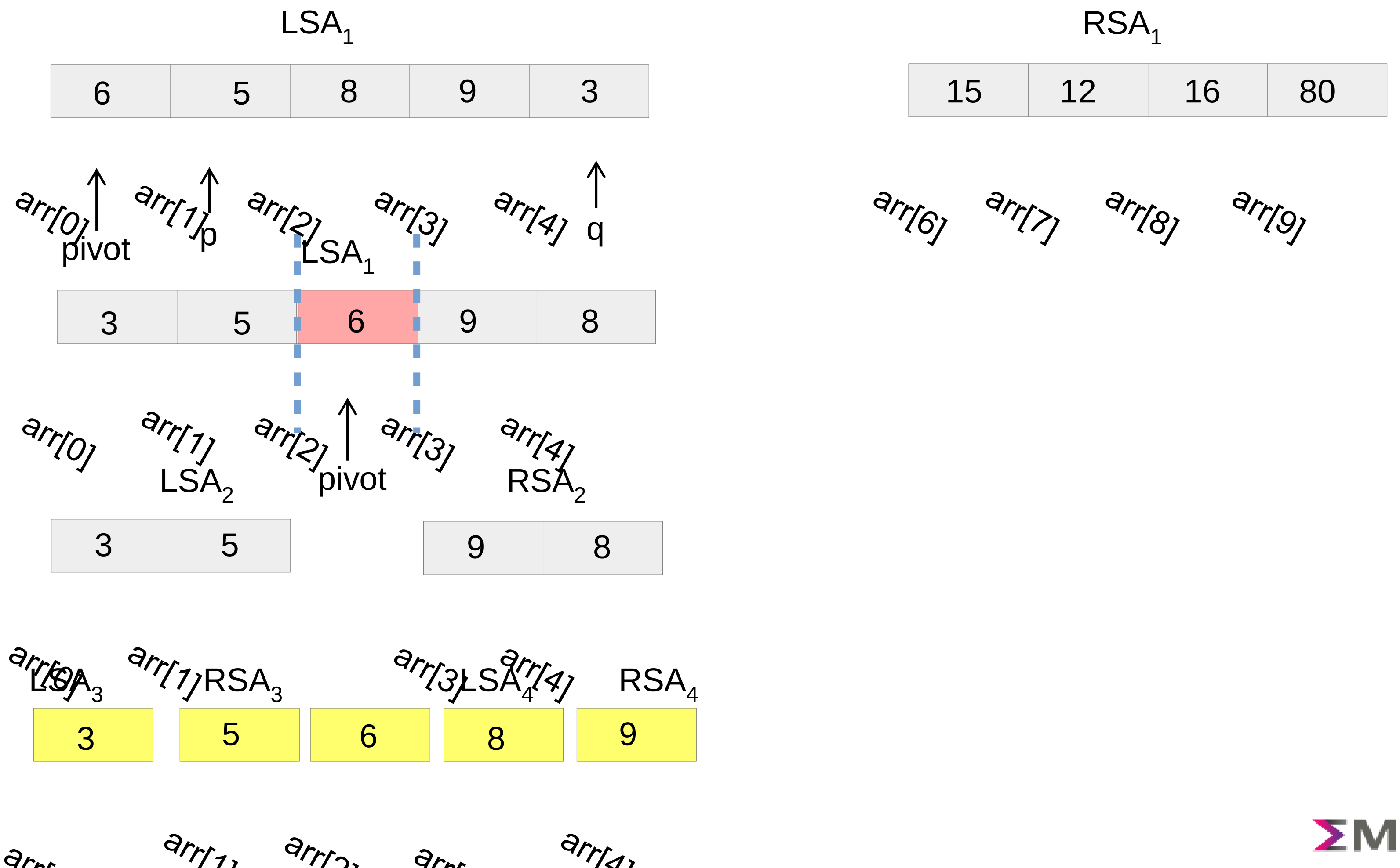
Data Structure – Sorting Techniques

Concept



Data Structure – Sorting Techniques

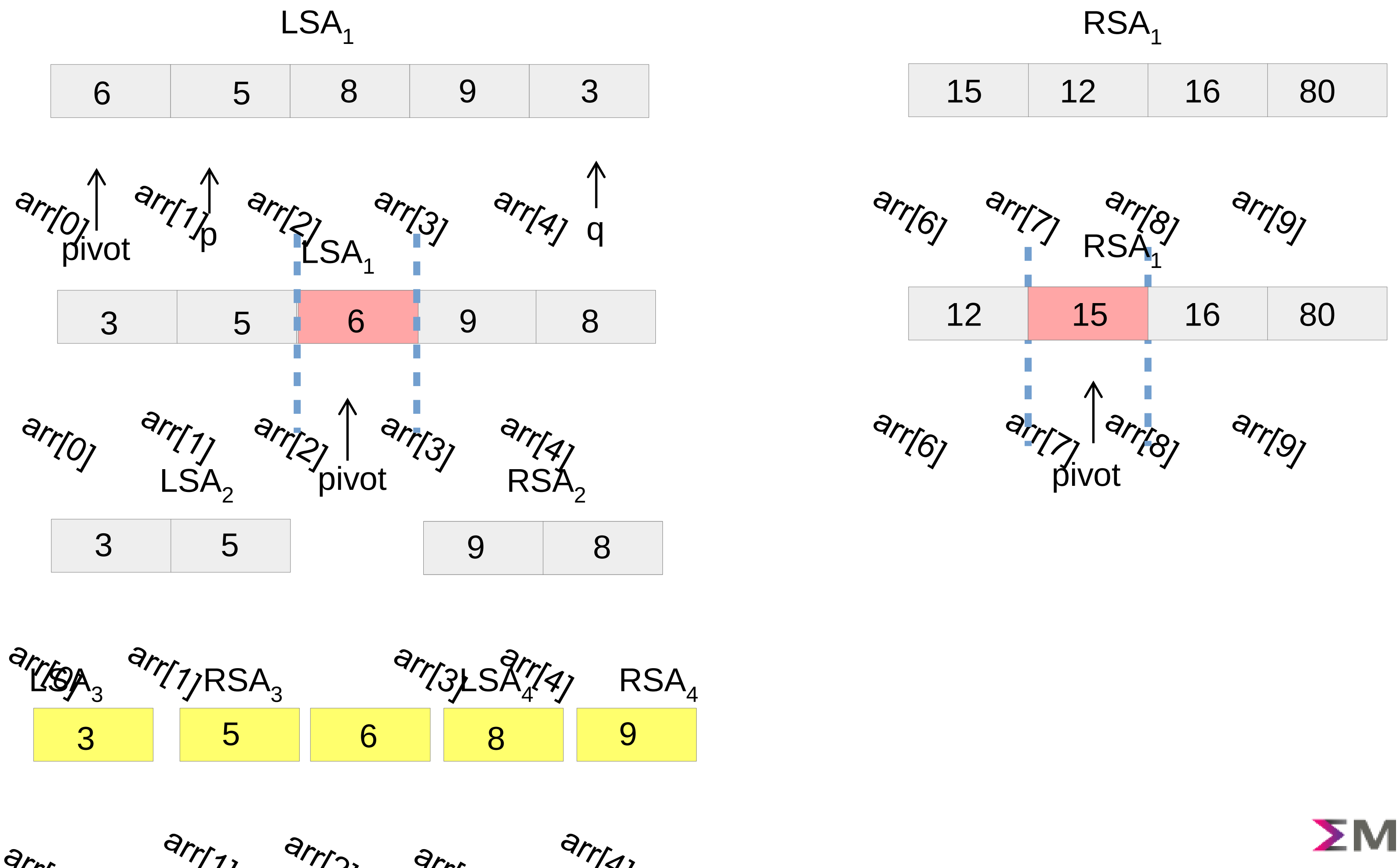
Concept



LSA1
6 5 8 9 3
arr[0] arr[1] arr[2] arr[3] arr[4]
pivot
p
q
RSA1
15 12 16 80
arr[6] arr[7] arr[8] arr[9]
LSA1
3 5 6 9 8
arr[0] arr[1] arr[2] arr[3] arr[4]
pivot
LSA2
3 5
RSA2
9 8
LSA3 RSA3 LSA4 RSA4
arr[0] arr[1] arr[3] arr[4]
3 5 6 8 9

Data Structure – Sorting Techniques

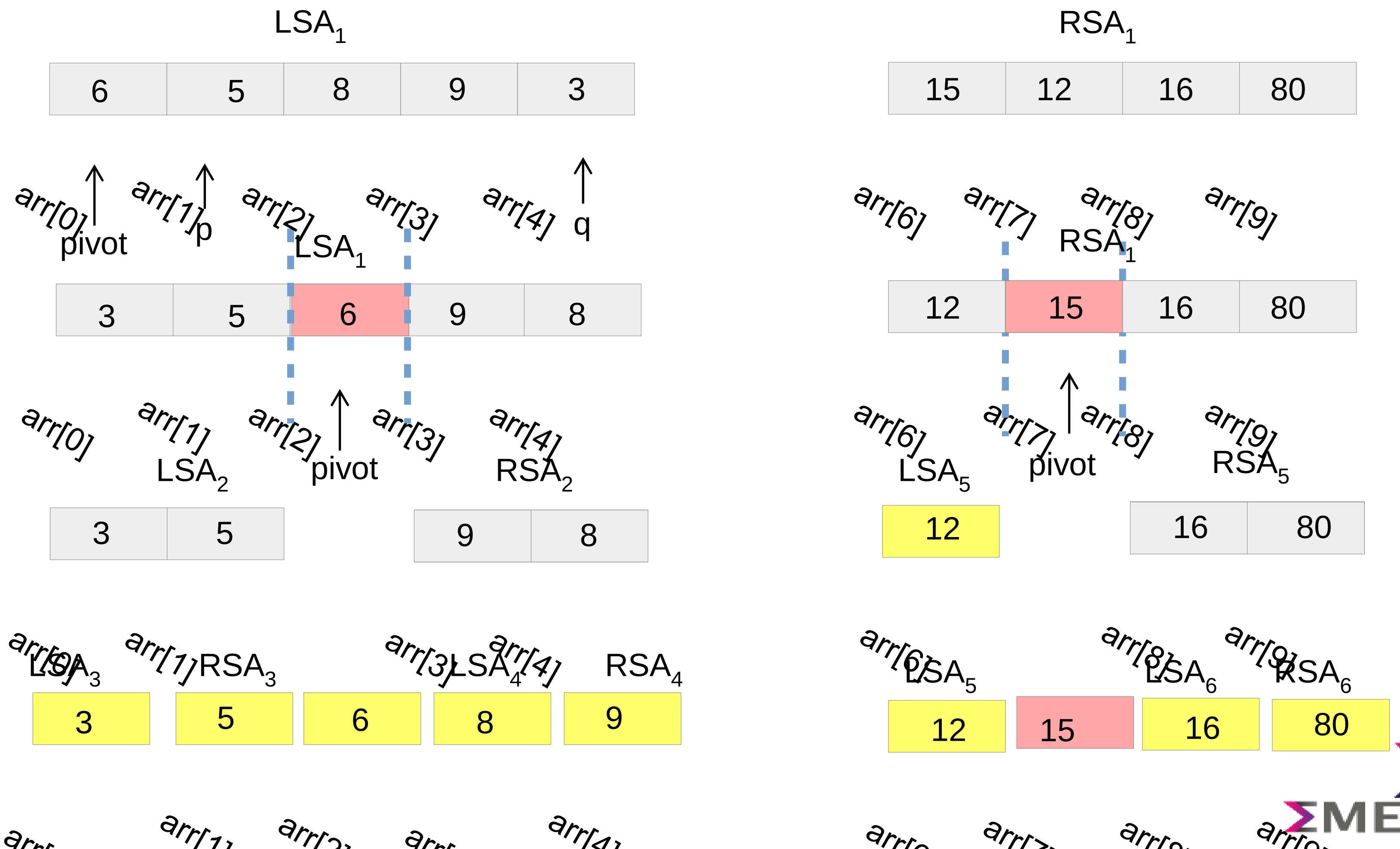
Concept



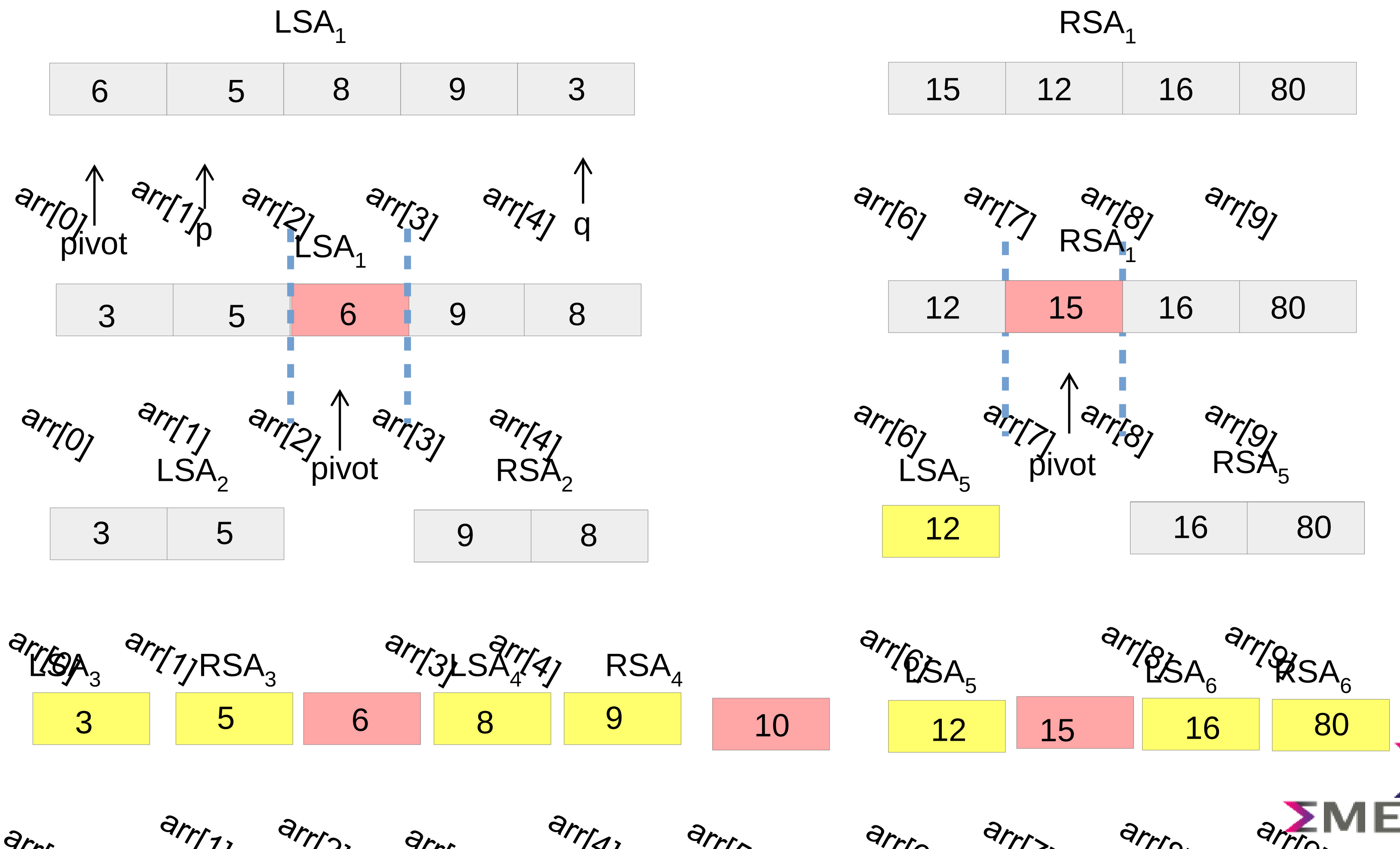
LSA1
6 5 8 9 3
arr[0] arr[1] arr[2] arr[3] arr[4]
pivot
p
q
RSA1
15 12 16 80
arr[6] arr[7] arr[8] arr[9]
LSA1
3 5 6 9 8
arr[0] arr[1] arr[2] arr[3] arr[4]
pivot
RSA1
12 15 16 80
arr[6] arr[7] arr[8] arr[9]
pivot
LSA2
3 5
RSA2
9 8
arr[0] arr[1] arr[3] arr[4]
LSA3 RSA3 LSA4 RSA4
3 5 6 8 9

Data Structure – Sorting Techniques

Concept



Concept


LSA1
6 5 8 9 3
RSA1
15 12 16 80
arr[0] pivot
arr[1] p
arr[2]
arr[3]
arr[4] q
arr[6]
arr[7]
arr[8]
arr[9]
LSA1
3 5 6 9 8
RSA1
12 15 16 80
arr[0]
arr[1]
arr[2]
pivot
arr[3]
arr[4]
LSA2
RSA2
3 5
9 8
arr[6]
arr[7]
pivot
arr[8]
arr[9]
LSA5
RSA5
12
16 80
arr[0] LSA3
arr[1] RSA3
arr[3] LSA4
arr[4]
RSA4
3 5 6 8 9 10
arr[6] LSA5
arr[8] LSA6
arr[9] RSA6
12 15 16 80
arr[1]

Algorithm

- quick_sort(arr,low,high)

```
If (low < high)
        index = partition(arr,low,high)
    quick_sort(arr ,low,index -1)
    quick_sort(arr,index+1,high)
```

- partition(arr,low,high)

```
pivot = low,p = low+1,q = high
while(p <= q )
    while(arr[p] < arr[pivot])
            Increment p
    while(arr[q] > arr[pivot])
            Increment q
    If( p < q )
            swap(arr[p],arr[q])
 swap(arr[q],arr[pivot])

Return q
```

Quick Sort

Advantages

- It is an In-place Sorting technique so space requirement is minimal
- It can be applied to large list.
- Time Complexity is less ,O(nlogn)

Quick Sort

Advantages

- It is an In-place Sorting technique so space requirement is minimal
- It can be applied to large list.
- Time Complexity is less , average case : O(nlogn)

Disadvantages

- It is not a Stable Sorting technique
- External Sorting is not possible

Code -Quick Sort