Revisiting Partition

Given any sequence of values which are comparable and stored in a Random Access Range [first, last) we can partition the values such that:

- there exists a special pivot value, pivot
- the new, rearranged sequence of values structures the elements this way

← Two-way partition

The two-way partition is pretty nice, but for algorithms like Quick Sort or Quick Select, there is a more efficient partitioning strategy.

Ideally, we'd like to group all values equal to the pivot together in the "middle". Since we have three groups instead of two, this is called a three-way partition

Three-Way Partition

- Key insight: keep track of where values < pivot ; values == pivot

```
partition ( first, last , pivot ):
    lo = first      // smaller values start @ beginning
    eq = first      // equal values could start @ beginning
    hi = last       // larger values start at the end
    while eq != hi :
        if *eq < pivot:                          } value < pivot
            swap lo with eq                        must move value into
            advance lo and eq                      smallest partition
        else if *eq > pivot:                     } value > pivot
            decrement hi                           must move value into
            swap eq with hi                        largest partition
        else:                                    } value == pivot
            advance eq                             advance "end" of equal
                                                   partition
```

Case 1

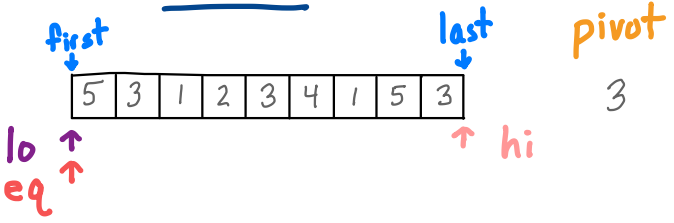
Case 2

Case 3

Tracing An Example

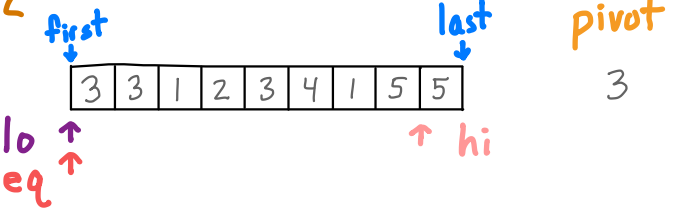
Note: we will always be inspecting *eq

Initial



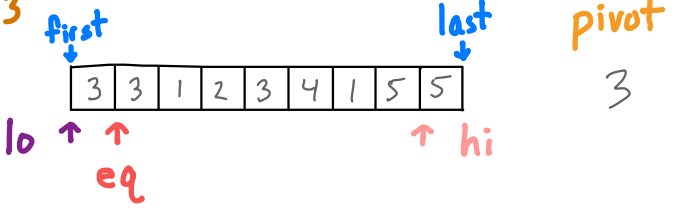
1. Case 2

- decr hi & move 5 to the end



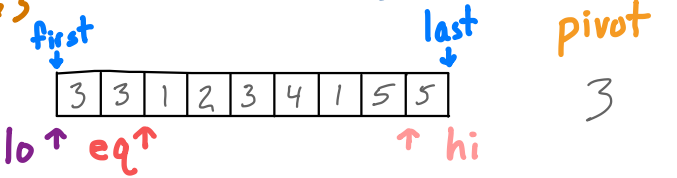
2. Case 3

- advance "eq"



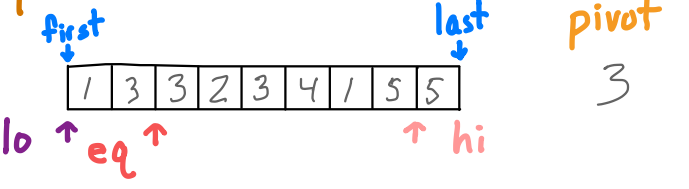
3. Case 3

- advance "eq"



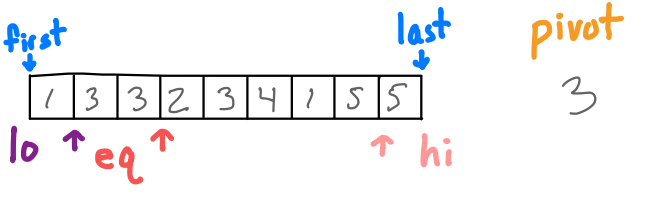
4a. Case 1

- swap "lo" with "eq"

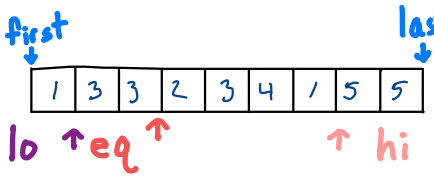


4b.

- advance both

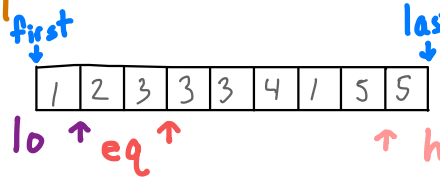


• state from last page



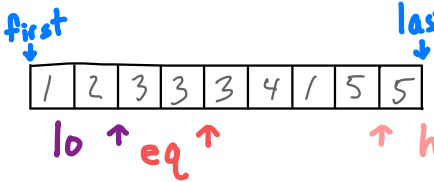
pivot
3

5a. Case 1 • swap lo with eq



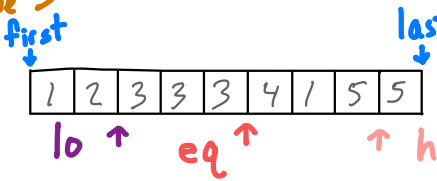
pivot
3

5b. • advance lo and eq



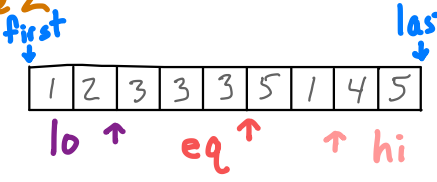
pivot
3

6. Case 3 • advance eq



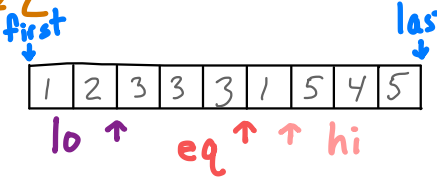
pivot
3

7. Case 2 • decr. hi ; swap eq with hi



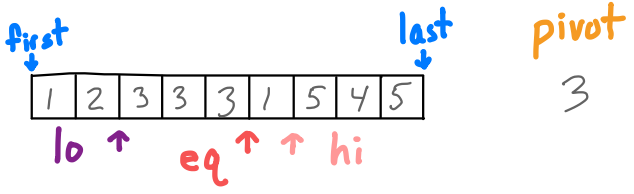
pivot
3

8. Case 2 • decr. hi ; swap eq with hi

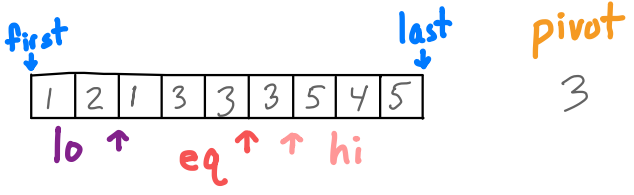


pivot
3

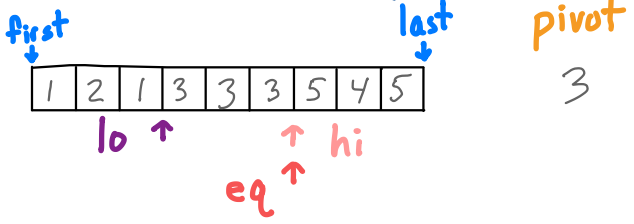
• State from last page



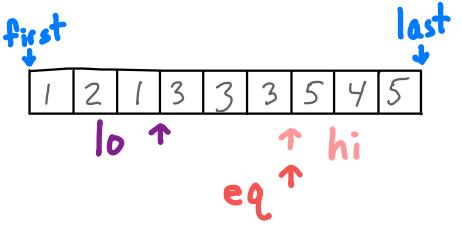
9a. Case 1 • swap lo with eq



9b. • advance lo & eq



10. • eq == hi — Done!



Resulting Ranges:

[first, lo) elements < pivot

[lo, hi) elements == pivot

[hi, last) elements > pivot