

# ECE 220

A special on Quicksort: Lecture x000B+



# Quick sort

# Quick sort

- One of the more faster sorting algorithms.

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.
  - Subdivide & repeat (recursive)

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.
  - Subdivide & repeat (recursive)
- Many varieties exist; this course cannot cover them all.

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.
  - Subdivide & repeat (recursive)
- Many varieties exist; this course cannot cover them all.
  - How to pick pivot?



# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.
  - Subdivide & repeat (recursive)
- Many varieties exist; this course cannot cover them all.
  - How to pick pivot?
    - First, last, mid, random, etc.

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.
  - Subdivide & repeat (recursive)
- Many varieties exist; this course cannot cover them all.
  - How to pick pivot?
    - First, last, mid, random, etc.
  - Recursive vs. iterative.

# Quick sort

- One of the more faster sorting algorithms.
- Key idea: choose a pivot element; then ...
  - Move all elements greater than pivot to right of it and smaller than pivot to left of it.
  - Subdivide & repeat (recursive)
- Many varieties exist; this course cannot cover them all.
  - How to pick pivot?
    - First, last, mid, random, etc.
  - Recursive vs. iterative.
- Main point: understand one variety and understand it well.



# Quick sort

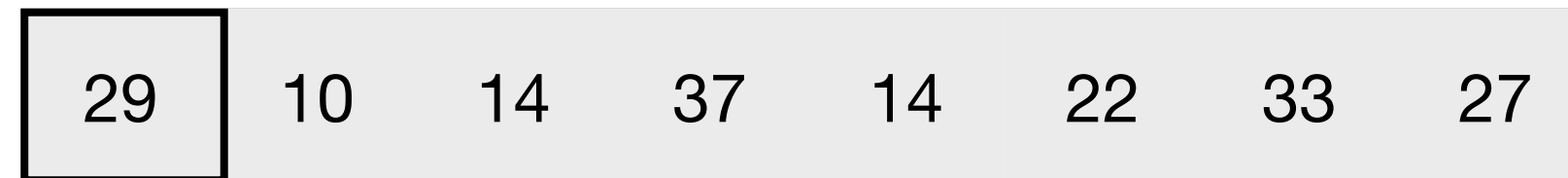
29 10 14 37 14 22 33 27

# Quick sort

29 10 14 37 14 22 33 27

1. Choose first element of given array as pivot.

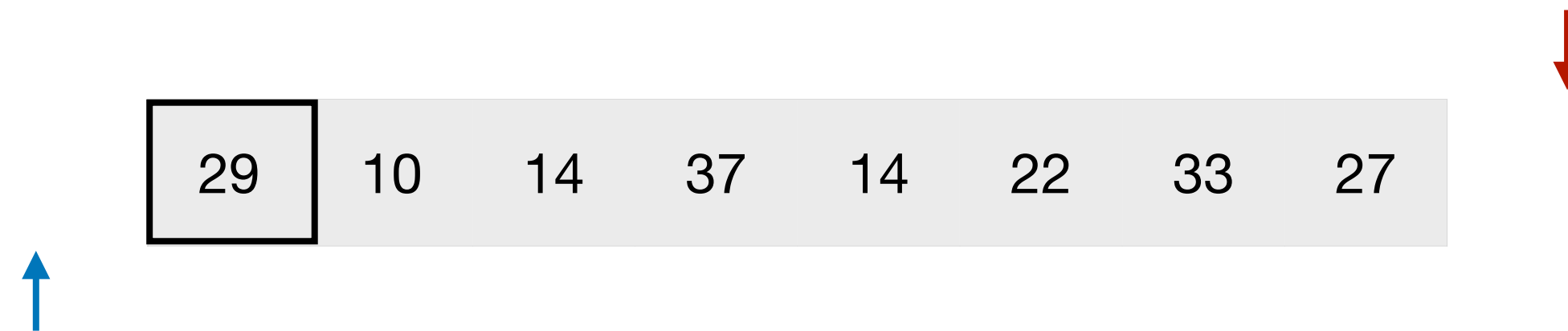
# Quick sort



1. Choose first element of given array as pivot.

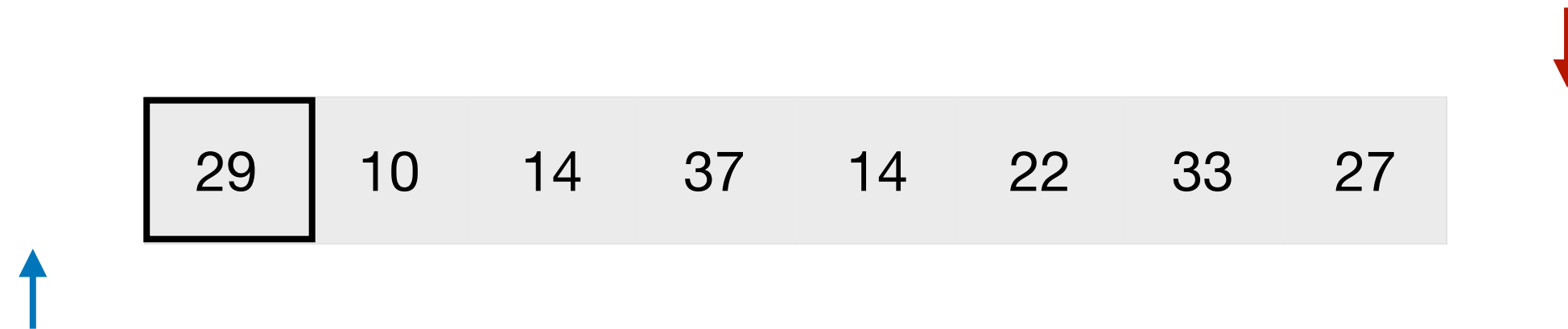


# Quick sort



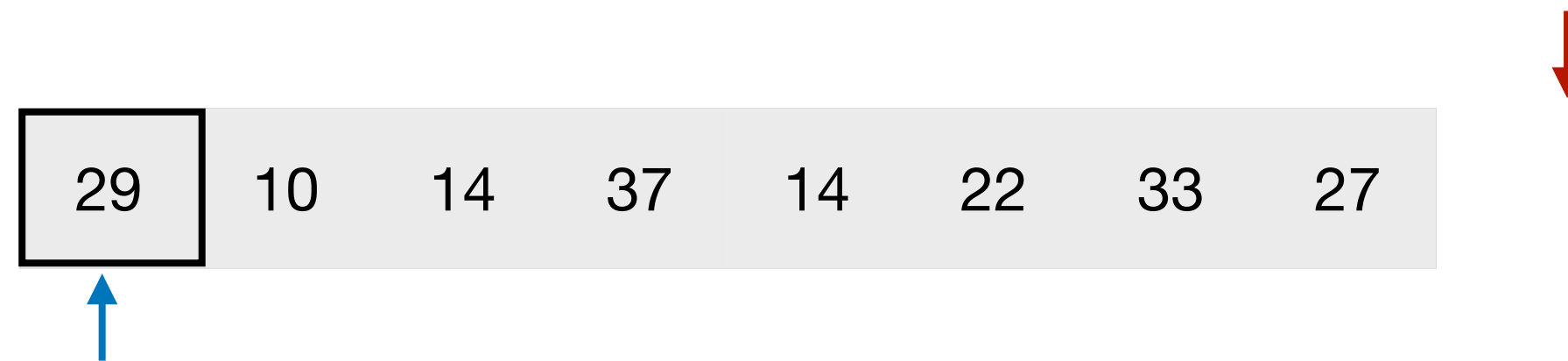
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot

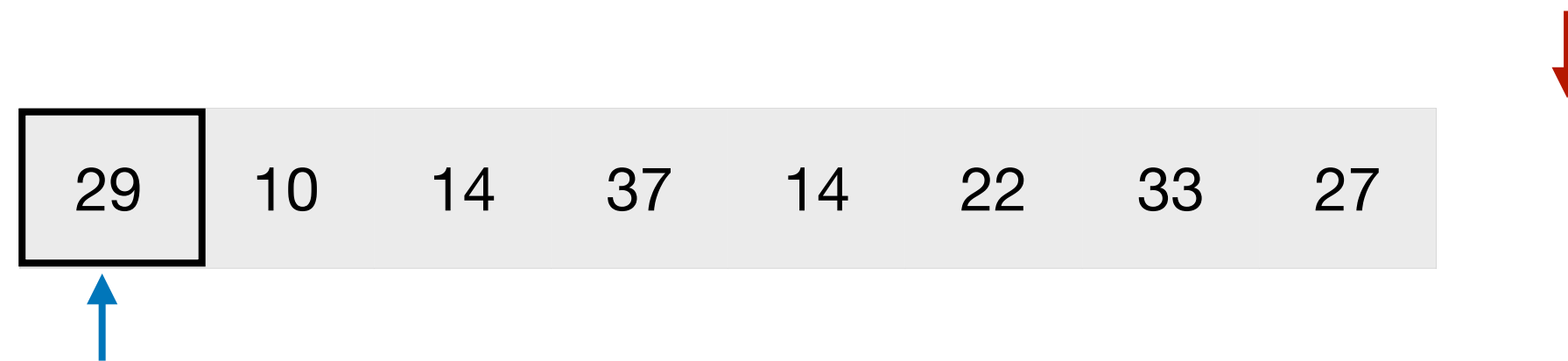
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot

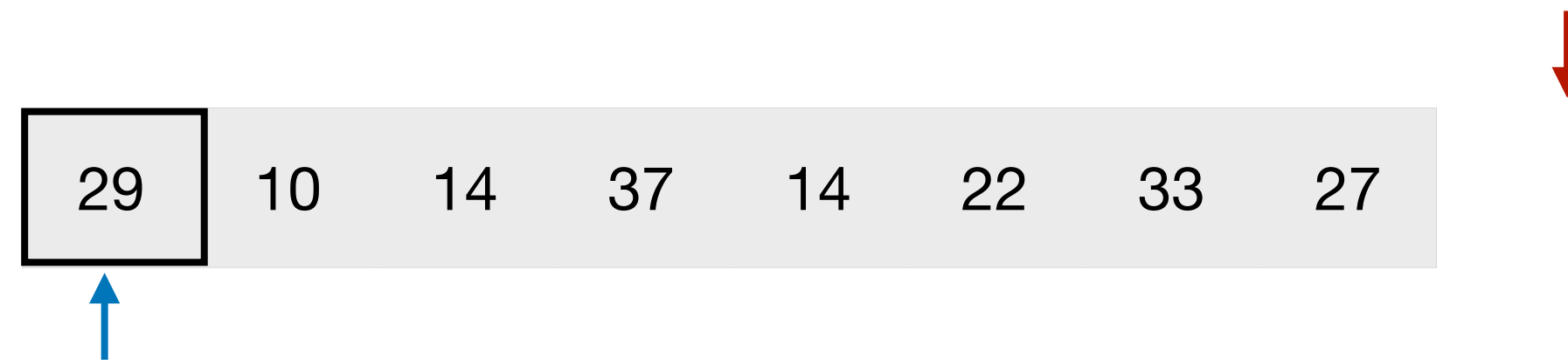


# Quick sort



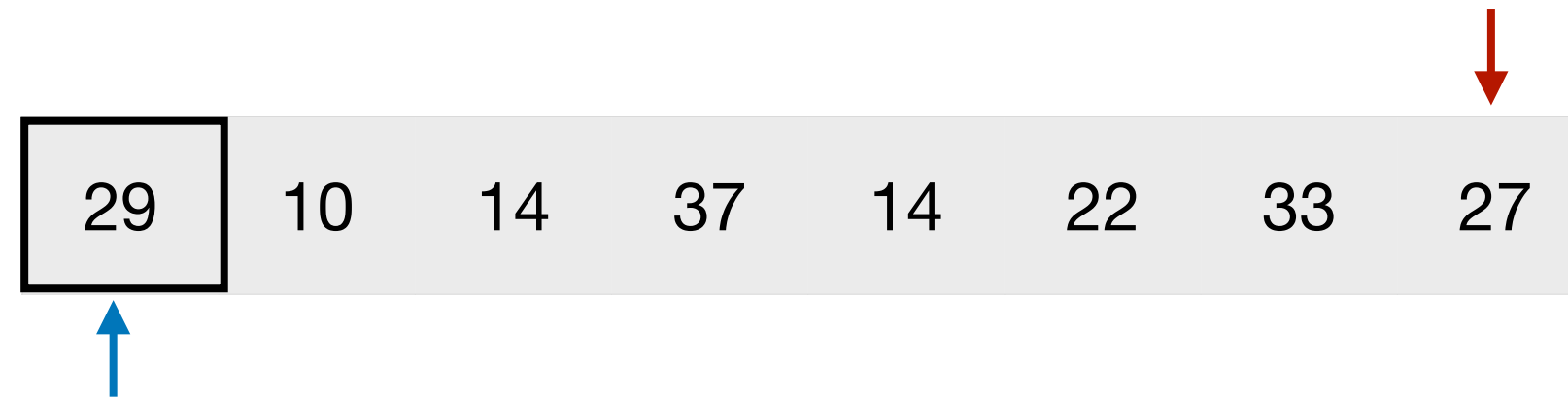
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.

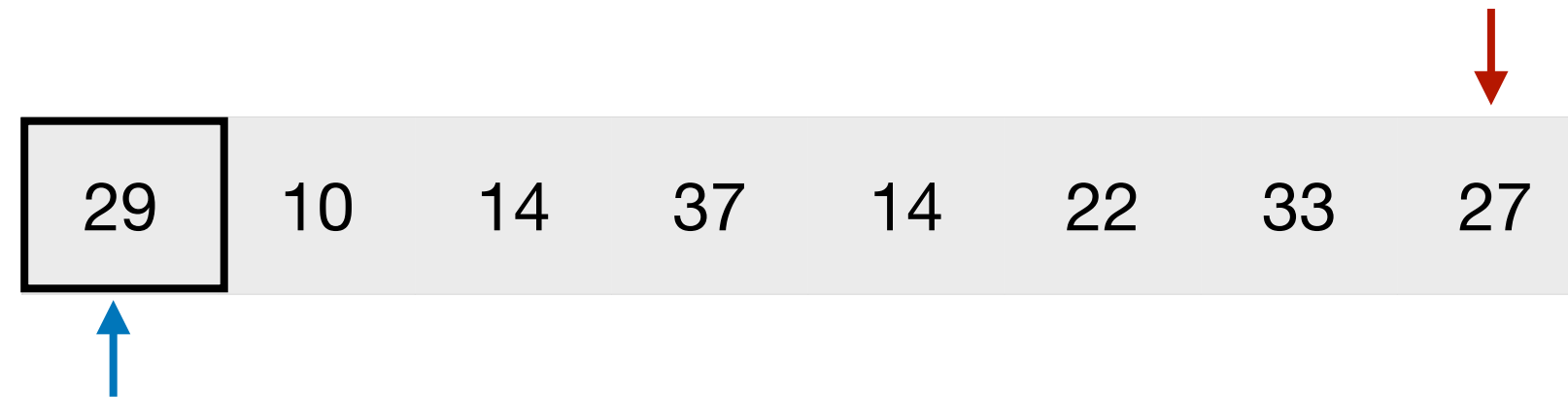
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.

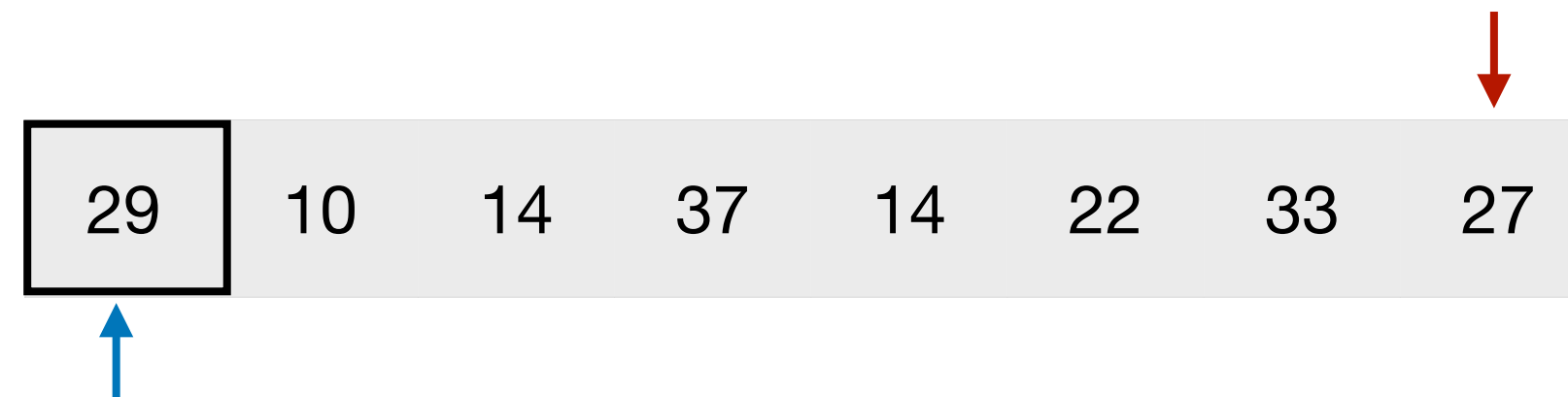


# Quick sort



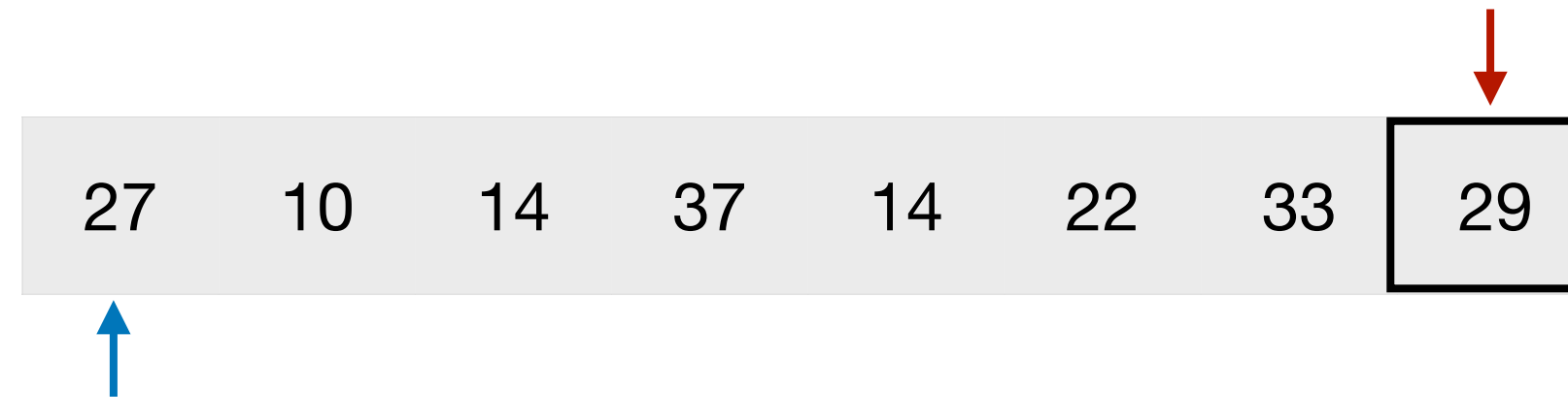
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.

# Quick sort



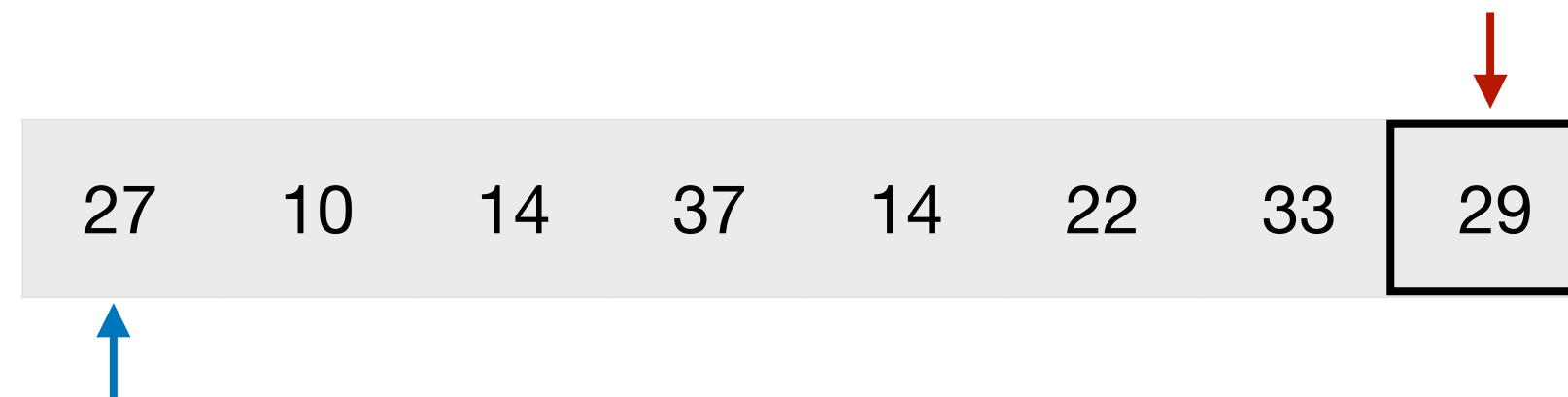
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.

# Quick sort



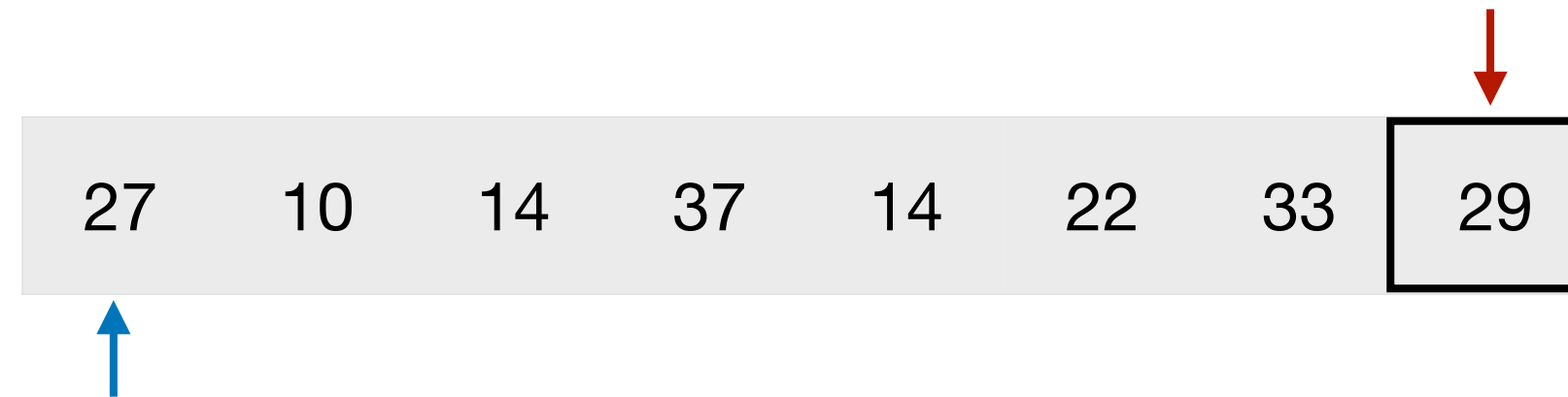
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

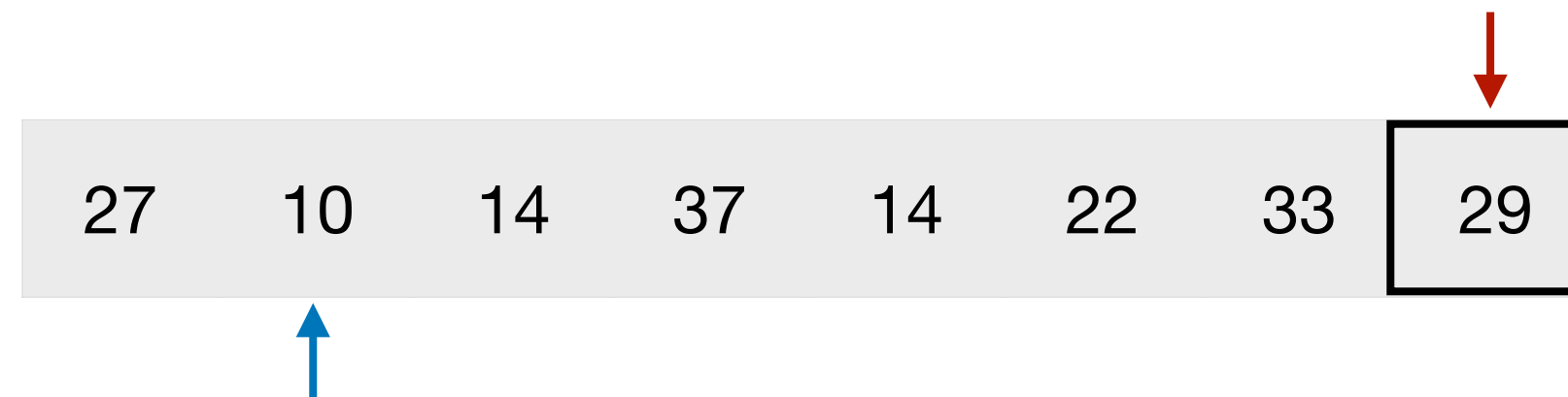
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

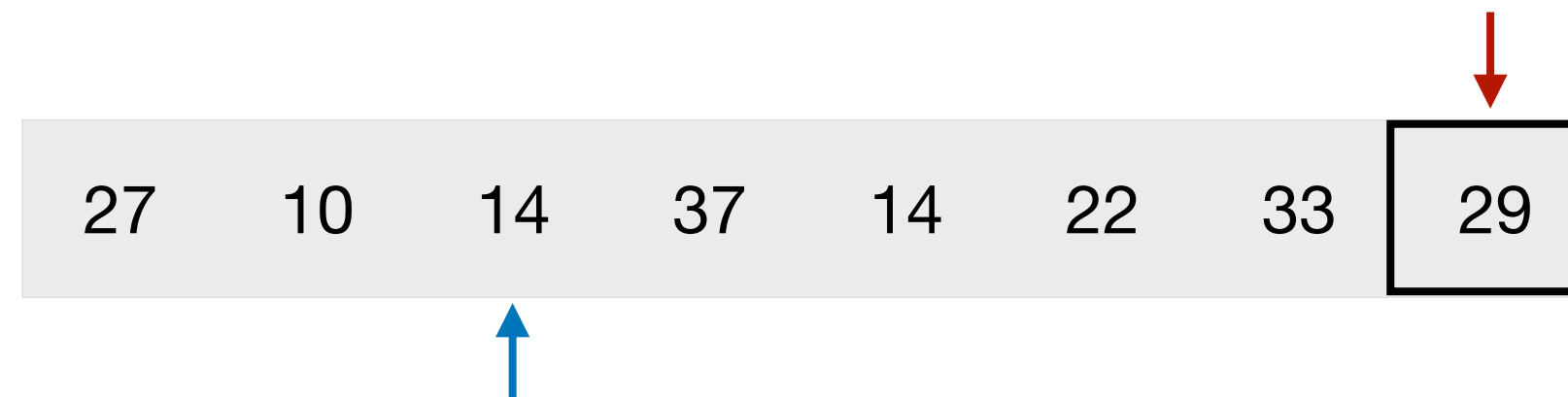


# Quick sort



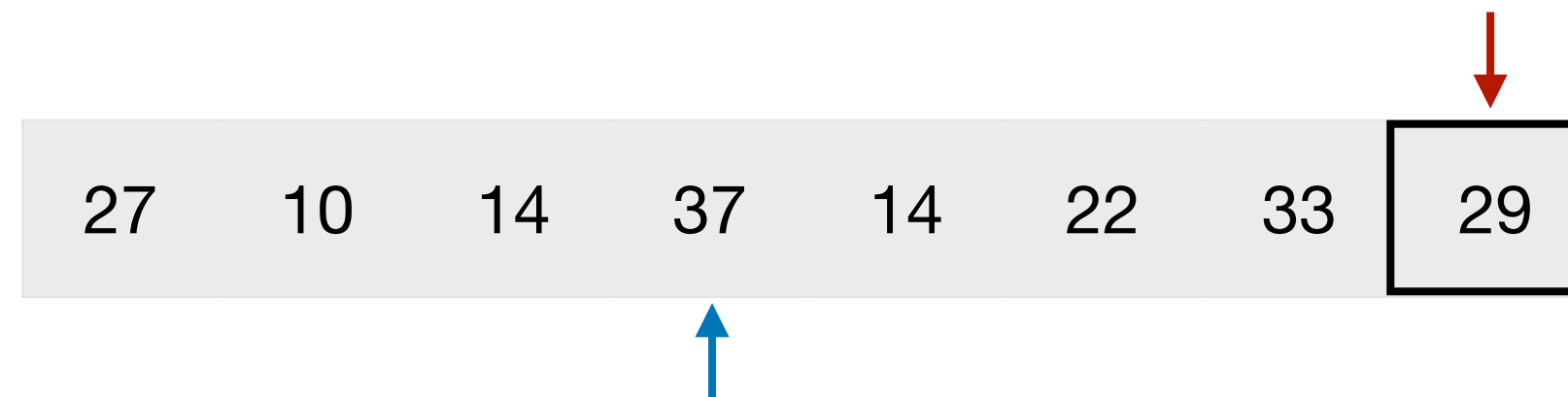
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



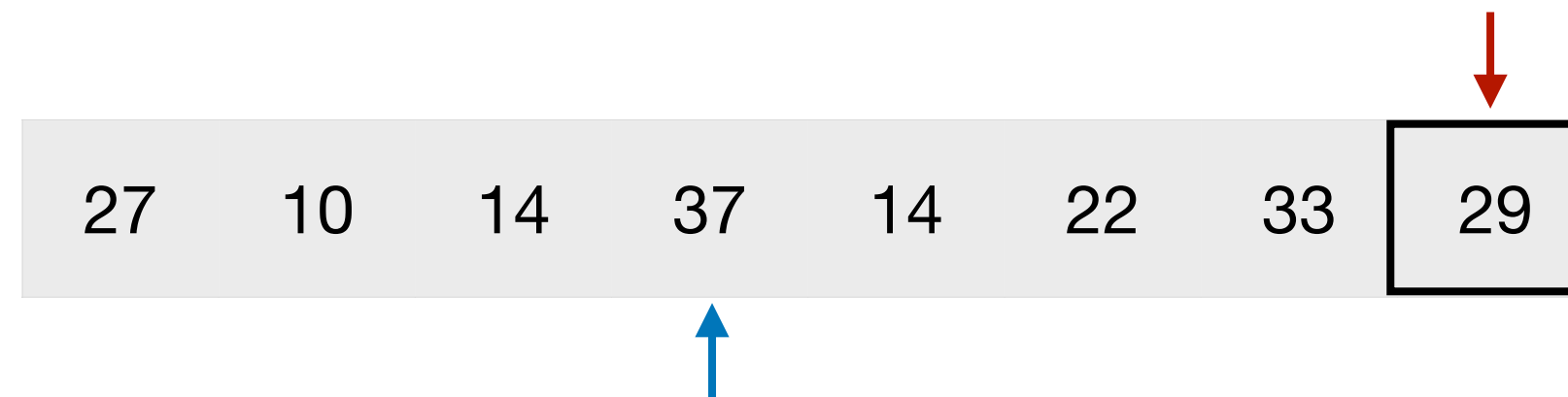
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



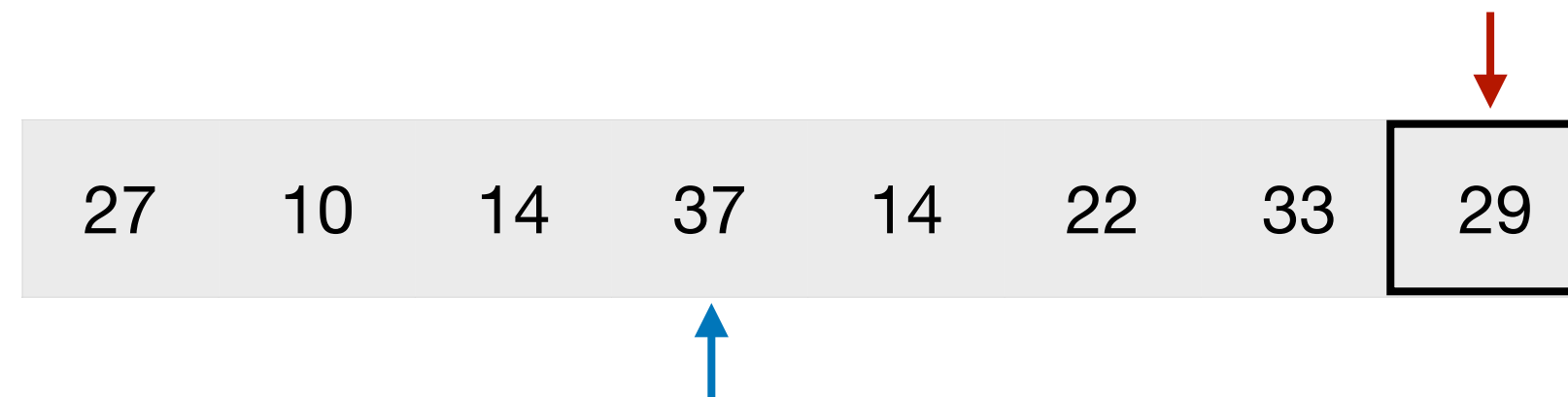
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



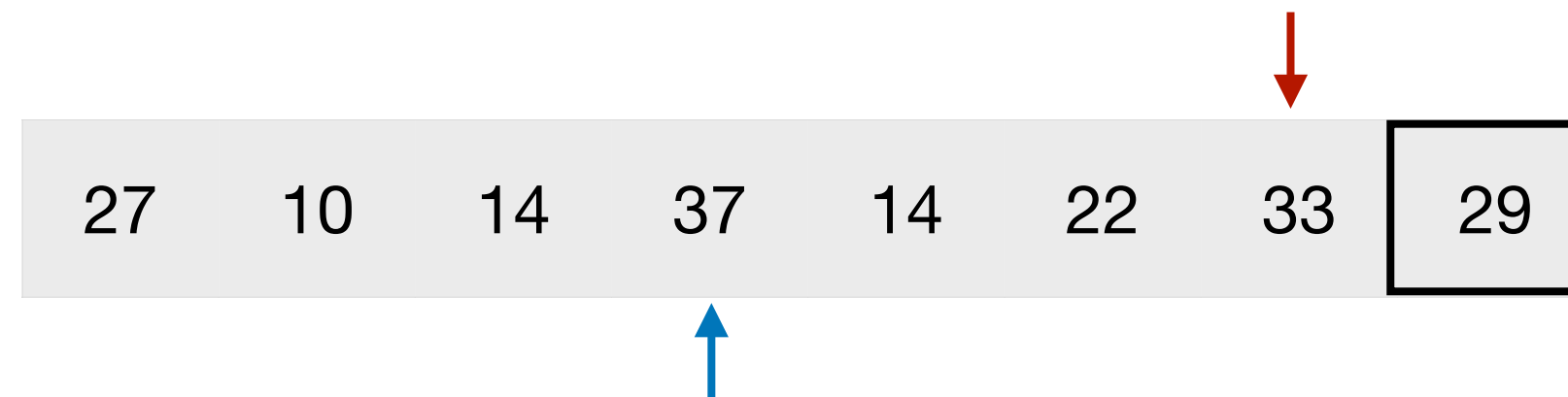
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

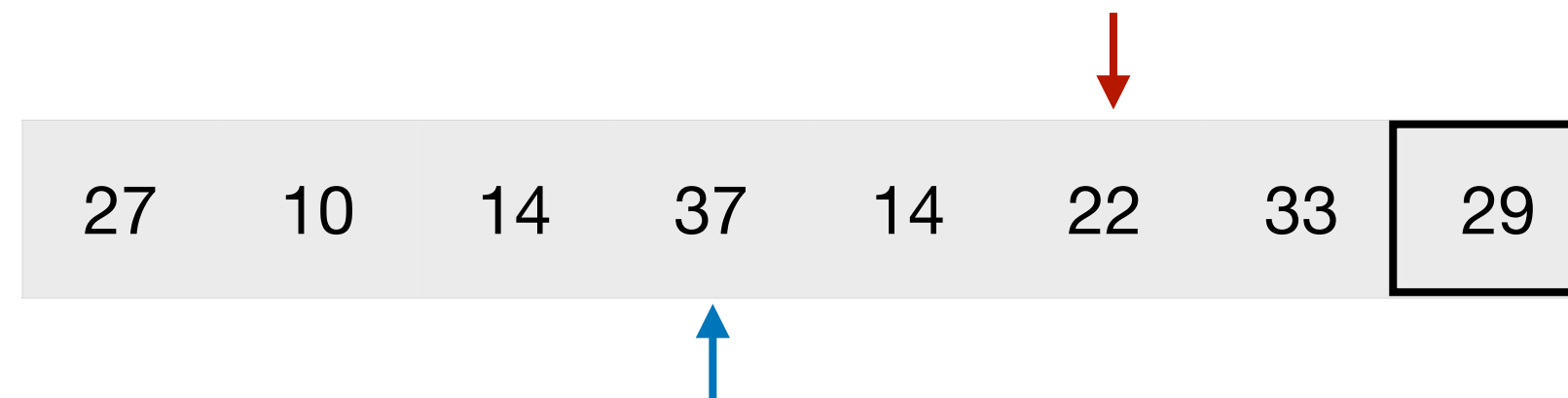
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

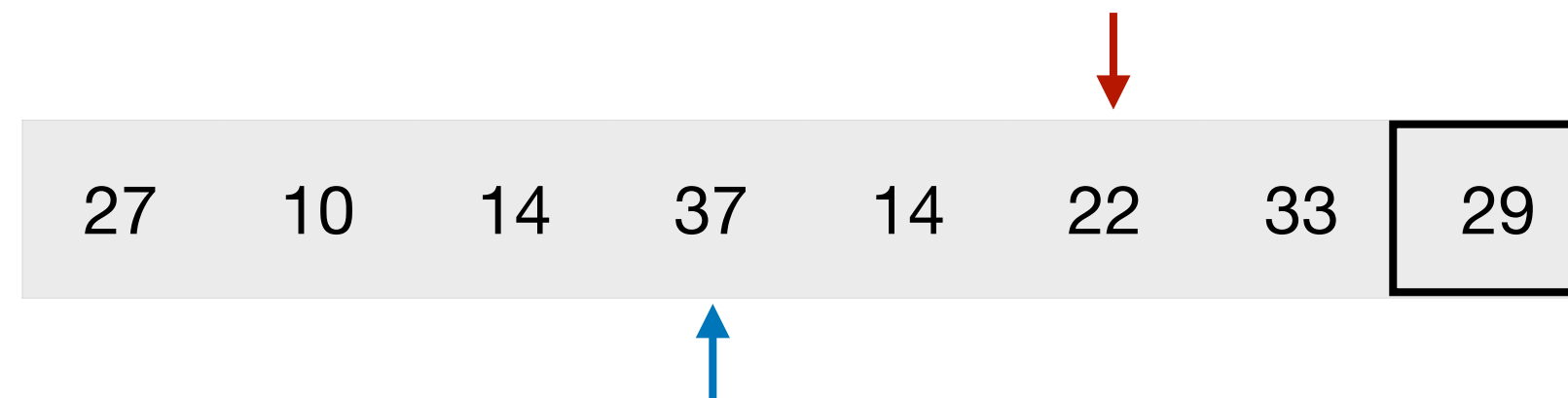


# Quick sort



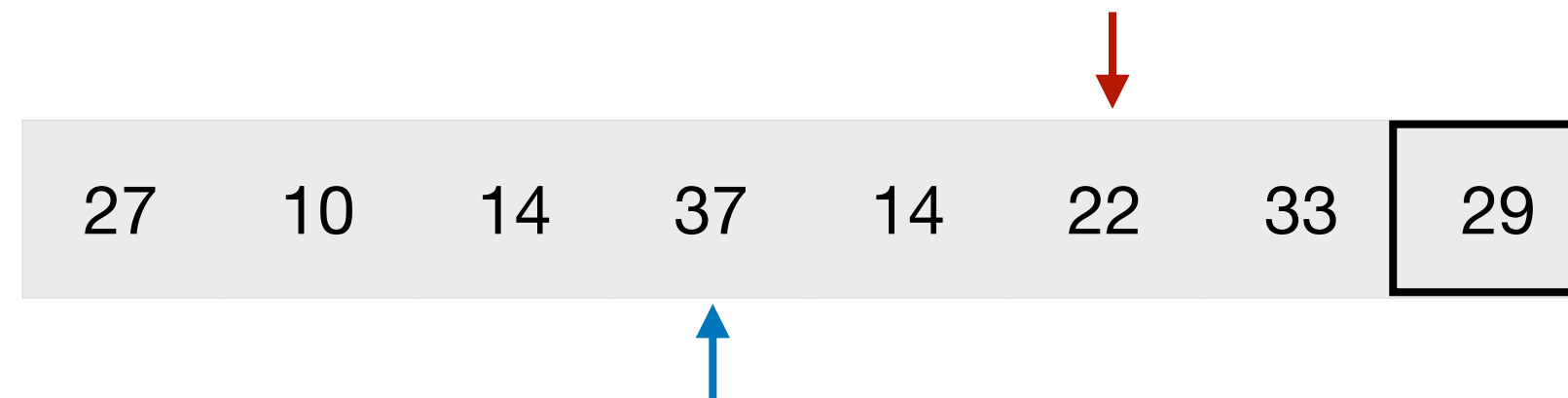
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



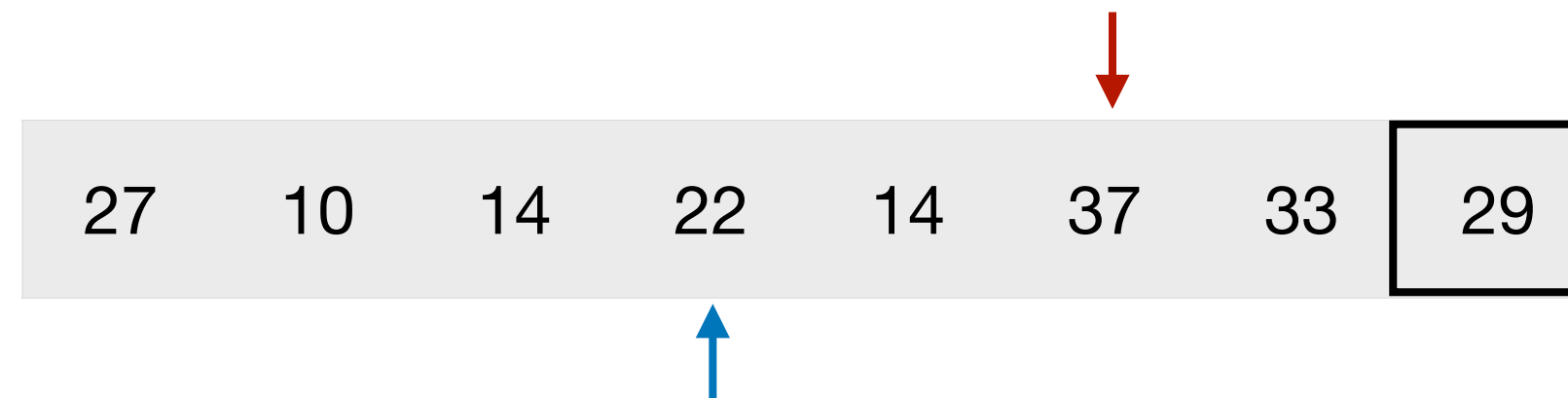
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



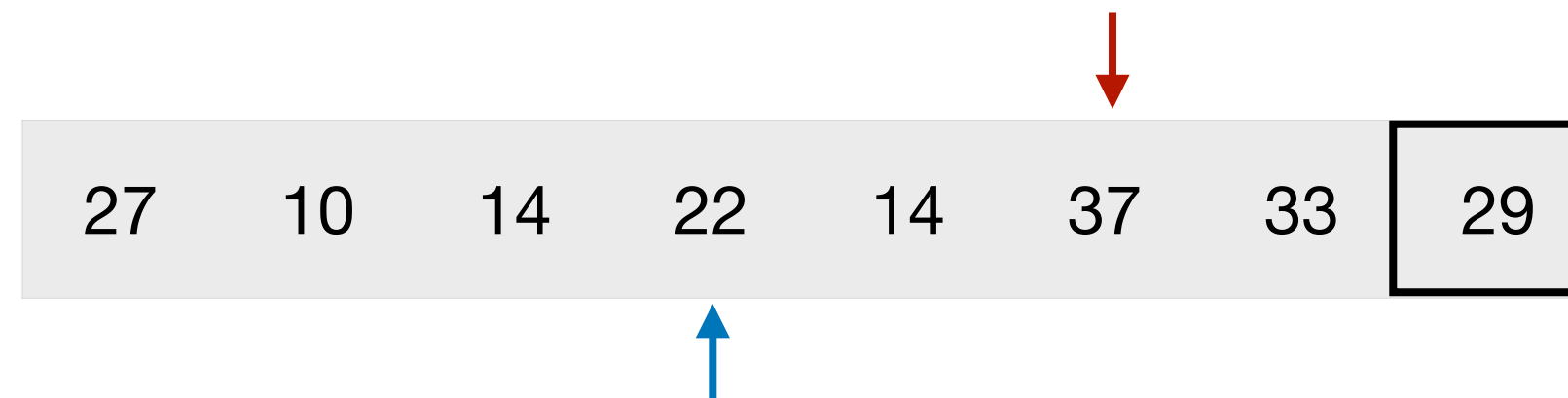
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
- 5. If neither pointers can move swap elements.**
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



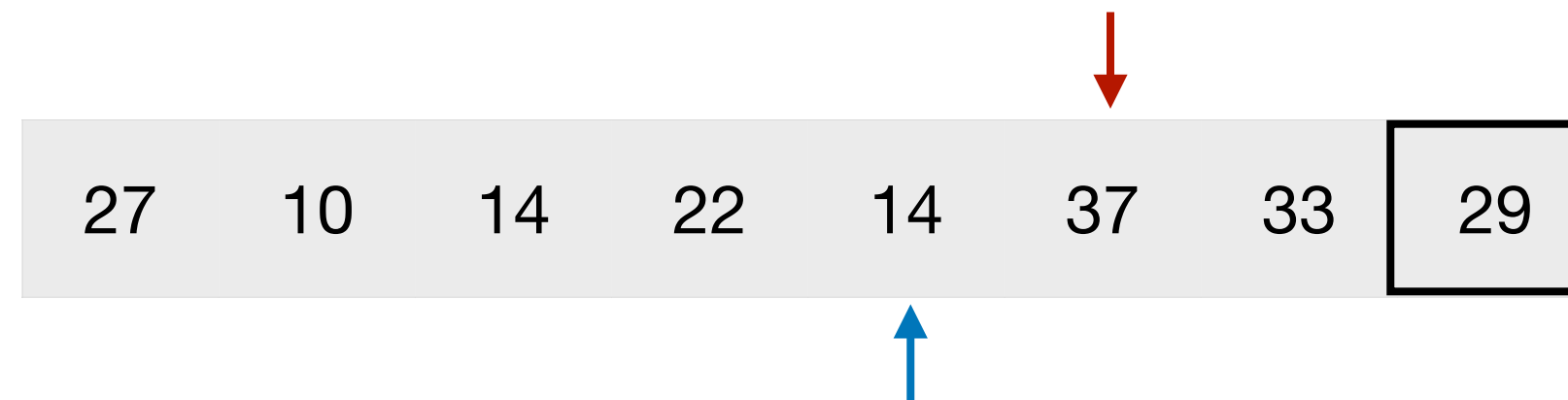
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
- 5. If neither pointers can move swap elements.**
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.



# Quick sort



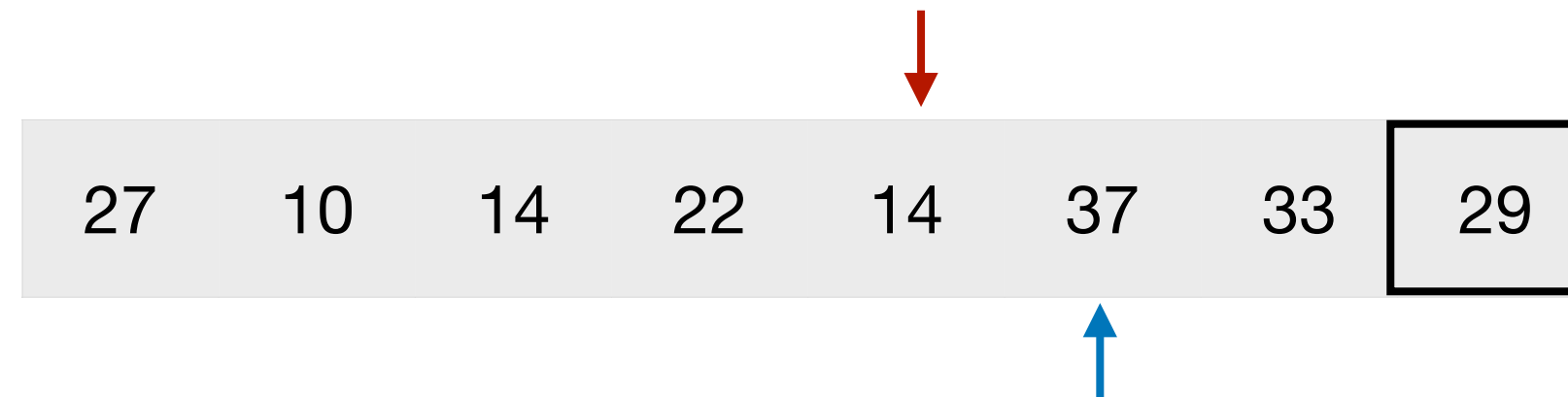
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



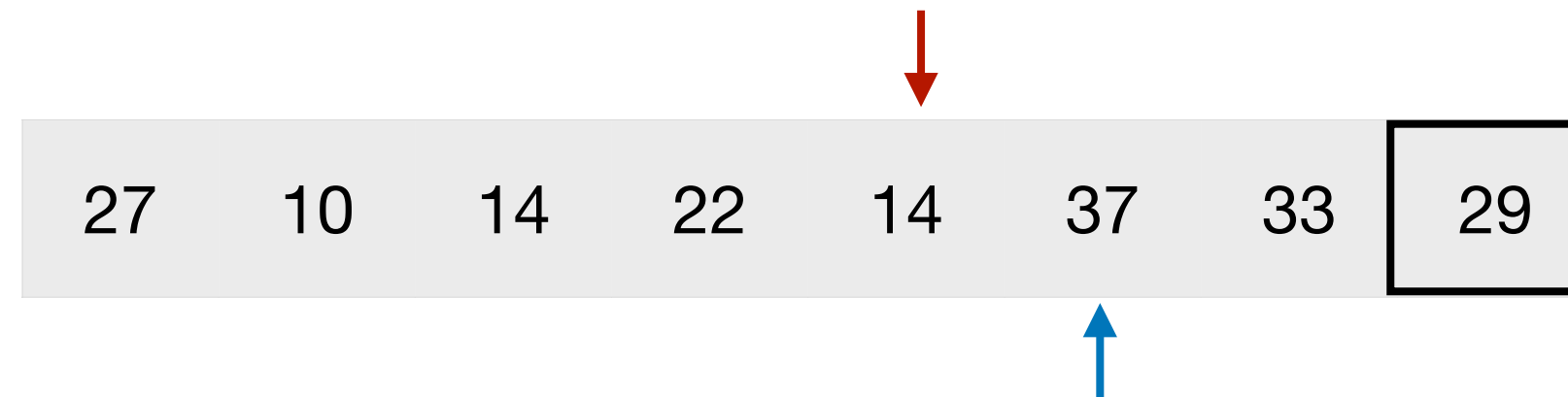
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



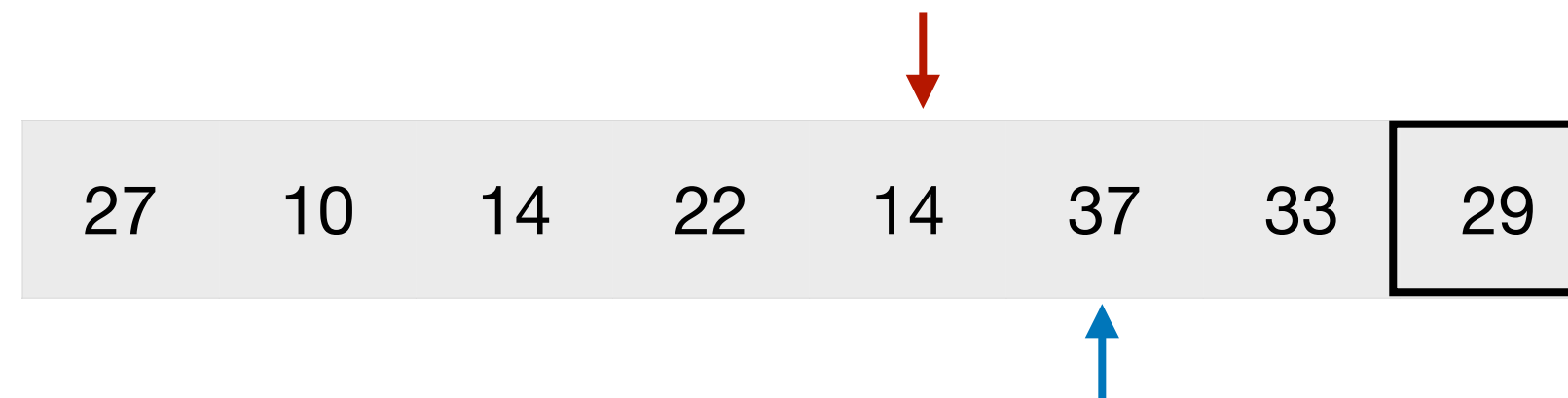
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



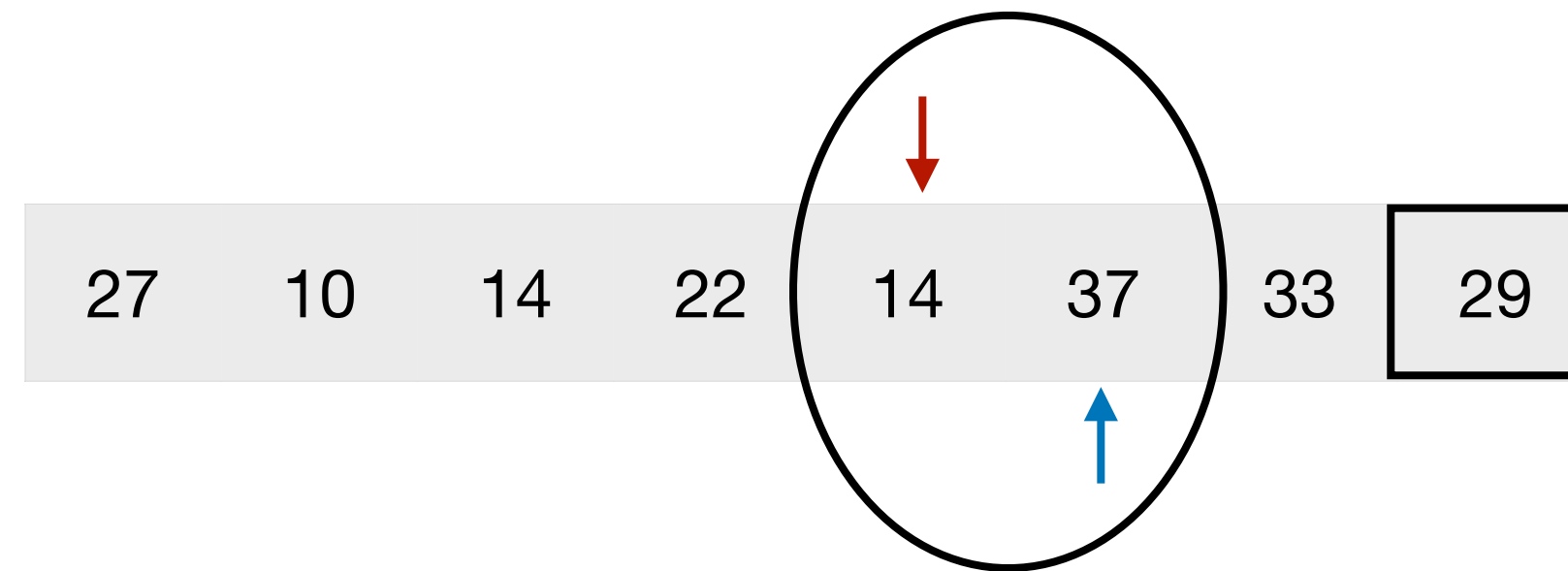
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer**.

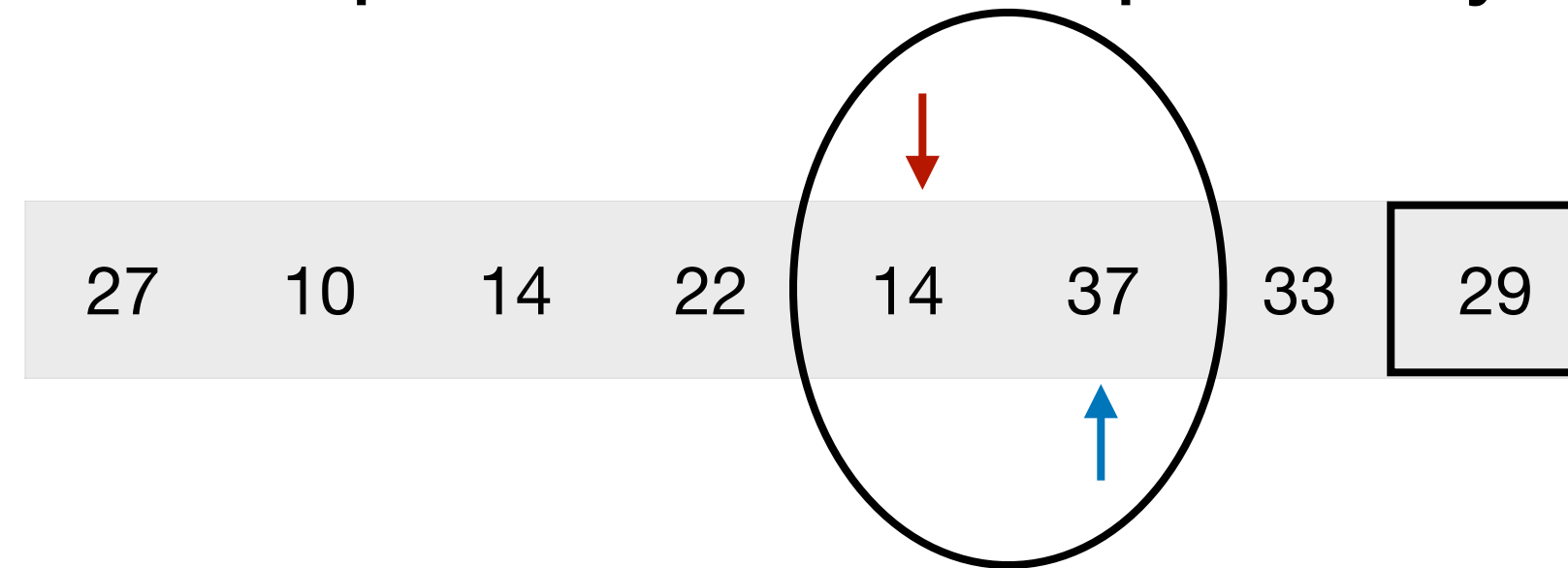
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer**.

# Quick sort

If pointers cross; split array & recurse on each.



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer**.



# Quick sort

27 10 14 22 14 37 33 29

1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. **If pointers cross/overlap, split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort

27	10	14	22	14	37	33	29
----	----	----	----	----	----	----	----

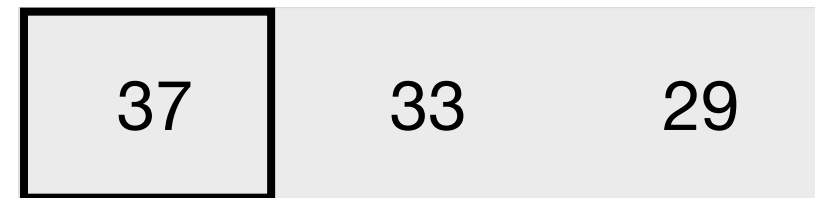
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. **If pointers cross/overlap, split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort

37 33 29

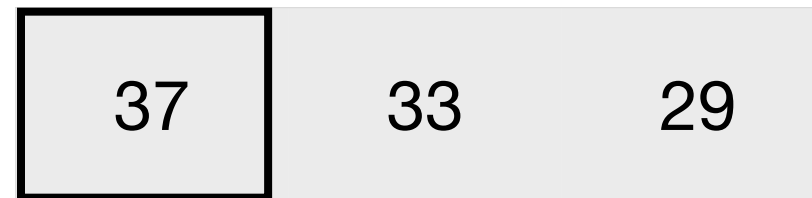
- 1. Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



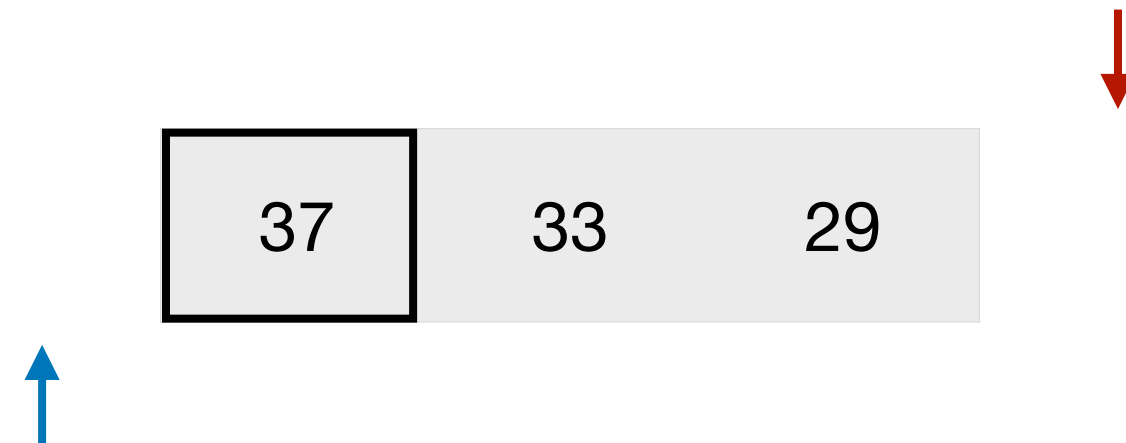
- 1. Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



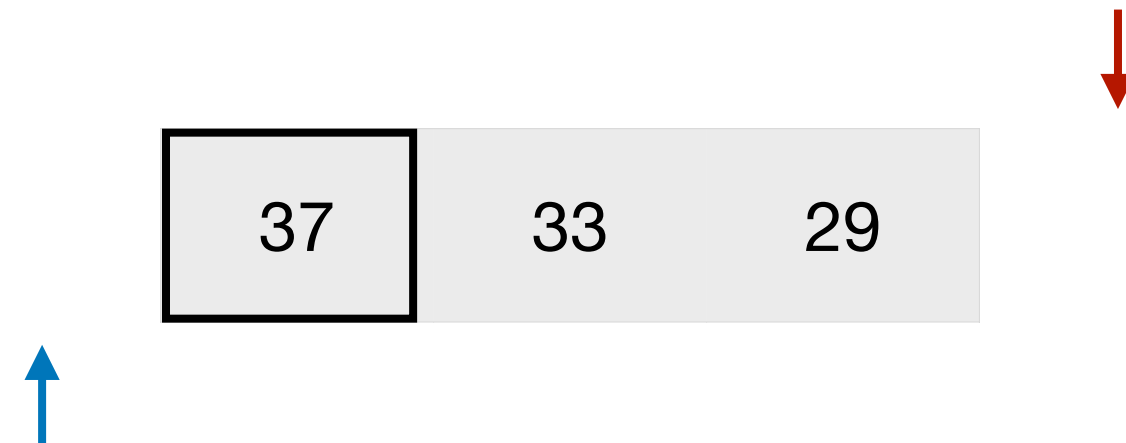
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

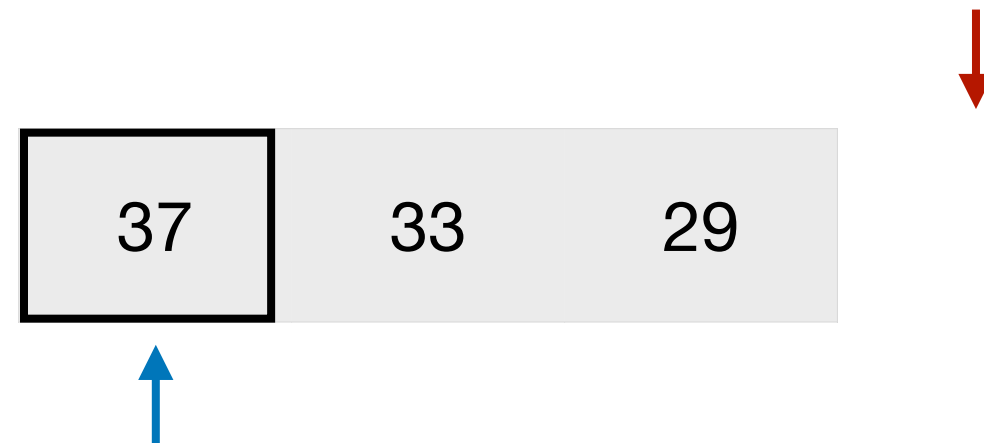
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

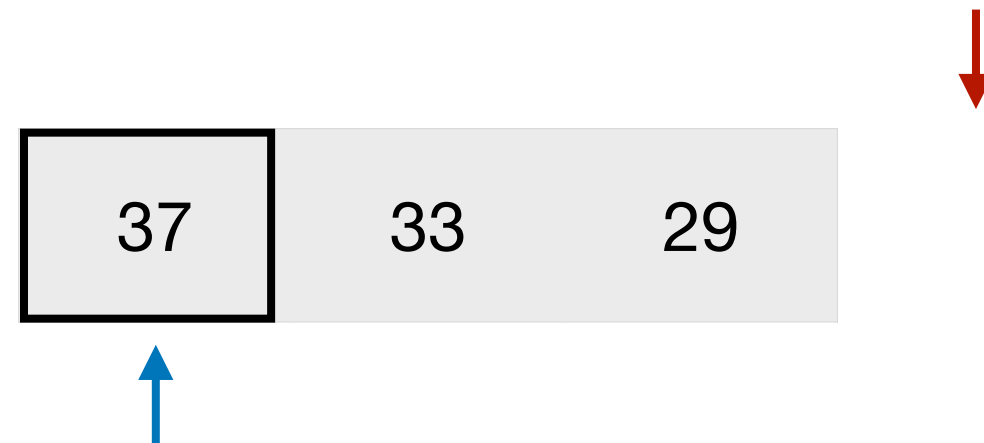


# Quick sort



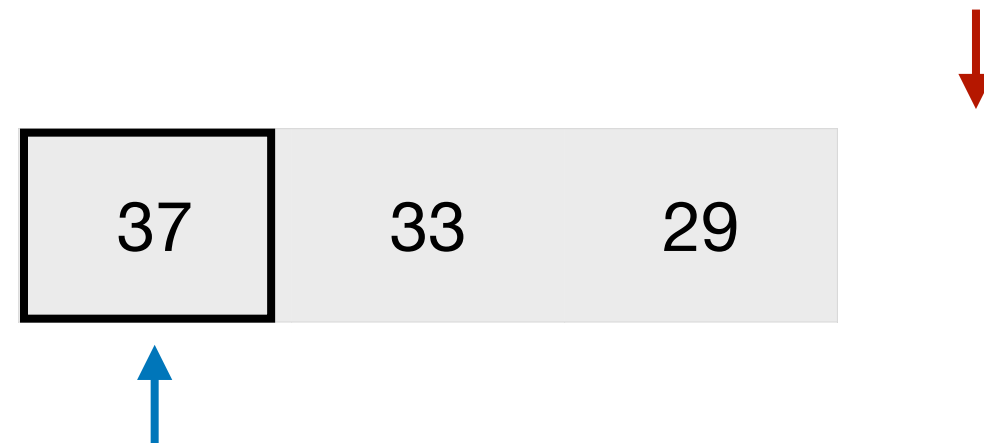
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



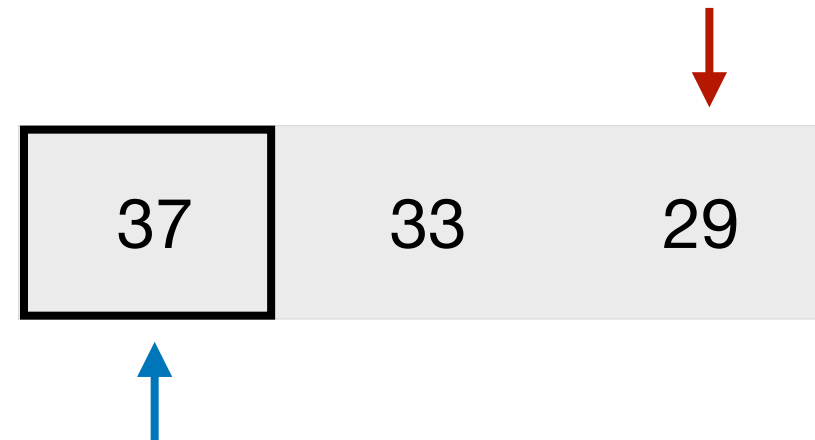
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



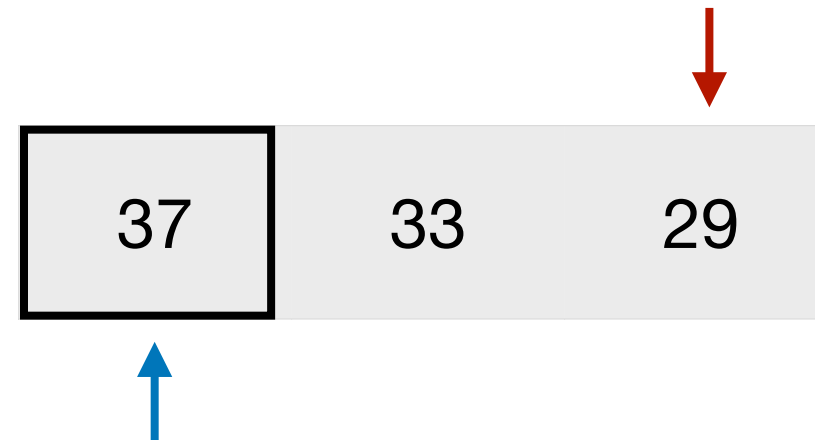
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



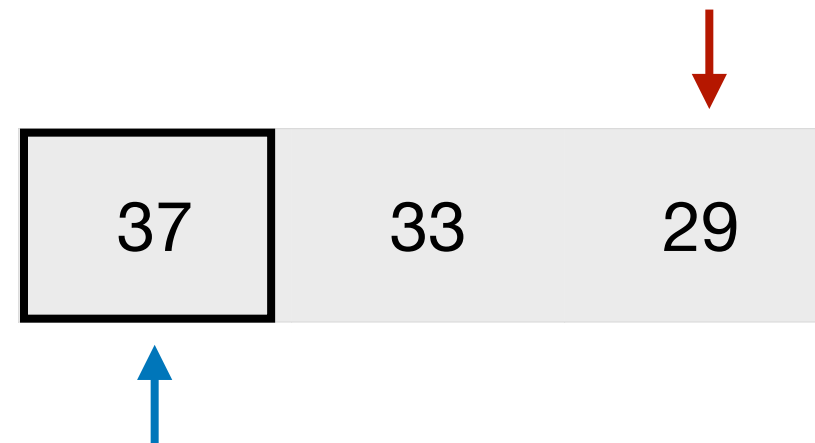
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



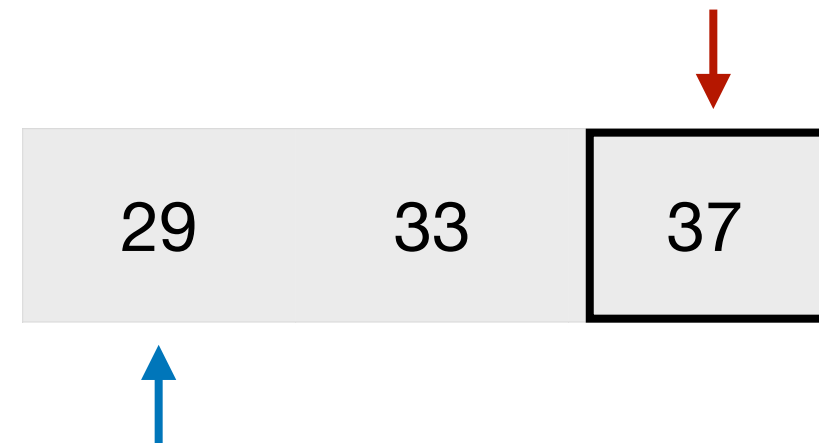
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



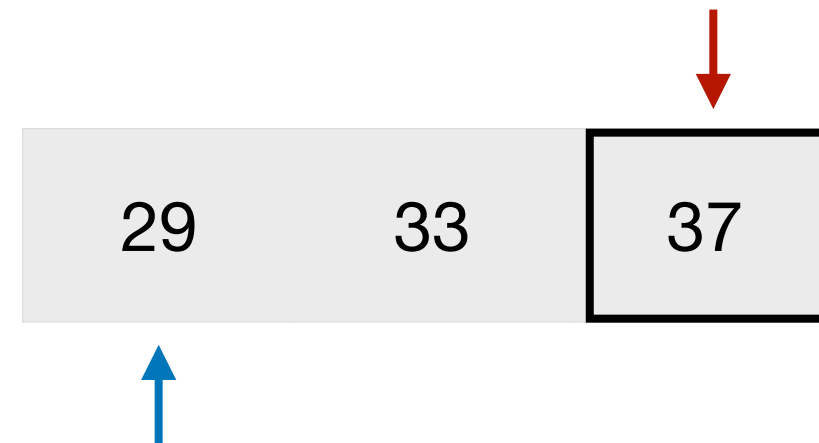
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
- 5. If neither pointers can move swap elements.**
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
- 5. If neither pointers can move swap elements.**
6. Repeat 3-5 while left pointer < right pointer.

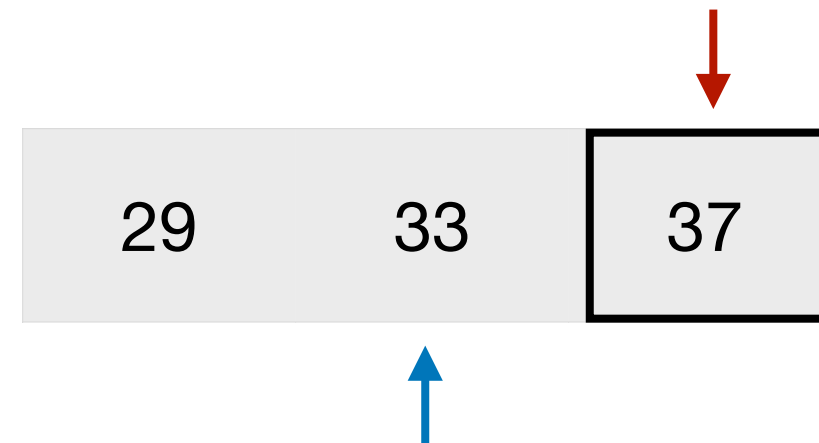
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

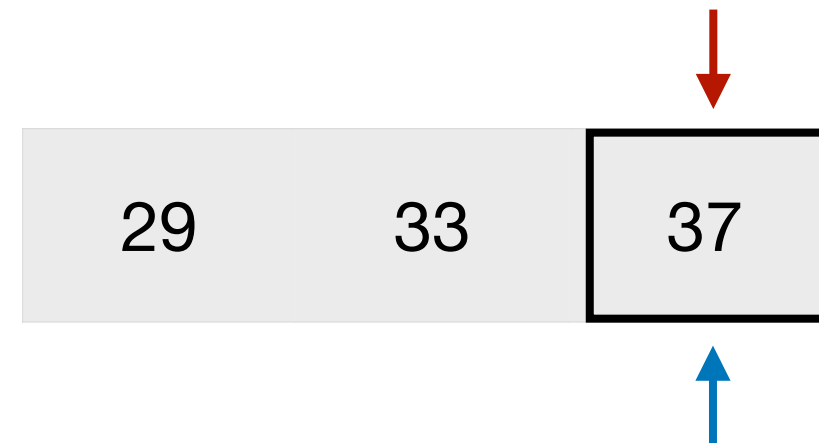


# Quick sort



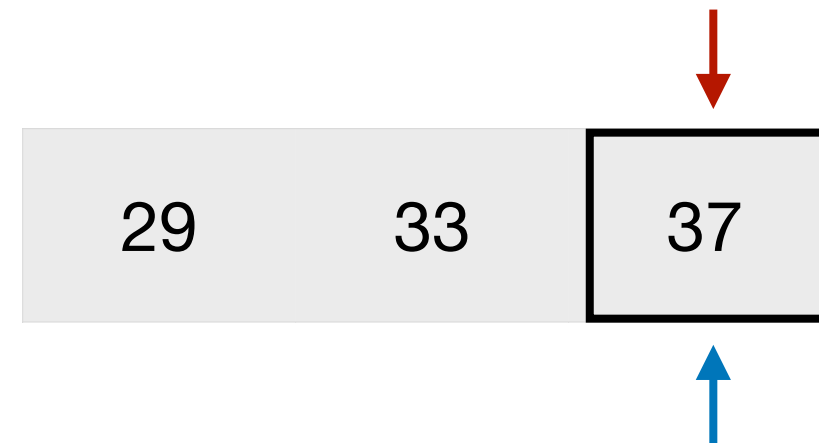
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



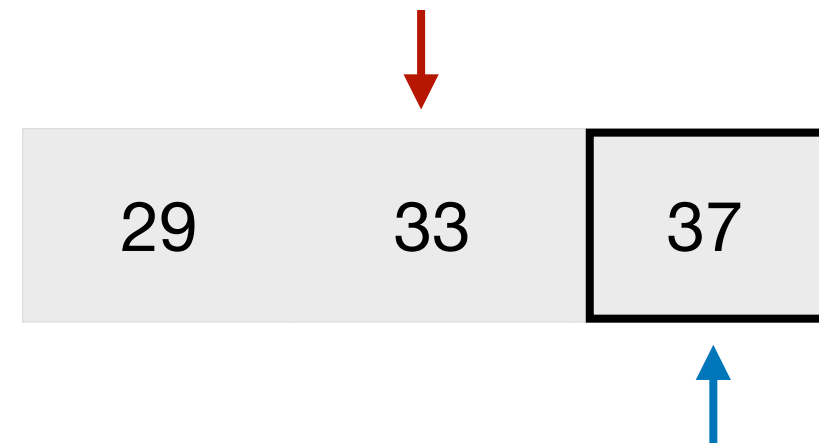
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



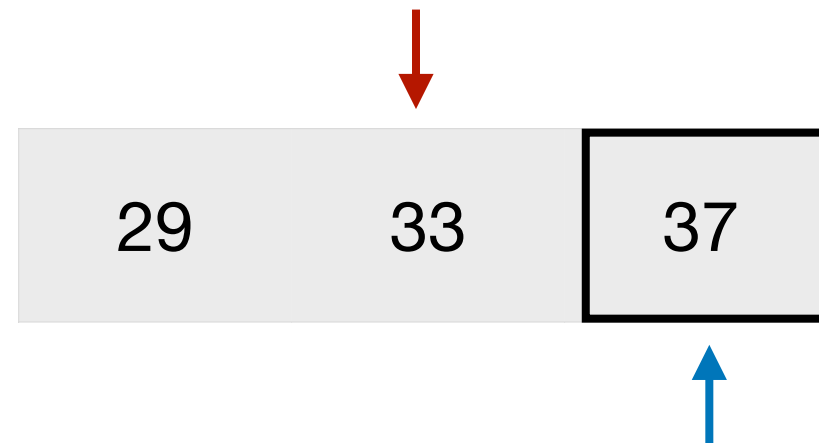
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



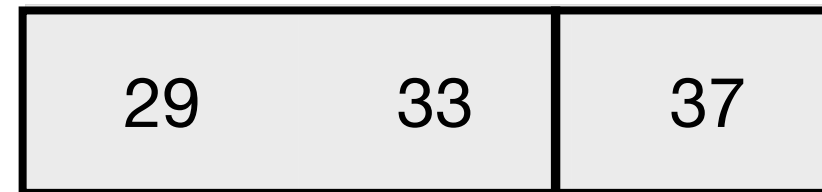
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. **If pointers cross/overlap, split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer.**

# Quick sort

29	33	37
----	----	----

1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. **If pointers cross/overlap, split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer.**

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. **If pointers cross/overlap, split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer.**

Single element arrays  
are considered sorted

# Quick sort

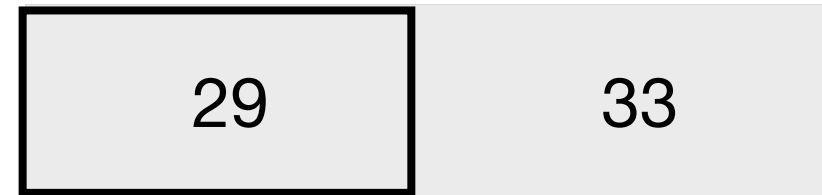
29

33

1. **Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

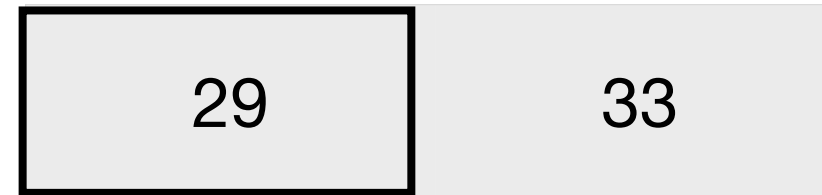


# Quick sort



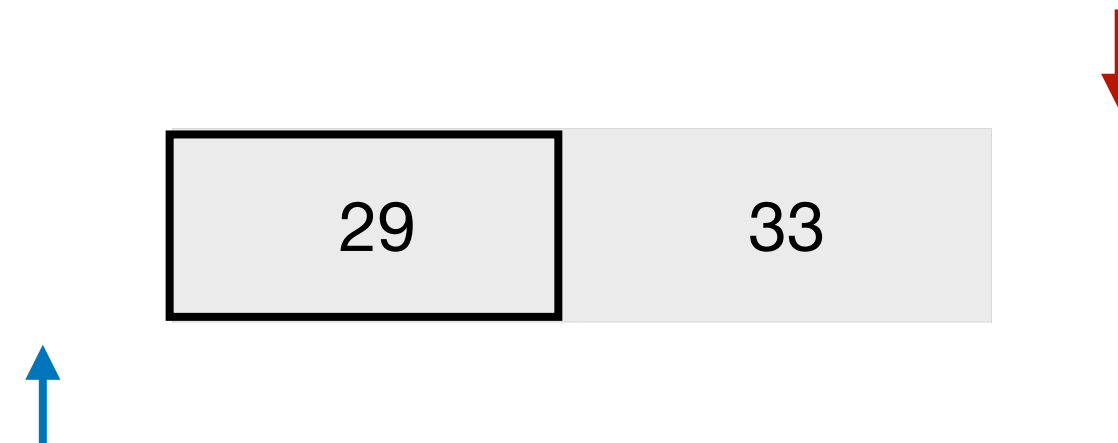
- 1. Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



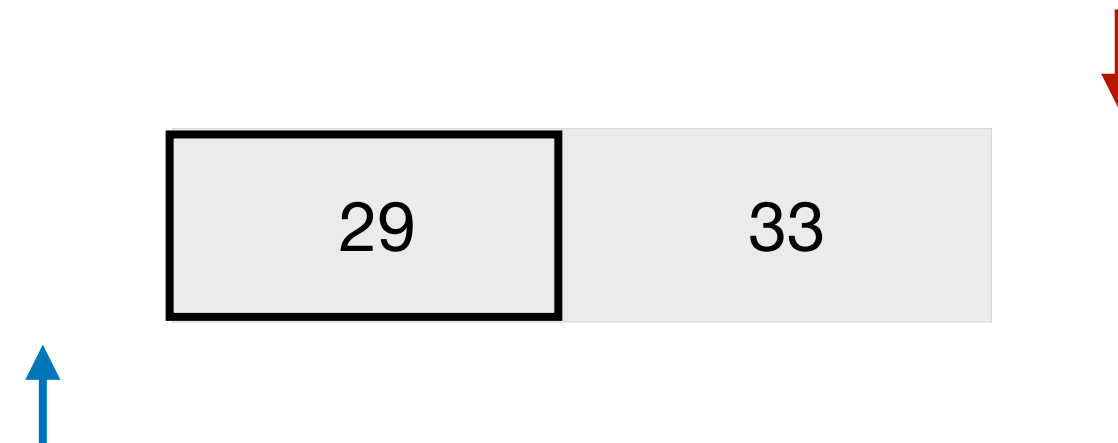
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



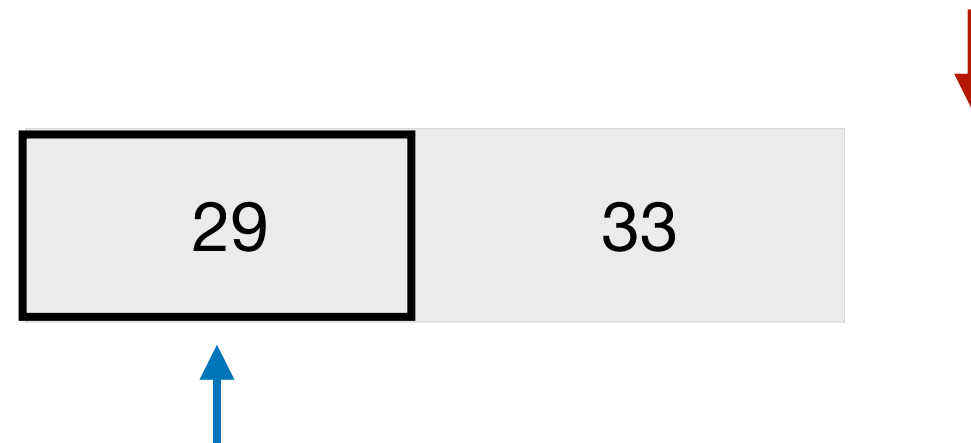
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



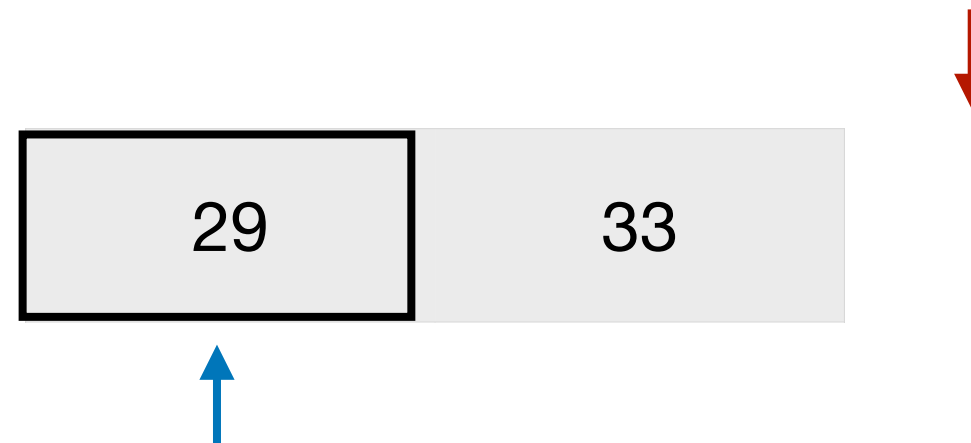
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



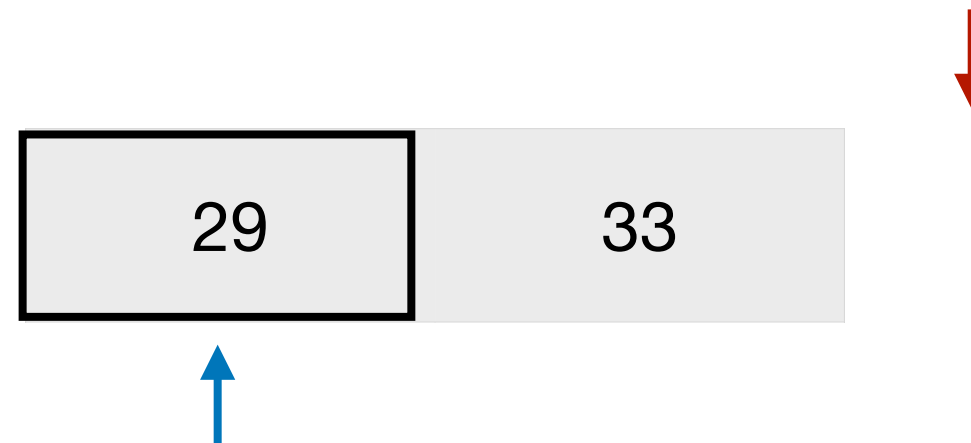
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



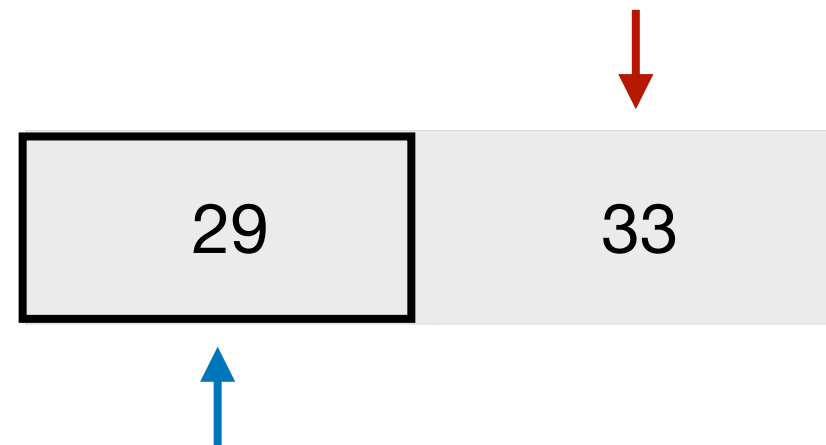
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

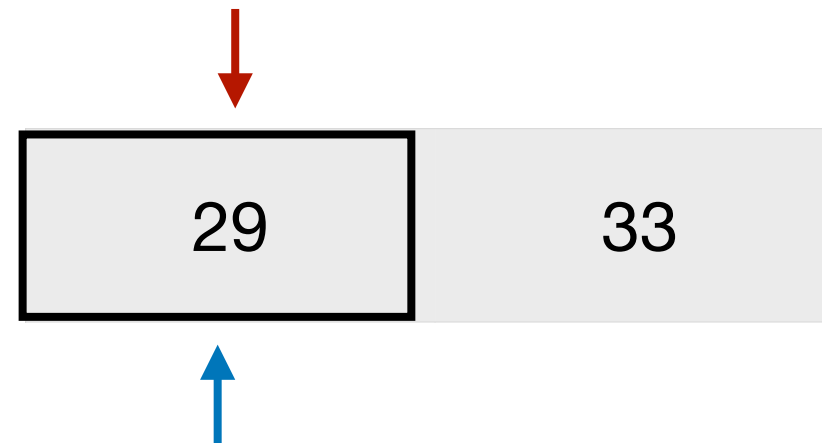
# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

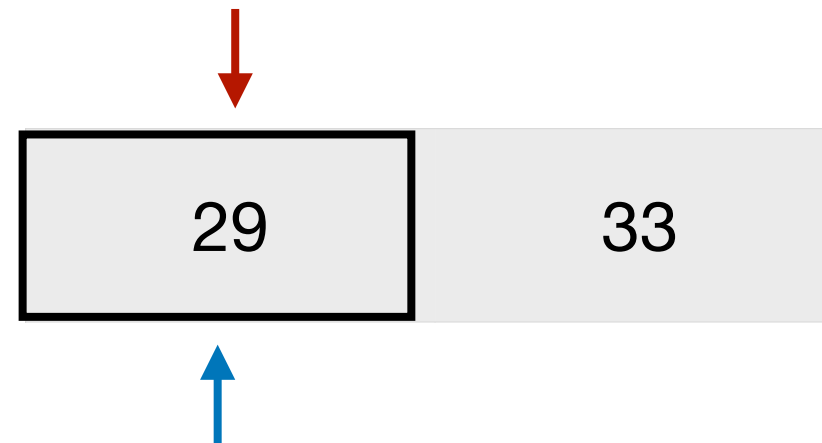


# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
  1. If pointers cross/overlap, split array & recurse on each subarray.
5. If neither pointers can move swap elements.
6. Repeat 3-5 while left pointer < right pointer.

# Quick sort



1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, **split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer.**

# Quick sort



Single element arrays  
are considered sorted

1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
  1. If pointers cross/overlap, **split array & recurse on each subarray.**
5. If neither pointers can move swap elements.
6. Repeat 3-5 while **left pointer < right pointer.**

# Quick sort

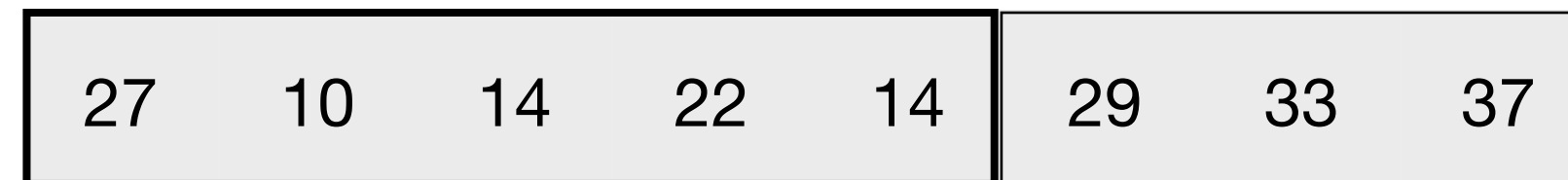
27 10 14 22 14 29 33 37



Repeat on the other array.

Follow the steps on this array yourself.

# Quick sort



Repeat on the other array.

Follow the steps on this array yourself.

# Implementation

```
void Swap(int* one, int* two){
    int temp = *one;
    *one = *two;
    *two = temp;
}

void QuickSort(int arr[], int start, int end){
    if (start < end){
        int pivotVal = partition(arr, start, end);
        QuickSort(arr, start, _____);
        QuickSort(arr, _____, end);
    }
}
```

```
int partition(int arr[], int start, int end){
    int pivotVal = _____;
    int i = _____;
    int j = _____;

    while(1){
        do i++;
        while (_____);

        do j--;
        while (_____);

        if (_____)
            return j;

        Swap(&arr[i], &arr[j]);
    }
}
```