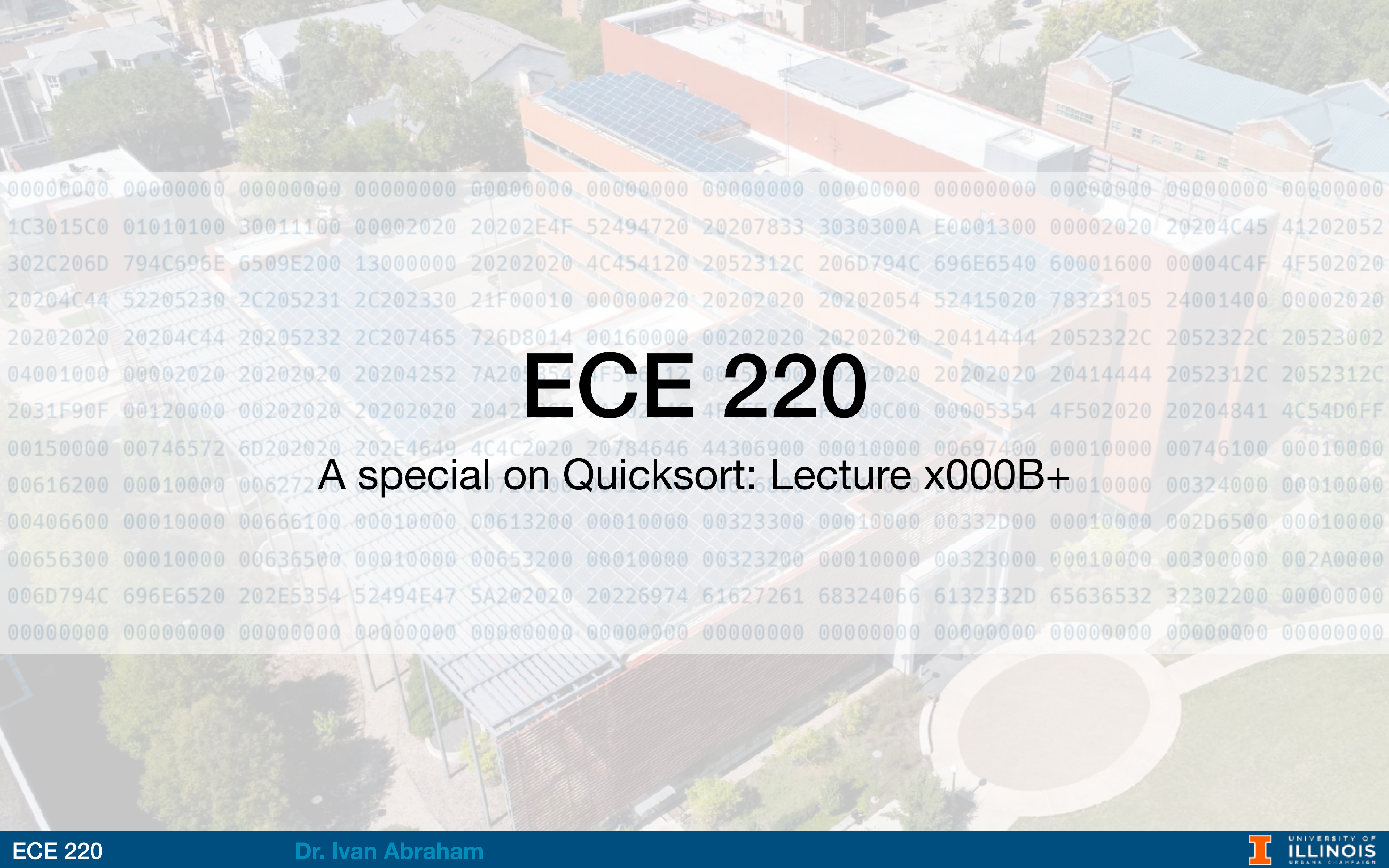




00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
1C3015C0 01010100 30011100 00002020 20202E4F 52494720 20207833 3030300A E0001300 00002020 20204C45 41202052
302C206D 794C696E 6509E200 13000000 20202020 4C454120 2052312C 206D794C 696E6540 60001600 00004C4F 4F502020
20204C44 52205230 2C205231 2C202330 21F00010 00000020 20202020 20202054 52415020 78323105 24001400 00002020
20202020 20204C44 20205232 2C207465 726D8014 00160000 00202020 20202020 20414444 2052322C 2052322C 20523002
04001000 00002020 20202020 20204252 7A20B54 F50612 00150000 00202020 20202020 20414444 2052312C 2052312C
2031F90F 00120000 00202020 20202020 2042B65 00170246 4F5000C00 00005354 4F502020 20204841 4C54D0FF
00150000 00746572 6D202020 202E4649 4C4C2020 20784646 44306900 00010000 00697400 00010000 00746100 00010000
00616200 00010000 00627200 00120000 00720100 00120000 00616800 00010000 00003200 00010000 00324000 00010000
00406600 00010000 00666100 00010000 00613200 00010000 00323300 00010000 00332D00 00010000 002D6500 00010000
00656300 00010000 00636500 00010000 00653200 00010000 00323200 00010000 00323000 00010000 00300000 002A0000
006D794C 696E6520 202E5354 52494E47 5A202020 20226974 61627261 68324066 6132332D 65636532 32302200 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

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

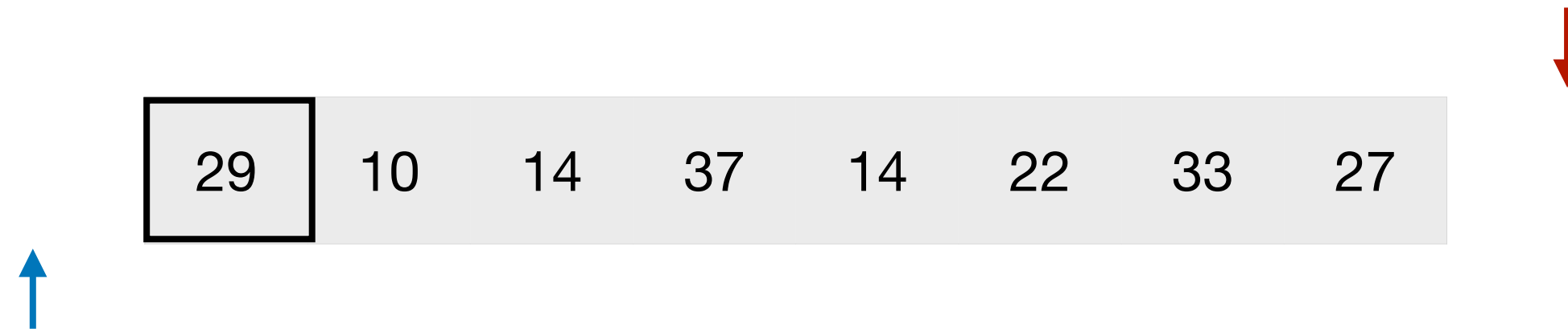
1. Choose first element of given array as pivot.

Quick sort

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

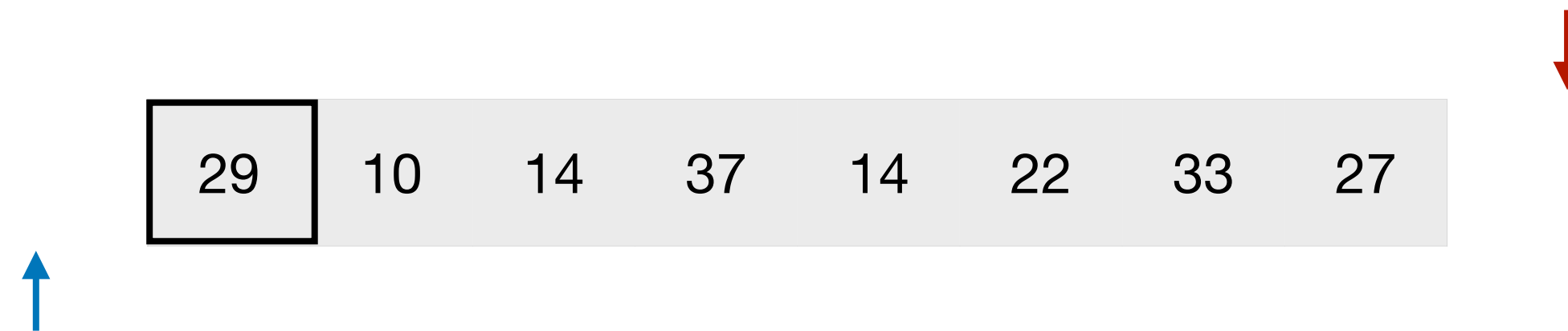
1. Choose first element of given array as pivot.

Quick sort



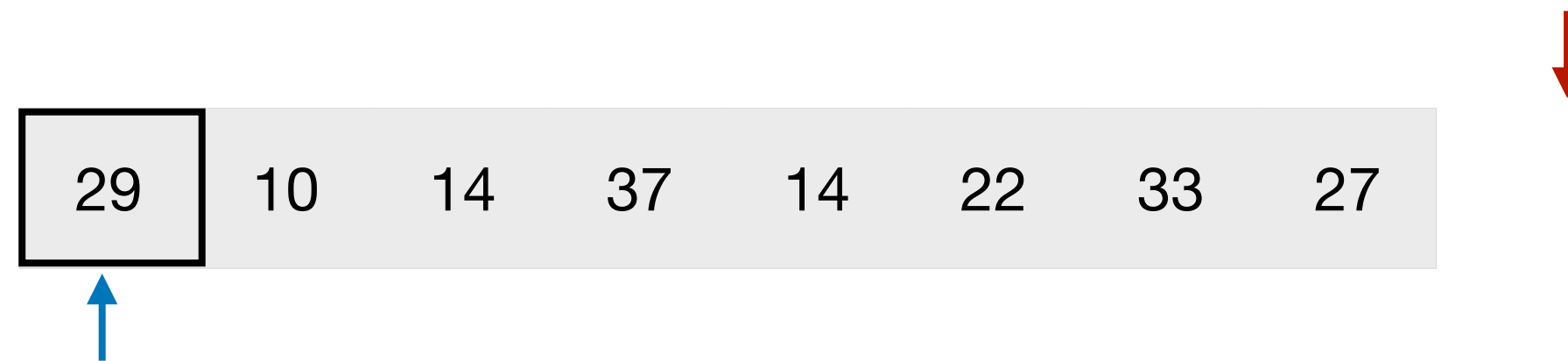
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.

Quick sort



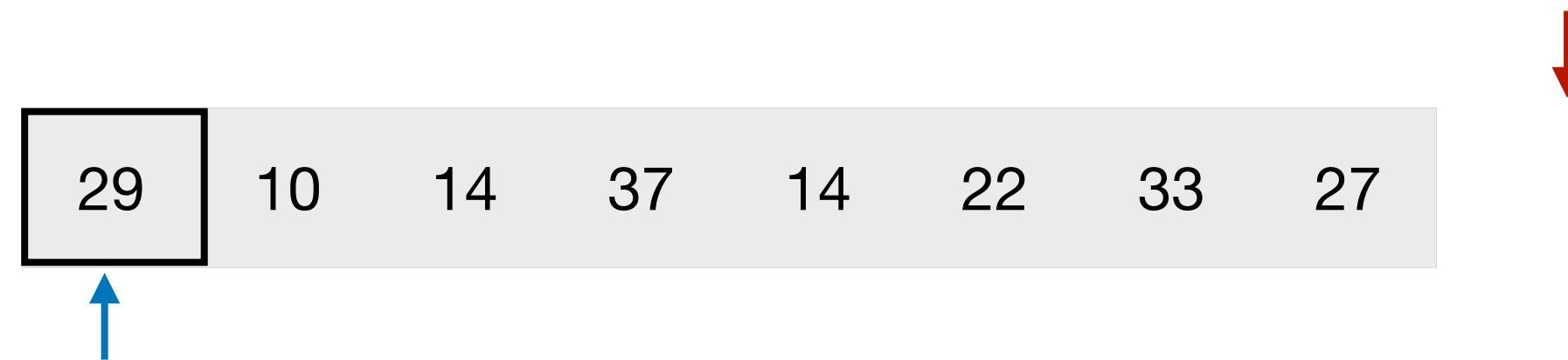
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot

Quick sort



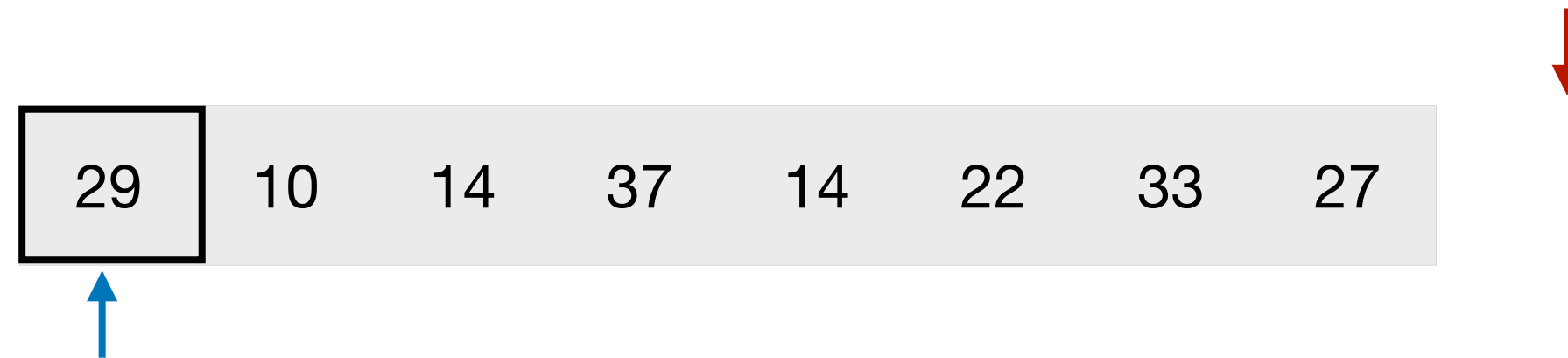
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot

Quick sort



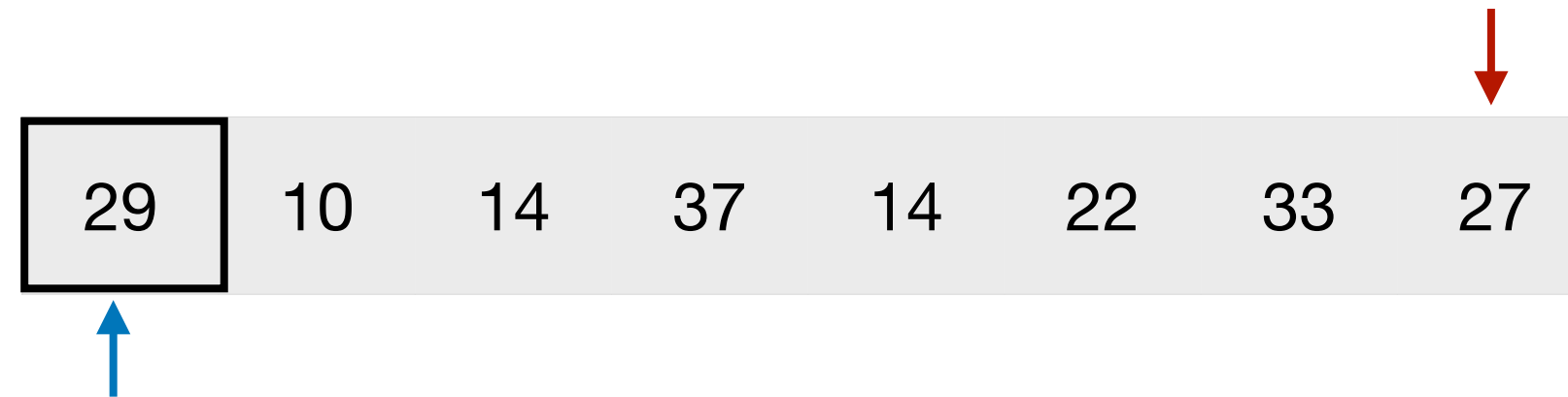
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot

Quick sort



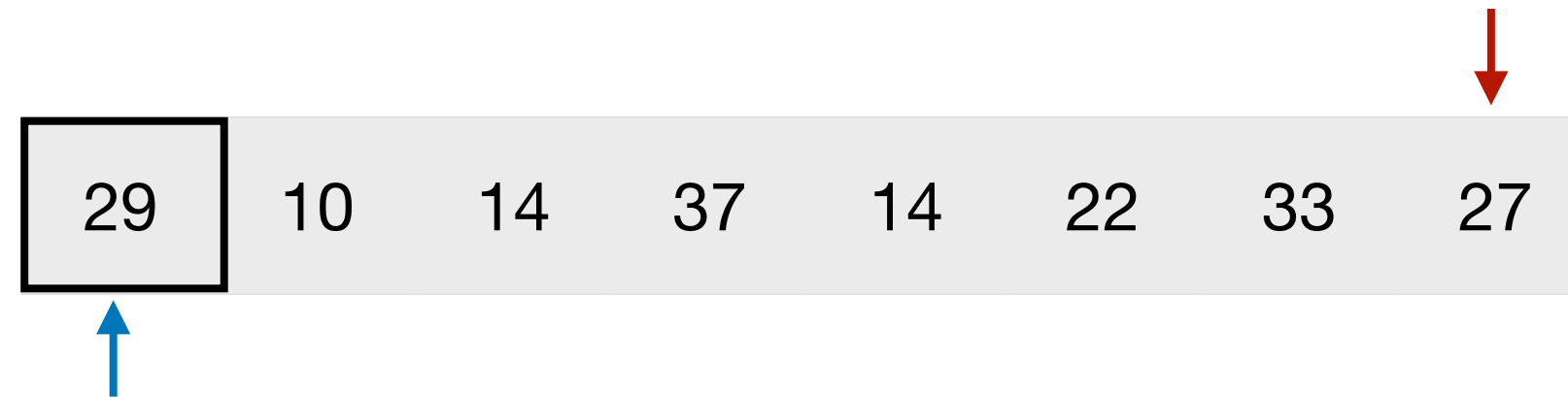
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.

Quick sort



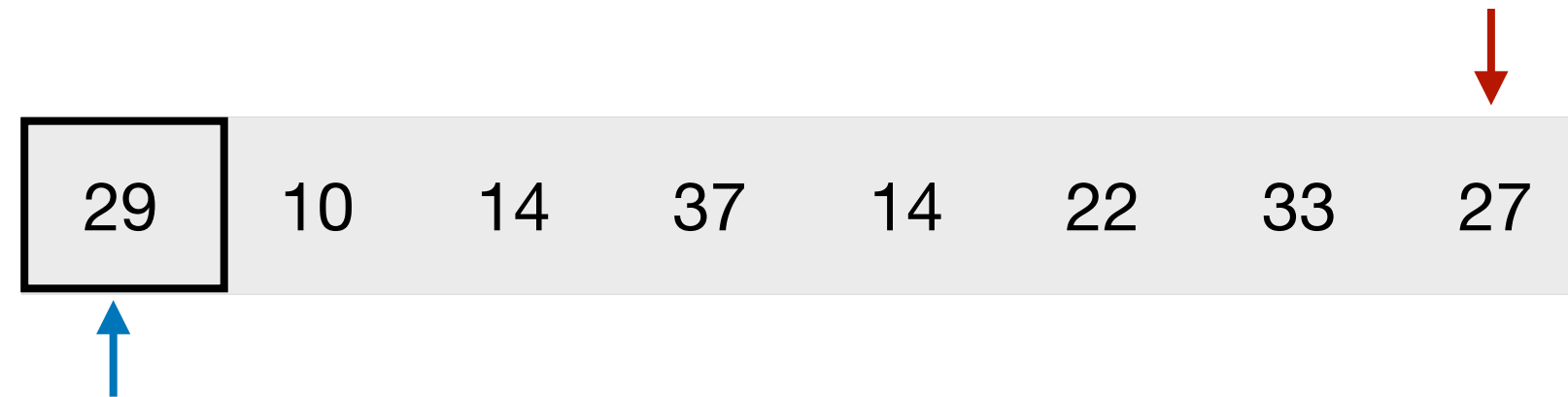
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.

Quick sort



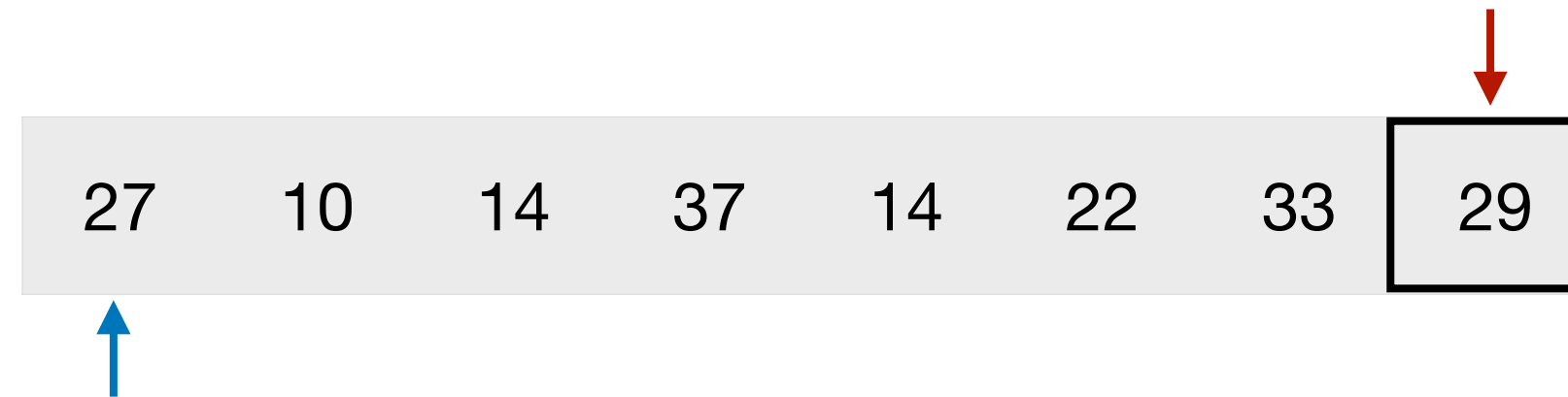
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.

Quick sort



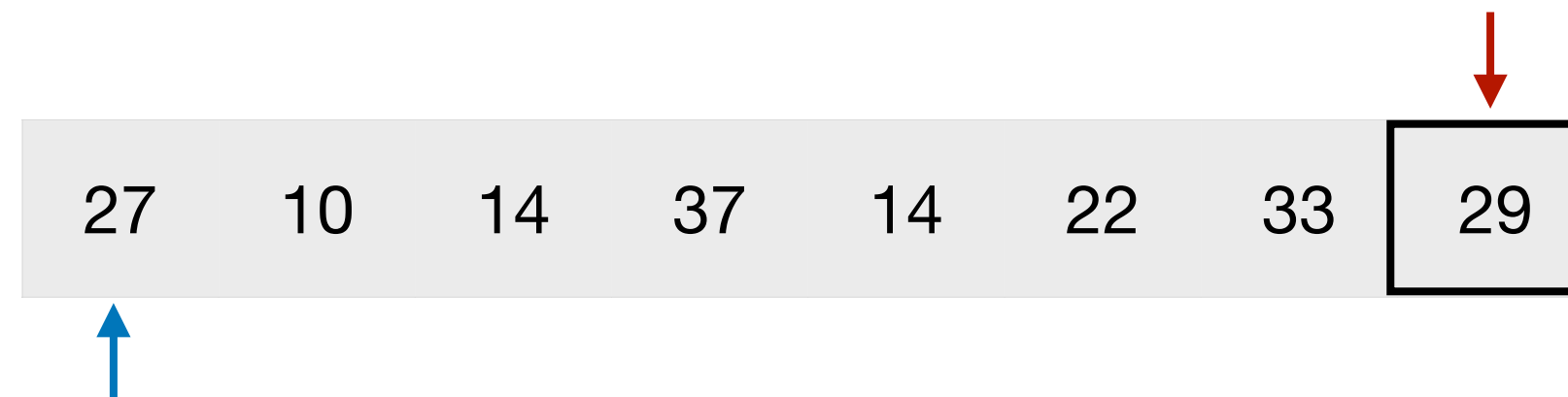
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.

Quick sort



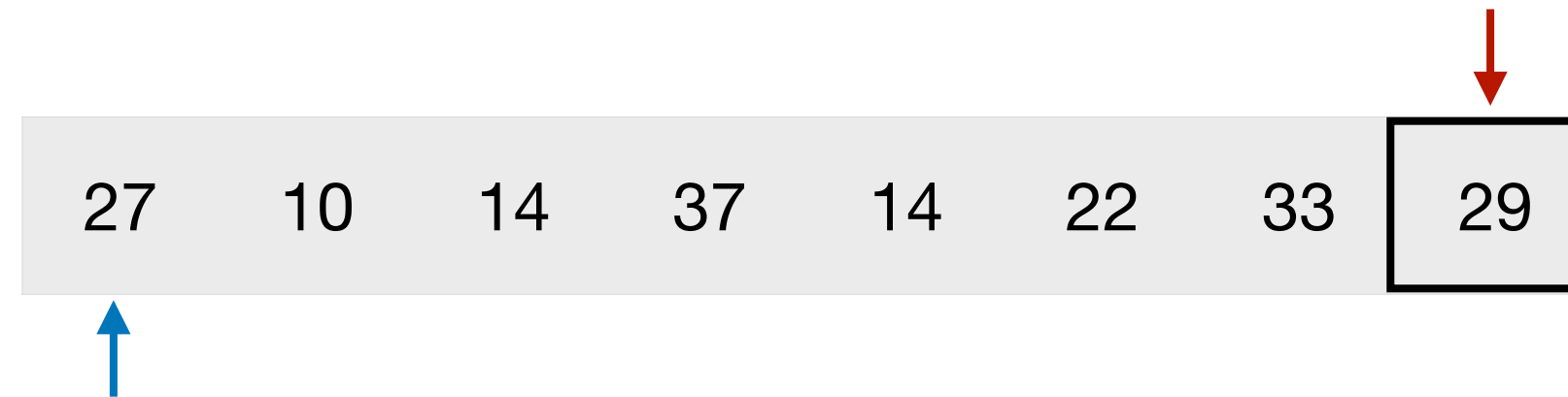
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.

Quick sort



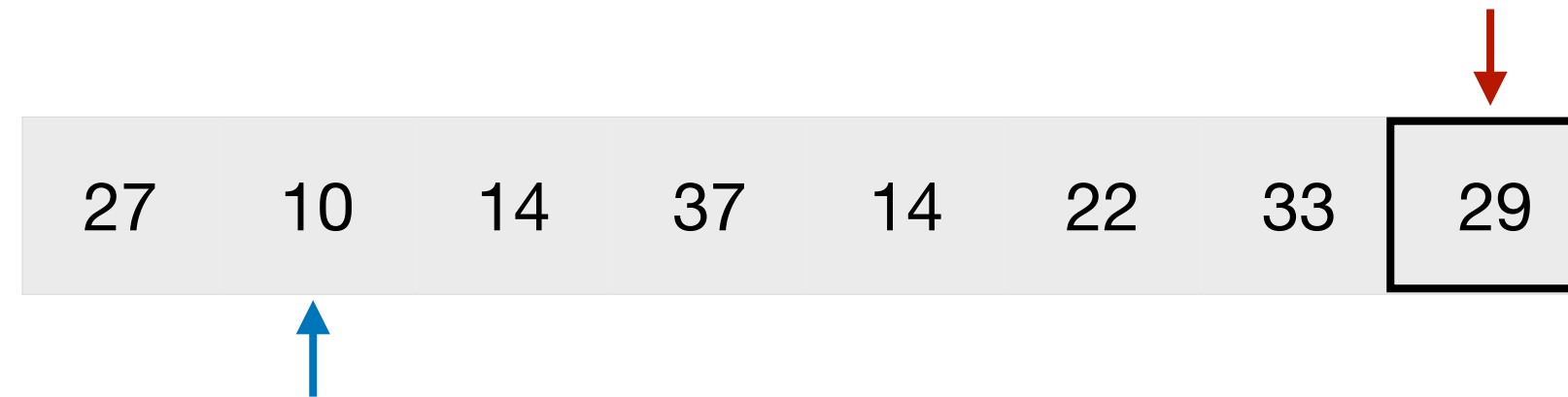
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.
6.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

Quick sort



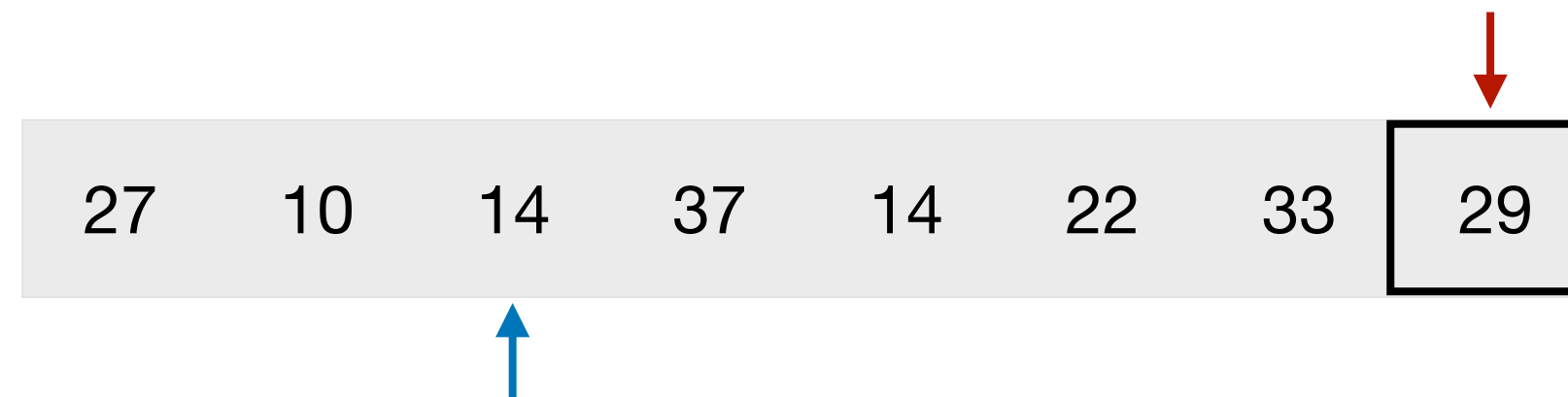
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



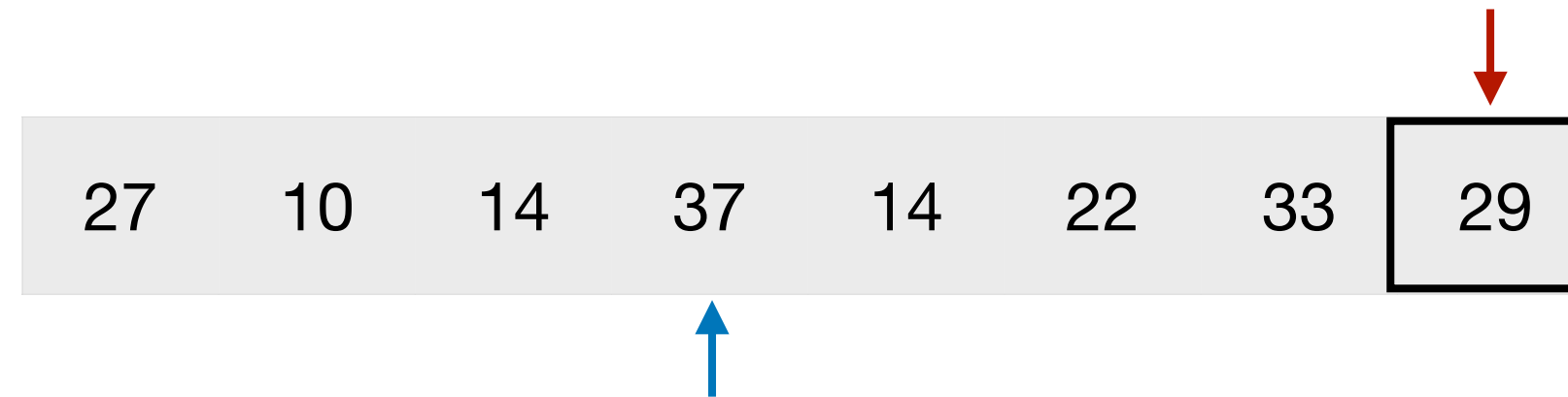
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



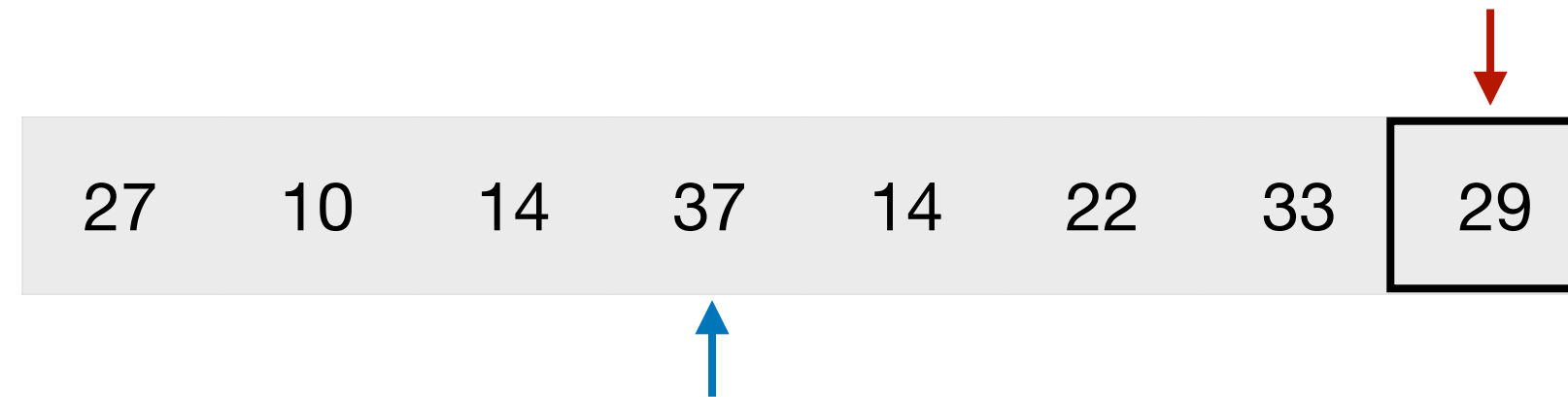
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



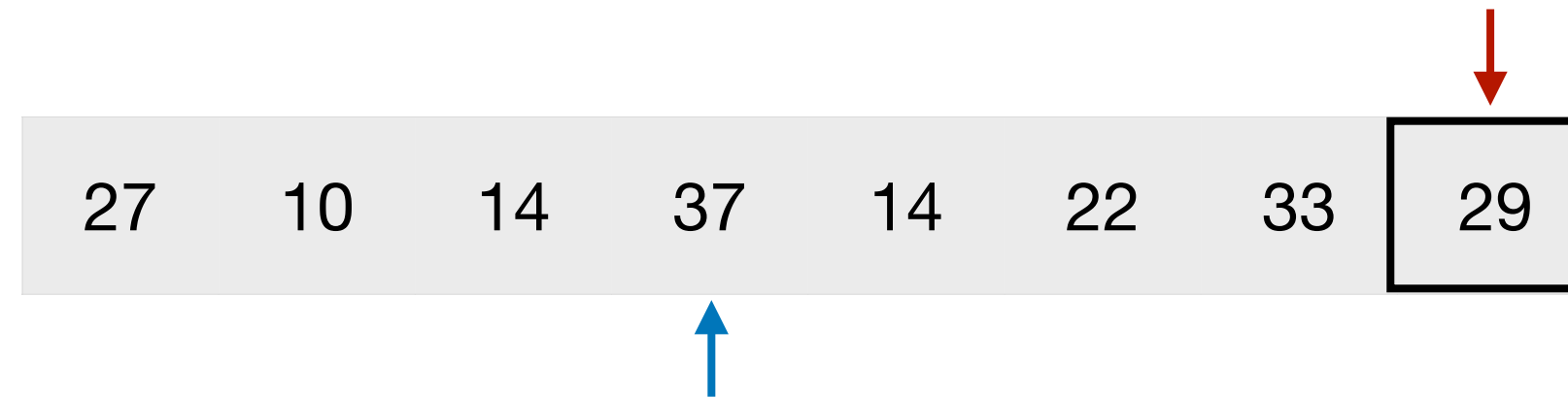
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



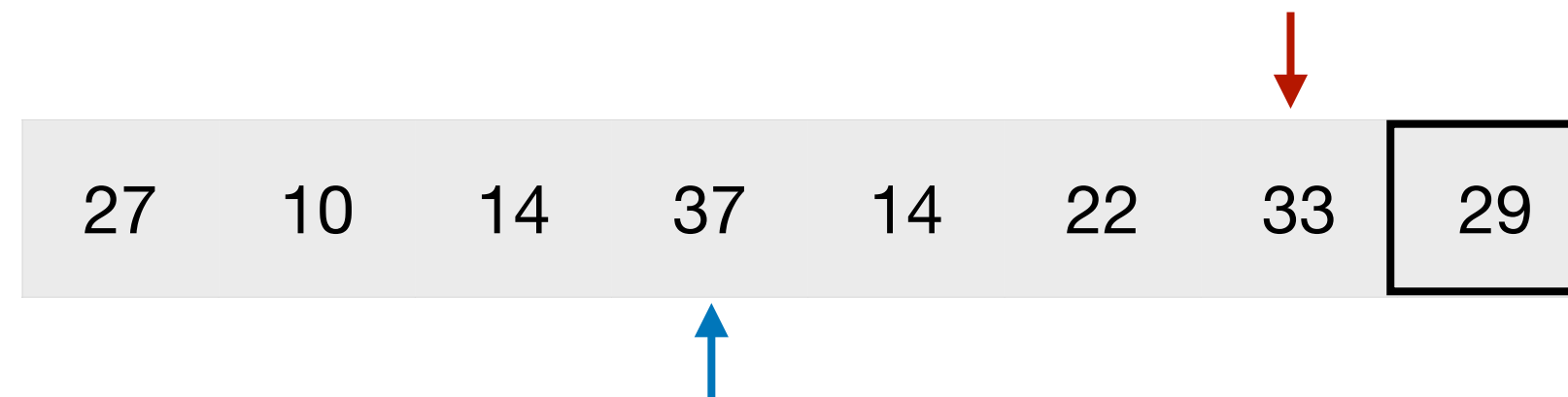
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



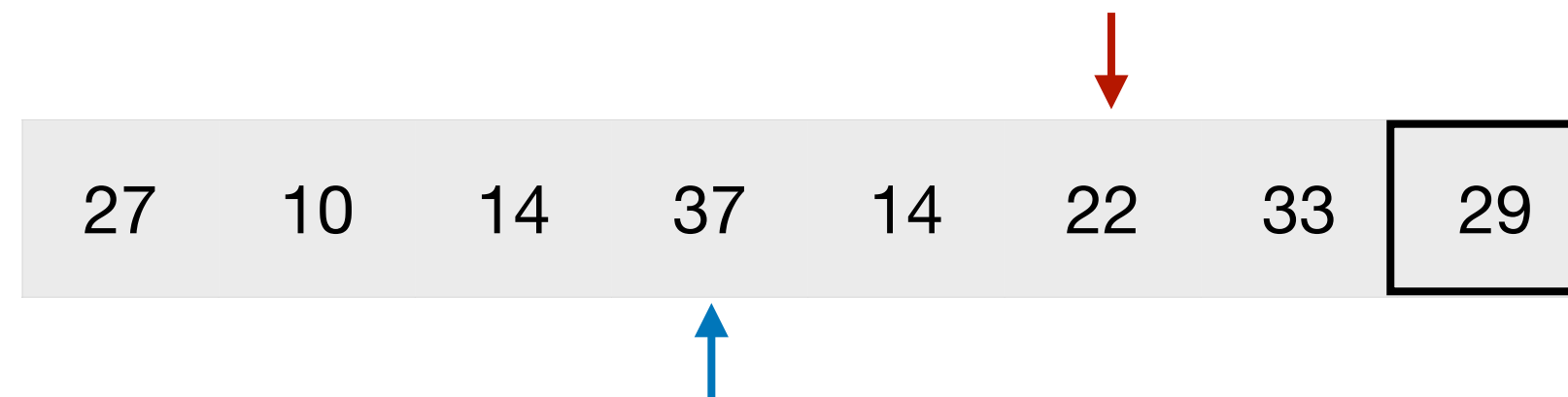
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

Quick sort



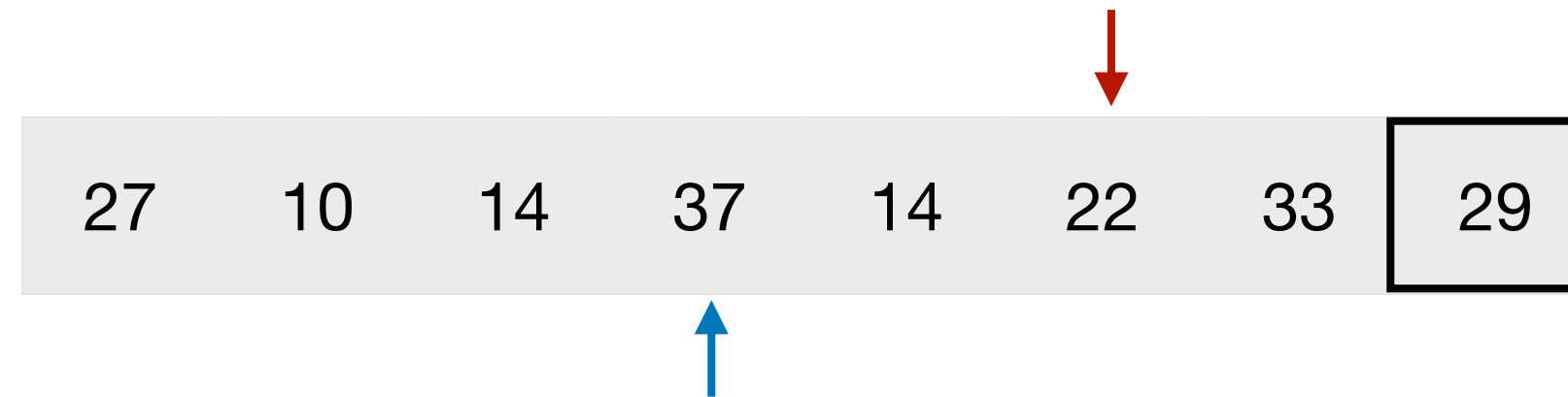
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

Quick sort



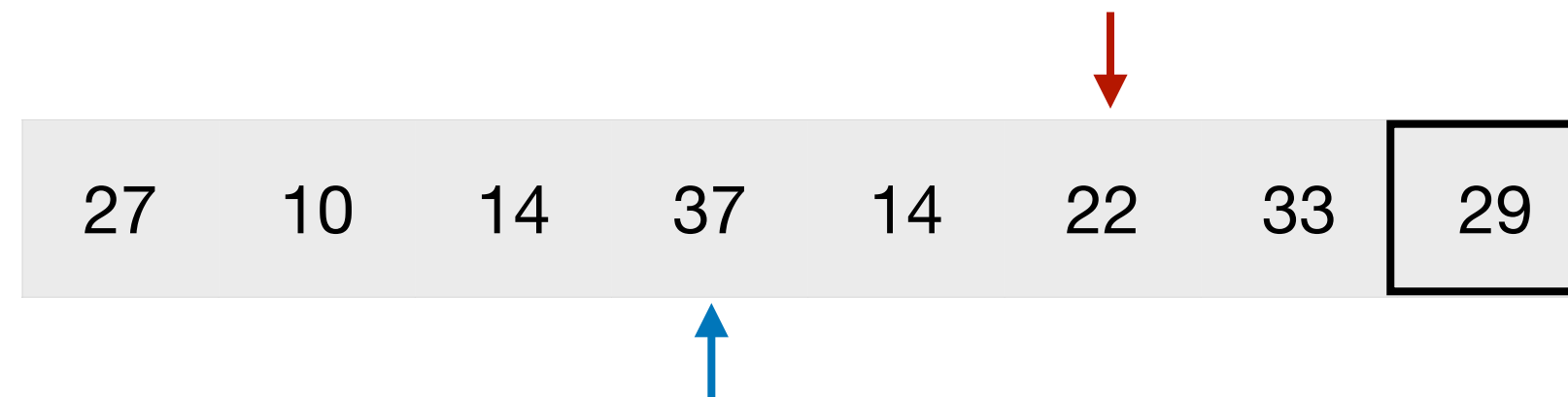
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

Quick sort



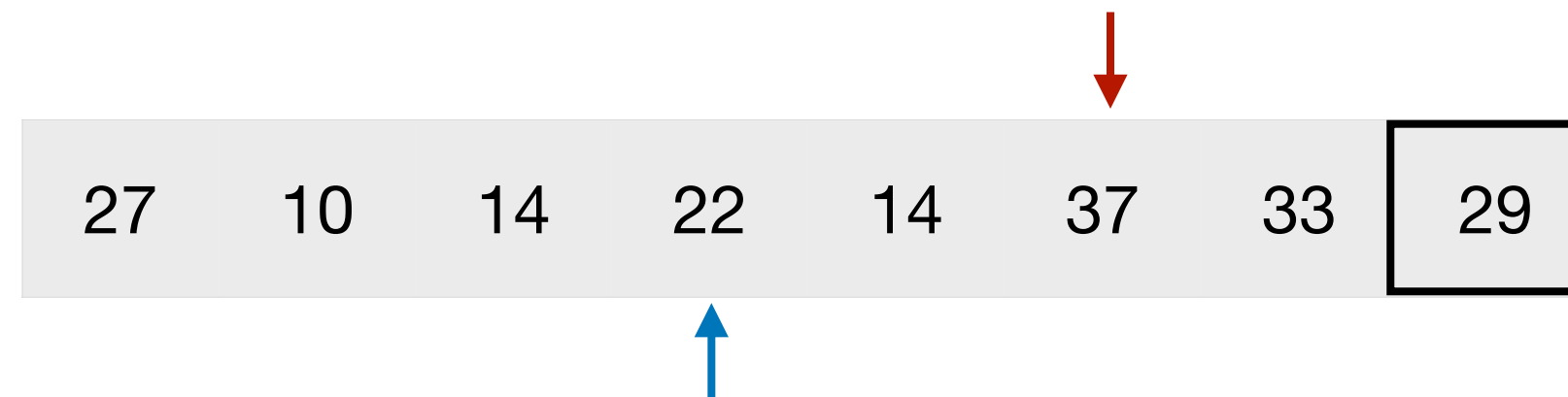
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

Quick sort



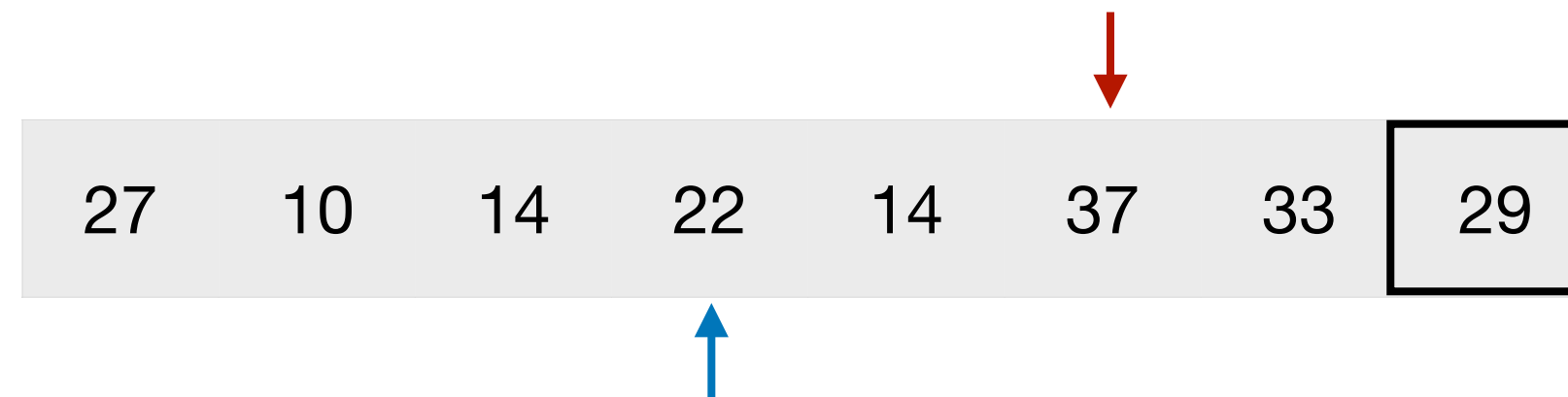
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
- 5. If neither pointers can move swap elements.**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

Quick sort



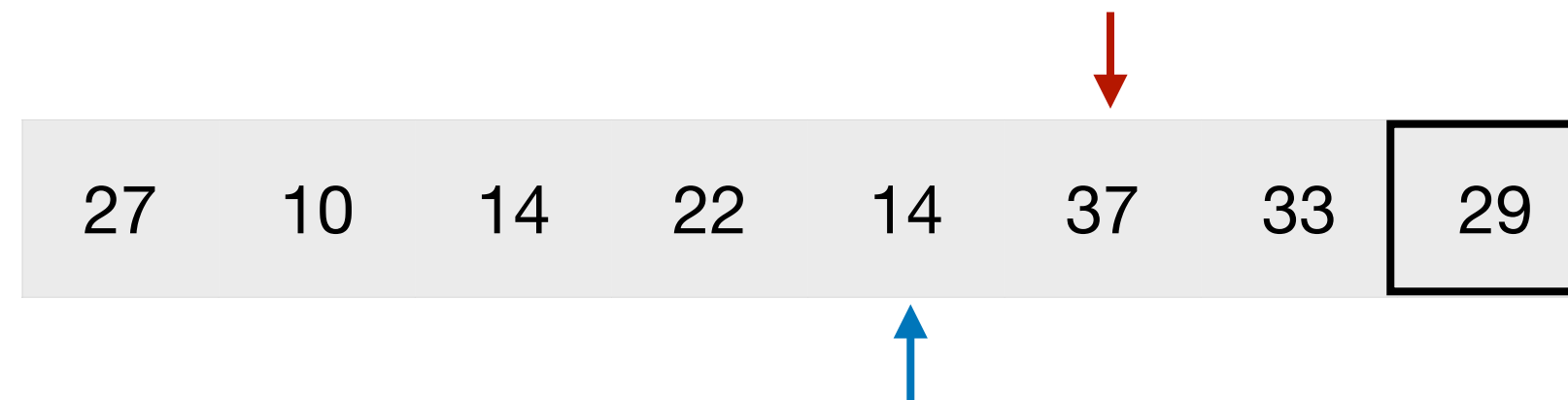
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
- 5. If neither pointers can move swap elements.**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



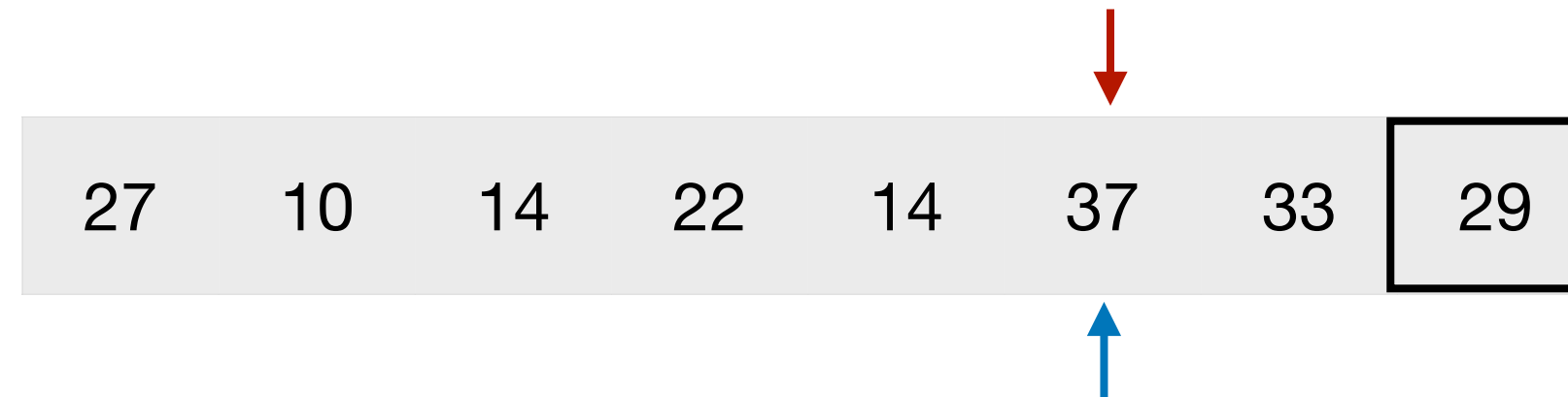
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



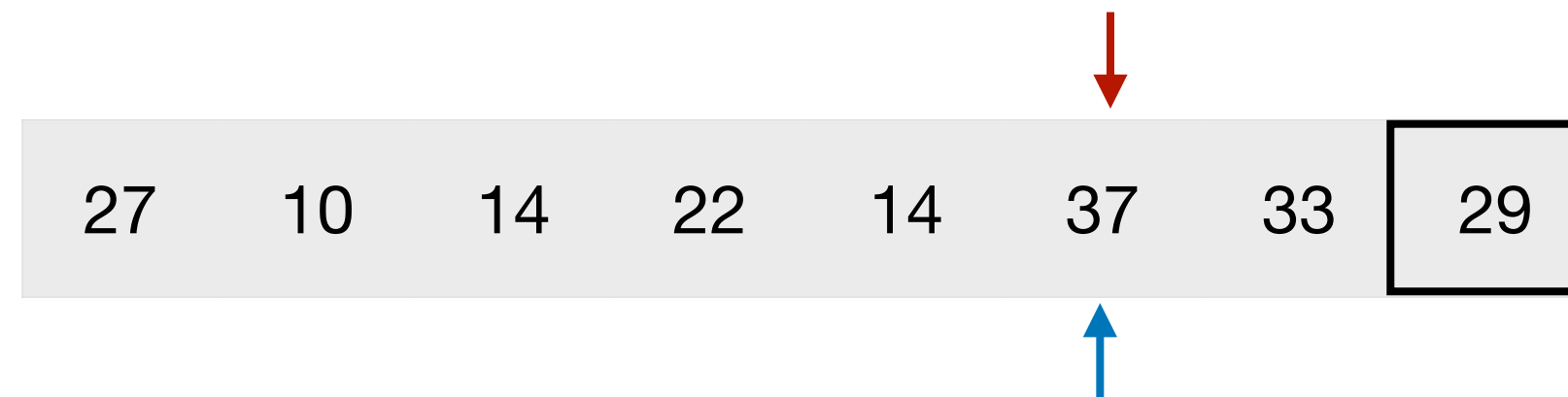
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



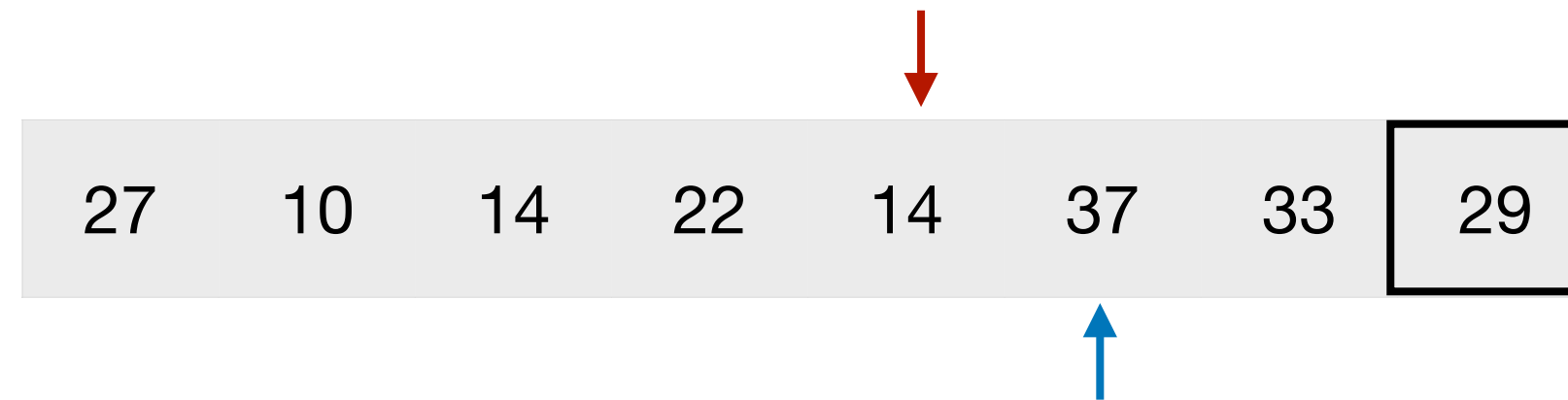
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
5. If neither pointers can move swap elements.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

Quick sort



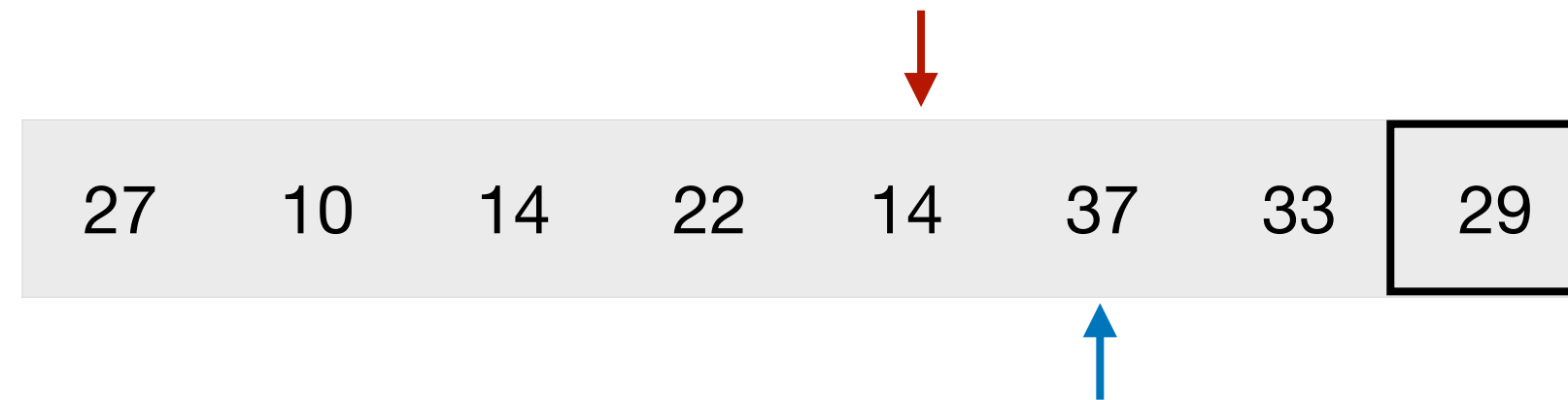
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

Quick sort



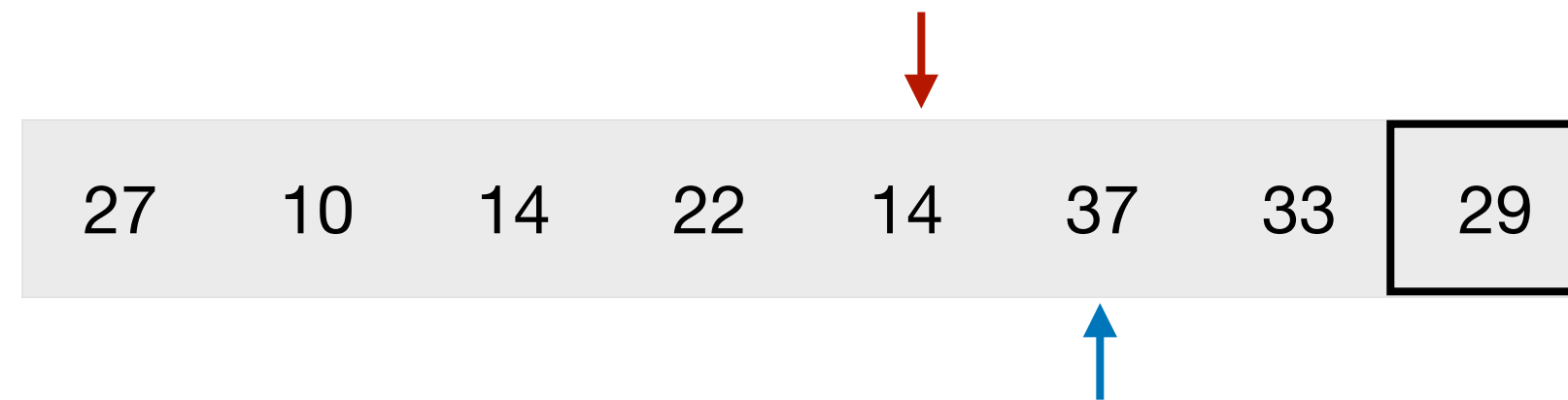
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

Quick sort



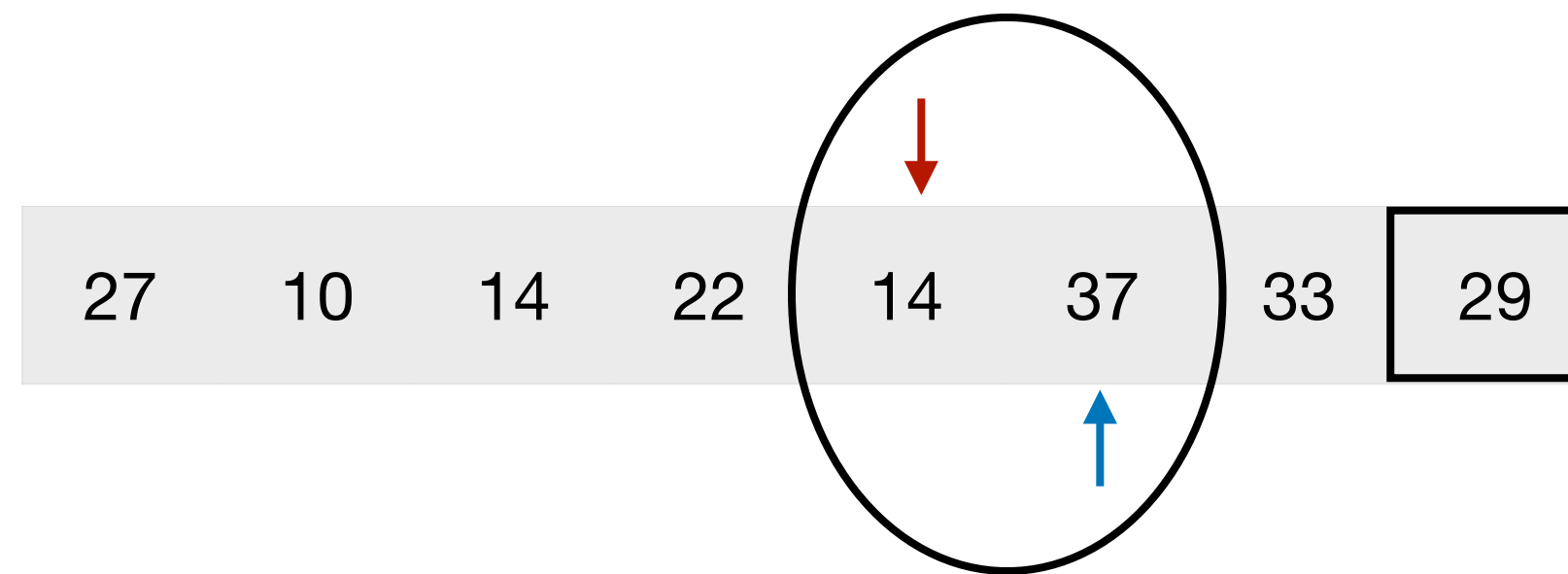
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
5. If neither pointers can move swap elements.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

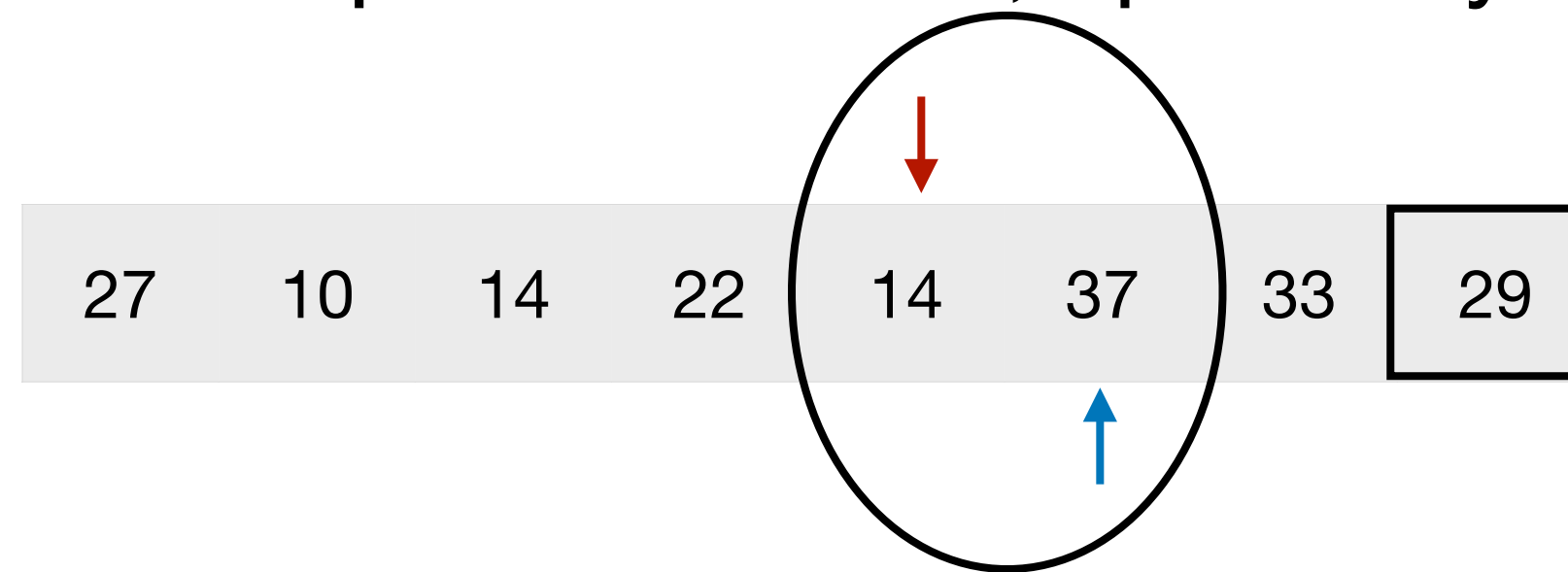
Quick sort



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
----	----	----	----	----	----	----	----

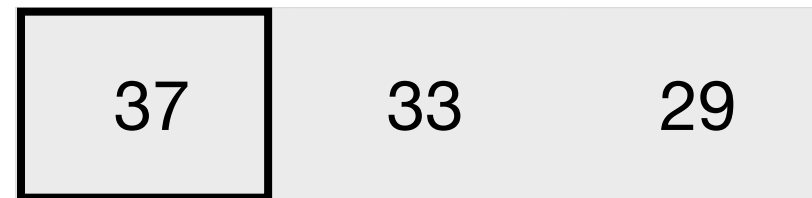
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. **If pointers cross/overlap, split array & recurse 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

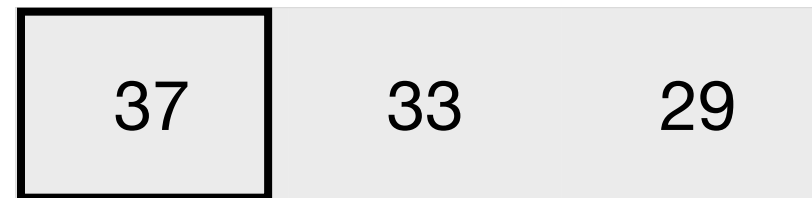
1. **Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



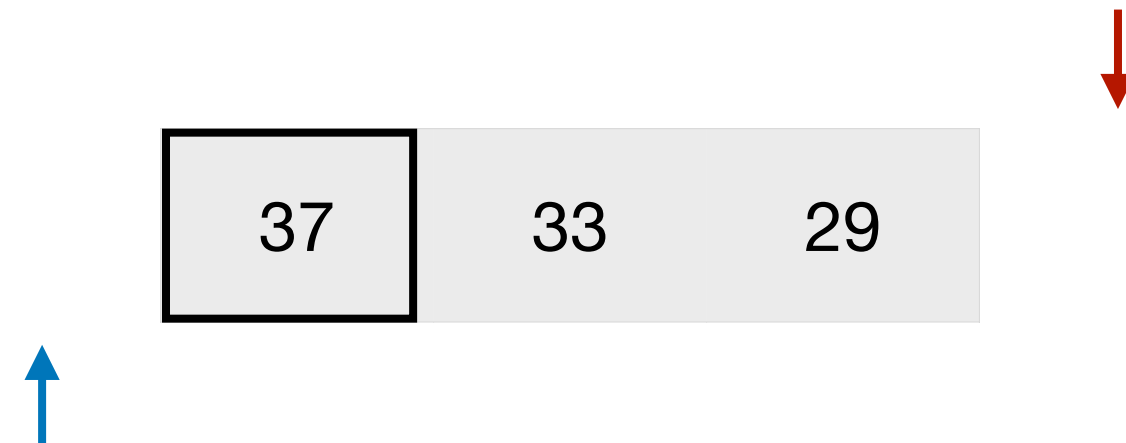
- 1. Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



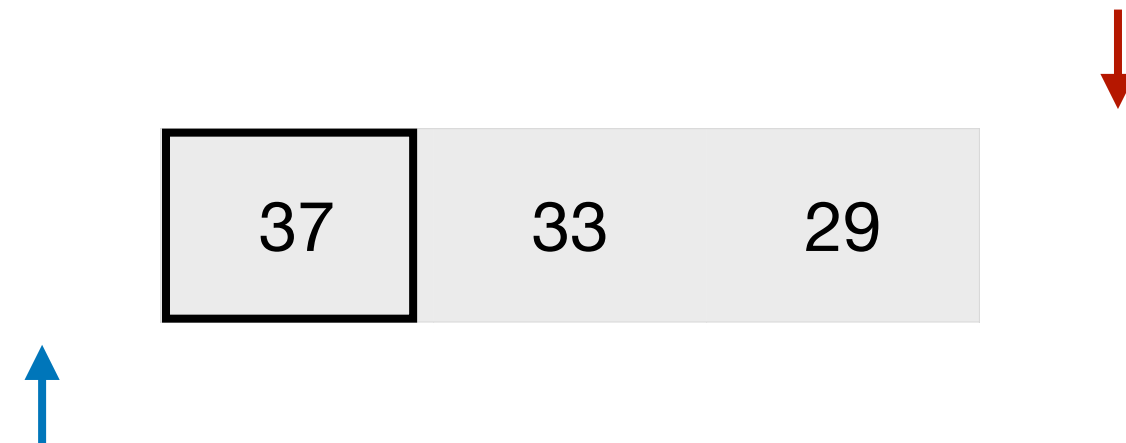
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



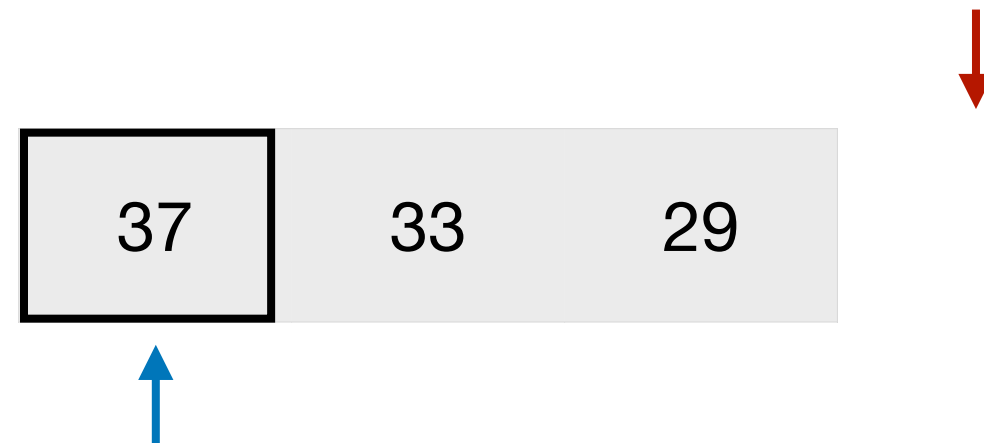
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



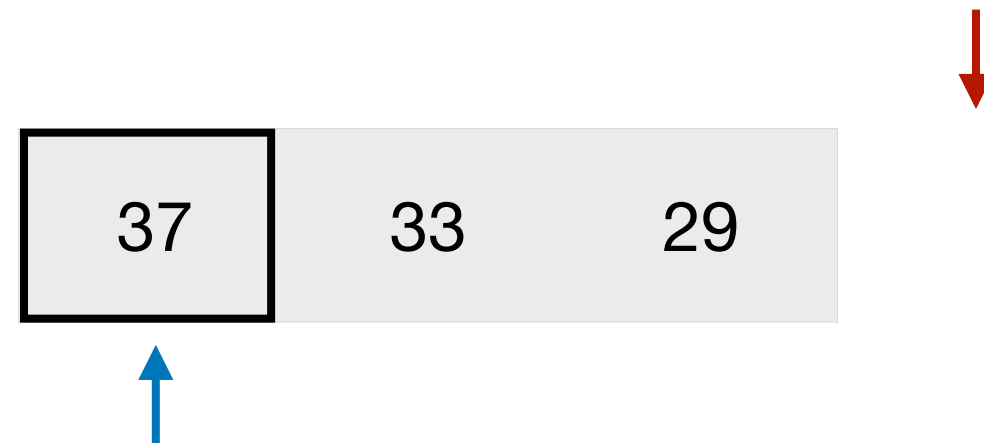
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



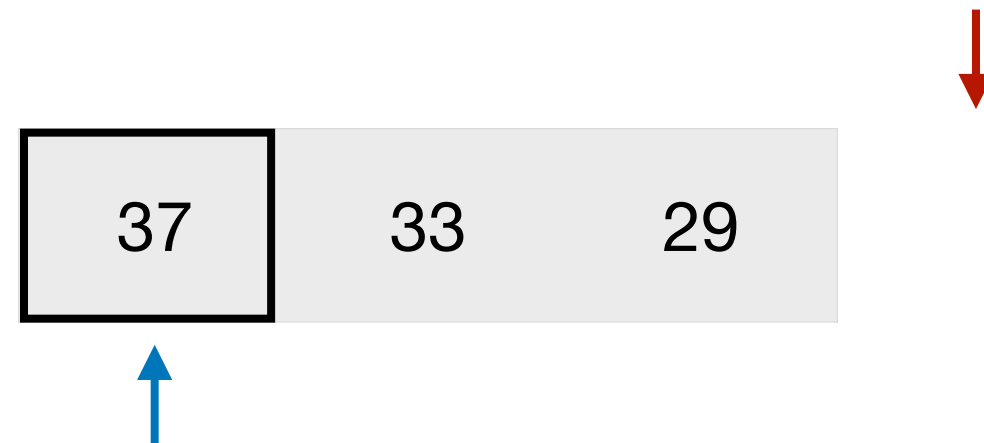
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



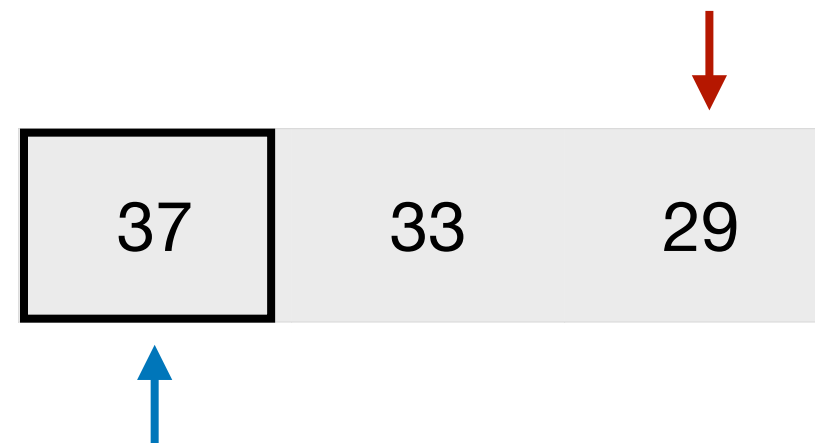
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



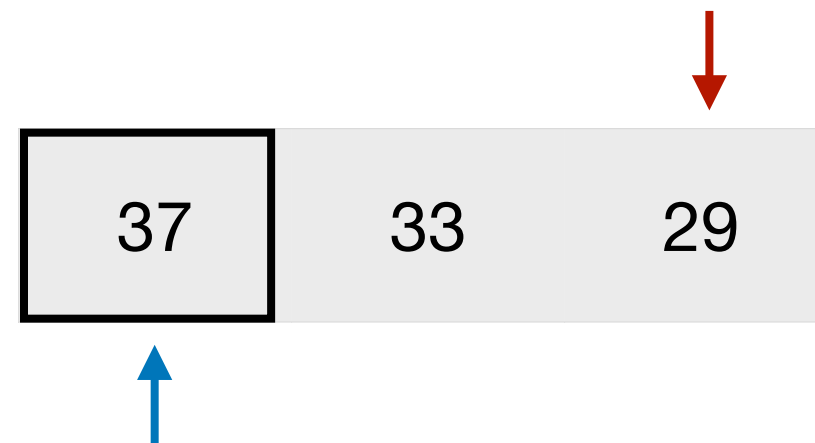
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
 1. If pointers cross/overlap, split array &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

Quick sort



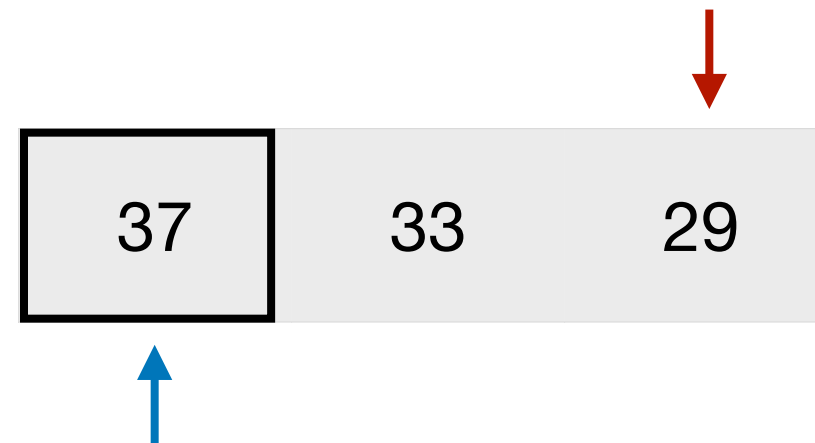
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
 1. If pointers cross/overlap, split array &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

Quick sort



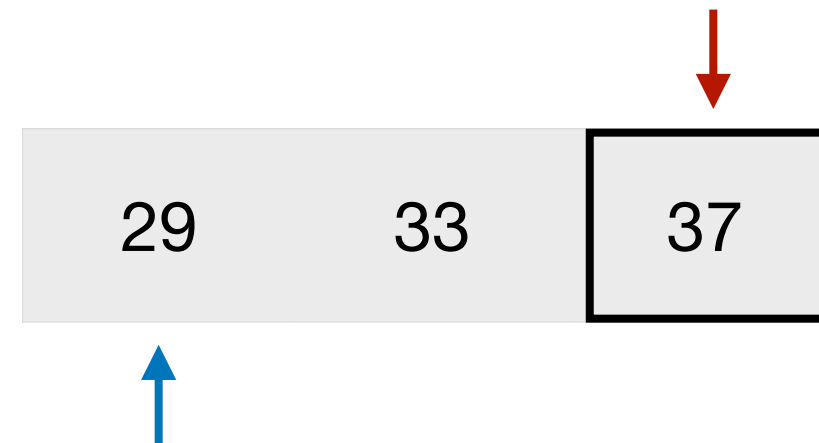
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
 1. If pointers cross/overlap, split array &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

Quick sort



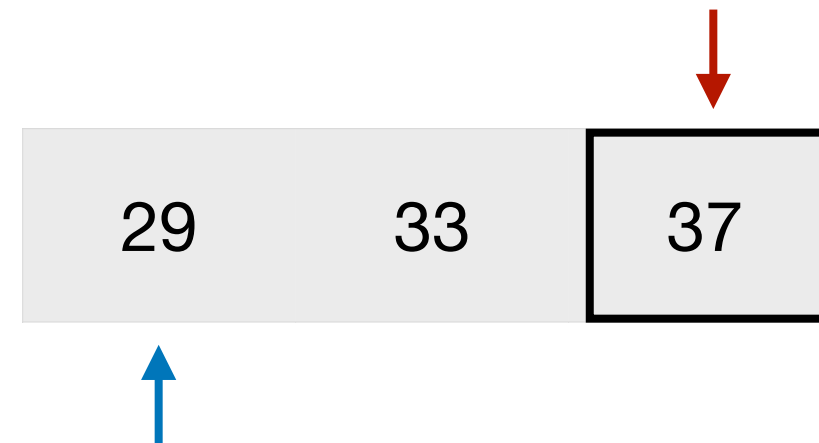
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array & recurse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

Quick sort



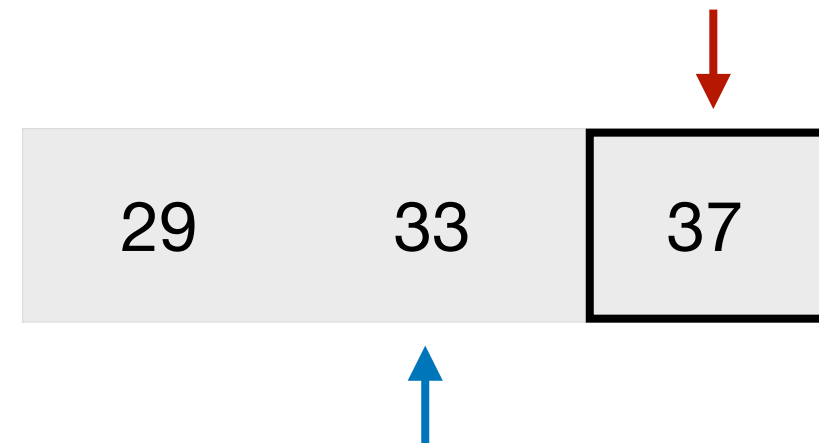
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array & recurse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

Quick sort



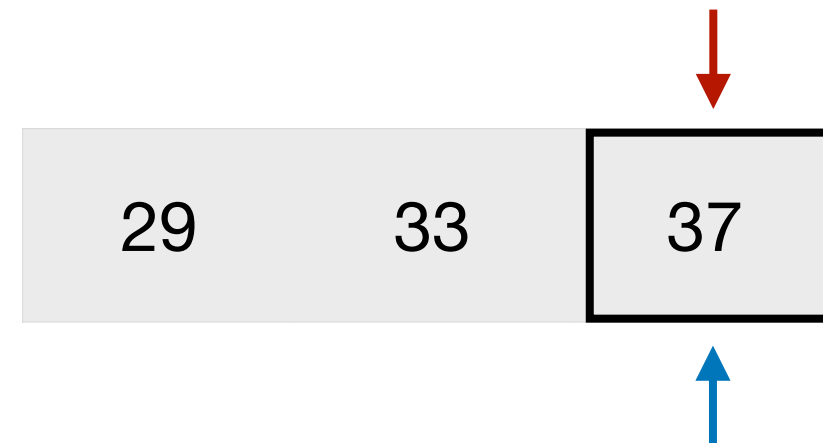
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



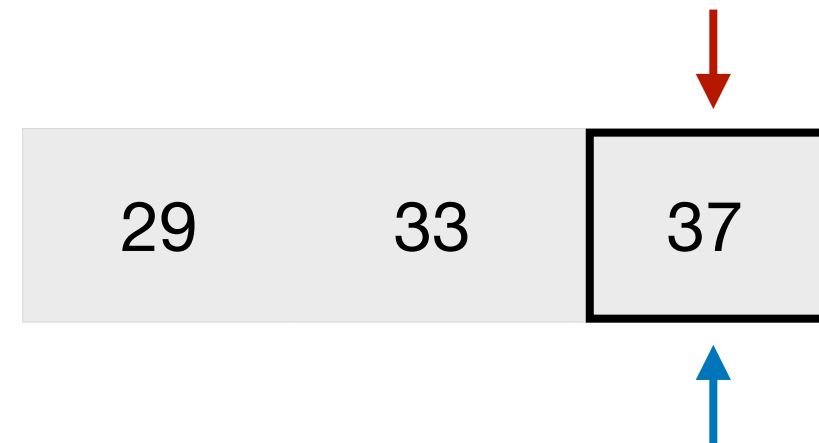
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



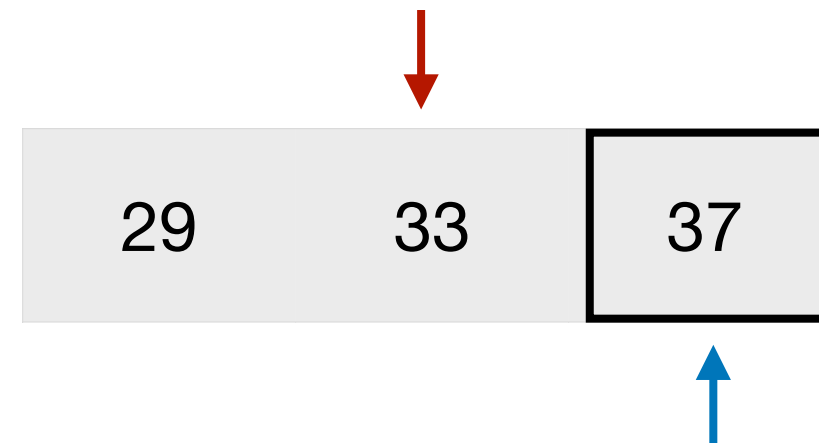
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



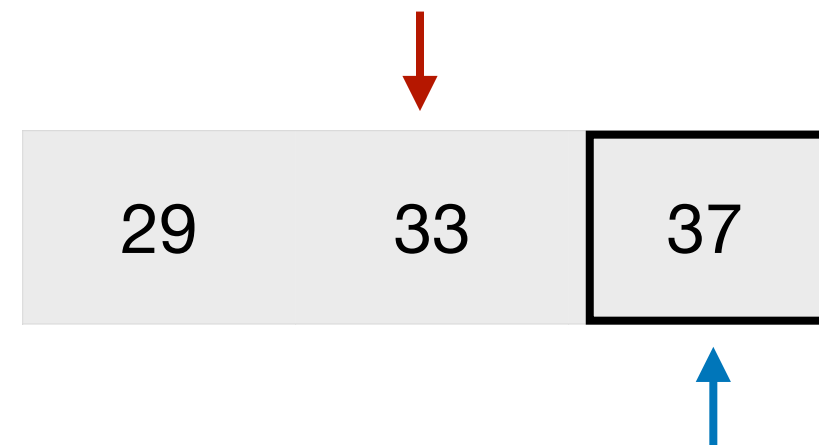
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
 1. If pointers cross/overlap, split array & 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



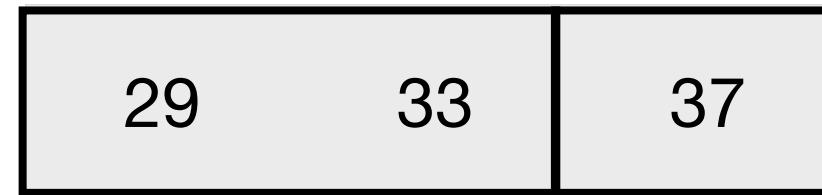
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. **If pointers cross/overlap, split array & recurse 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

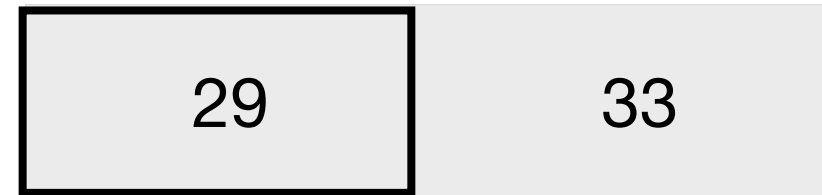
Quick sort

29

33

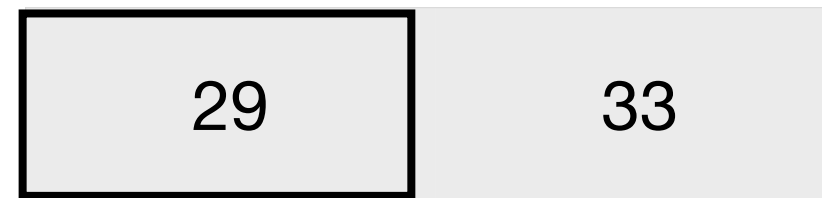
1. **Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



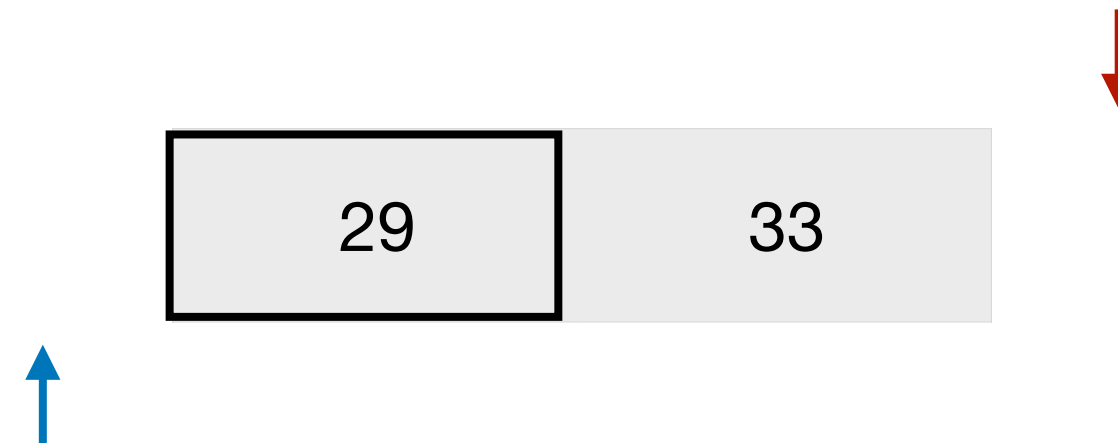
- 1. Choose first element of given array as pivot.**
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



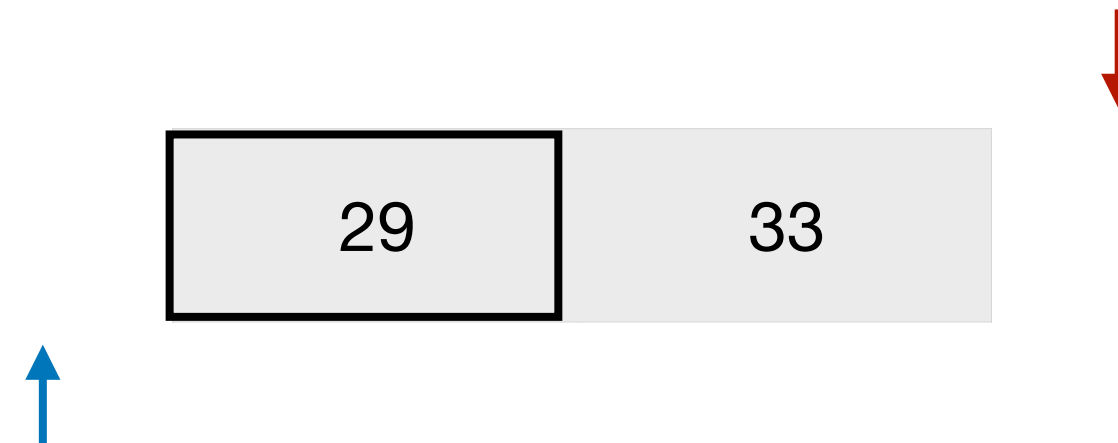
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



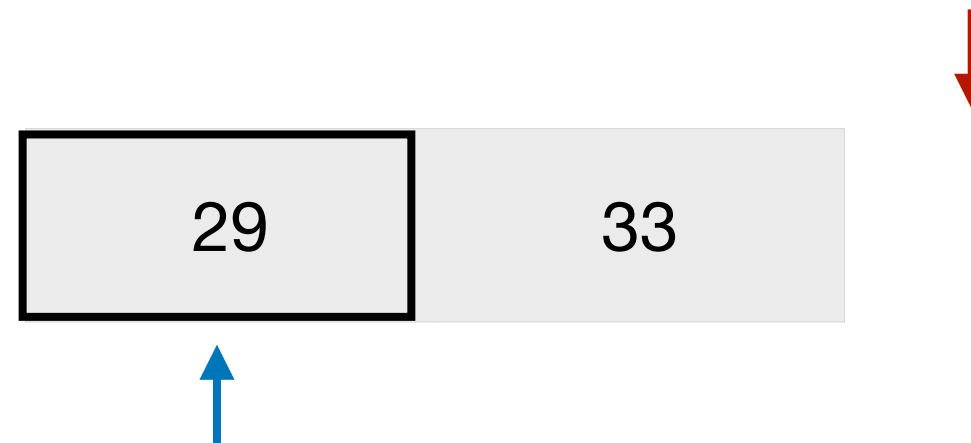
1. Choose first element of given array as pivot.
- 2. Maintain pointers from the left and right.**
3. Increment left pointer while the element it points to is less than pivot
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



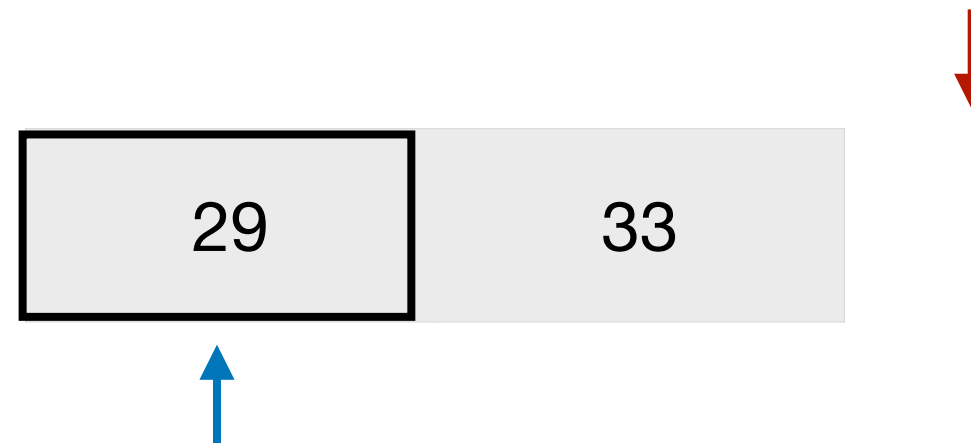
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



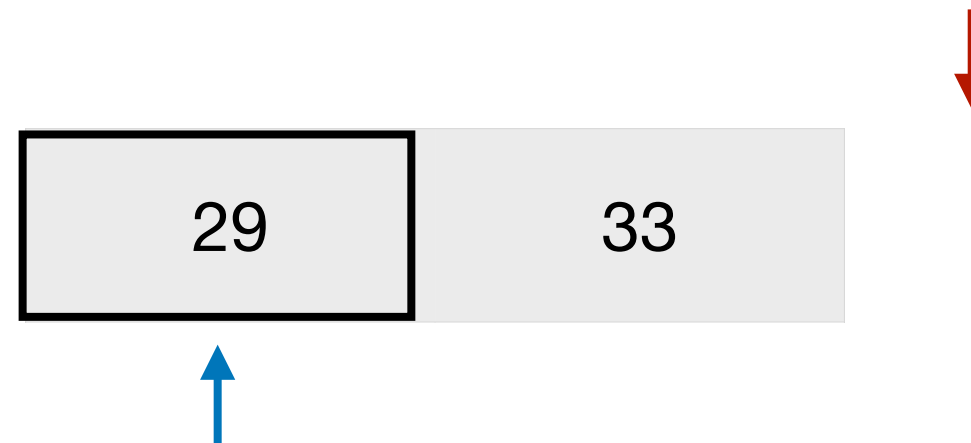
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



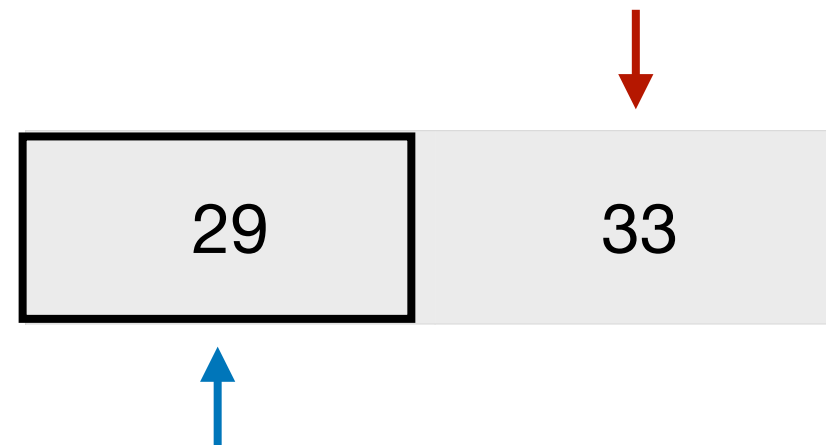
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
- 3. Increment left pointer while the element it points to is less than pivot**
4. Decrement right pointer while the element it points to is greater than pivot.
 1. If pointers cross/overlap, split array &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

Quick sort



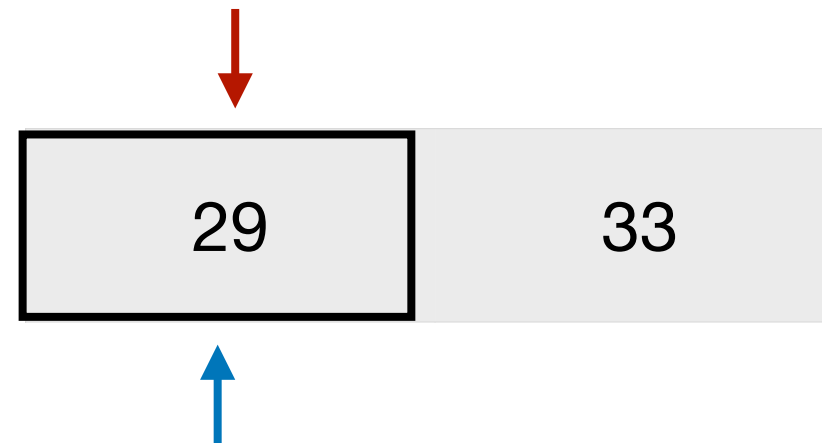
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
 1. If pointers cross/overlap, split array &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

Quick sort



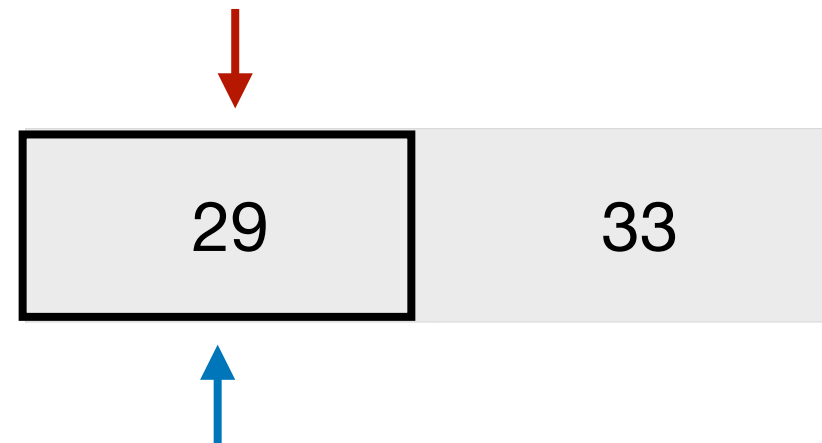
1. Choose first element of given array as pivot.
2. Maintain pointers from the left and right.
3. Increment left pointer while the element it points to is less than pivot
4. **Decrement right pointer while the element it points to is greater than pivot.**
 1. If pointers cross/overlap, split array & 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

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



Repeat on the other array.

Follow the steps on this array yourself.

Implementation

- Need mechanism to keep track of left/right pointers as we repeat on sub arrays — *use a STACK!*
- See https://github.com/iabraham/ece220-fa25/blob/main/lec11_1002/advanced/quicksort.c for an iterative version using stack.
- Usually implemented with *recursion*: Topic of next lecture!