

Quicksort Algorithm

Given an array of n elements (e.g., integers):

- If array only contains one element, return
- Else
 - pick one element to use as *pivot*.
 - Partition elements into two sub-arrays:
 - Elements less than or equal to pivot
 - Elements greater than pivot
 - Quicksort two sub-arrays
 - Return results

Example

We are given array of n integers to sort:

40	20	10	80	60	50	7	30	10 0
----	----	----	----	----	----	---	----	---------

Pick Pivot Element

There are a number of ways to pick the pivot element. In this example, we will use the first element in the array:

40	20	10	80	60	50	7	30	10 0
----	----	----	----	----	----	---	----	---------

Partitioning Array

Given a pivot, partition the elements of the array such that the resulting array consists of:

1. One sub-array that contains elements $< \text{pivot}$
2. Another sub-array that contains elements $\geq \text{pivot}$

The sub-arrays are stored in the original data array.

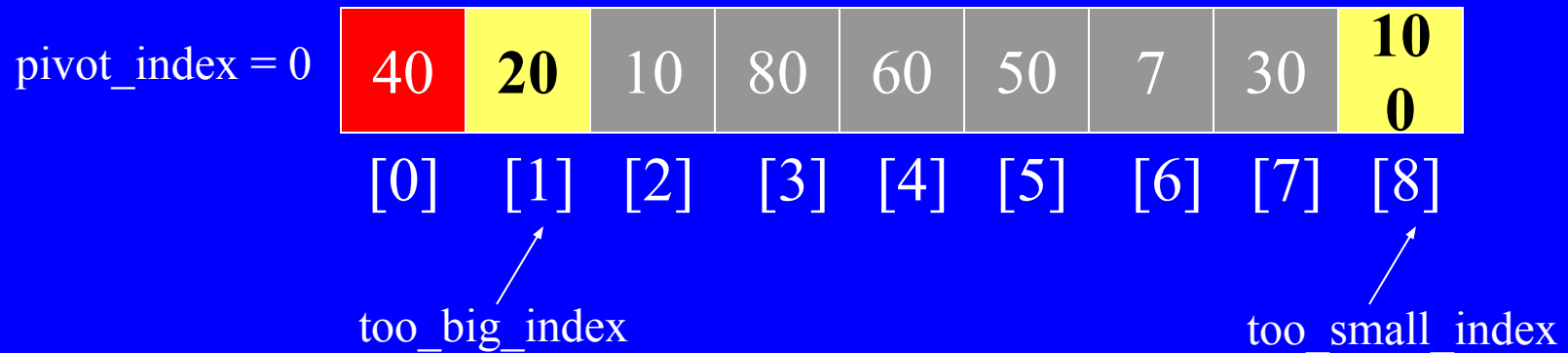
pivot_index = 0

40	20	10	80	60	50	7	30	10 0
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

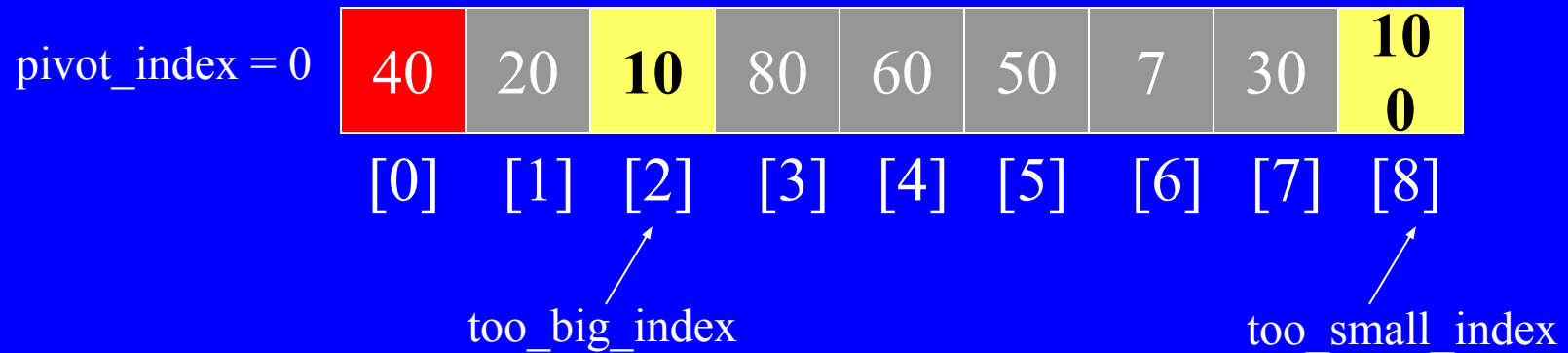
too_big_index

too_small_index

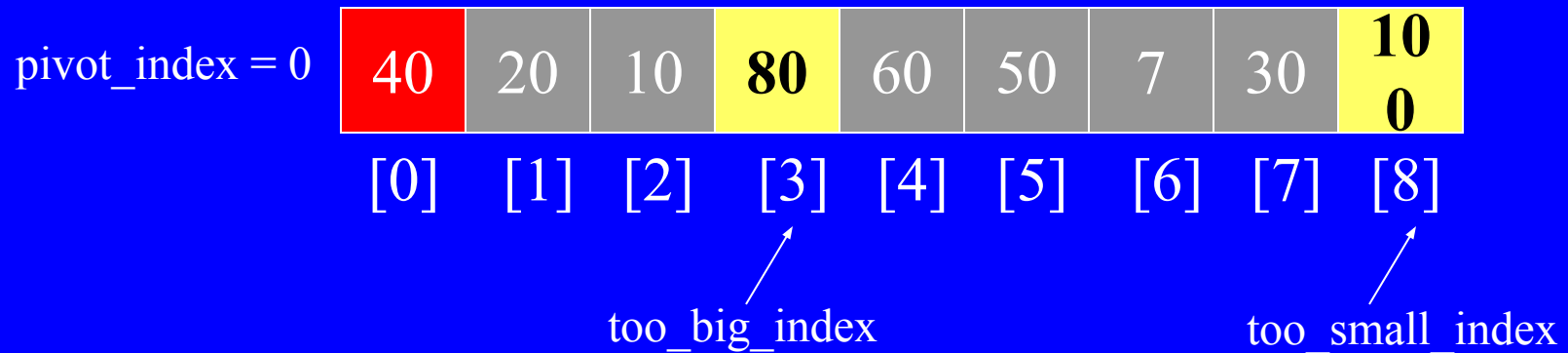
1. While `data[too_big_index] <= data[pivot]`
 `++too_big_index`



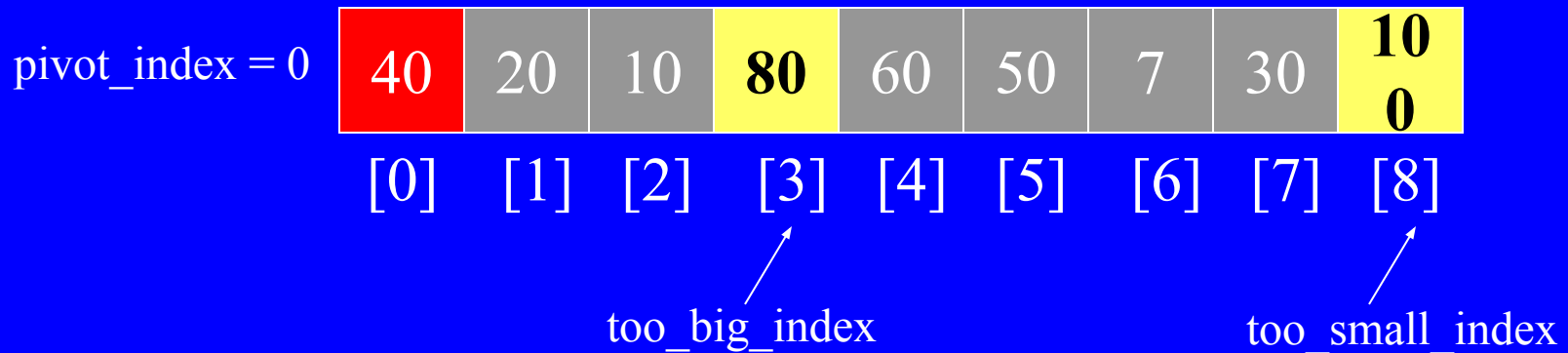
1. While `data[too_big_index] <= data[pivot]`
 `++too_big_index`



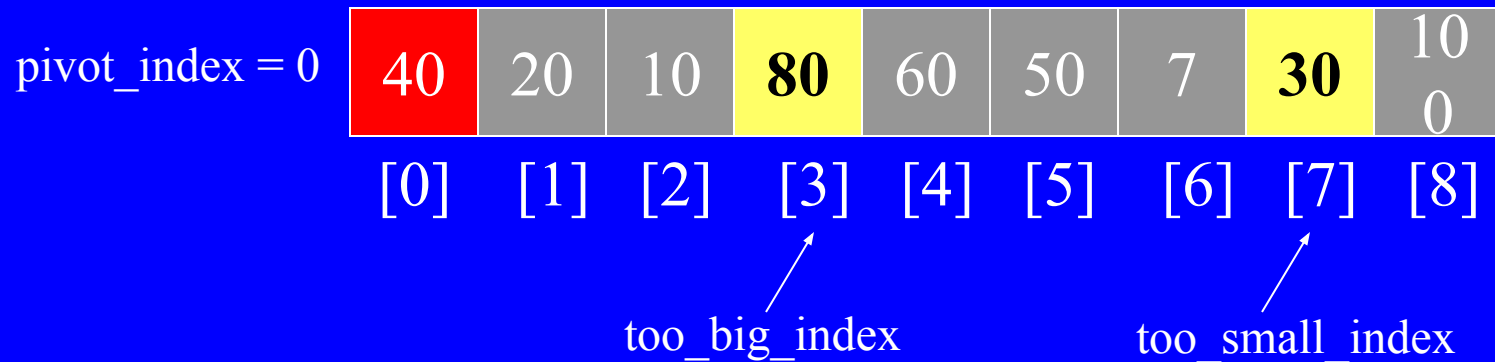
1. While `data[too_big_index] <= data[pivot]`
 `++too_big_index`



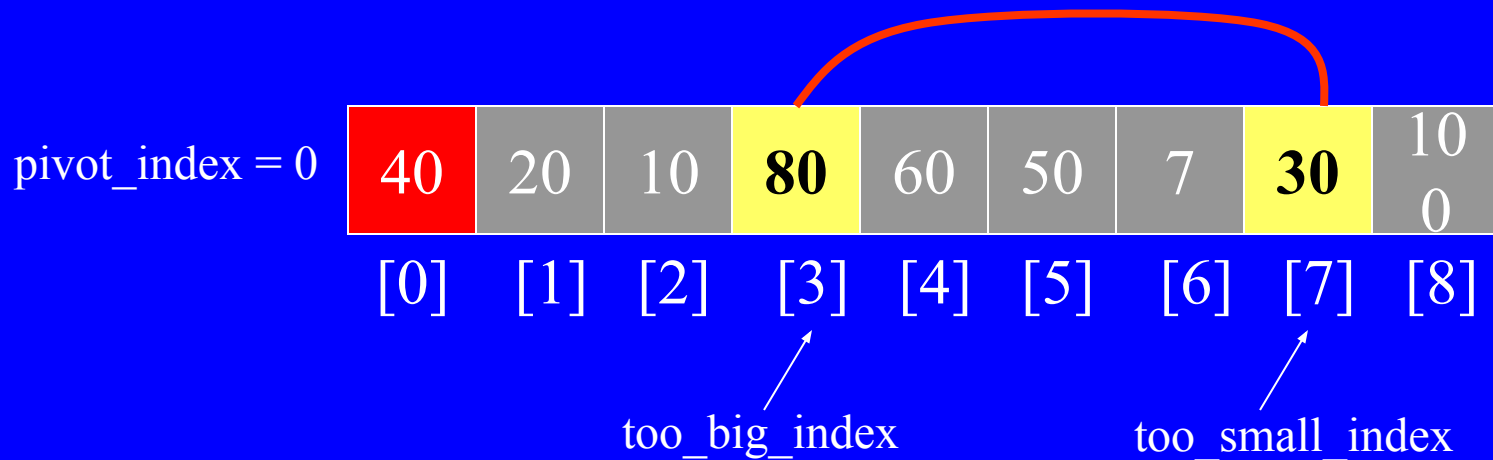
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index



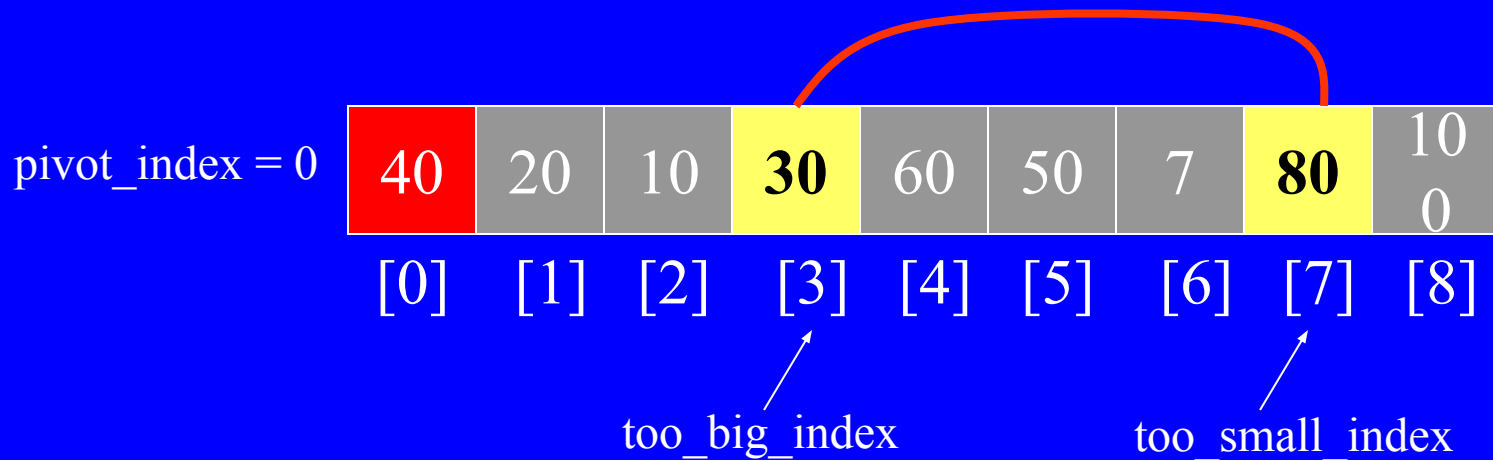
1. While `data[too_big_index] <= data[pivot]`
 `++too_big_index`
2. While `data[too_small_index] > data[pivot]`
 `--too_small_index`



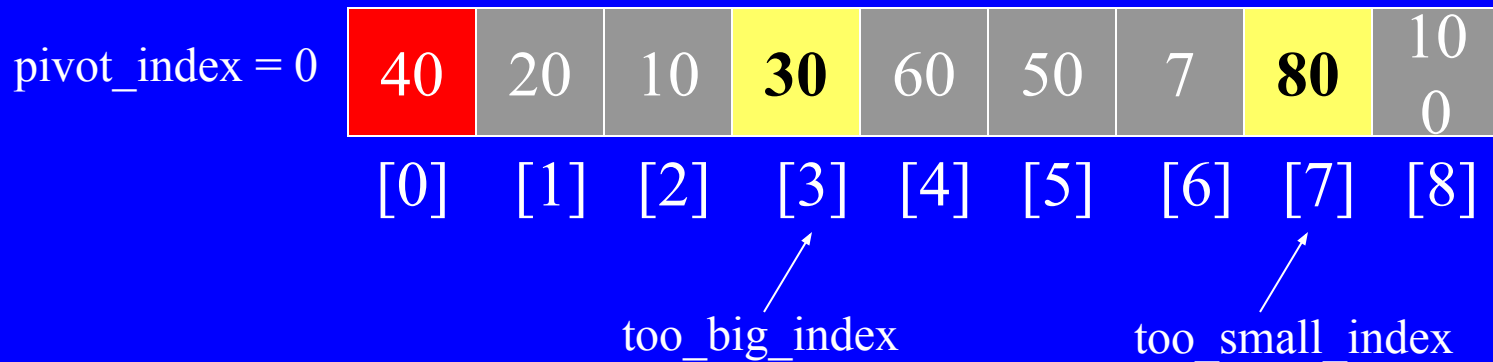
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 $++\text{too_big_index}$
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 $--\text{too_small_index}$
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



- 1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 $++\text{too_big_index}$
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 $--\text{too_small_index}$
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.

$\text{pivot_index} = 0$

40	20	10	30	60	50	7	80	100
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

\nearrow too_big_index \nearrow too_small_index

- 1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.

$\text{pivot_index} = 0$

40	20	10	30	60	50	7	80	10
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

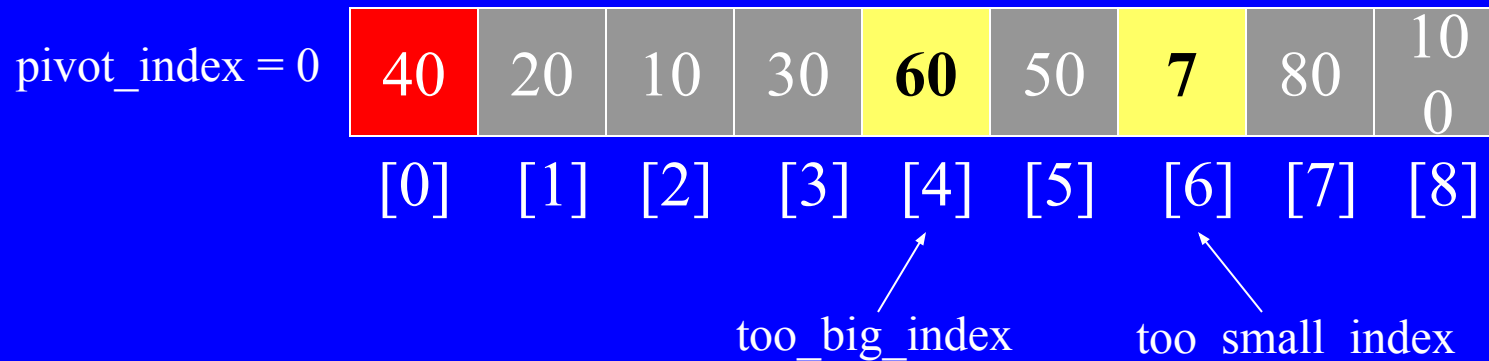
 ↑ ↑
 too_big_index too_small_index

1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
- 2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.

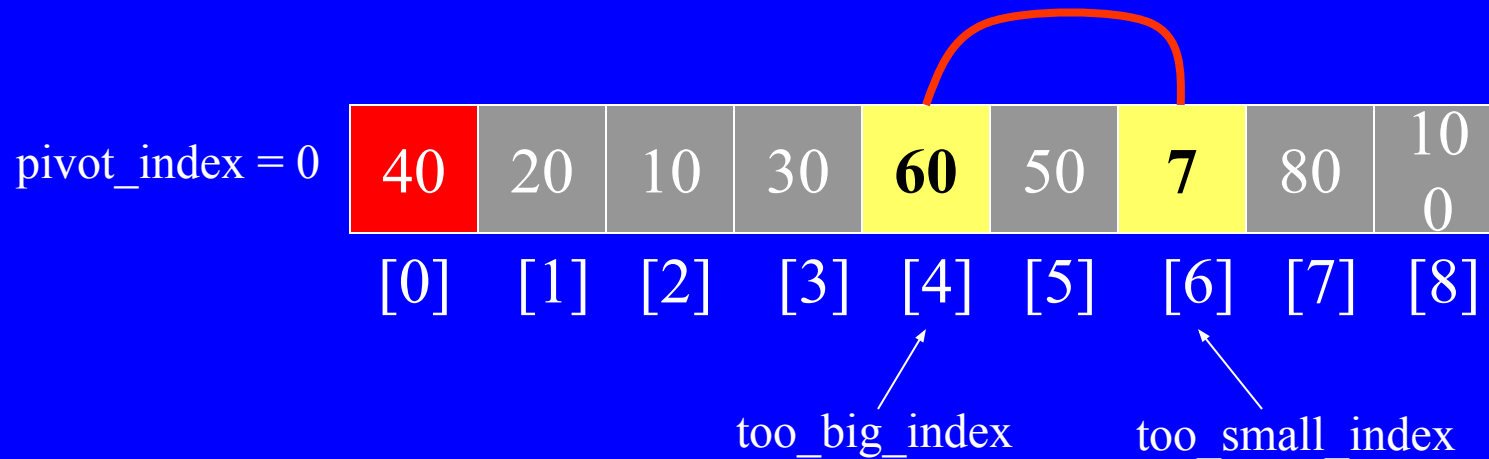
$\text{pivot_index} = 0$

40	20	10	30	60	50	7	80	100
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
				\nearrow too_big_index			\nearrow too_small_index	

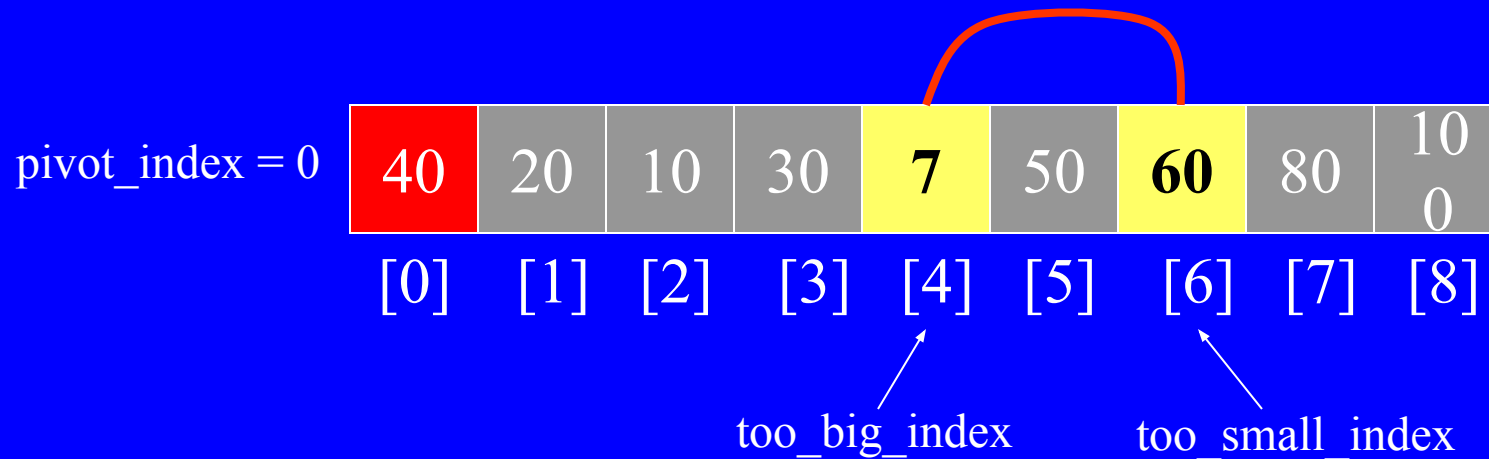
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
- 2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



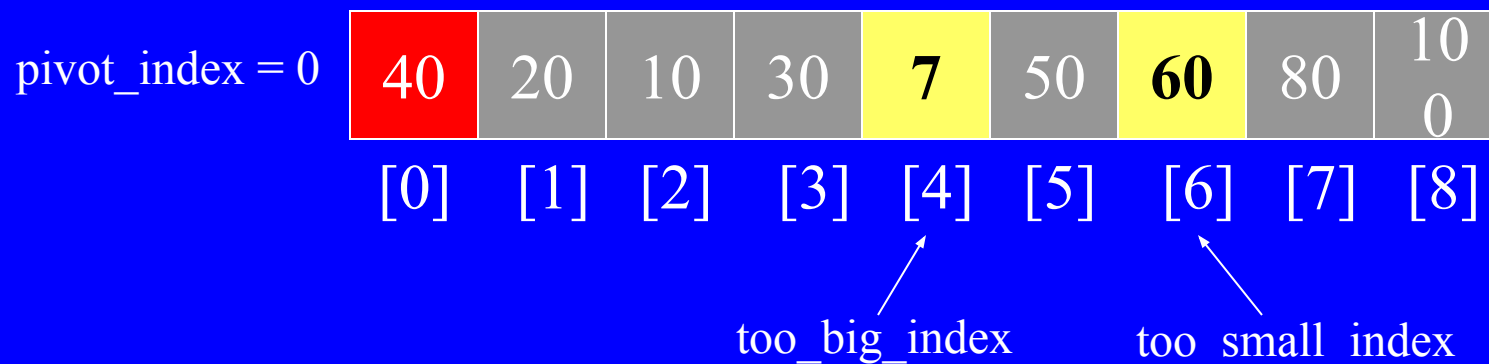
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
- 3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



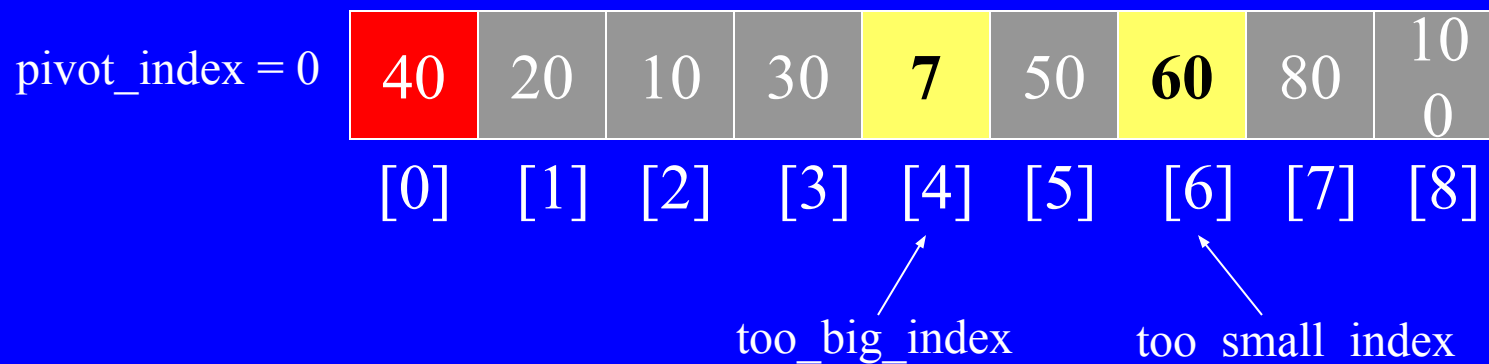
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
- 3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



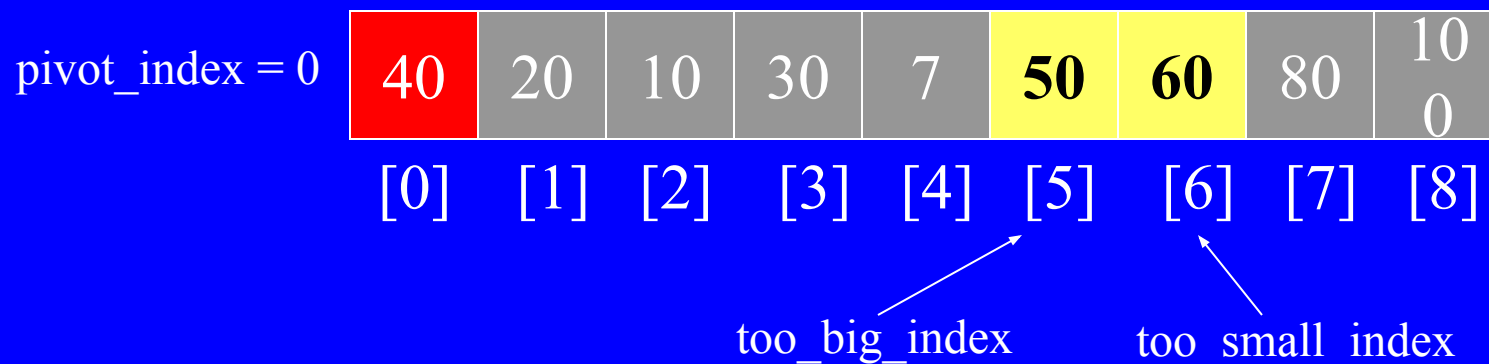
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
- 4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



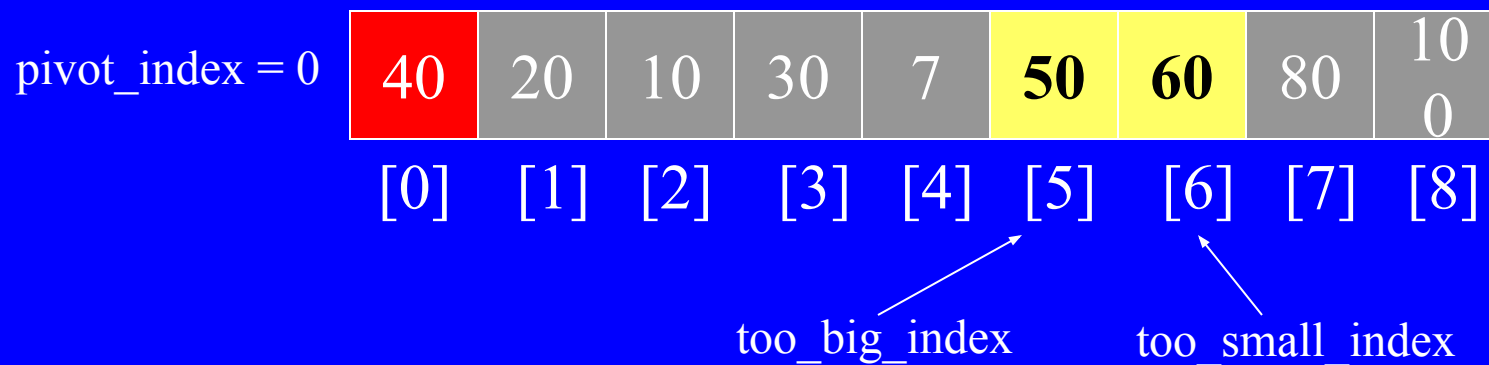
- 1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



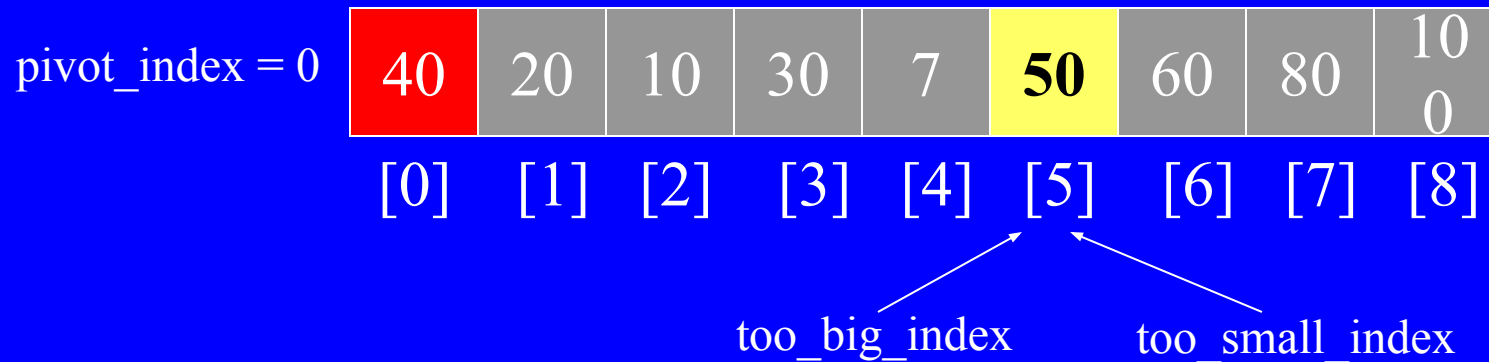
- 1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



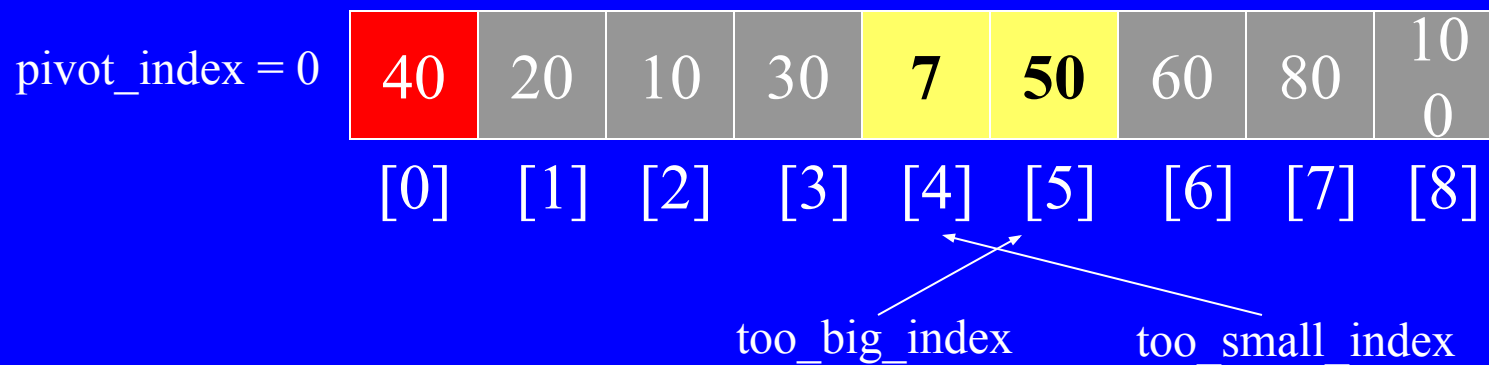
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
- 2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



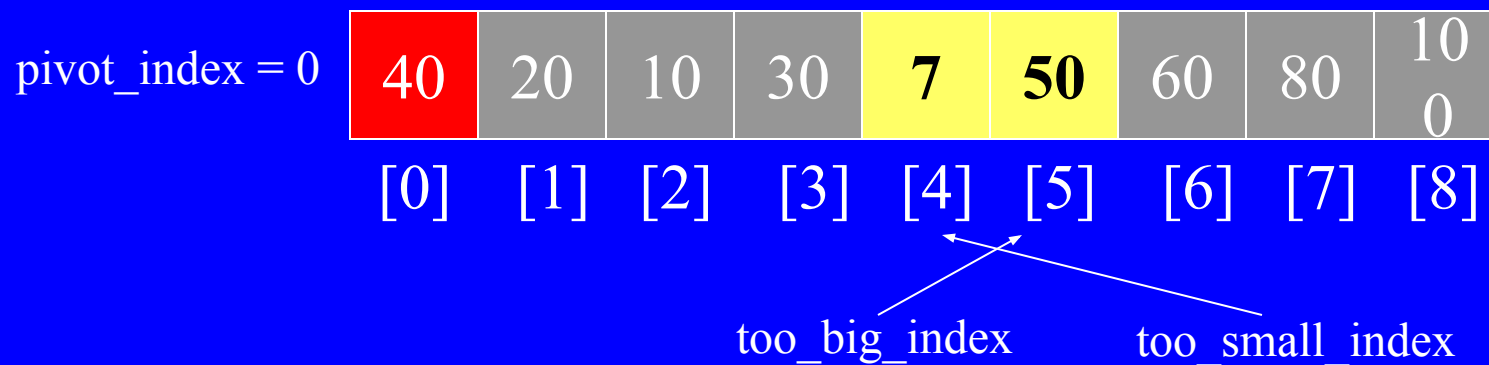
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 $++\text{too_big_index}$
- 2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 $--\text{too_small_index}$
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



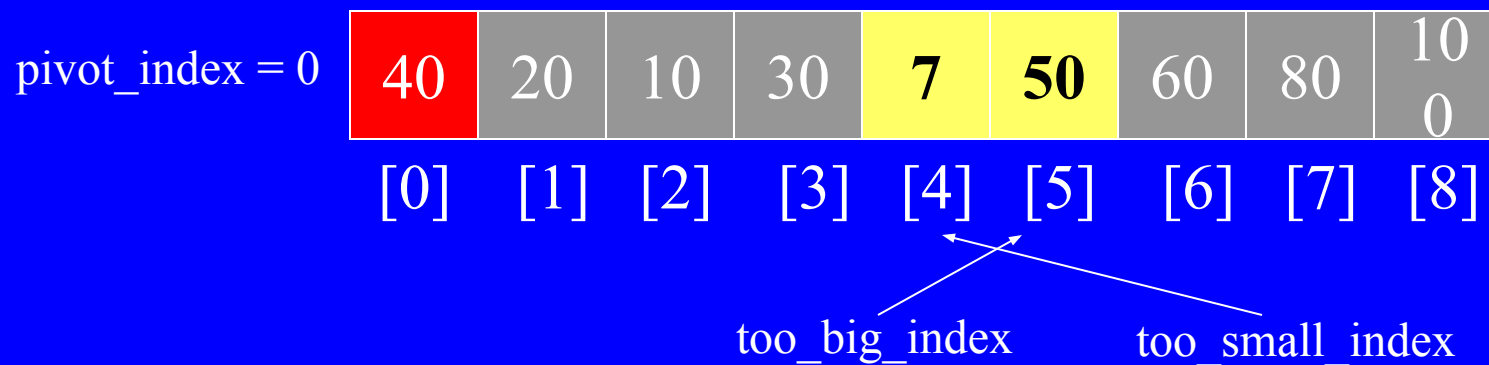
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
- 2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



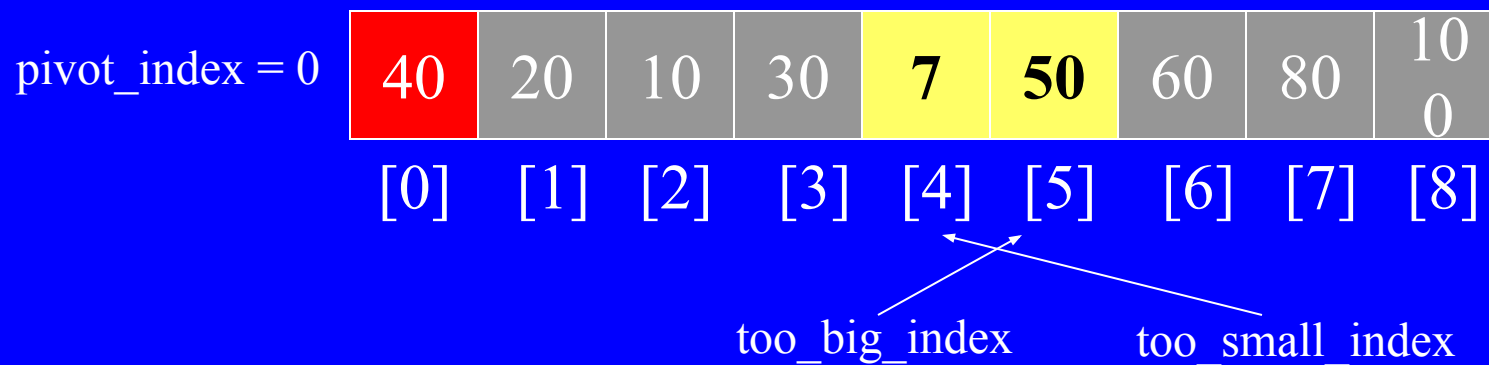
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
- 3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
- 4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
- 5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
- 5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$

pivot_index = 4

7	20	10	30	40	50	60	80	100
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]

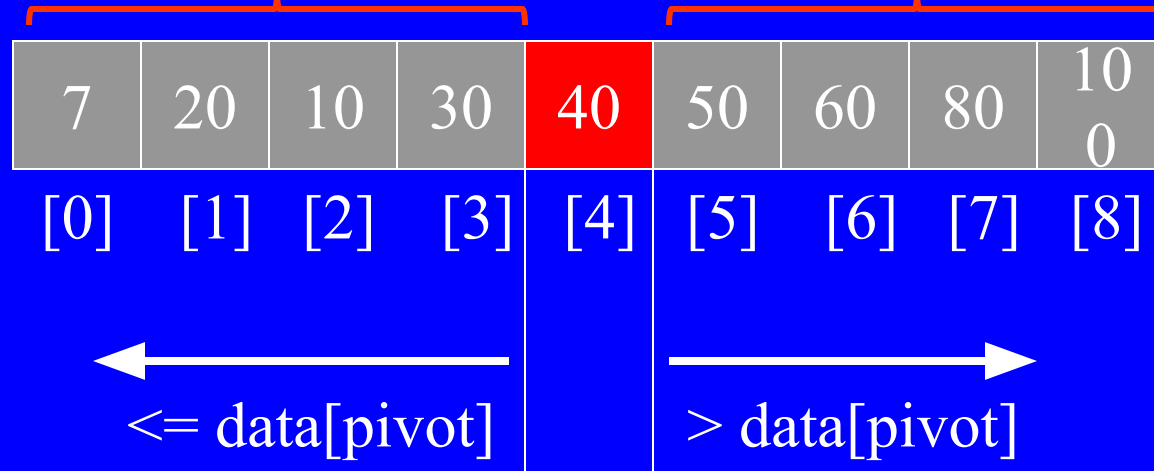
too_big_index

too_small_index

Partition Result

7	20	10	30	40	50	60	80	100
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
← ≤ data[pivot]					→ > data[pivot]			

Recursion: Quicksort Sub-arrays



Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- What is best case running time?

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- What is best case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays of size $n/2$
 2. Quicksort each sub-array

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- What is best case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays of size $n/2$
 2. Quicksort each sub-array
 - Depth of recursion tree?

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- What is best case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays of size $n/2$
 2. Quicksort each sub-array
 - Depth of recursion tree? $O(\log_2 n)$

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- What is best case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays of size $n/2$
 2. Quicksort each sub-array
 - Depth of recursion tree? $O(\log_2 n)$
 - Number of accesses in partition?

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- What is best case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays of size $n/2$
 2. Quicksort each sub-array
 - Depth of recursion tree? $O(\log_2 n)$
 - Number of accesses in partition? $O(n)$

Quicksort Analysis

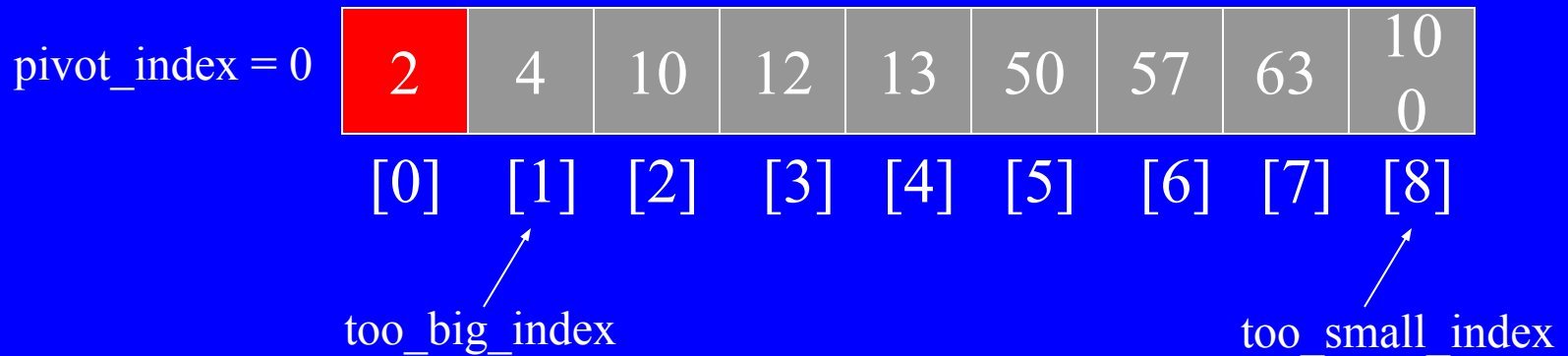
- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$

Quicksort Analysis

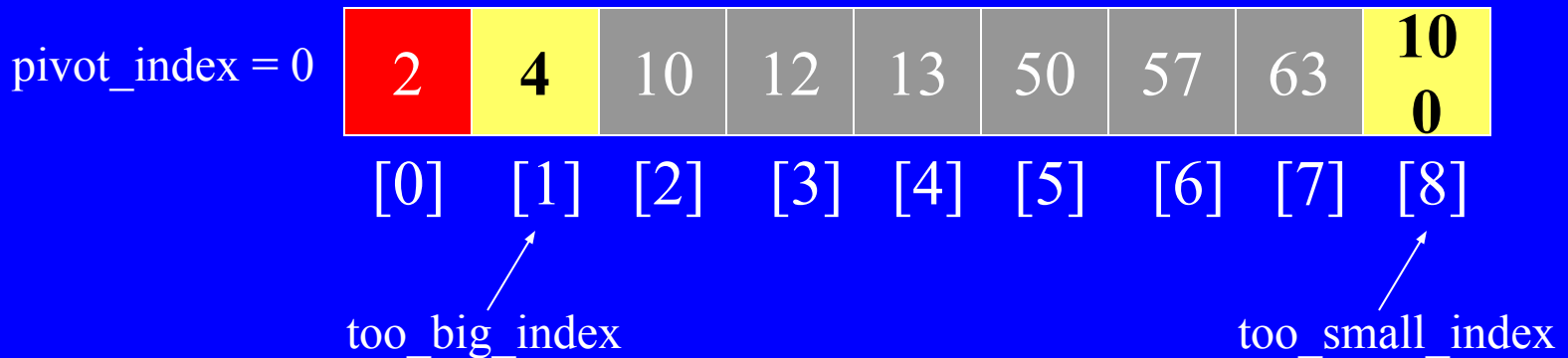
- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time?

Quicksort: Worst Case

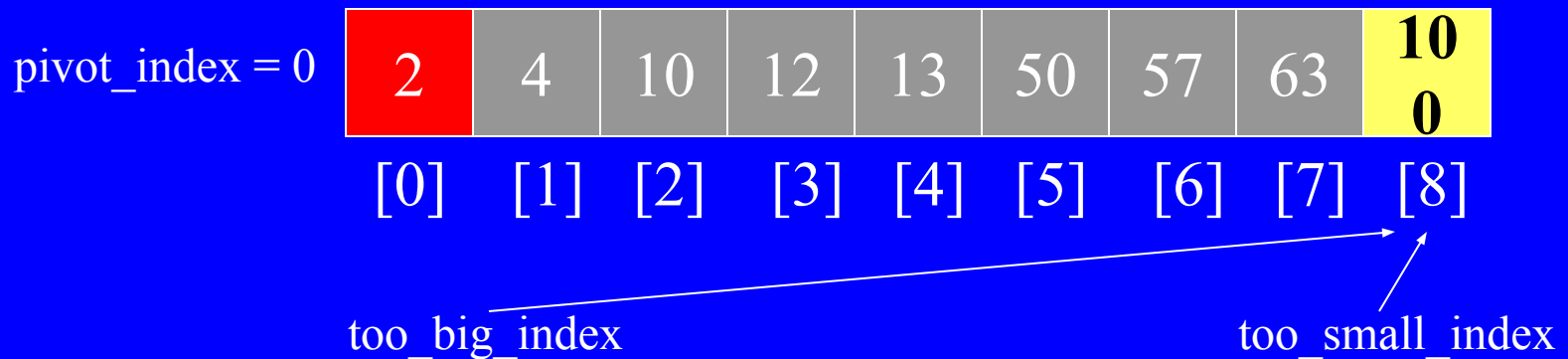
- Assume first element is chosen as pivot.
- Assume we get array that is already in order:



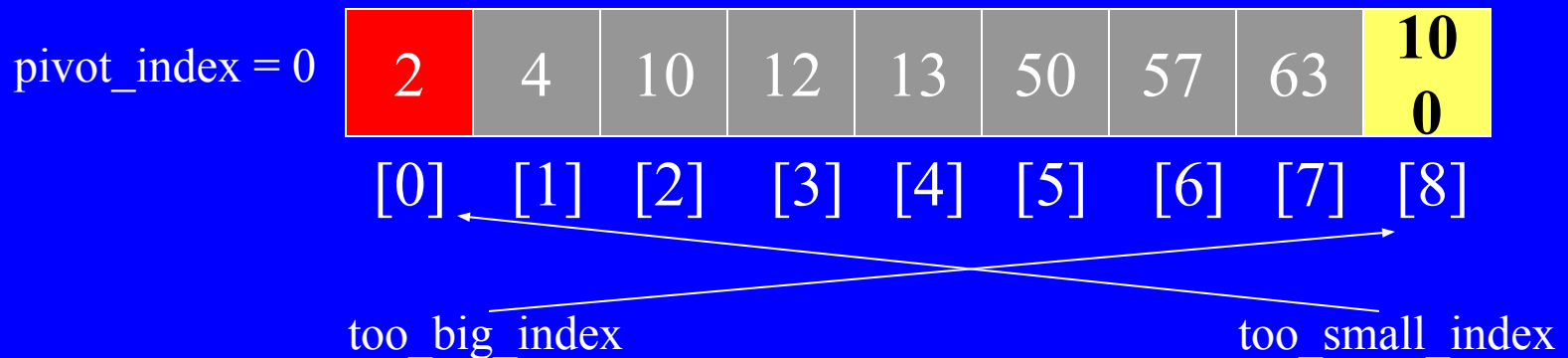
- 1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



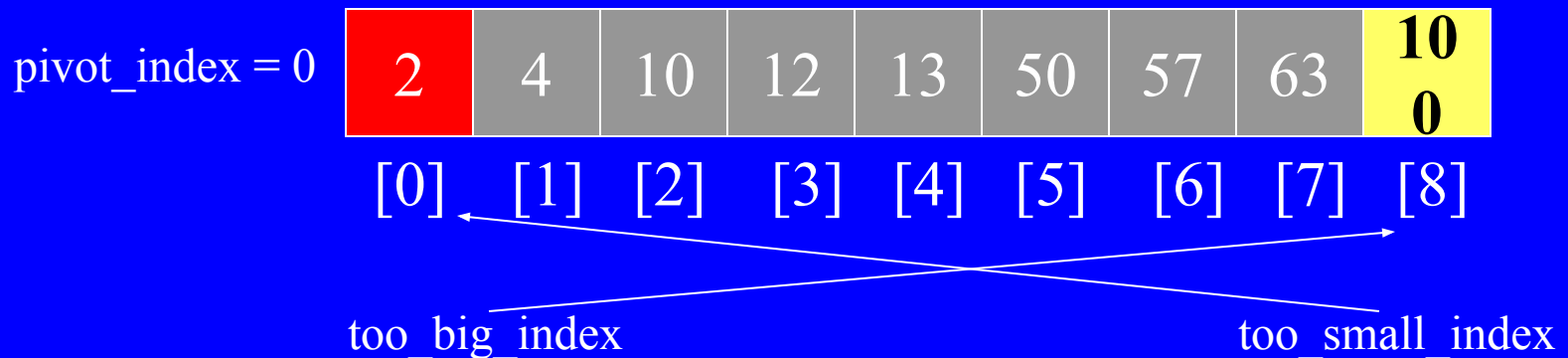
- 1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



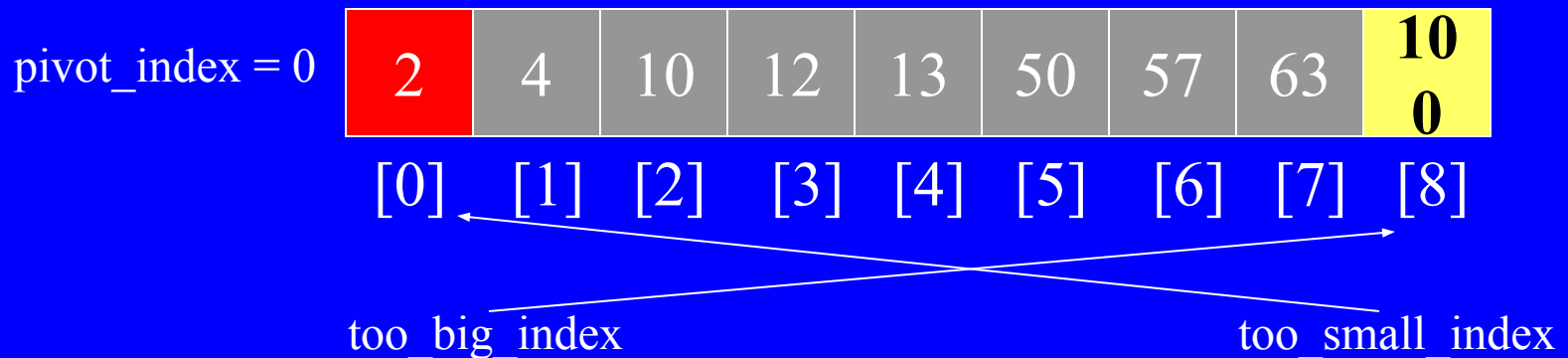
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
- 2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



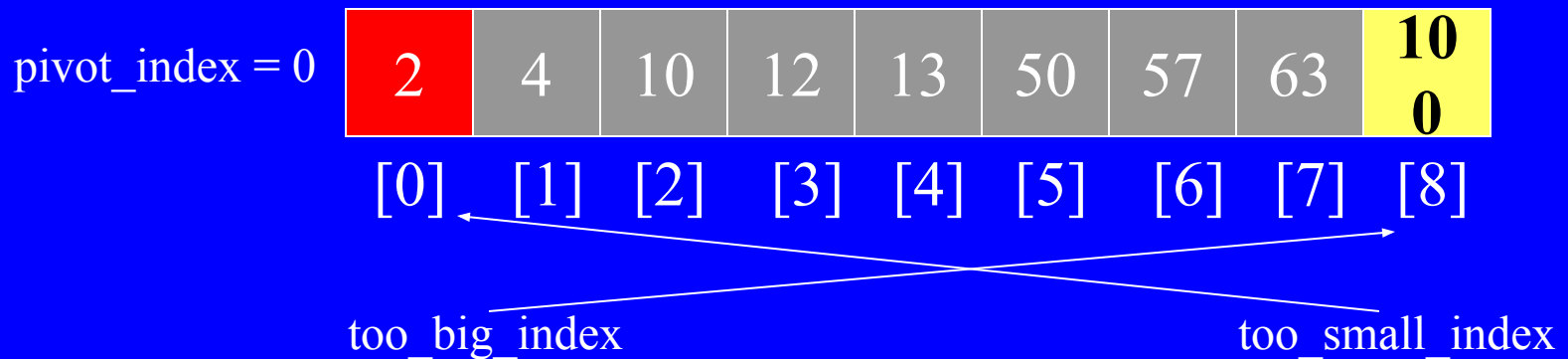
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
- 3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



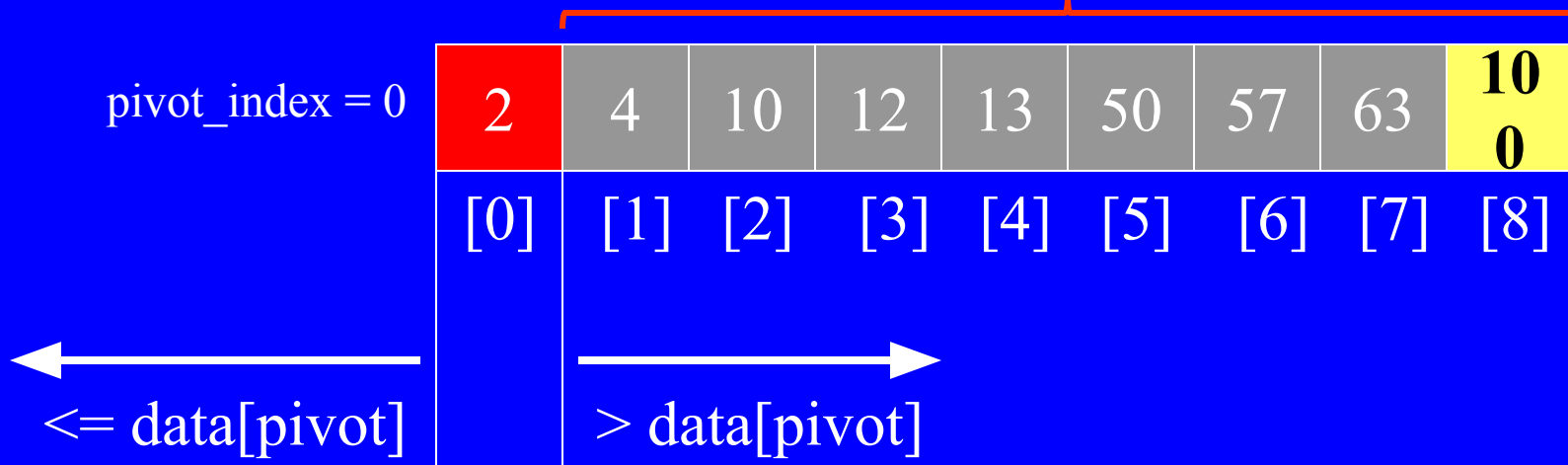
1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
- 4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
- 5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



1. While $\text{data}[\text{too_big_index}] \leq \text{data}[\text{pivot}]$
 ++too_big_index
2. While $\text{data}[\text{too_small_index}] > \text{data}[\text{pivot}]$
 --too_small_index
3. If $\text{too_big_index} < \text{too_small_index}$
 swap $\text{data}[\text{too_big_index}]$ and $\text{data}[\text{too_small_index}]$
4. While $\text{too_small_index} > \text{too_big_index}$, go to 1.
- 5. Swap $\text{data}[\text{too_small_index}]$ and $\text{data}[\text{pivot_index}]$



Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays:
 - one sub-array of size 0
 - the other sub-array of size $n-1$
 2. Quicksort each sub-array
 - Depth of recursion tree?

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays:
 - one sub-array of size 0
 - the other sub-array of size $n-1$
 2. Quicksort each sub-array
 - Depth of recursion tree? $O(n)$

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays:
 - one sub-array of size 0
 - the other sub-array of size $n-1$
 2. Quicksort each sub-array
 - Depth of recursion tree? $O(n)$
 - Number of accesses per partition?

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time?
 - Recursion:
 1. Partition splits array in two sub-arrays:
 - one sub-array of size 0
 - the other sub-array of size $n-1$
 2. Quicksort each sub-array
 - Depth of recursion tree? $O(n)$
 - Number of accesses per partition? $O(n)$

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time: $O(n^2)$!!!

Quicksort Analysis

- Assume that keys are random, uniformly distributed.
- Best case running time: $O(n \log_2 n)$
- Worst case running time: $O(n^2)$!!!
- What can we do to avoid worst case?

Improved Pivot Selection

Pick median value of three elements from data array:
data[0], data[n/2], and data[n-1].

Use this median value as pivot.