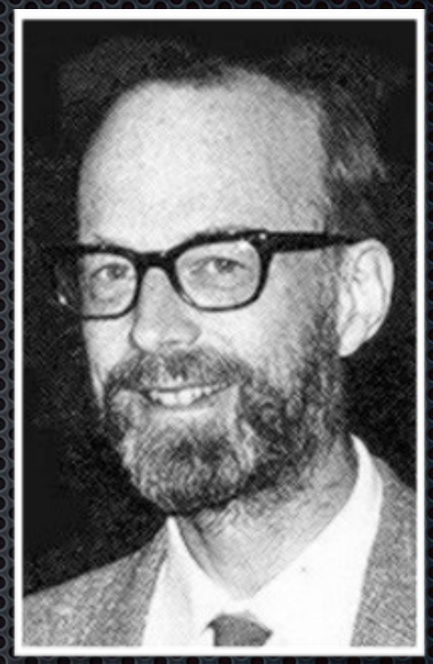
Quicksort

Baochun Li
Department of Electrical and Computer Engineering
University of Toronto

Quicksort



- Proposed by C.A.R. Hoare in 1960
- One of top 10 algorithms of 20th century in science and engineering

Quicksort

- Proposed by C.A.R. Hoare in 1960
- One of top 10 algorithms of 20th century in science and engineering

Selection sort		Quicksort	
10^3	10^6	10^3	10^6
2.8 hours	317 years	0.6 sec	12 min

Quicksort

- Proposed by C.A.R. Hoare in 1960
- One of top 10 algorithms of 20th century in science and engineering

Selection sort		Quicksort	
10^3	10^6	10^3	10^6
2.8 hours	317 years	0.6 sec	12 min

Divide-and-conquer

Divide-and-conquer

- Divide the problem into a number of subproblems that are smaller instances of the same problem

Divide-and-conquer

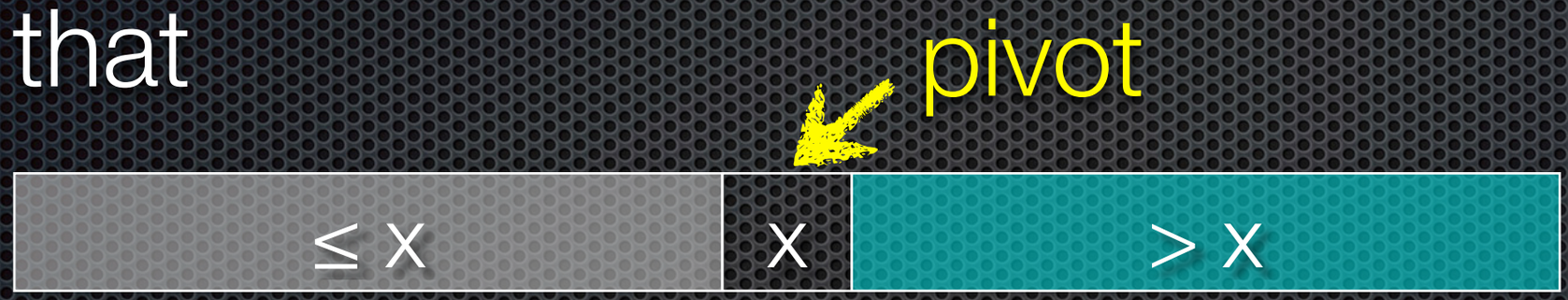
- Divide the problem into a number of subproblems that are smaller instances of the same problem
- Conquer the subproblems

Divide-and-conquer

- Divide the problem into a number of subproblems that are smaller instances of the same problem
- Conquer the subproblems
- Combine the solutions to the subproblems into the solution for the original problem

Basic idea

- (Divide) Partition the array into two subarrays, such that

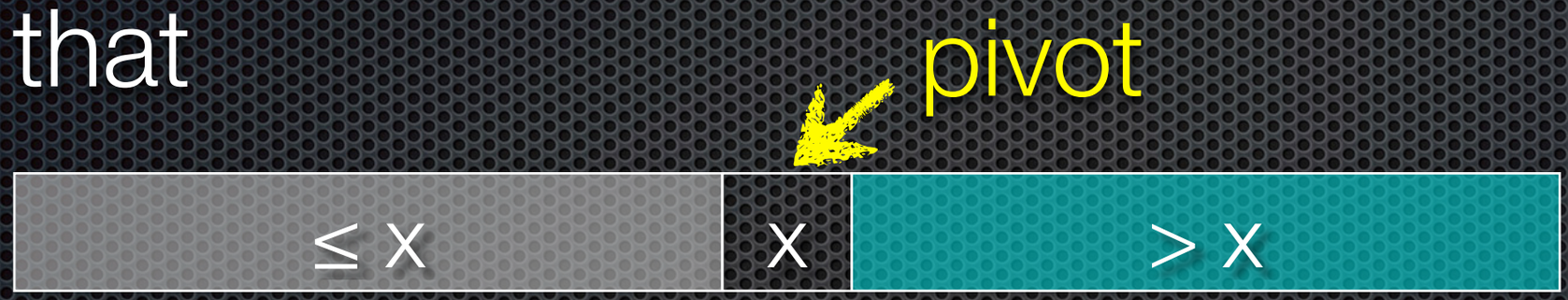


- (Conquer) Sort the two subarrays recursively

Input	61	46	12	63	52	91	27	55	74	14	71	37	87
Partition	55	46	12	57	52	14	27	61	74	91	71	63	87
Sort left	12	14	27	37	46	52	55	61	74	91	71	63	87
Sort right	12	14	27	37	46	52	55	61	63	71	74	87	91
Result	12	14	27	37	46	52	55	61	63	71	74	87	91

Basic idea

- (Divide) Partition the array into two subarrays, such that

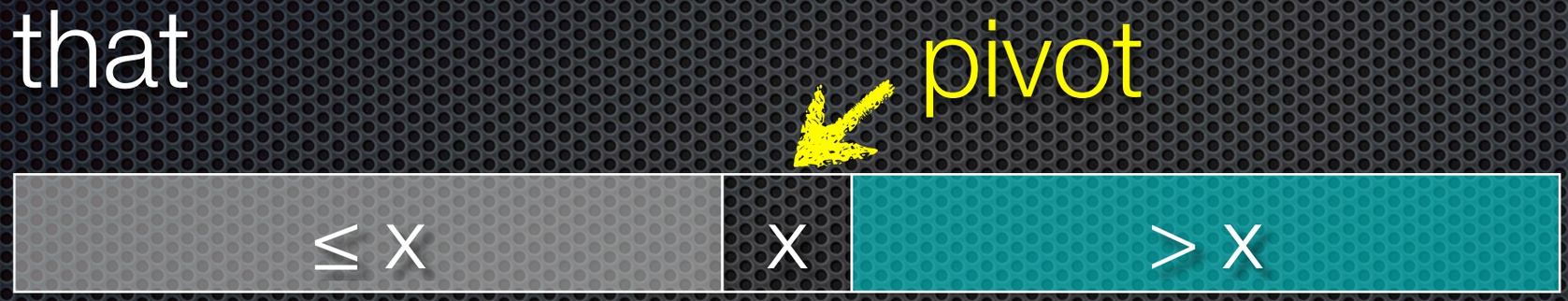


- (Conquer) Sort the two subarrays recursively

Input	61	46	12	63	52	91	27	55	74	14	71	37	87
Partition	55	46	12	57	52	14	27	61	74	91	71	63	87
Sort left	12	14	27	37	46	52	55	61	74	91	71	63	87
Sort right	12	14	27	37	46	52	55	61	63	71	74	87	91
Result	12	14	27	37	46	52	55	61	63	71	74	87	91

Basic idea

- (Divide) Partition the array into two subarrays, such that

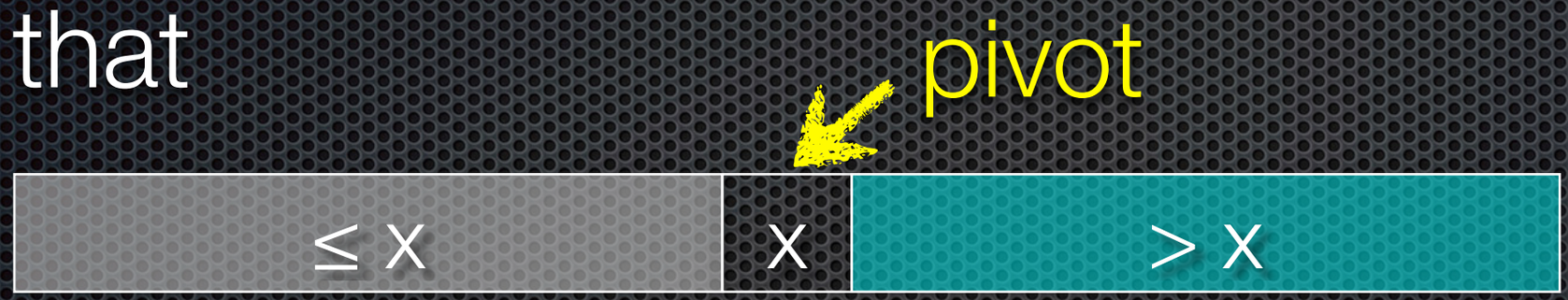


- (Conquer) Sort the two subarrays recursively

Input	61	46	12	63	52	91	27	55	74	14	71	37	87
Partition	55	46	12	57	52	14	27	61	74	91	71	63	87
Sort left	12	14	27	37	46	52	55	61	74	91	71	63	87
Sort right	12	14	27	37	46	52	55	61	63	71	74	87	91
Result	12	14	27	37	46	52	55	61	63	71	74	87	91

Basic idea

- (Divide) Partition the array into two subarrays, such that



- (Conquer) Sort the two subarrays recursively

Input	61	46	12	63	52	91	27	55	74	14	71	37	87
Partition	55	46	12	57	52	14	27	61	74	91	71	63	87
Sort left	12	14	27	37	46	52	55	61	74	91	71	63	87
Sort right	12	14	27	37	46	52	55	61	63	71	74	87	91
Result	12	14	27	37	46	52	55	61	63	71	74	87	91

Partitioning the array

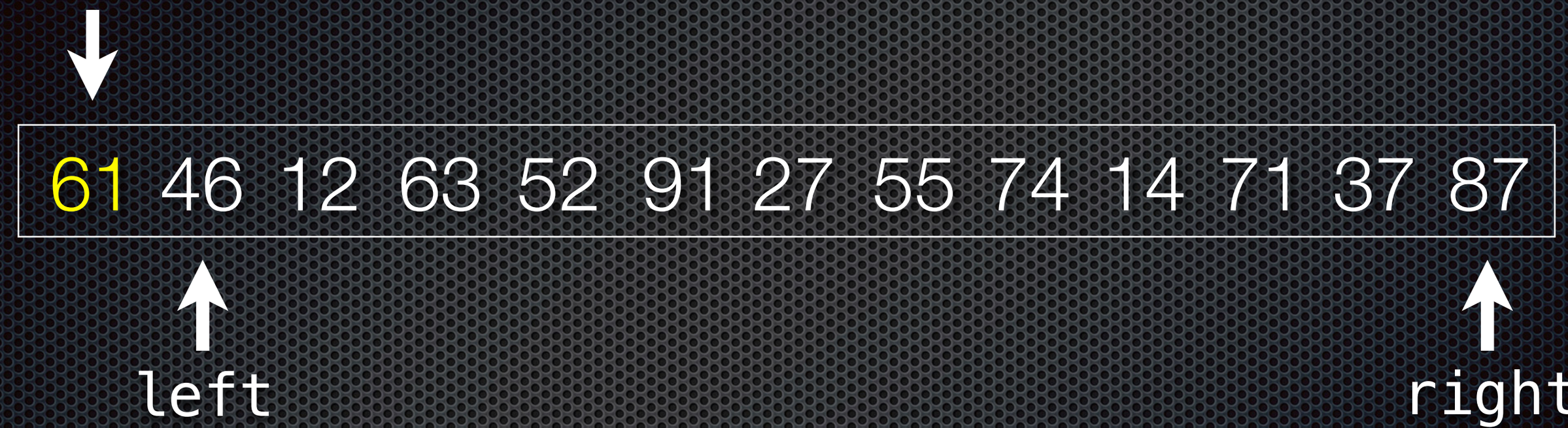
61 46 12 63 52 91 27 55 74 14 71 37 87

Partitioning the array

pivot

61	46	12	63	52	91	27	55	74	14	71	37	87

Partitioning the array



Partitioning the array

`pivot`

61	46	12	63	52	91	27	55	74	14	71	37	87

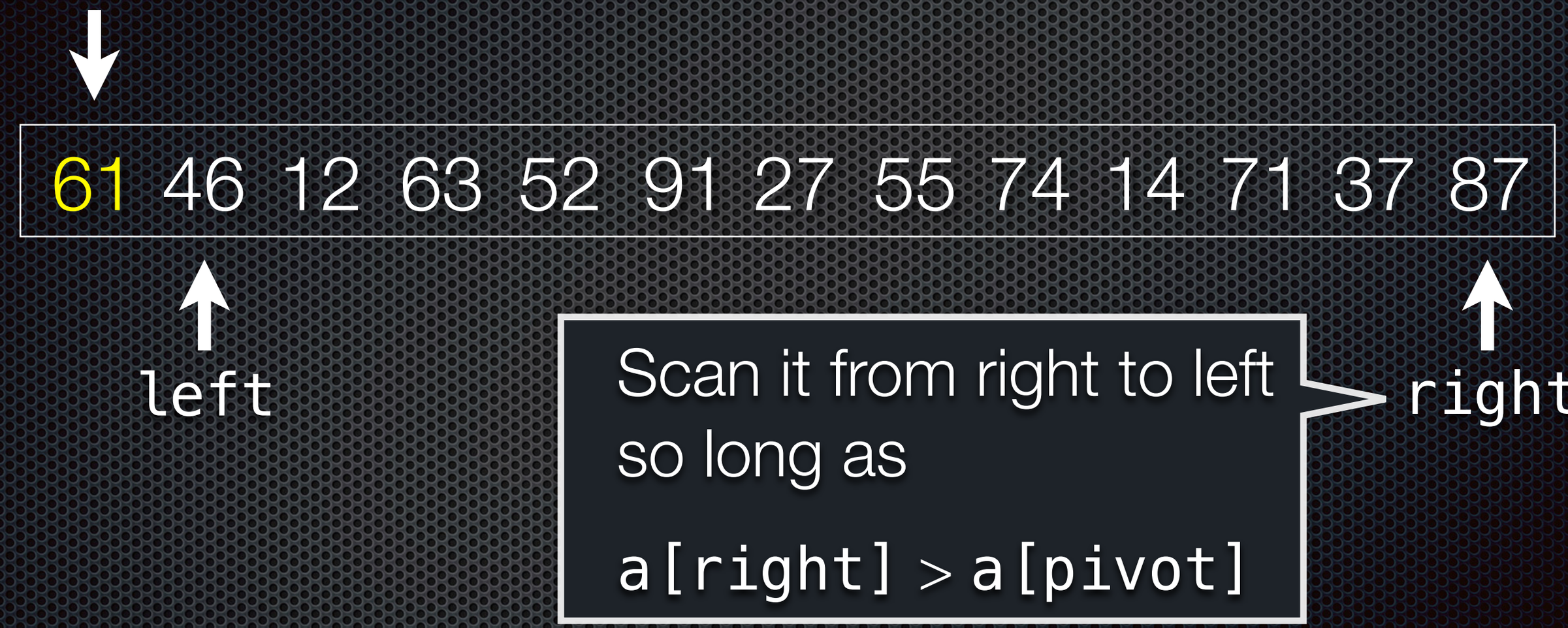
`left`

Scan it from left to right so long as

`a[left]` < `a[pivot]`

`right`

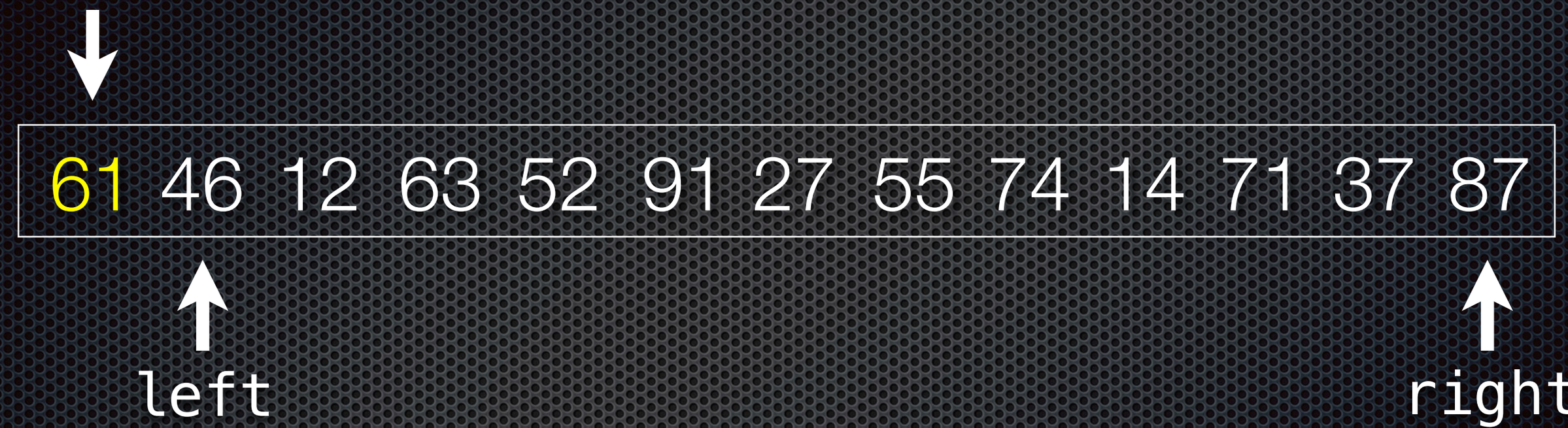
Partitioning the array



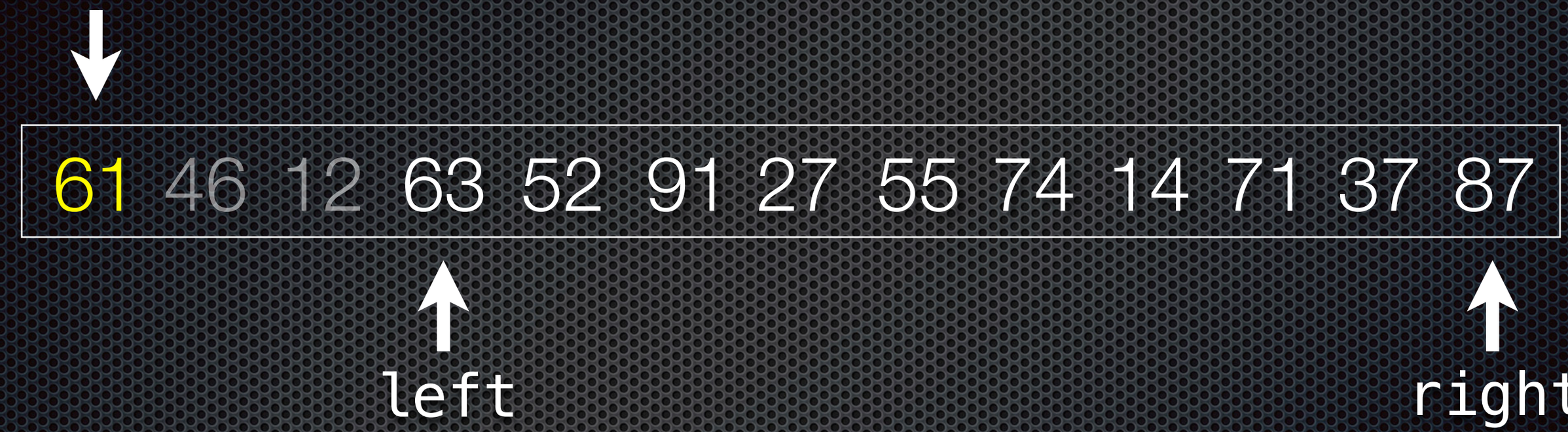
Partitioning the array

Exchange `a[left]` with `a[right]`
if `left` and `right` are not crossed

Partitioning the array



Partitioning the array

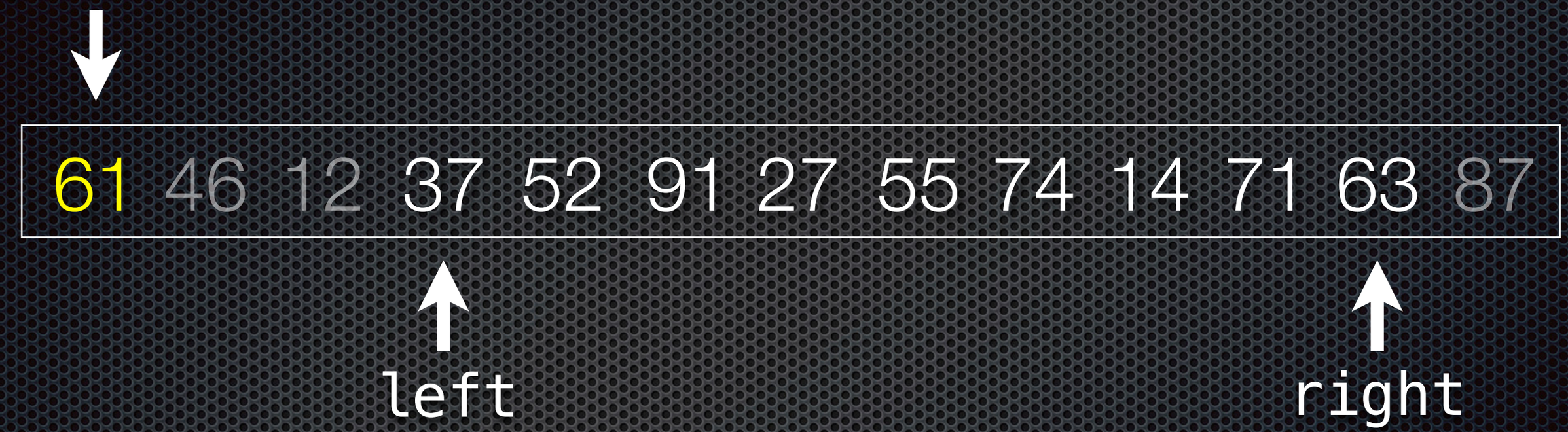


Stop `left` scan as `a[left]` ≥ `a[pivot]`

Partitioning the array

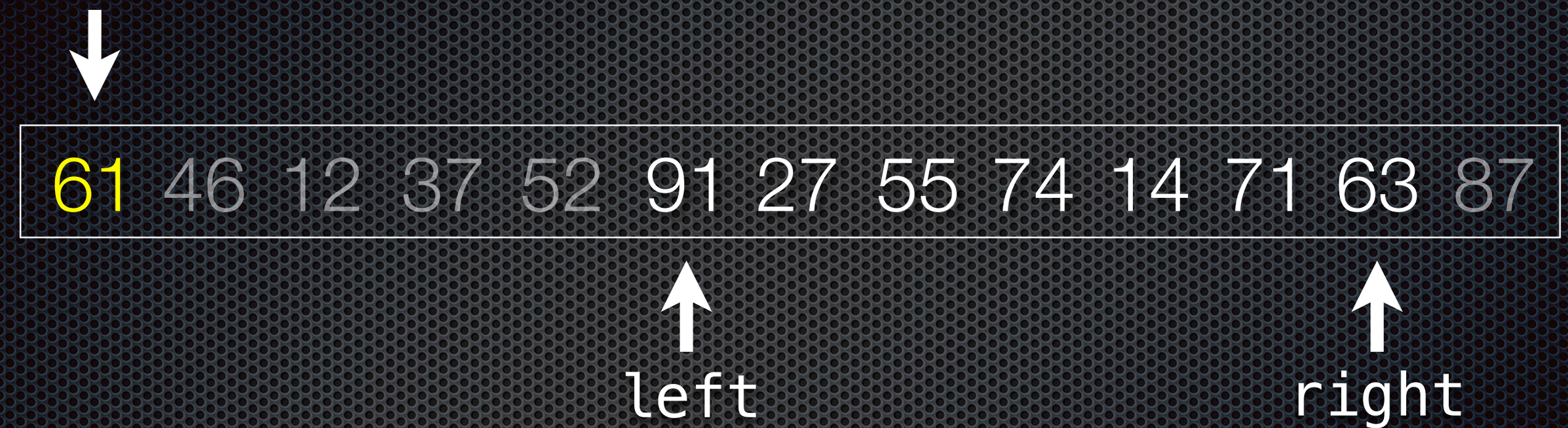
Stop `right` scan and exchange `a[left]` with `a[right]`

Partitioning the array



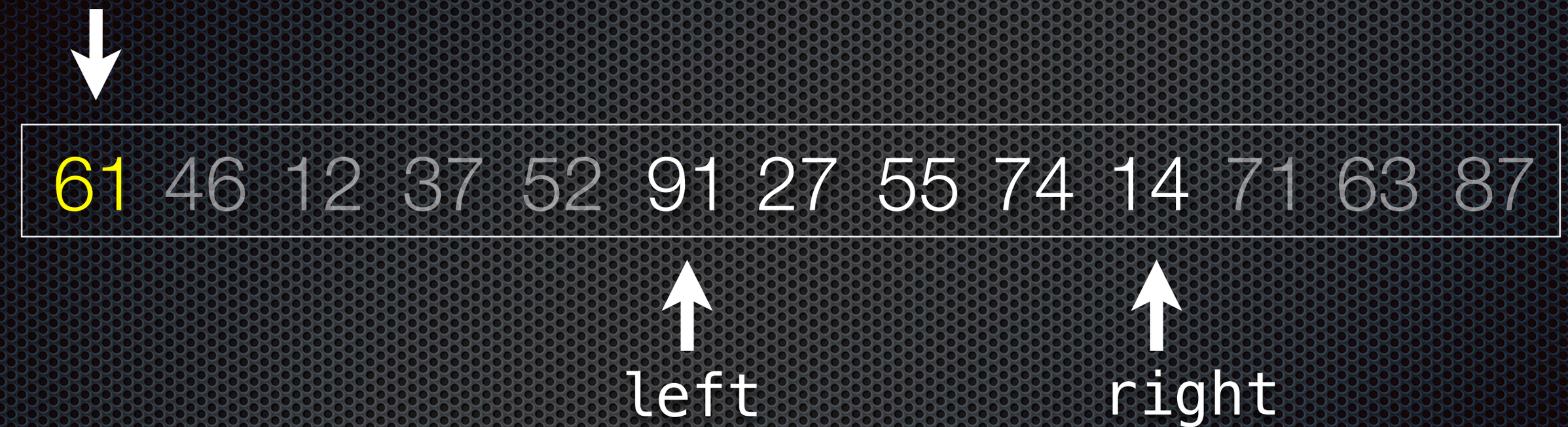
Stop `right` scan and exchange `a[left]` with `a[right]`

Partitioning the array



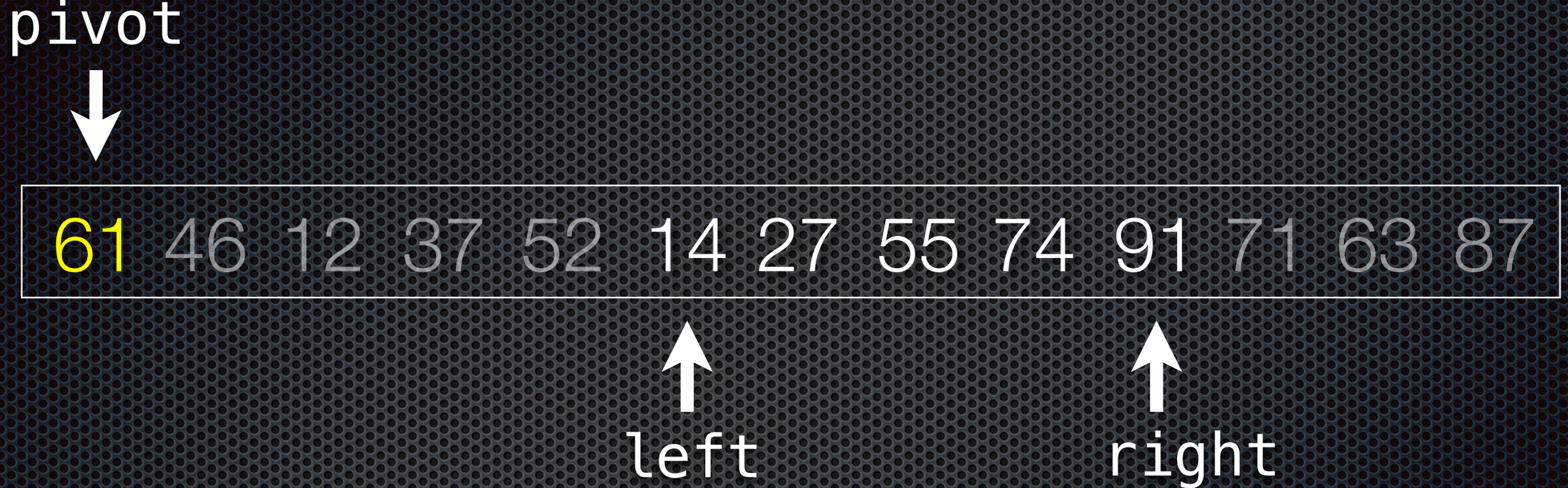
Stop `left` scan as `a[left]` ≥ `a[pivot]`

Partitioning the array



Stop `right` scan and exchange `a[left]` with `a[right]`

Partitioning the array

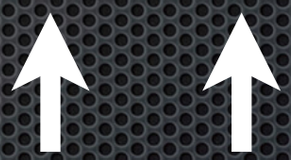


Stop `right` scan and exchange `a[left]` with `a[right]`

Partitioning the array

pivot

61	46	12	37	52	14	27	55	74	91	71	63	87



left right

Stop `left` scan as `a[left]` ≥ `a[pivot]`

Partitioning the array

`pivot`

61	46	12	37	52	14	27	55	74	91	71	63	87

`rightleft`

Stop `right` scan

Partitioning the array

Exchange `a[pivot]` with `a[right]`

Partitioning the array

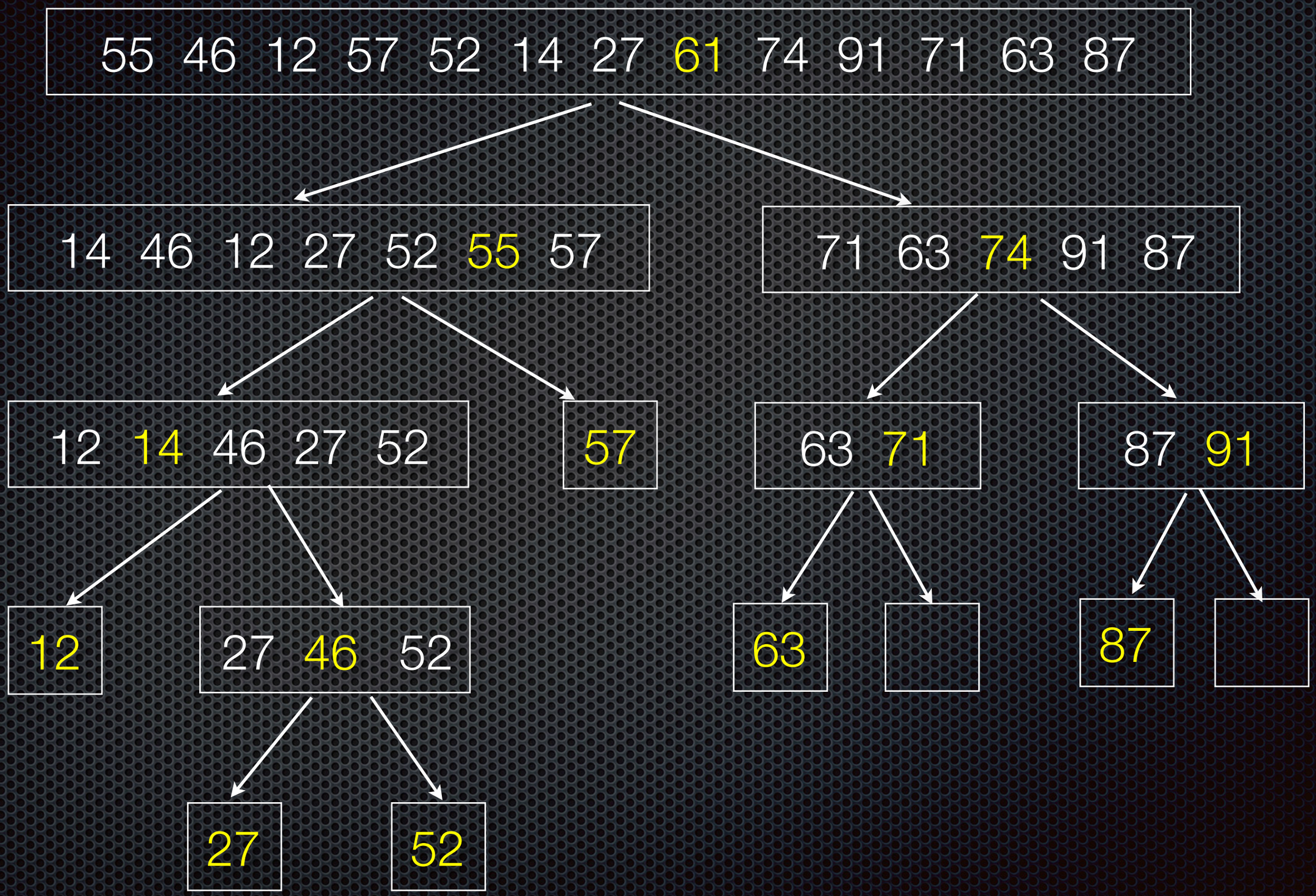
pivot

55	46	12	37	52	14	27	61	74	91	71	63	87

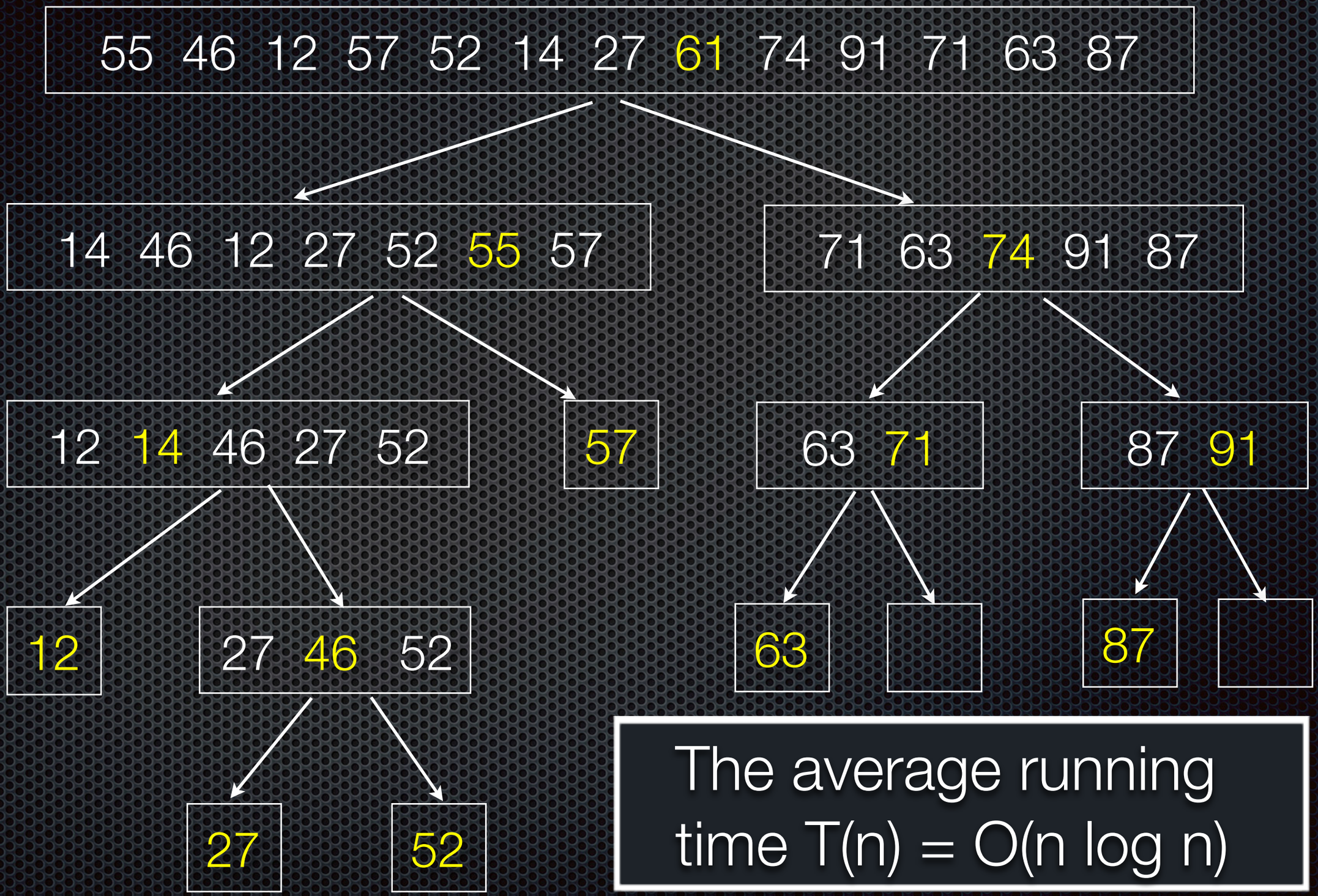
right left

Partitioned!

Recursion tree



Recursion tree

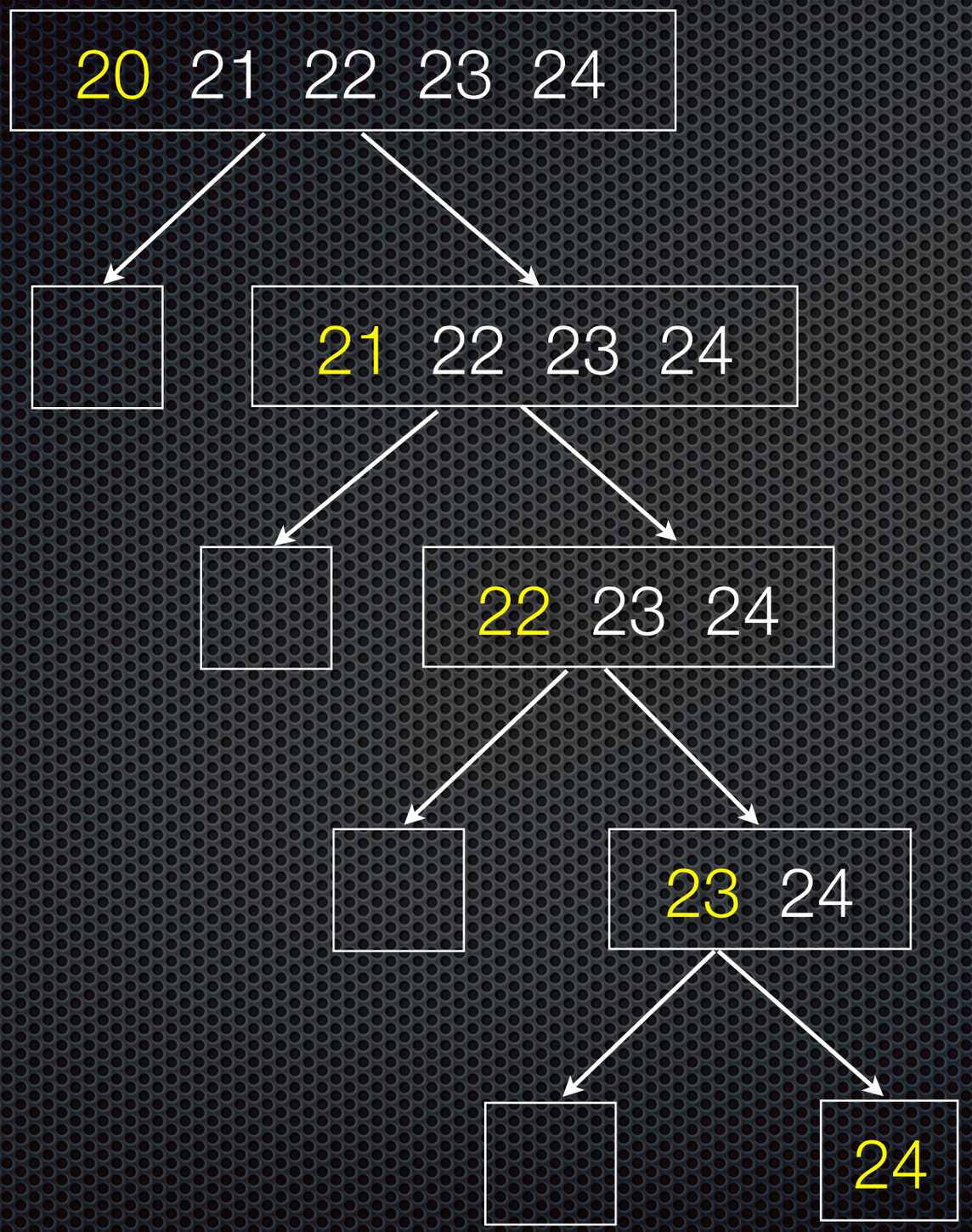


The average running time T(n) = O(n log n)

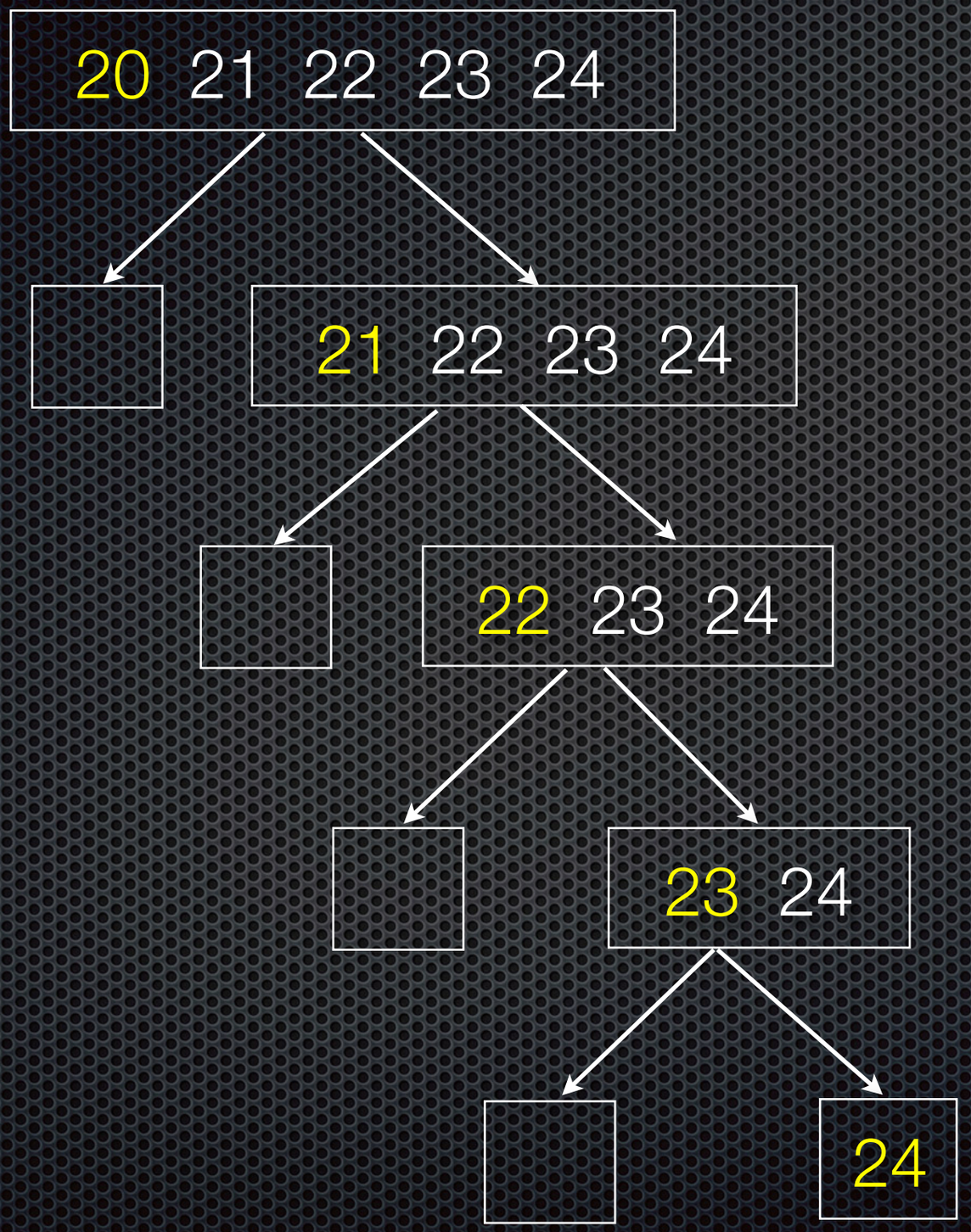
Another example

20 21 22 23 24

Another example



Another example



The worst-case running time $T(n) = O(n^2)$