Quick sort

This sorting algorithm uses the idea of divide and conquer.

It finds the element called pivot which divides the array into two halves in such a way that elements in the left half are smaller than pivot and elements in the right half are greater than pivot.

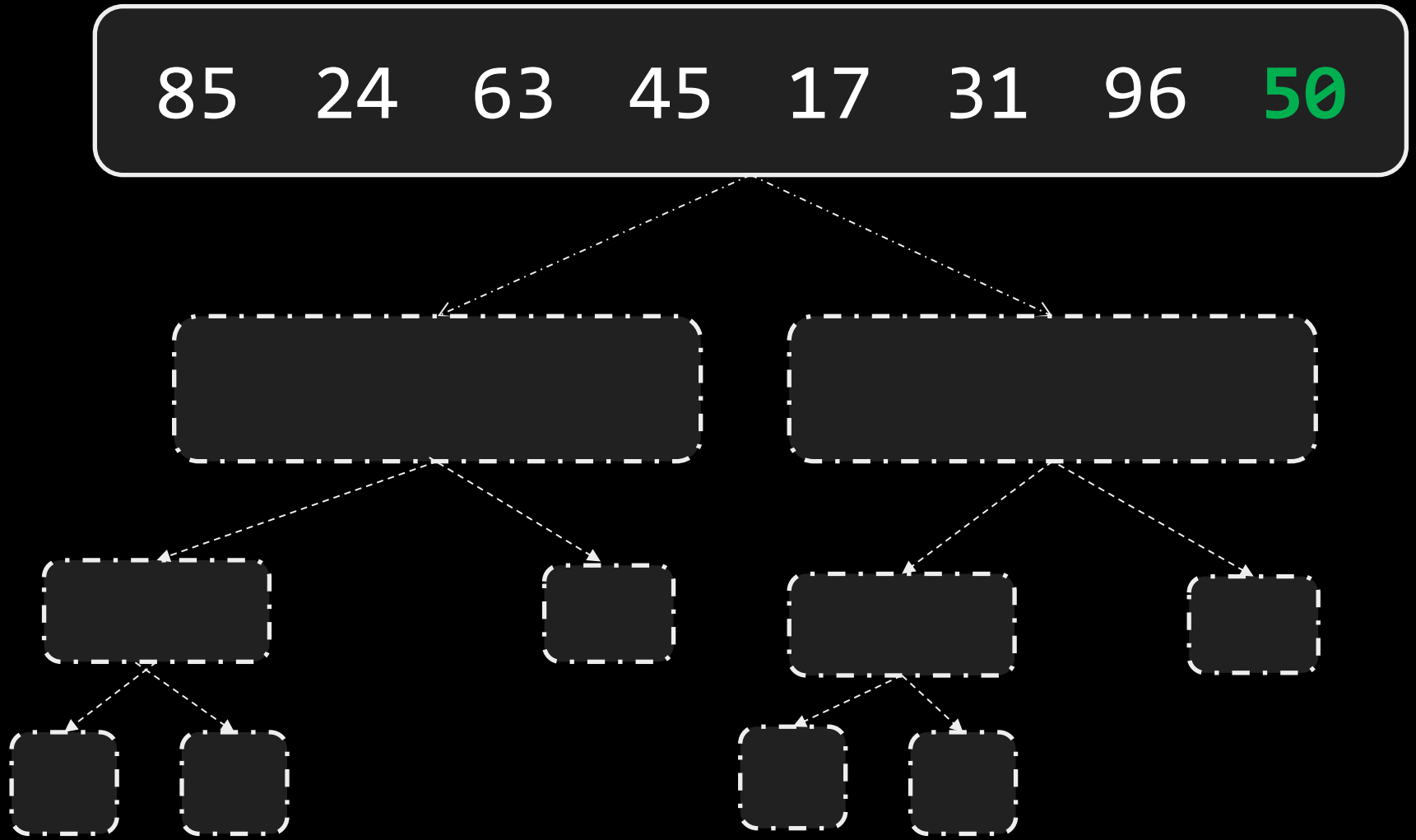
There are many different versions of quicksort that pick pivot in different ways:

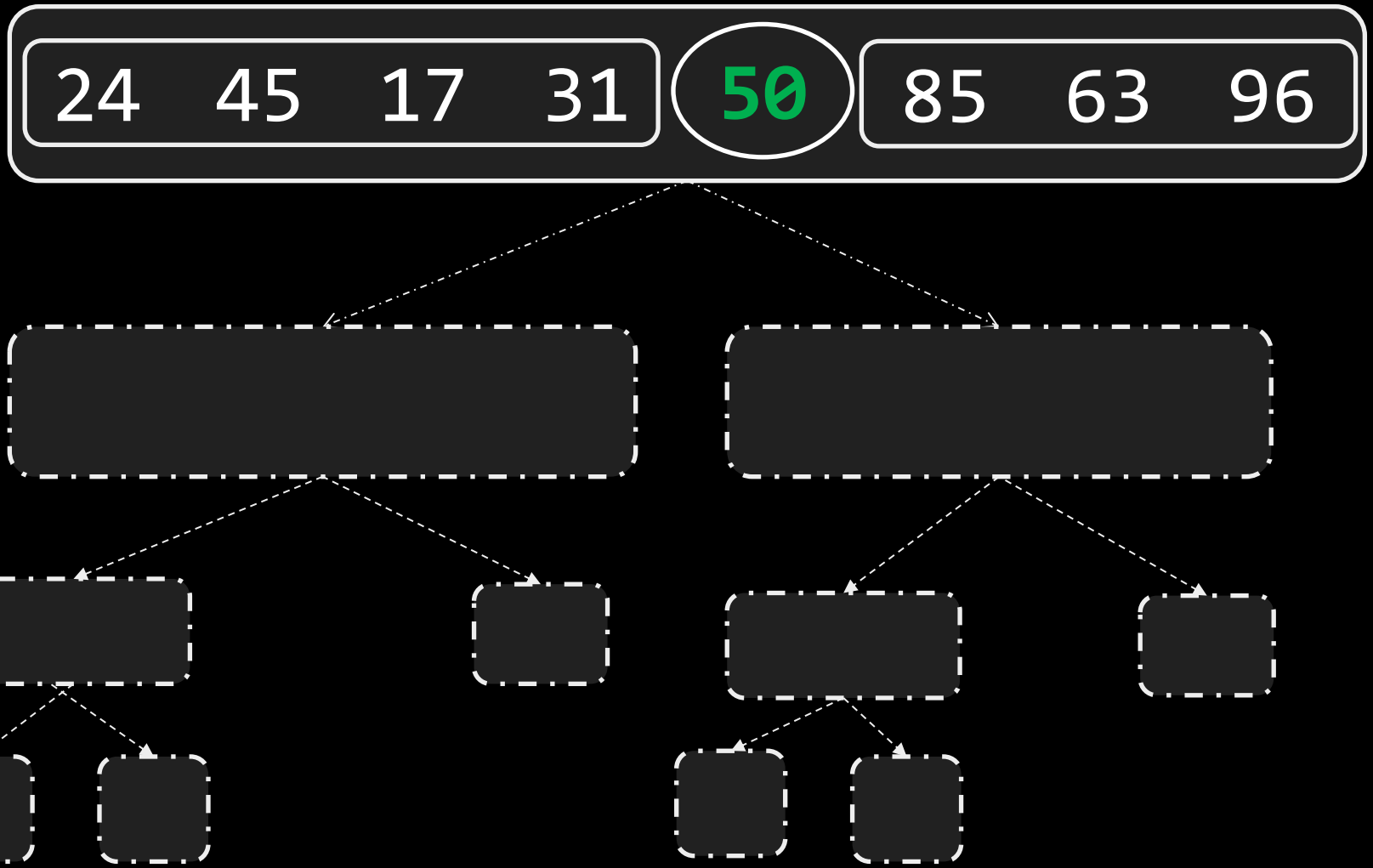
a. Always pick first element as pivot.
b. Always pick last element as pivot (We will use this)
c. Pick a random element as pivot.
d. Pick median as pivot.

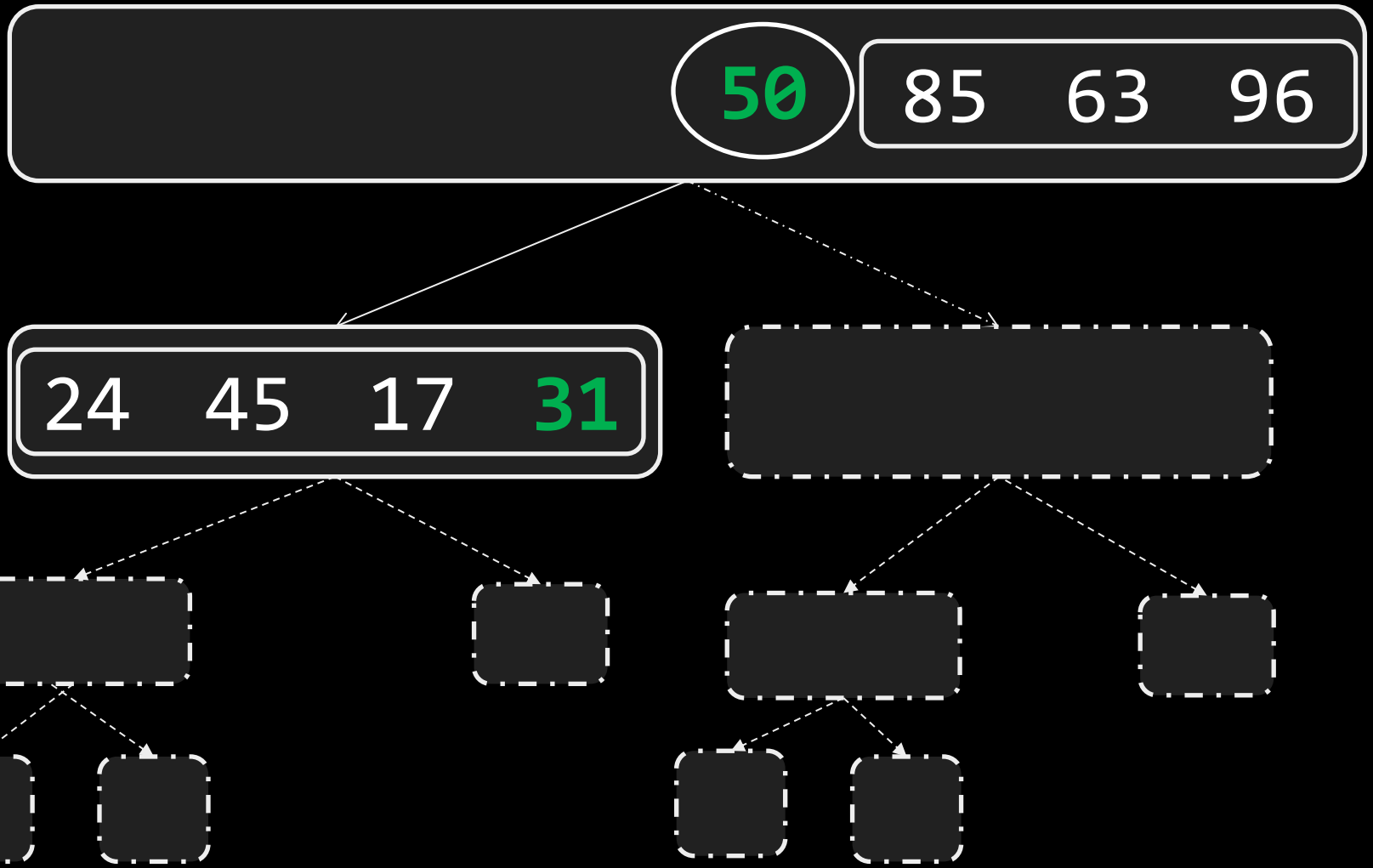
Algorithm

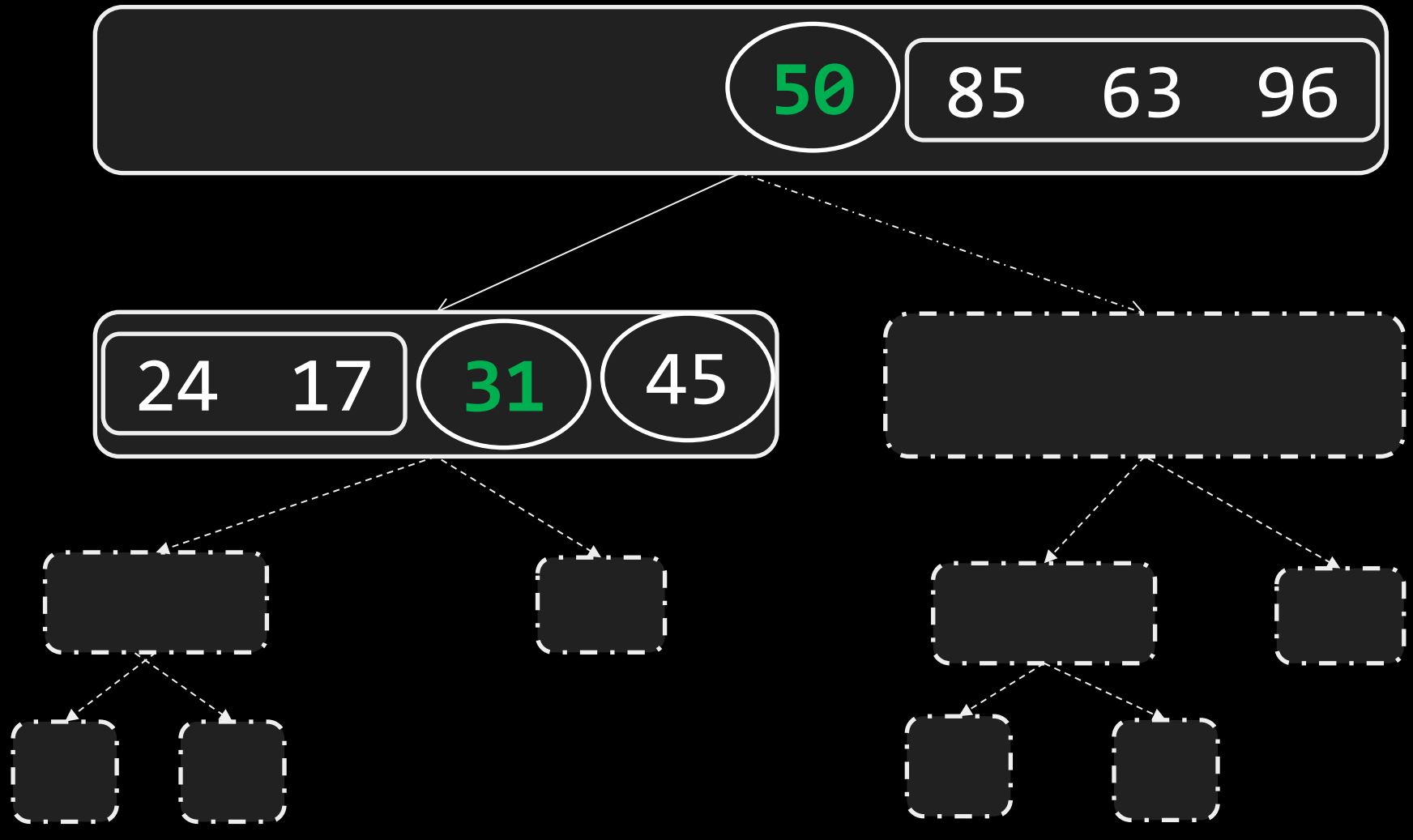
- Divide: If S has at least two elements(nothing needs to be done if S has zero or one element), select a specific element x from S, which is called the pivot. As is common practice, choose the pivot x to be the last element in S. Remove all the elements from S and put them into three sequences:
  - L, storing the elements in S less than x
  - E, storing the elements in S equal to x
  - G, storing the elements in S greater than x
- Conquer: Recursively sort sequences L and G
- Combine: Put back the elements into S in order by first inserting the elements of L, then those of E, and finally those of G.

85 24 63 45 17 31 96 50

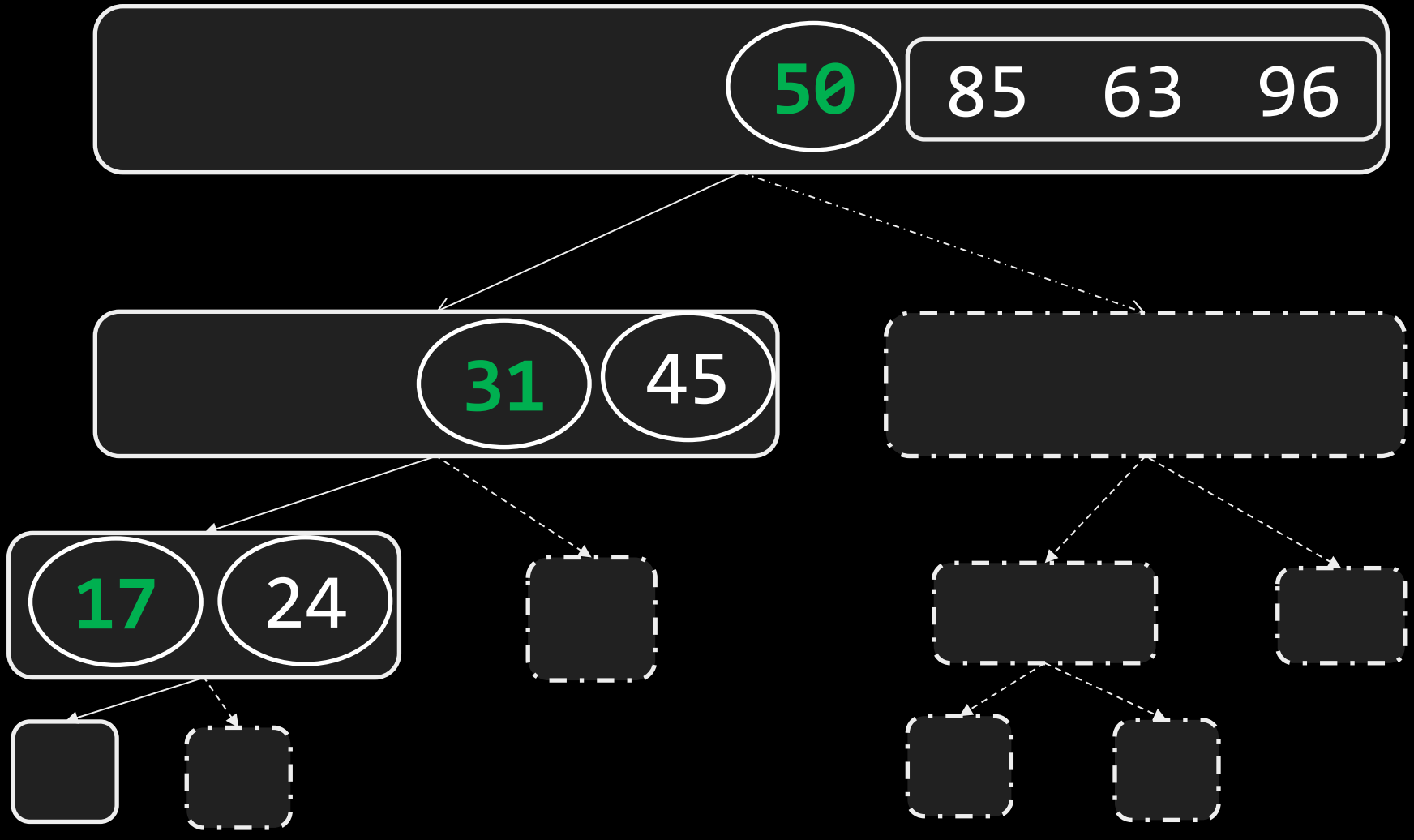

85 24 63 45 17 31 96 50

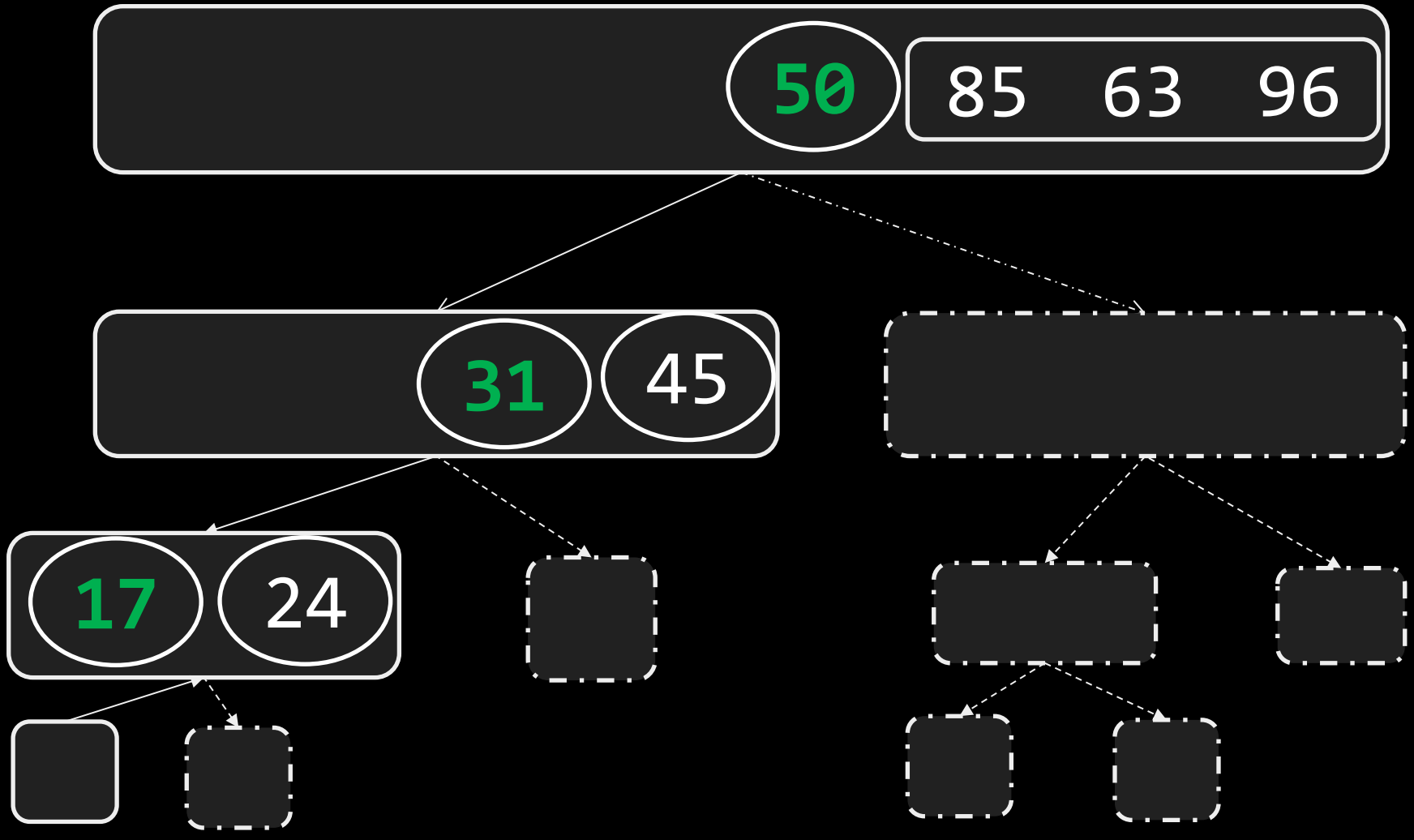

24 45 17 31
50
85 63 96

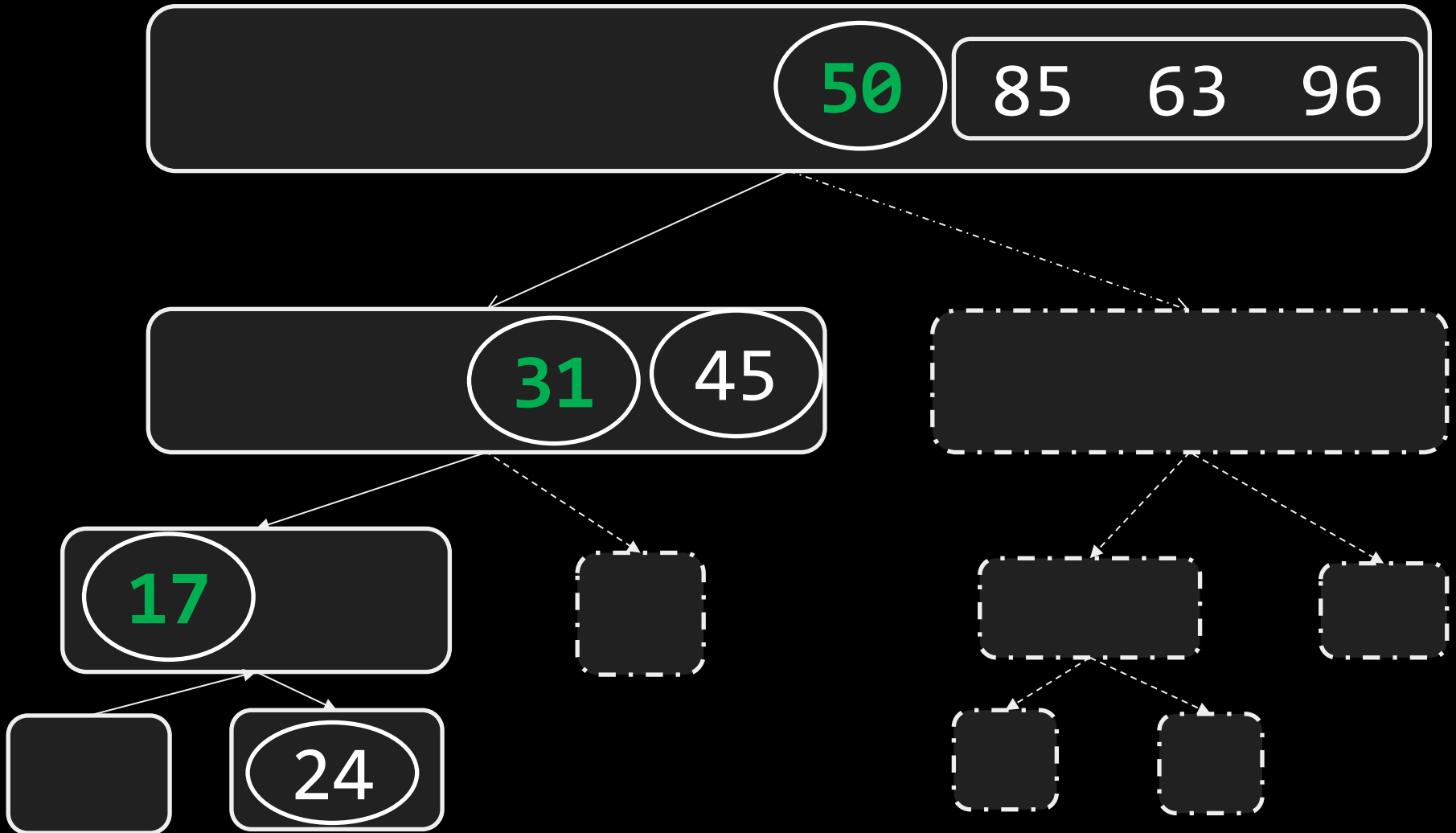

50
85
63
96
24
45
17
31

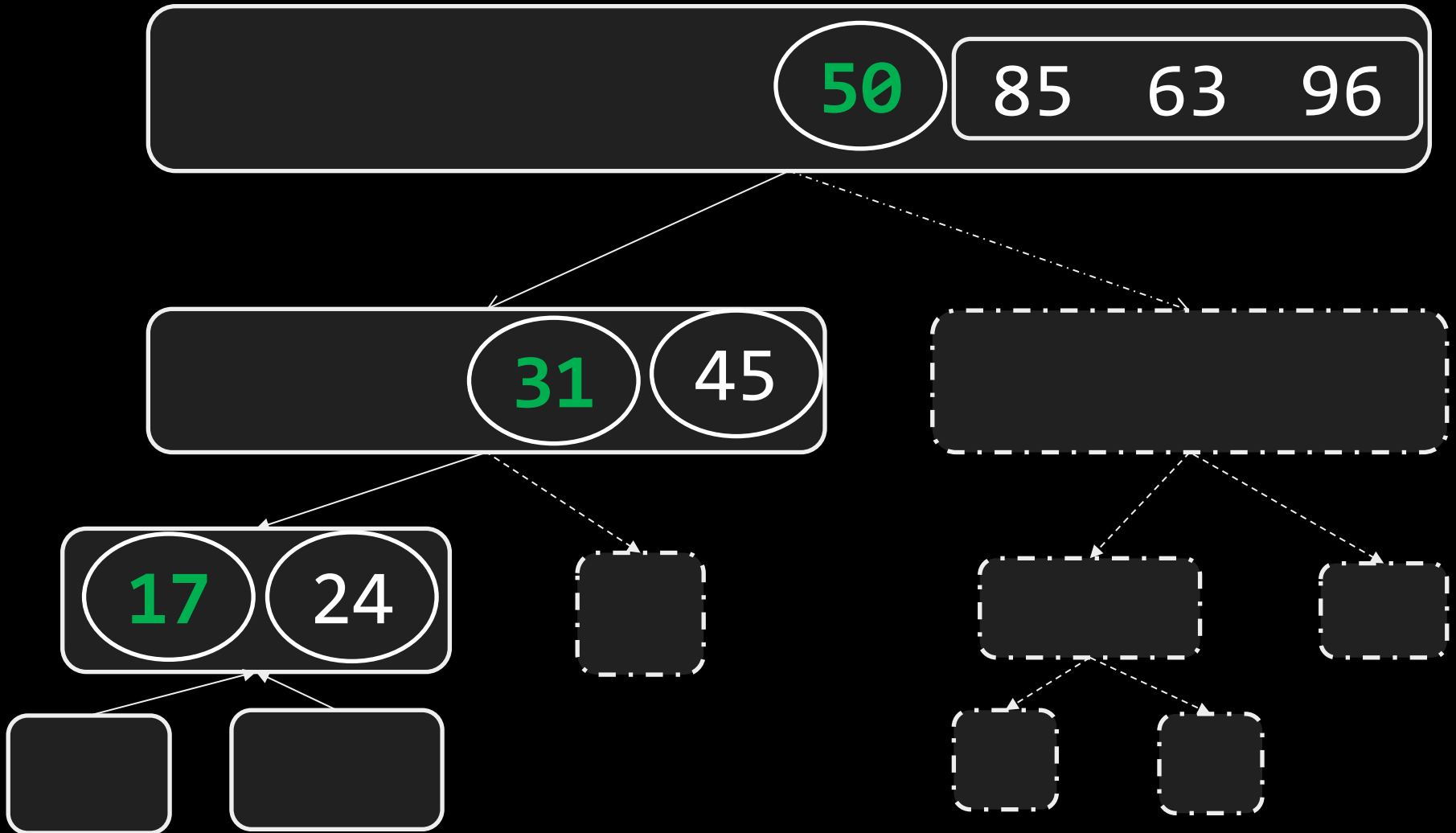

50
85
63
96
24
17
31
45


50
85
63
96
31
45
24
17

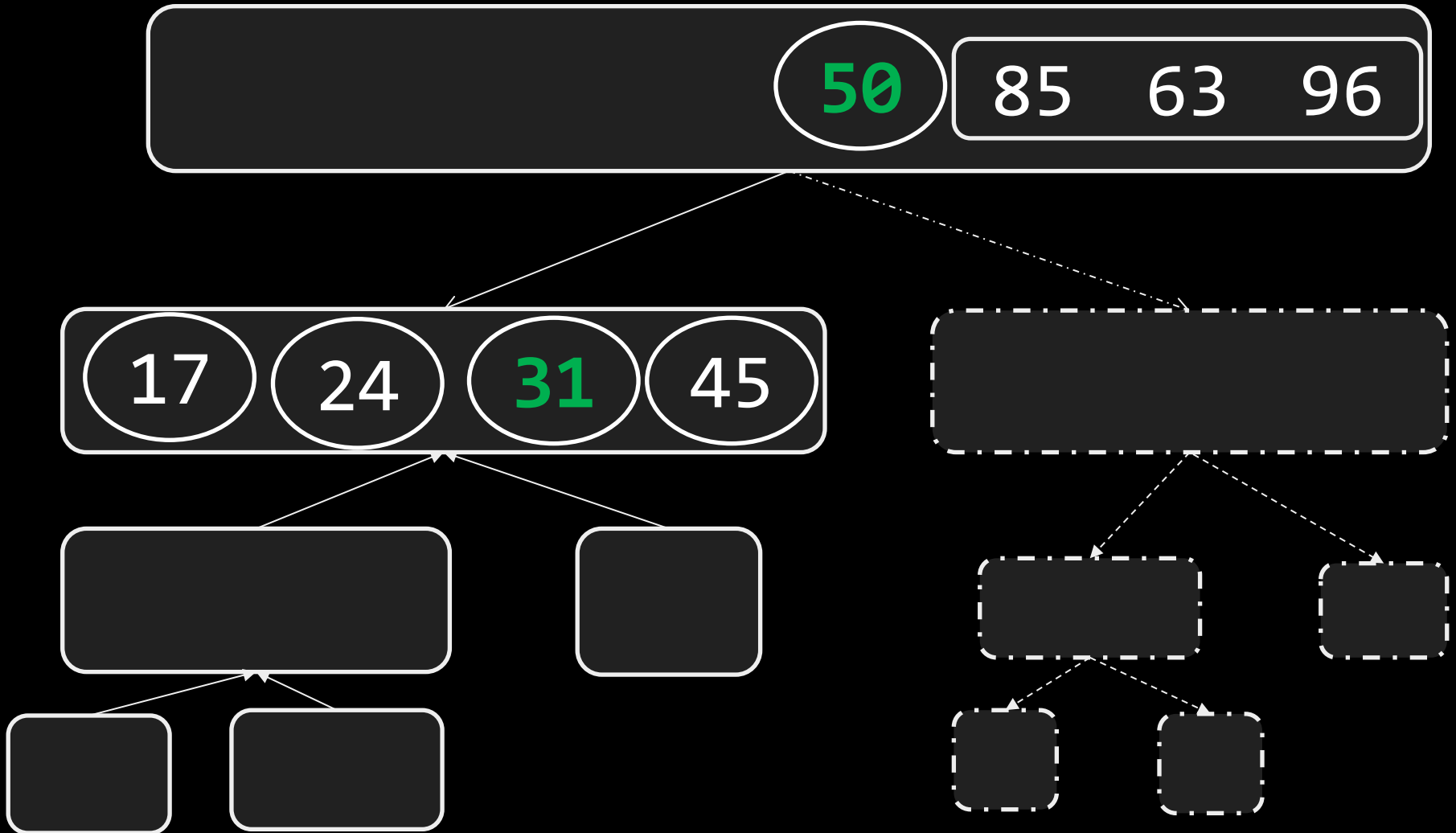

50
85
63
96
31
45
17
24


50
85
63
96
31
45
17
24

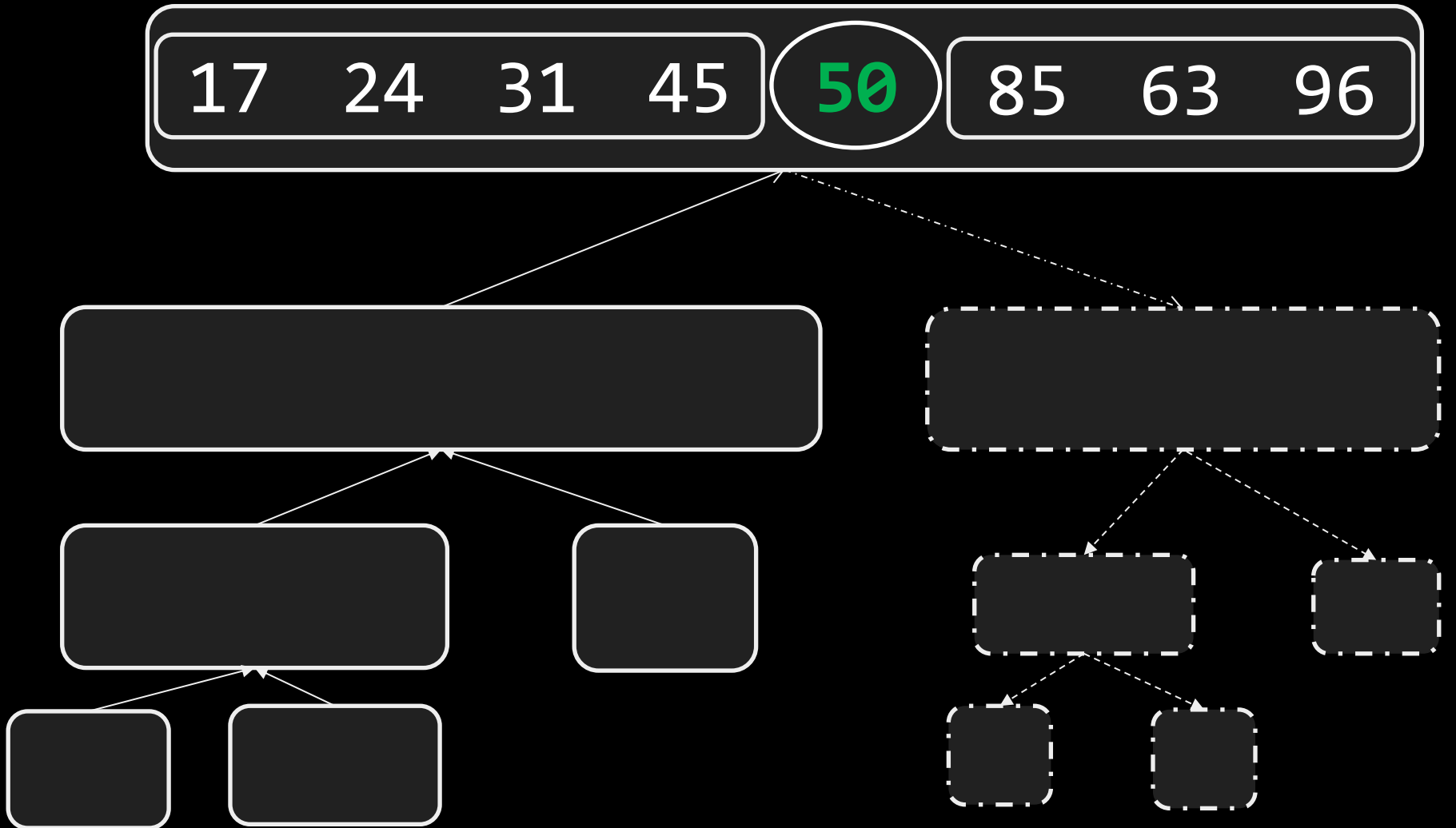

50
85
63
96
31
45
17
24

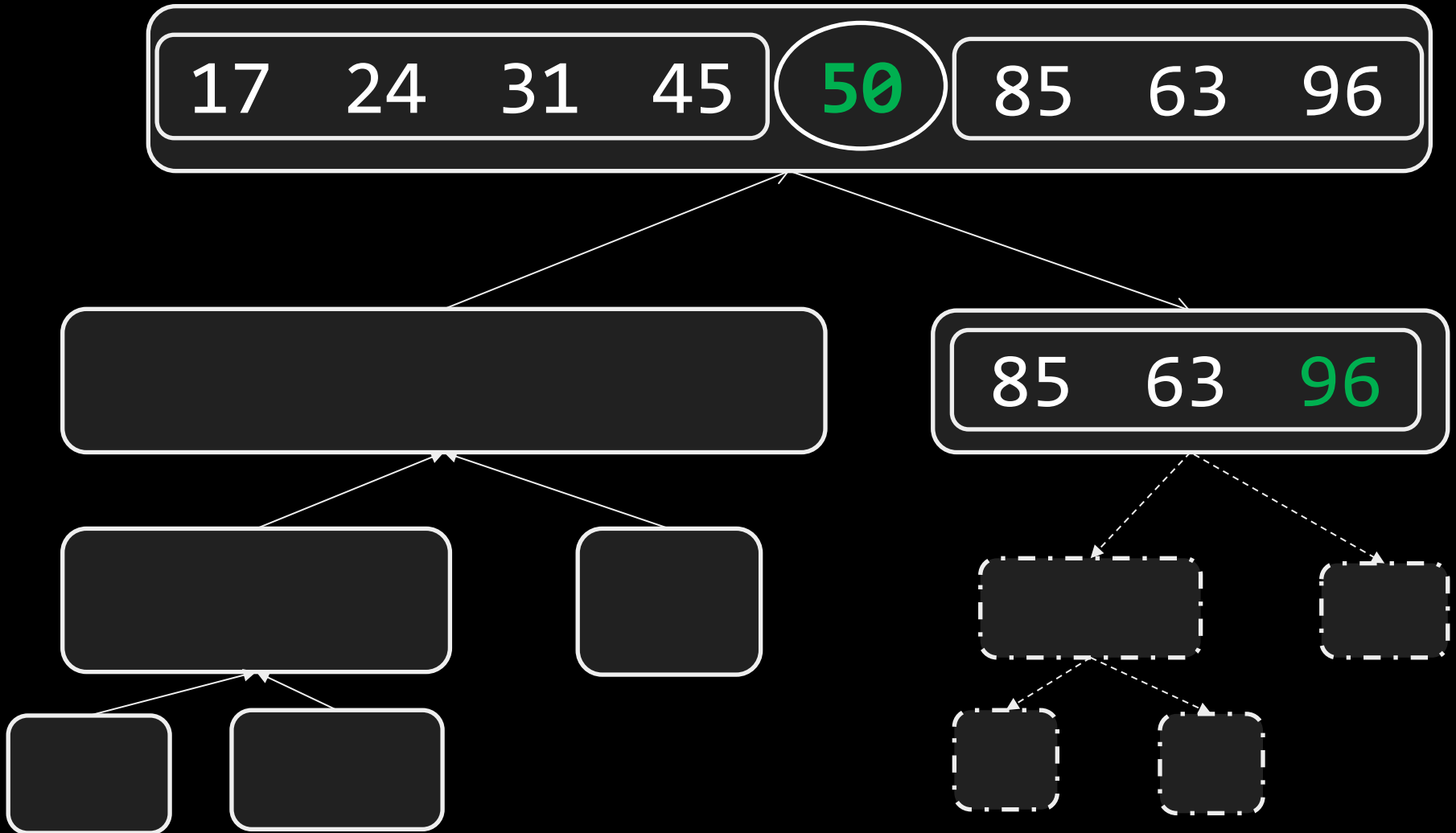

50
85
63
96
31
45
17
24

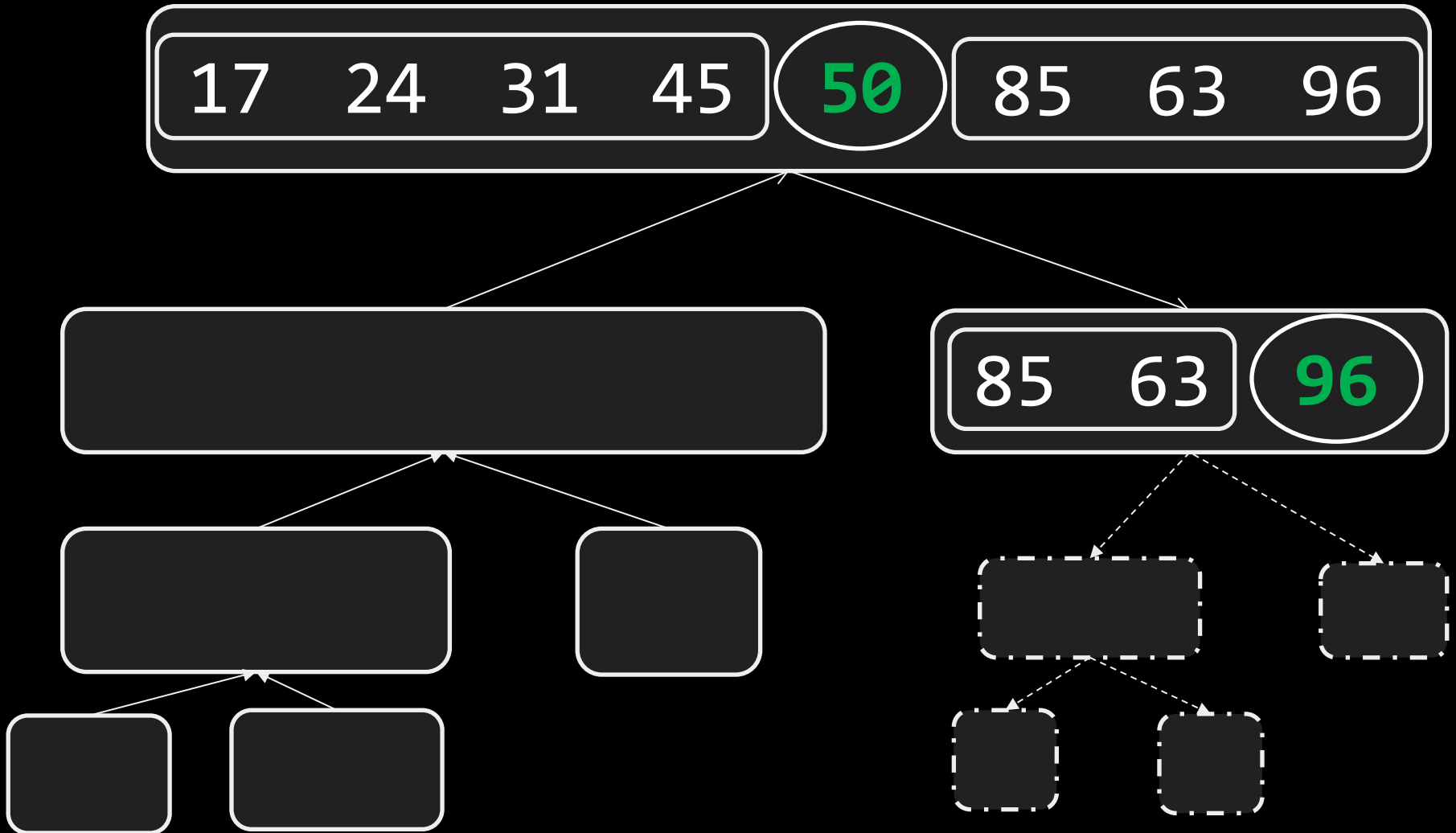

50
85
63
96
17
24
31
45

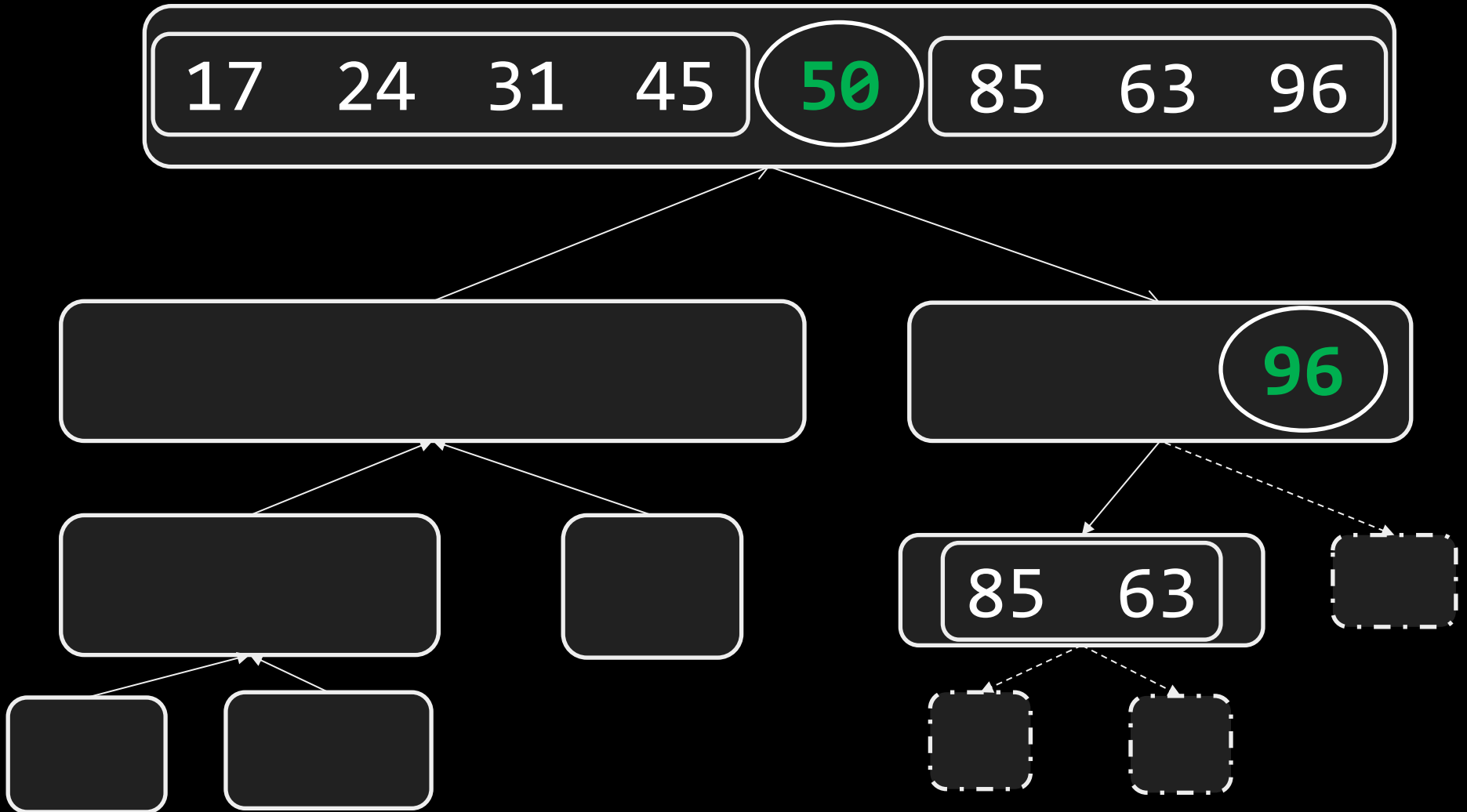

50
85
63
96
17
24
31
45

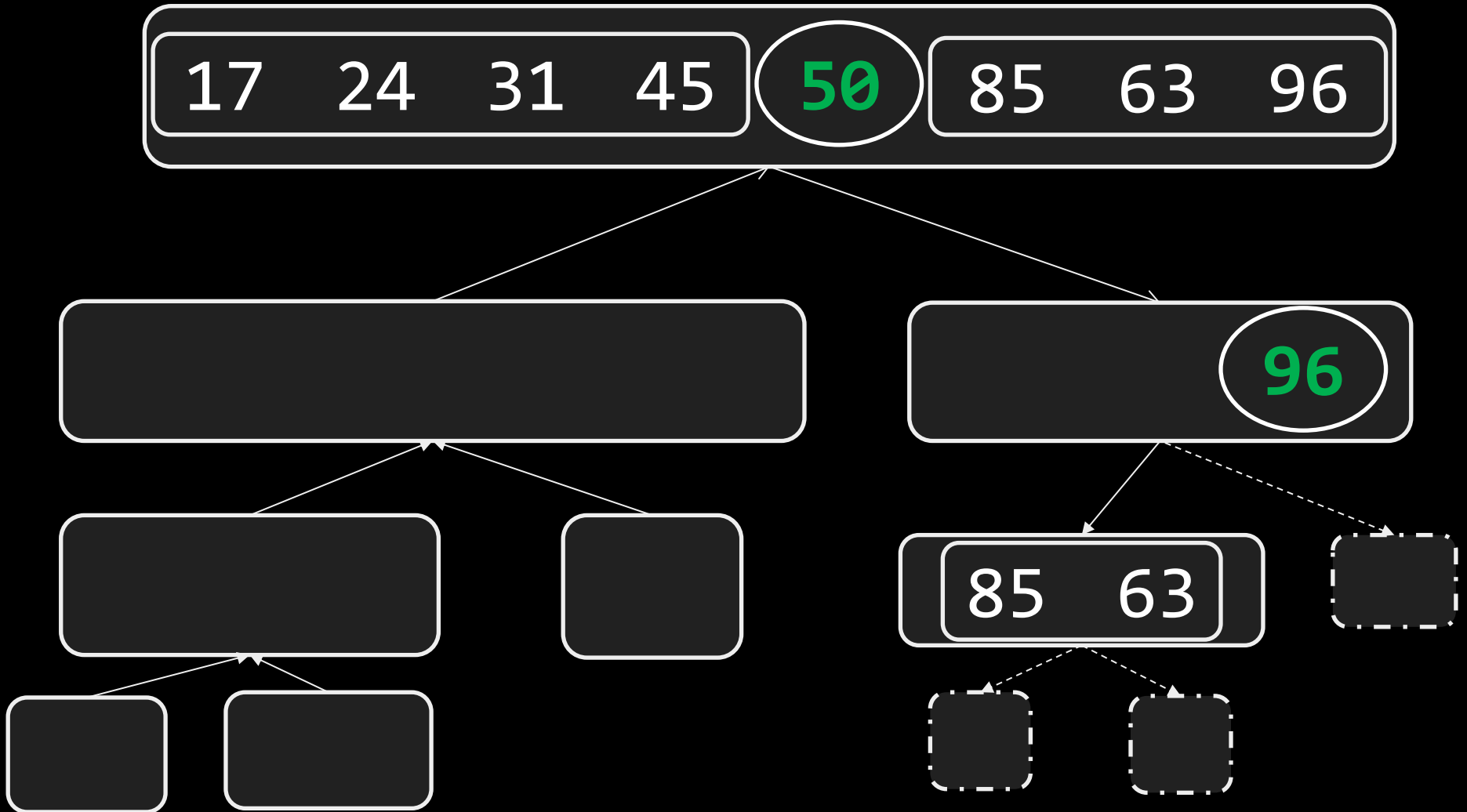

50
85 63 96
17 24 31 45

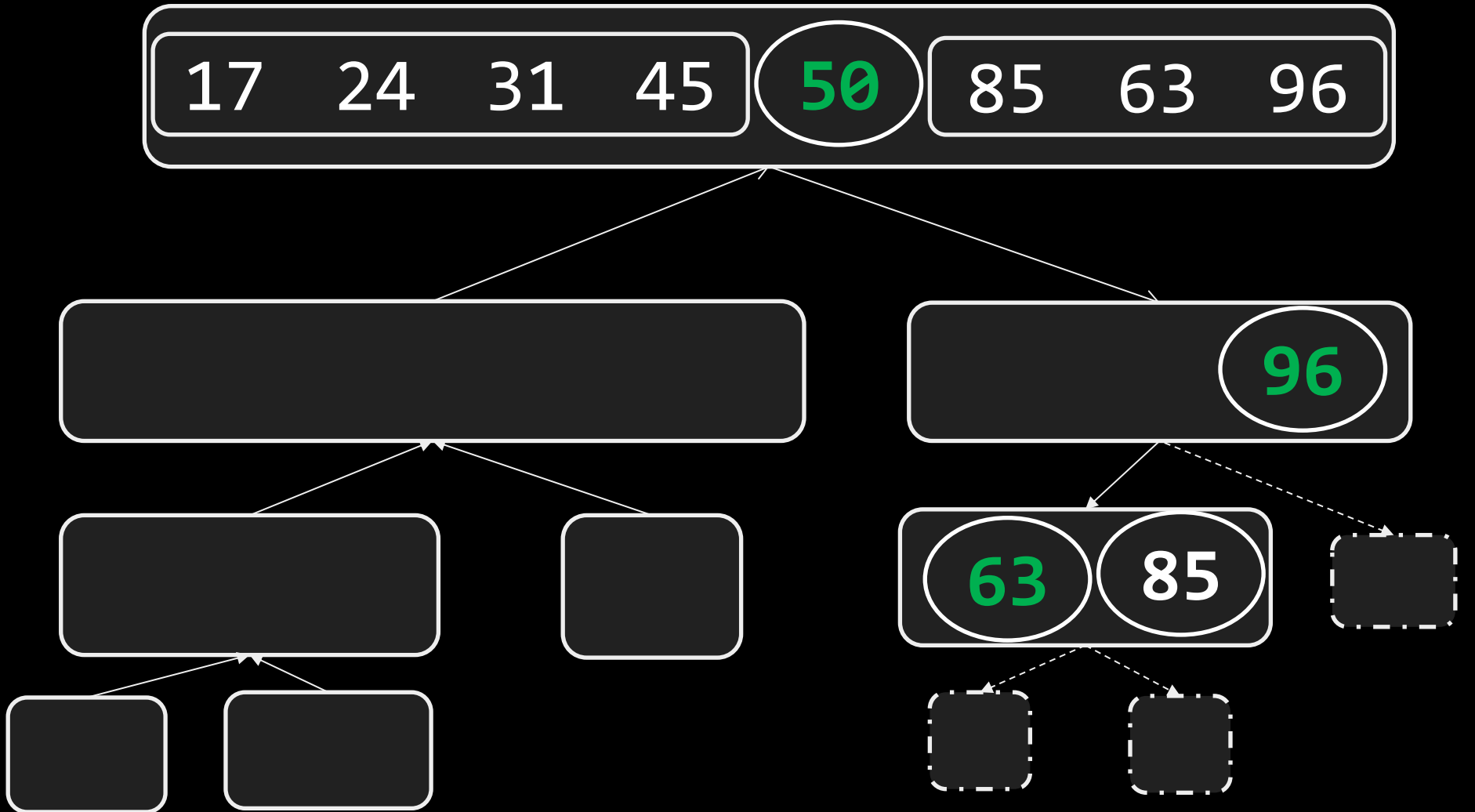

17 24 31 45
50
85 63 96

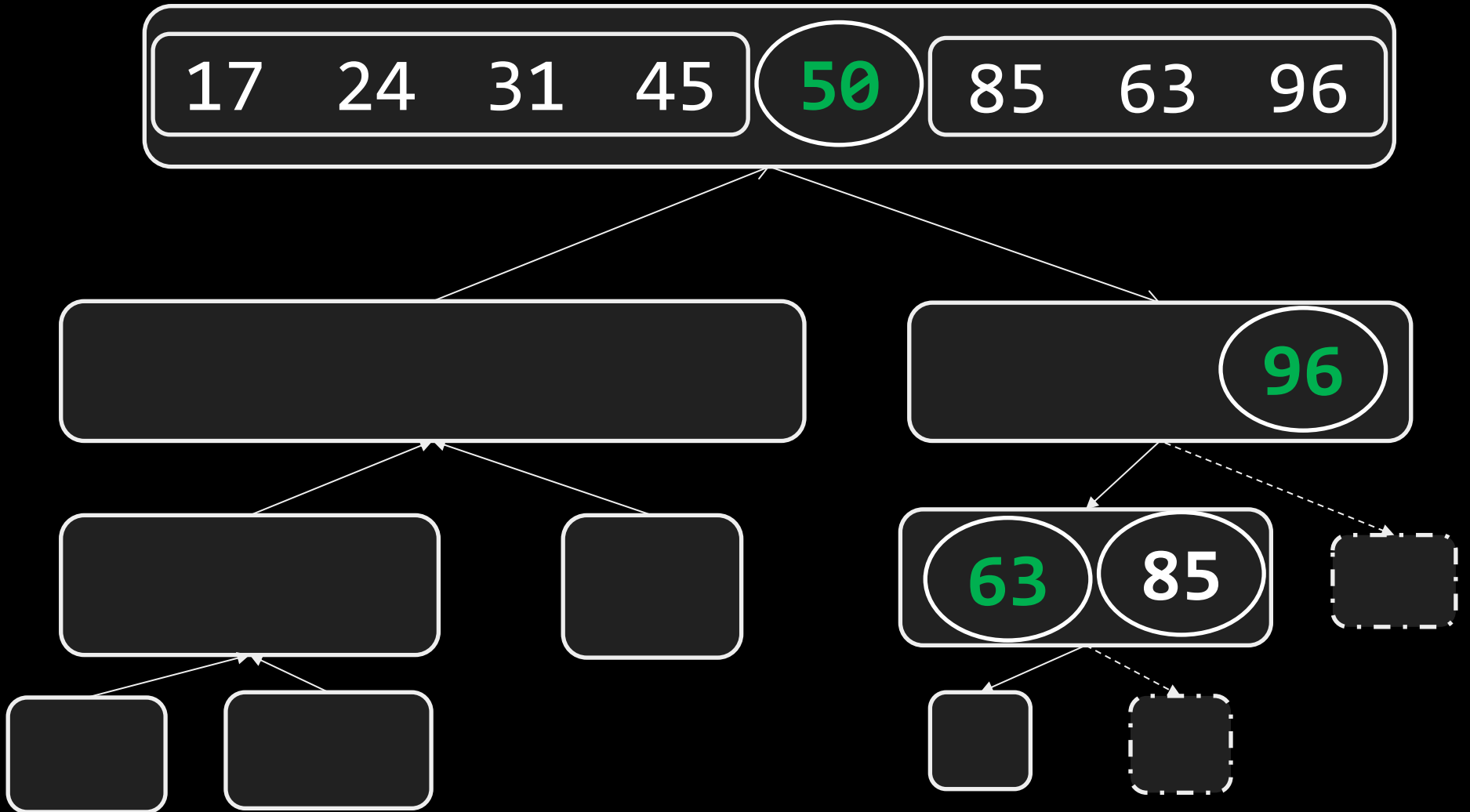

17 24 31 45
50
85 63 96
85 63 96

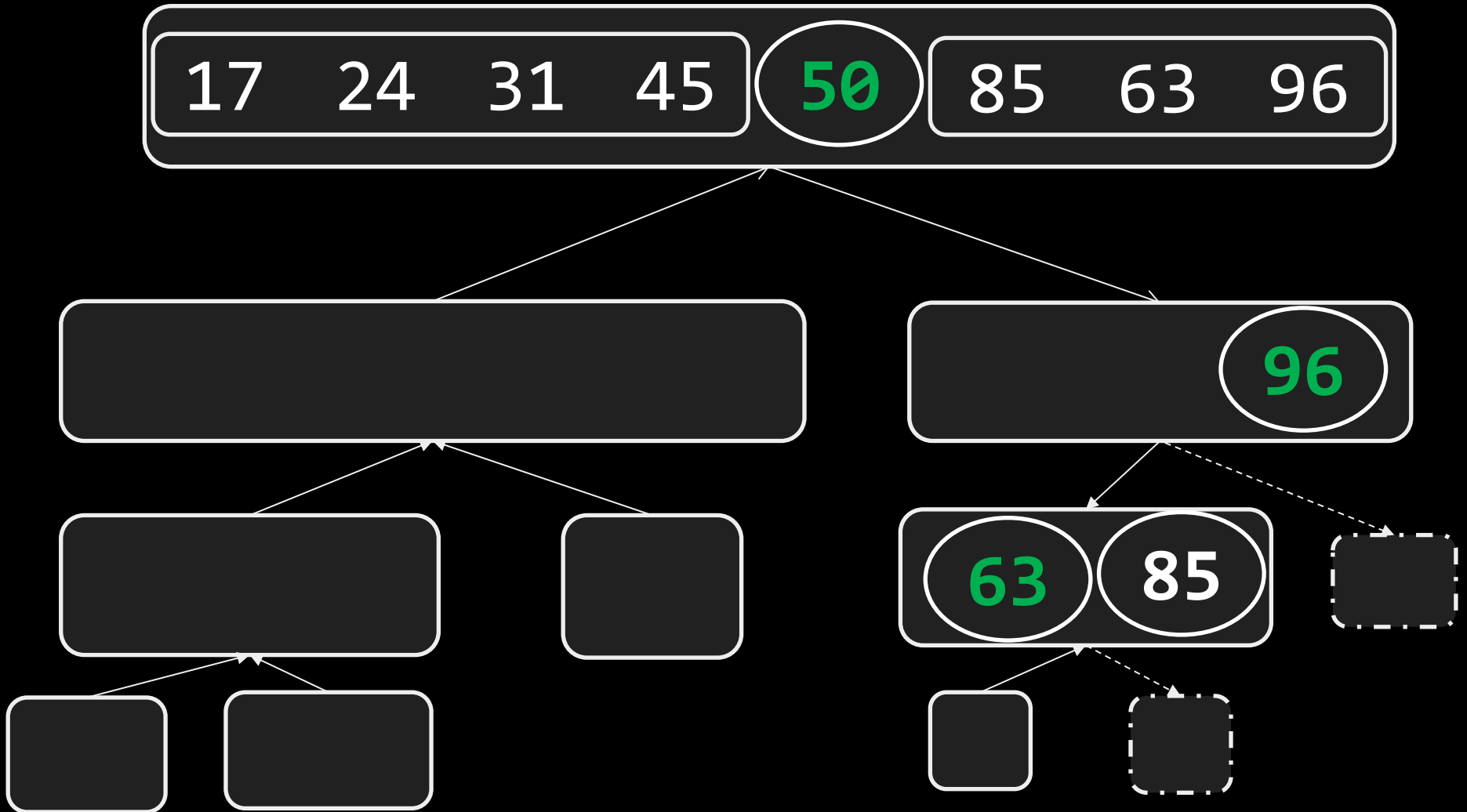

17 24 31 45
50
85 63 96
85 63
96

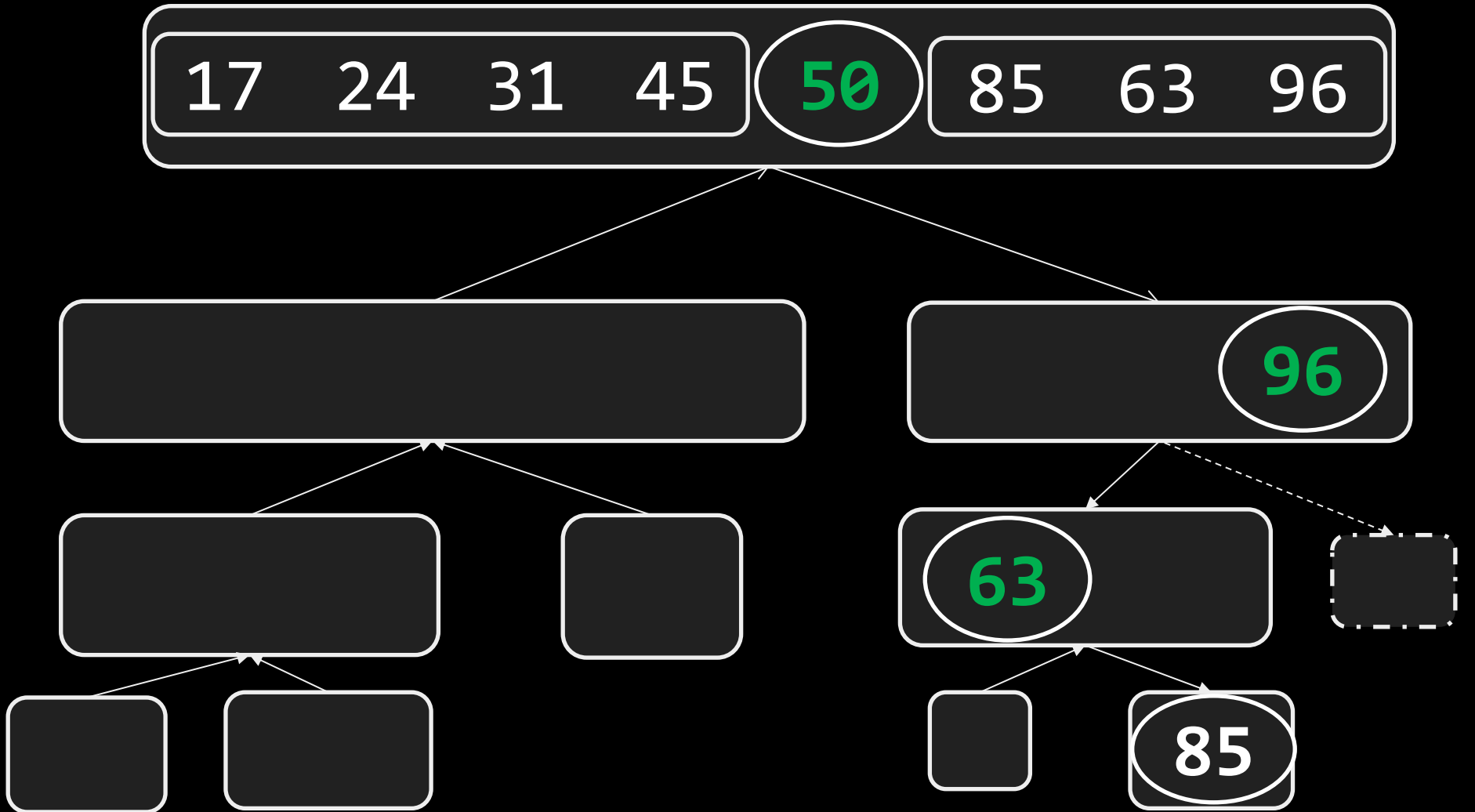

17 24 31 45
50
85 63 96
96
85 63

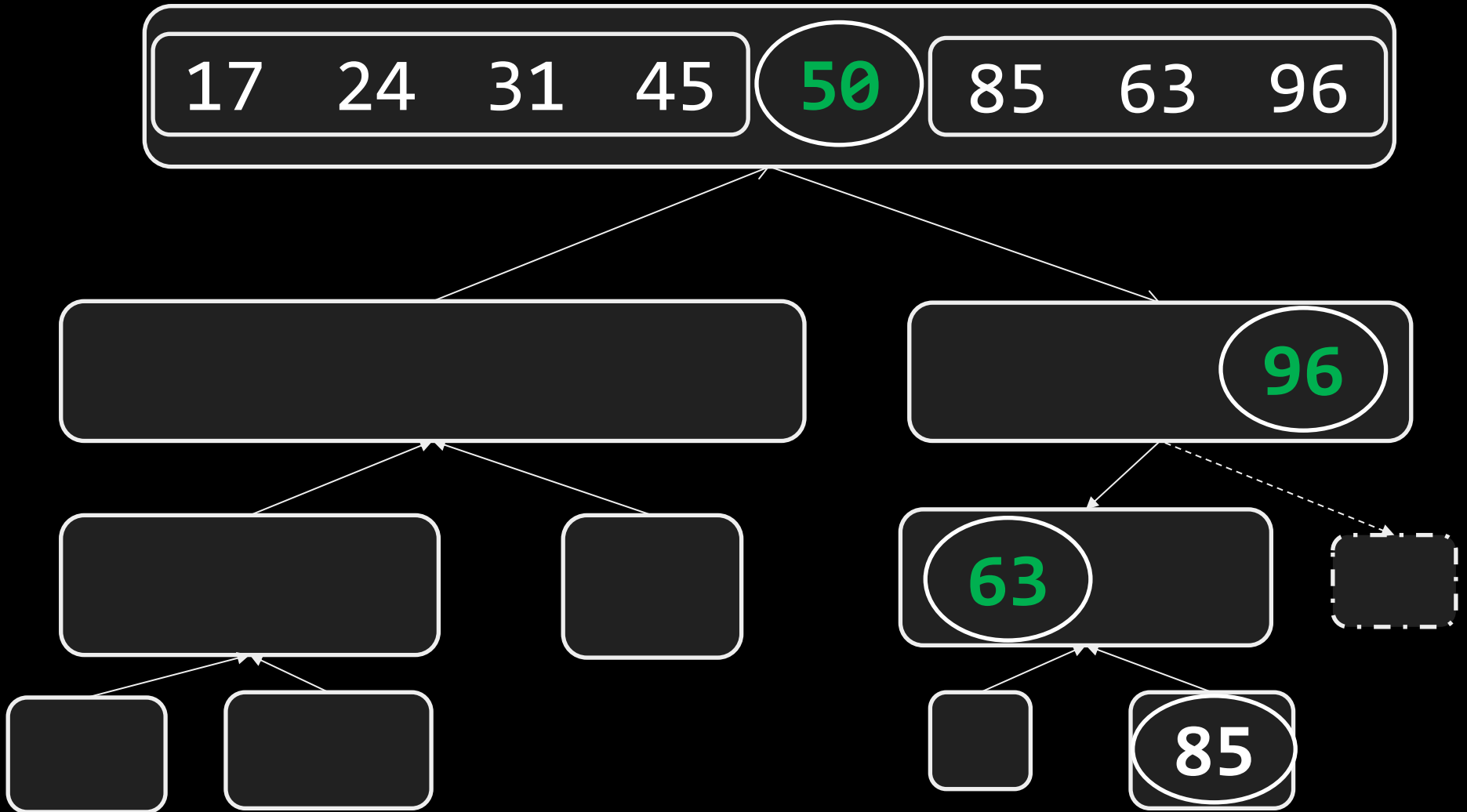

17 24 31 45
50
85 63 96
96
85 63

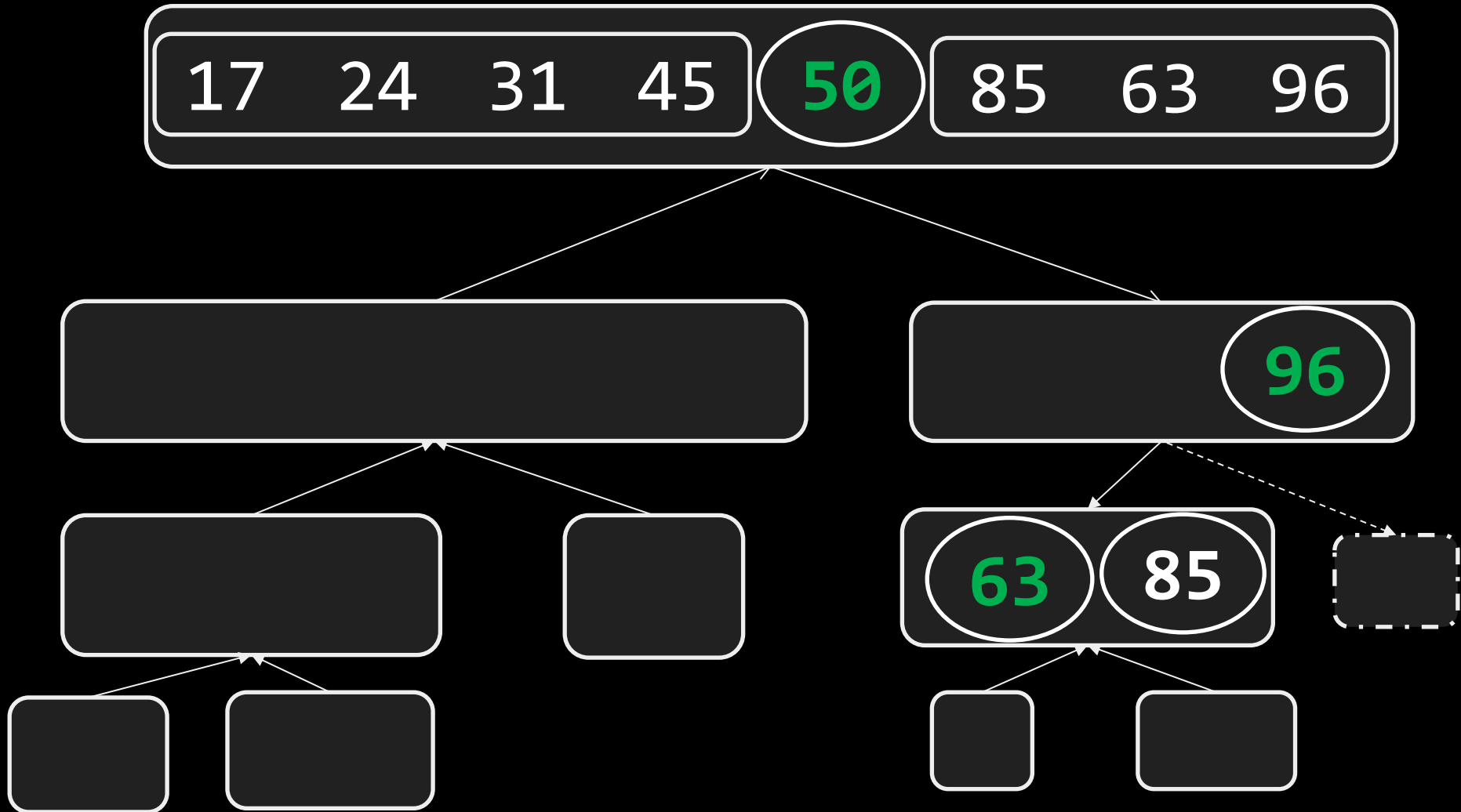

17 24 31 45
50
85 63 96
96
63
85

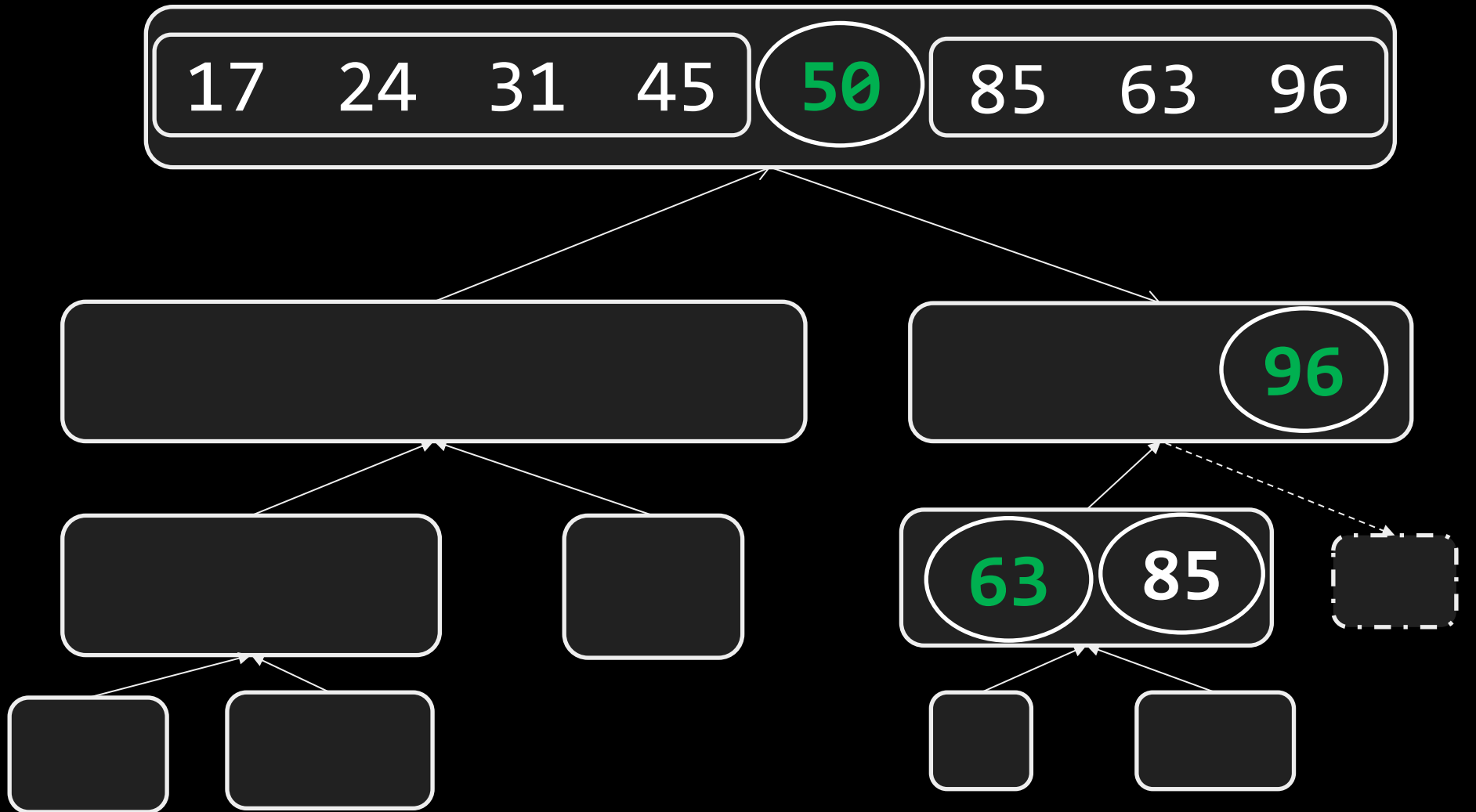

17 24 31 45
50
85 63 96
96
63
85

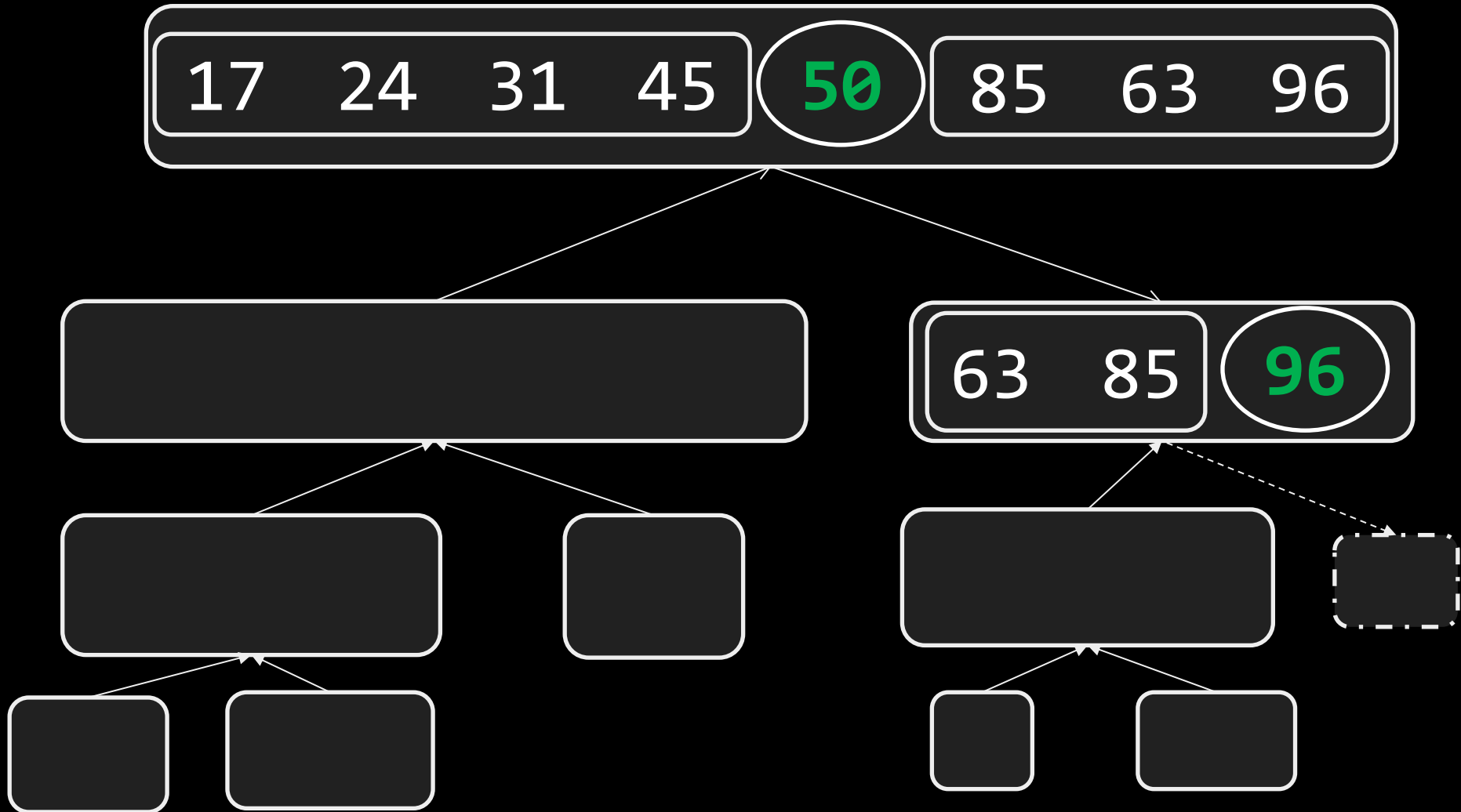

17 24 31 45
50
85 63 96
96
63
85

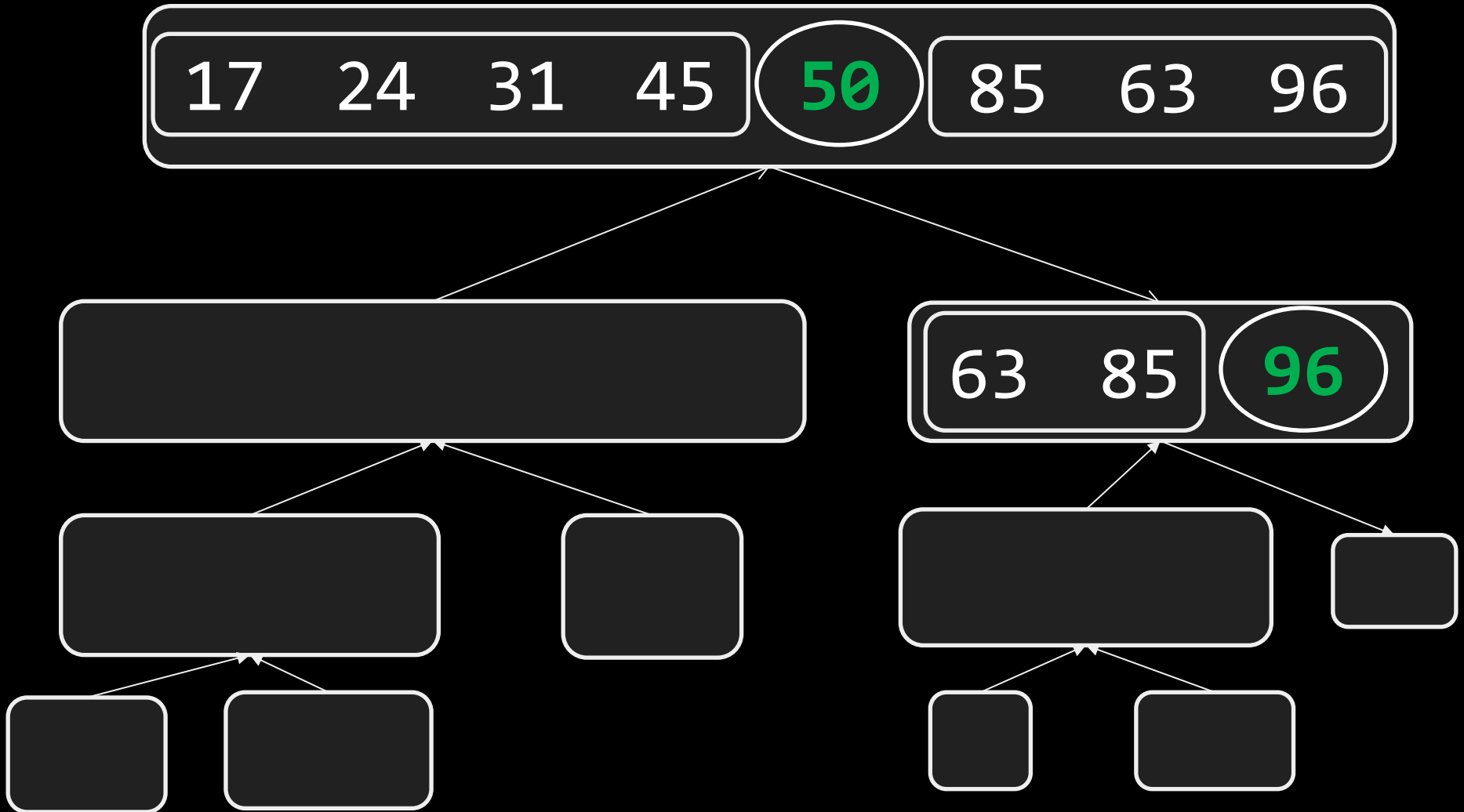

17 24 31 45
50
85 63 96
96
63
85

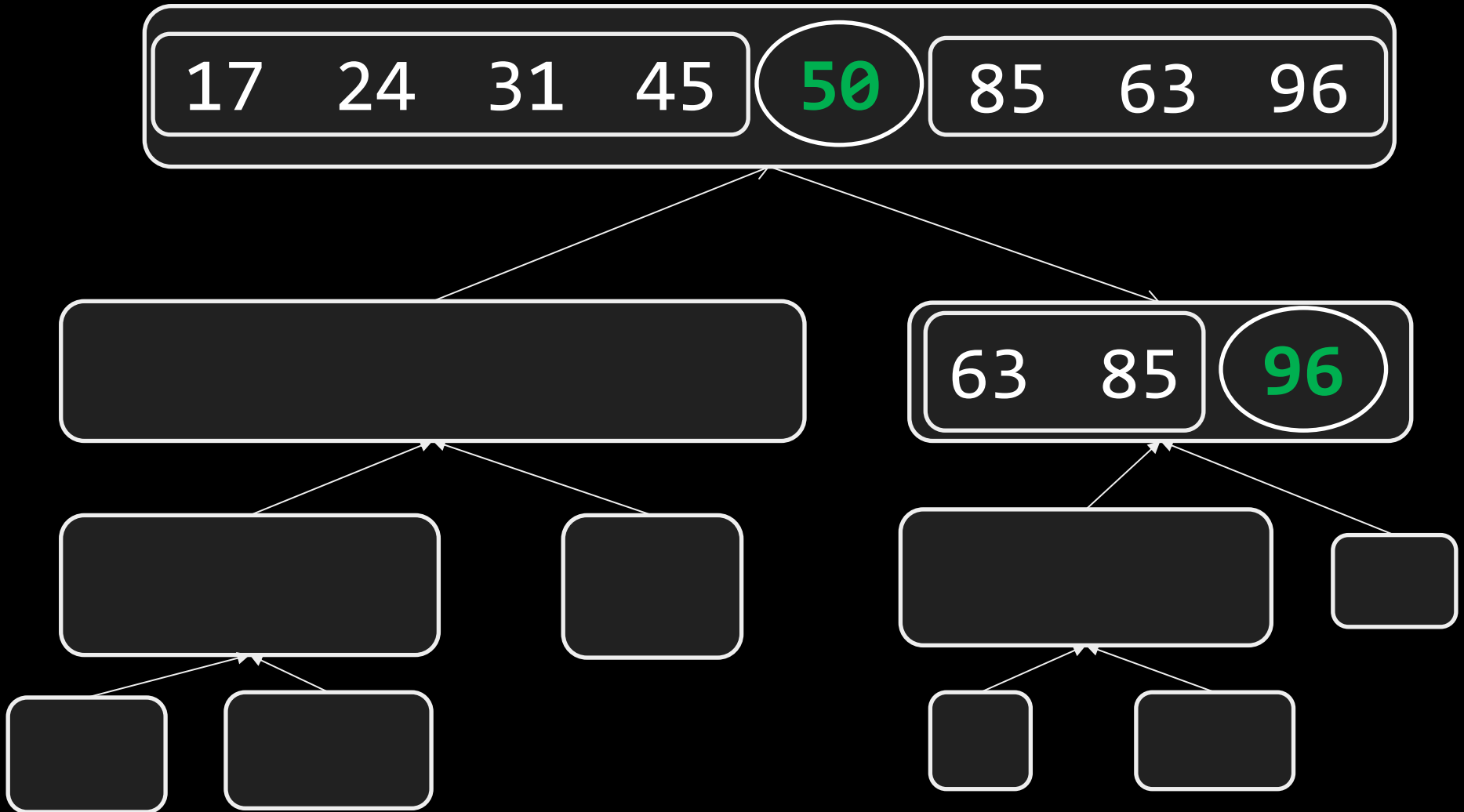

17 24 31 45
50
85 63 96
96
63
85

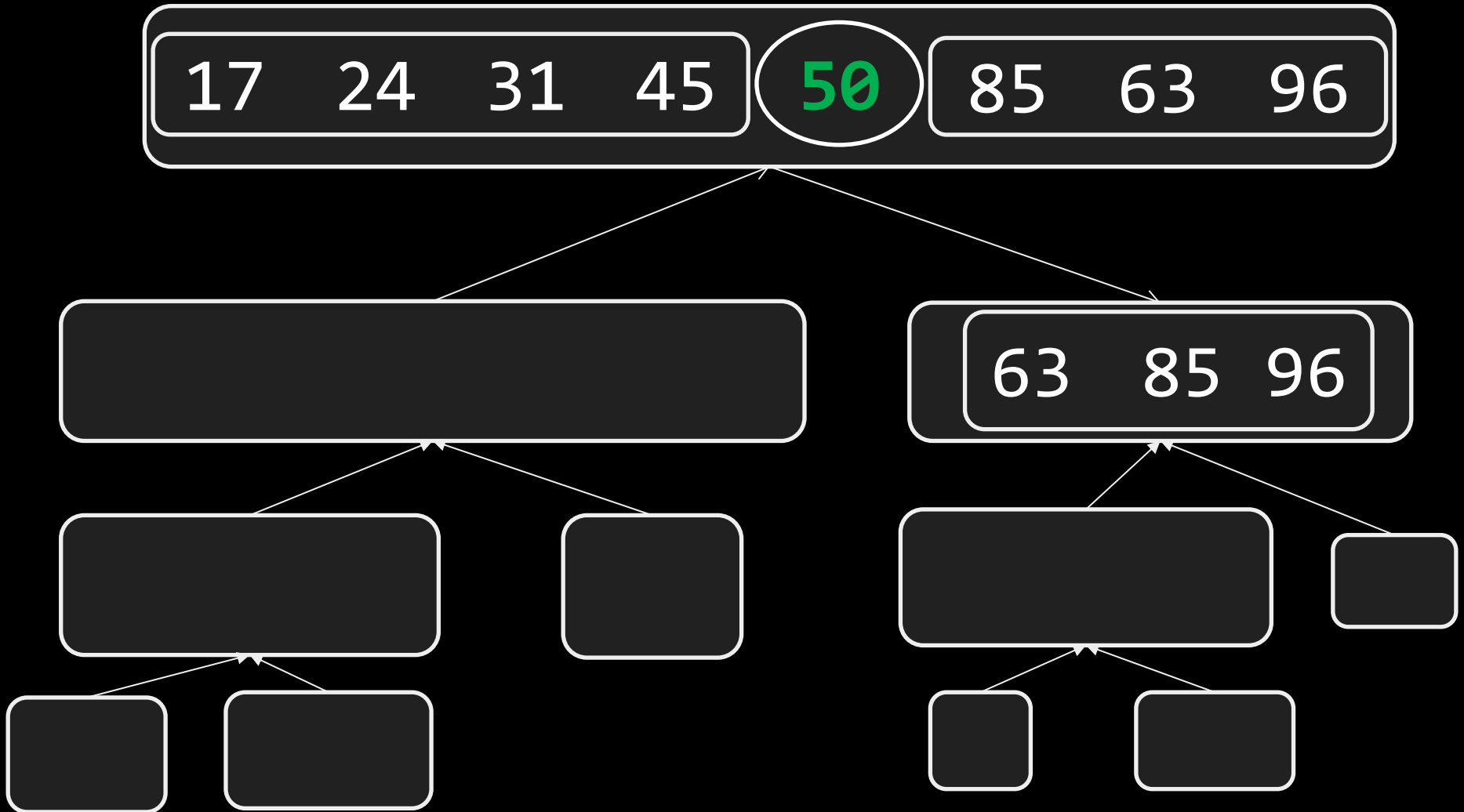

17 24 31 45
50
85 63 96
96
63
85

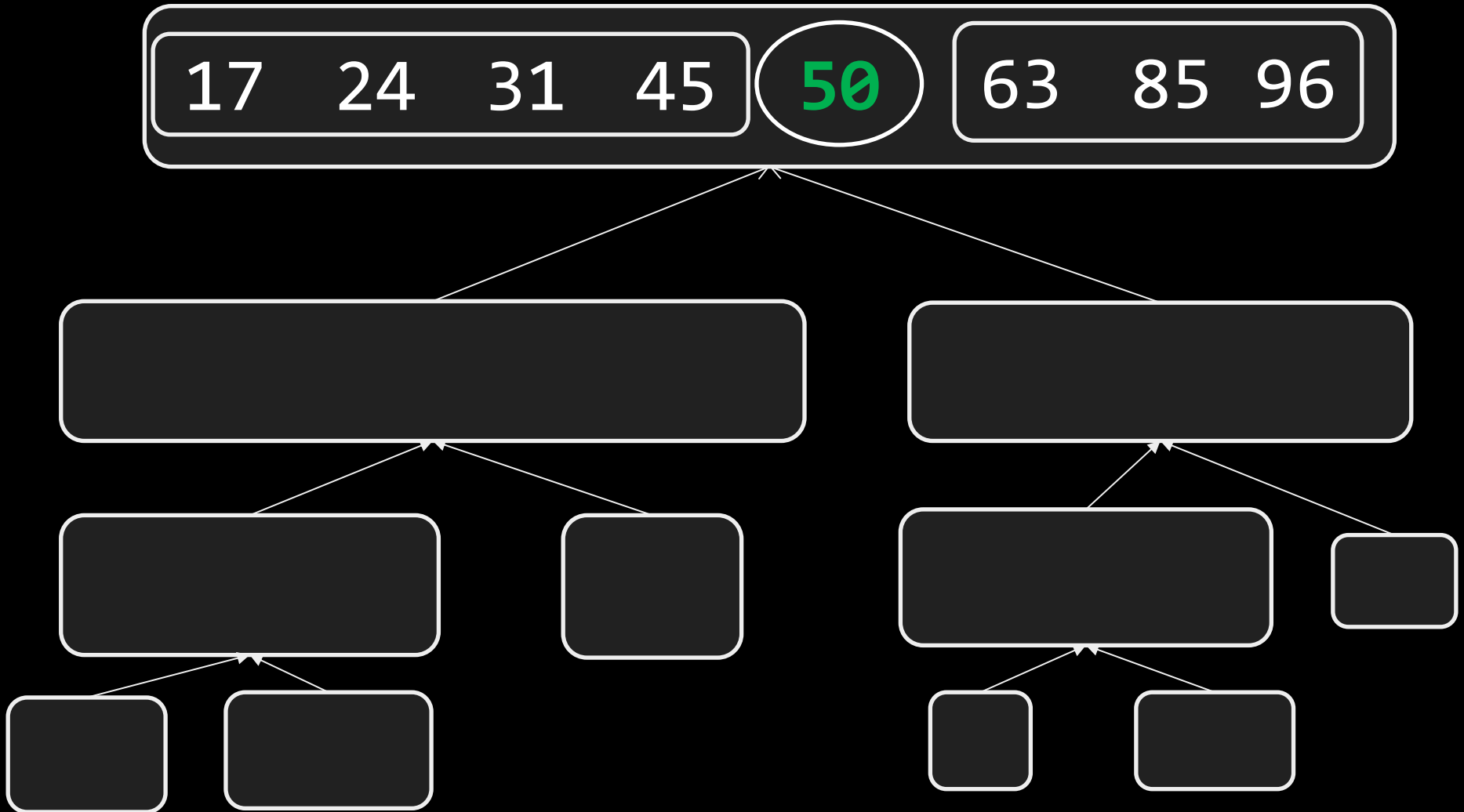

17 24 31 45
50
85 63 96
96
63
85

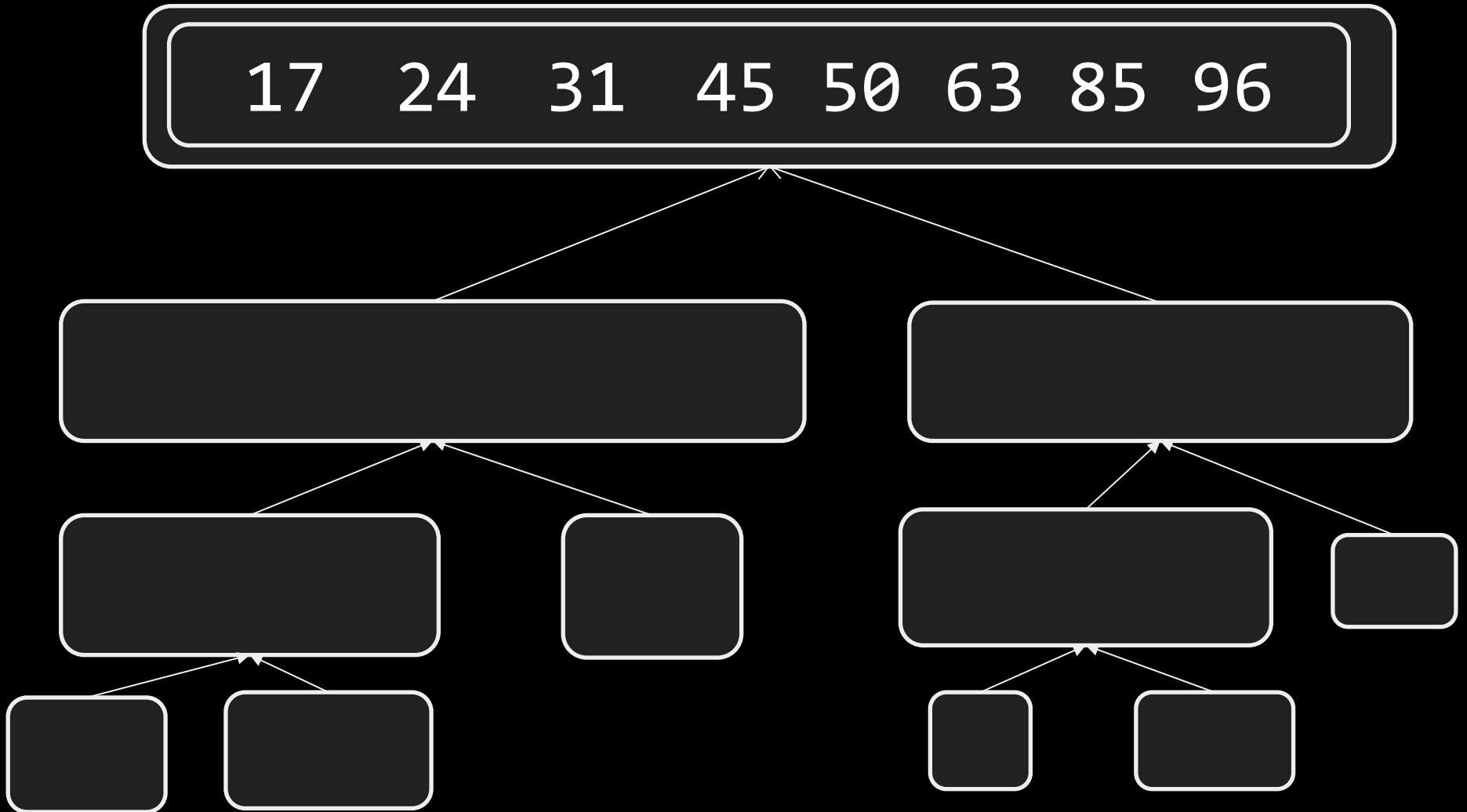

17 24 31 45
50
85 63 96
63 85
96


17 24 31 45
50
85 63 96
63 85
96


17 24 31 45
50
85 63 96
63 85
96


17 24 31 45
50
85 63 96
63 85 96


17 24 31 45
50
63 85 96


17 24 31 45 50 63 85 96

$O(n^2)$	bubble sort, selection sort, insertion sort, shell sort
$O(n \log n)$	merge sort
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

$O(n^2)$	bubble sort, selection sort, insertion sort, shell sort, quick sort
$O(n \log n)$	merge sort
$O(n)$	linear search
$O(\log n)$	binary search
$O(1)$	

$\Omega(n^2)$	selection sort, insertion sort
$\Omega(n \log n)$	merge sort, shell sort
$\Omega(n)$	bubble sort
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search

$\Omega(n^2)$	selection sort, insertion sort
$\Omega(n \log n)$	merge sort, shell sort, quick sort
$\Omega(n)$	bubble sort
$\Omega(\log n)$	
$\Omega(1)$	linear search, binary search