MergeSort

- The desirable features of Mergesort
  - It performs in O (n log n) in the worst case
  - Selection Sort was O(n^2), Bubble Sort was also O(n^2).
  - It is stable
  - It is quite independent of the way the initial list is organized
  - Good for linked lists. Can be implemented in such a way that data is accessed sequentially
- Drawbacks
  - It may require an array of up to the size of the original list
  - This can be avoided but the algorithms becomes significantly more complicated making it not worth while
  - Instead of making it complicated we can use heapsort which is also O(n log n)

MergeSort - Algorithm

mergesort ()

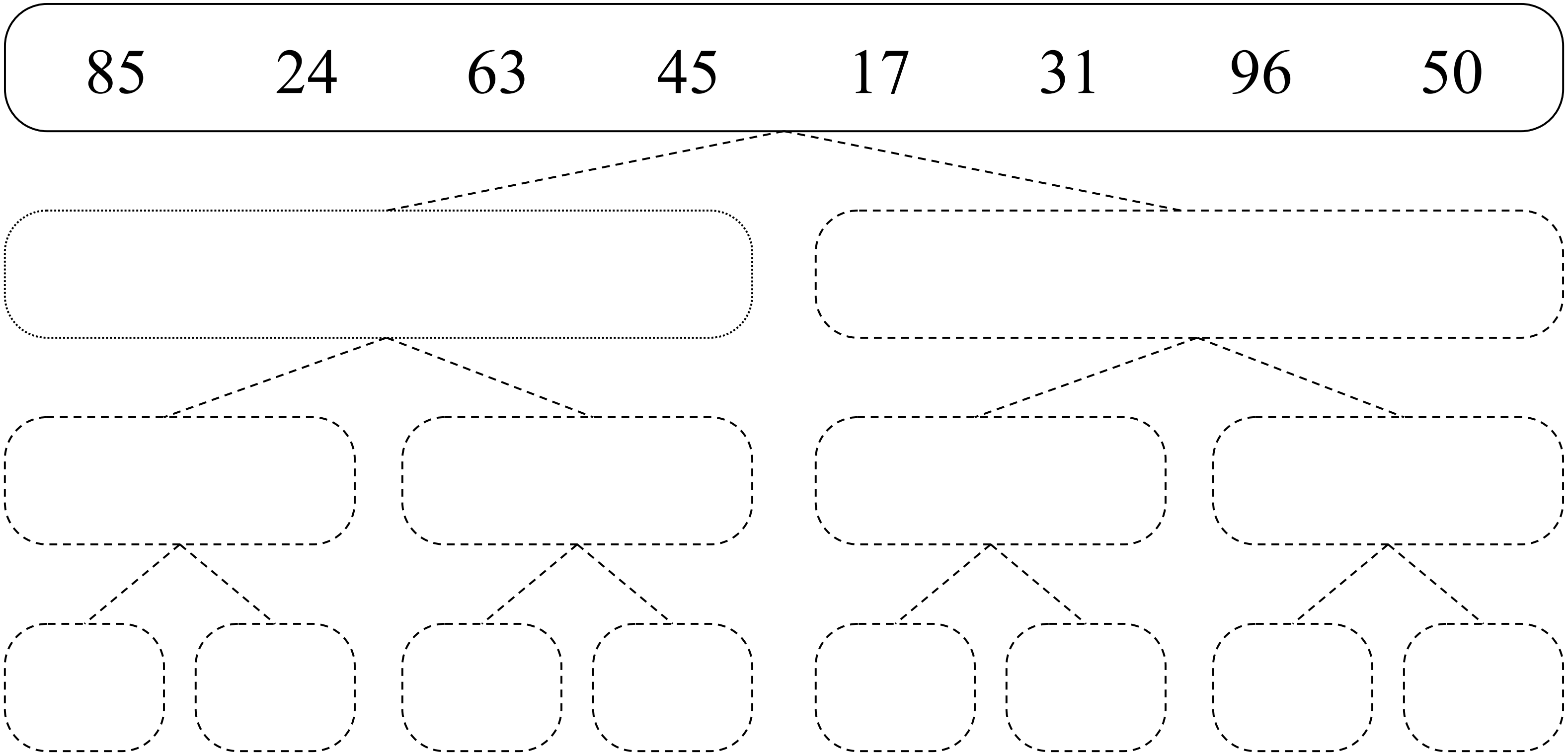
```
Item aux[MAXN];

void mergesort(Item a[], int left, int right) {
  int mid = (right + left) / 2;
  if (right <= left)
    return;
  mergesort(a, left, mid);
  mergesort(a, mid + 1, right);
  merge(a, left, mid, right);
}
```

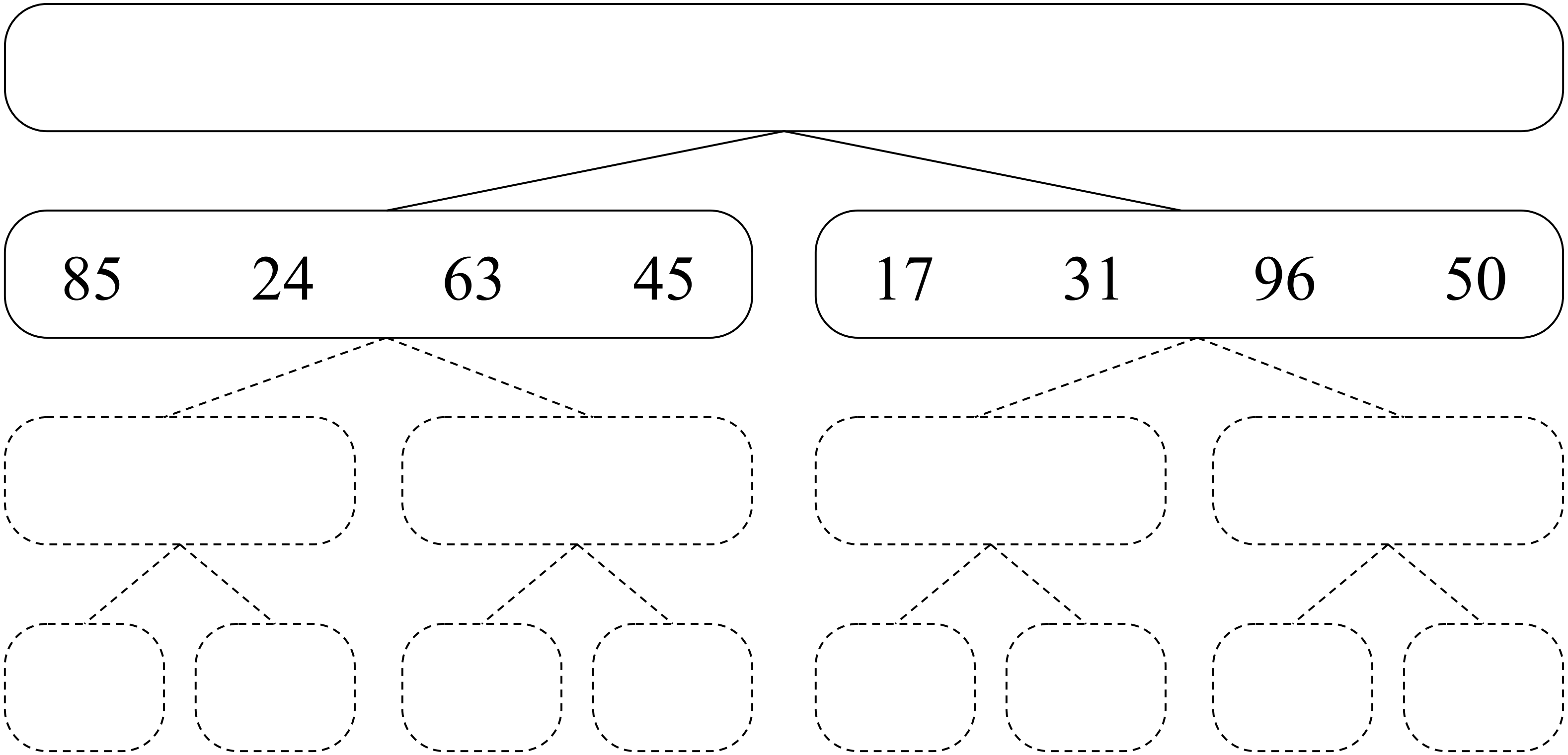
MergeSort - Algorithm

```
mergeAB(Item c[], Item a[], int N, Item b[], int M )
 { int i, j, k;
   for (i = 0, j = 0, k = 0; k < N+M; k++)
     {
       if (i == N) { c[k] = b[j++]; continue; }
       if (j == M) { c[k] = a[i++]; continue; }
       c[k] = (less(a[i], b[j])) ? a[i++] : b[j++];
     }
 }
```

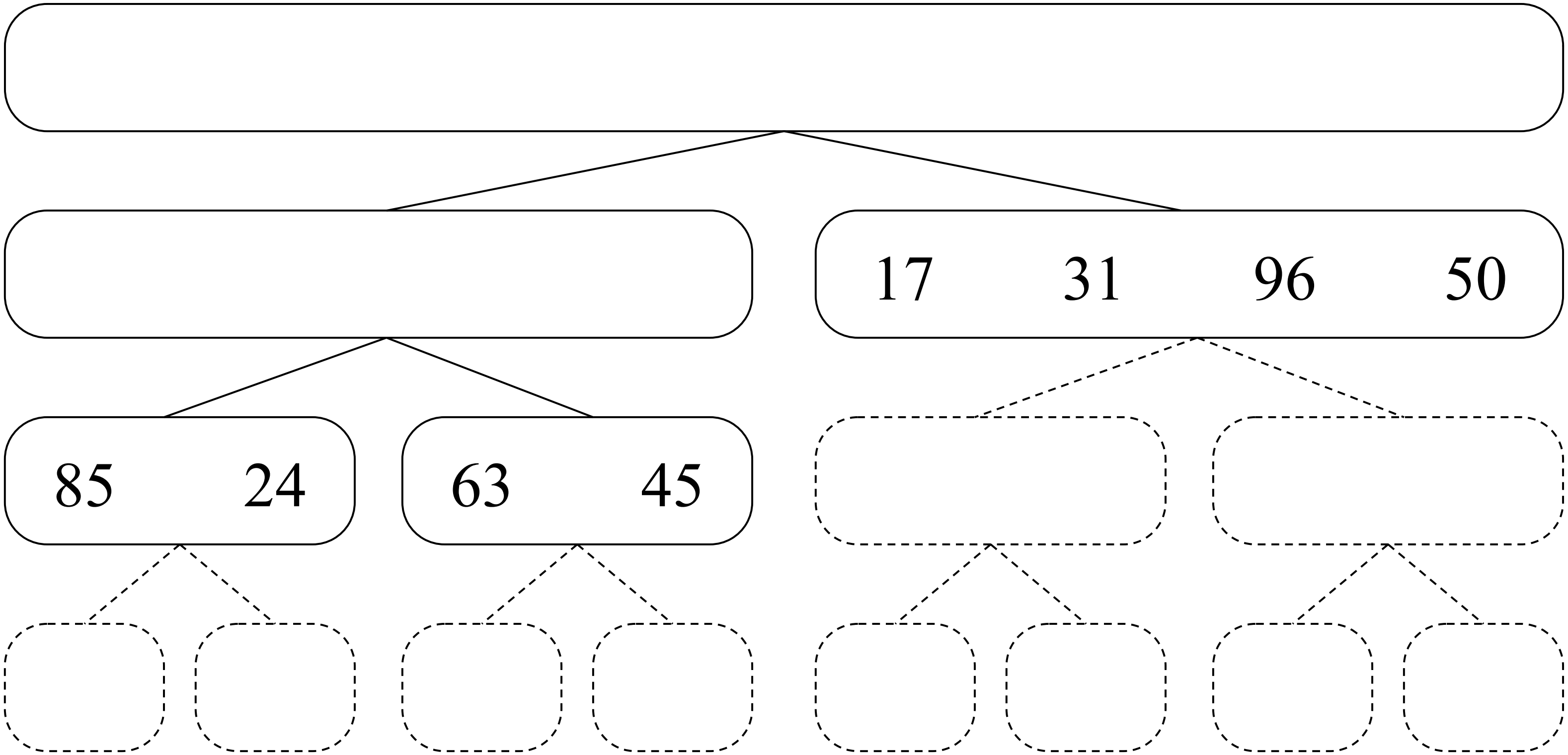
Mergesort: Illustration



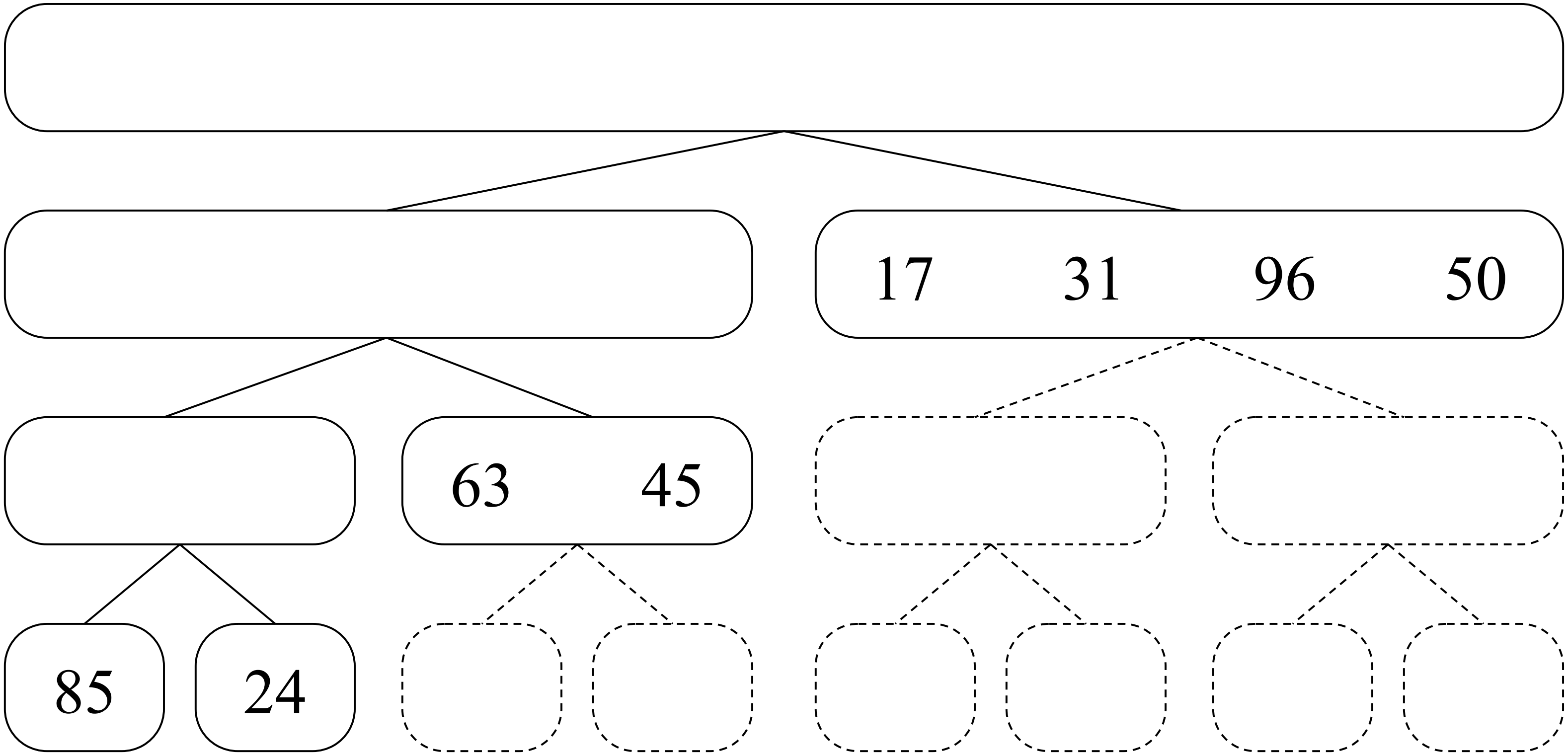
Mergesort: Illustration



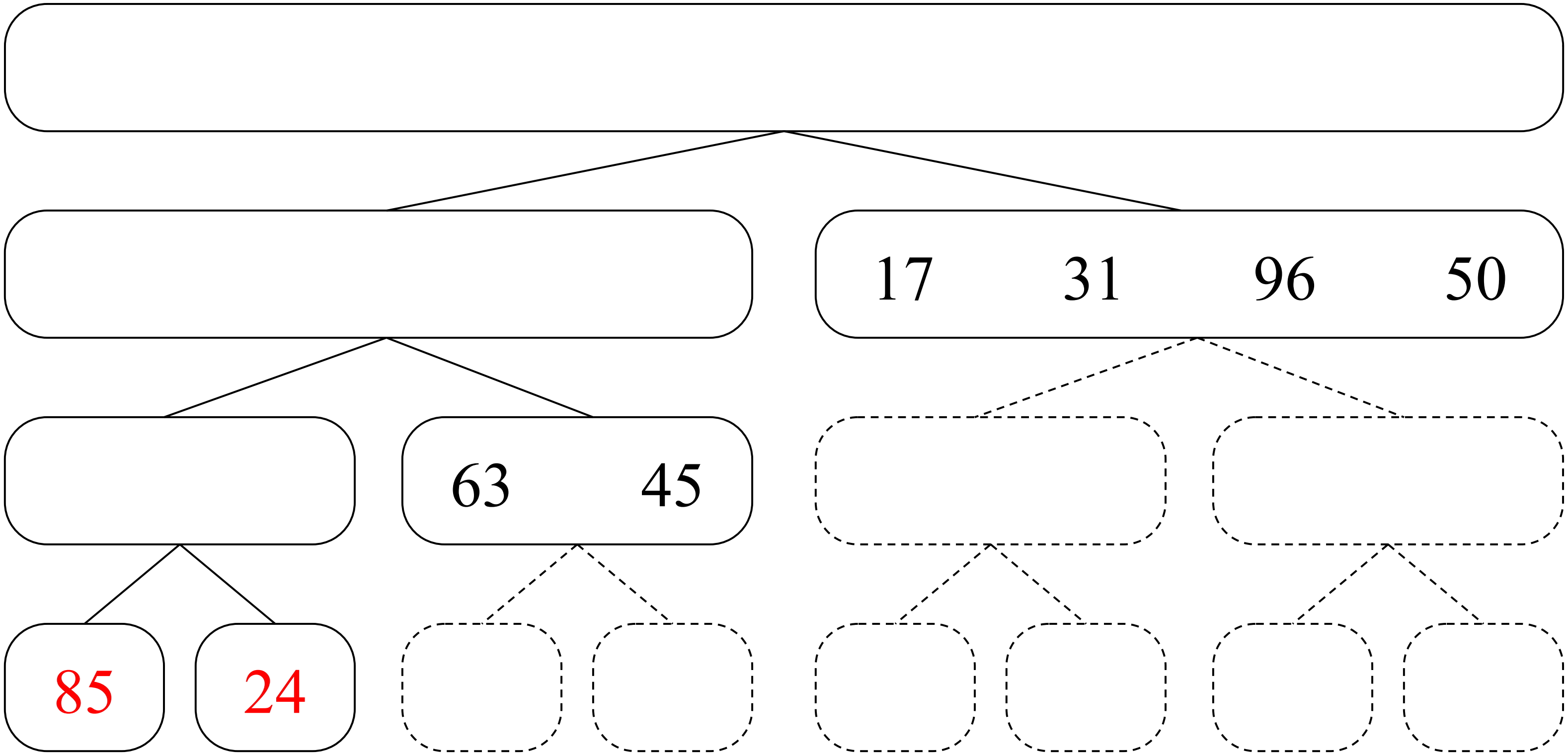
Mergesort: Illustration



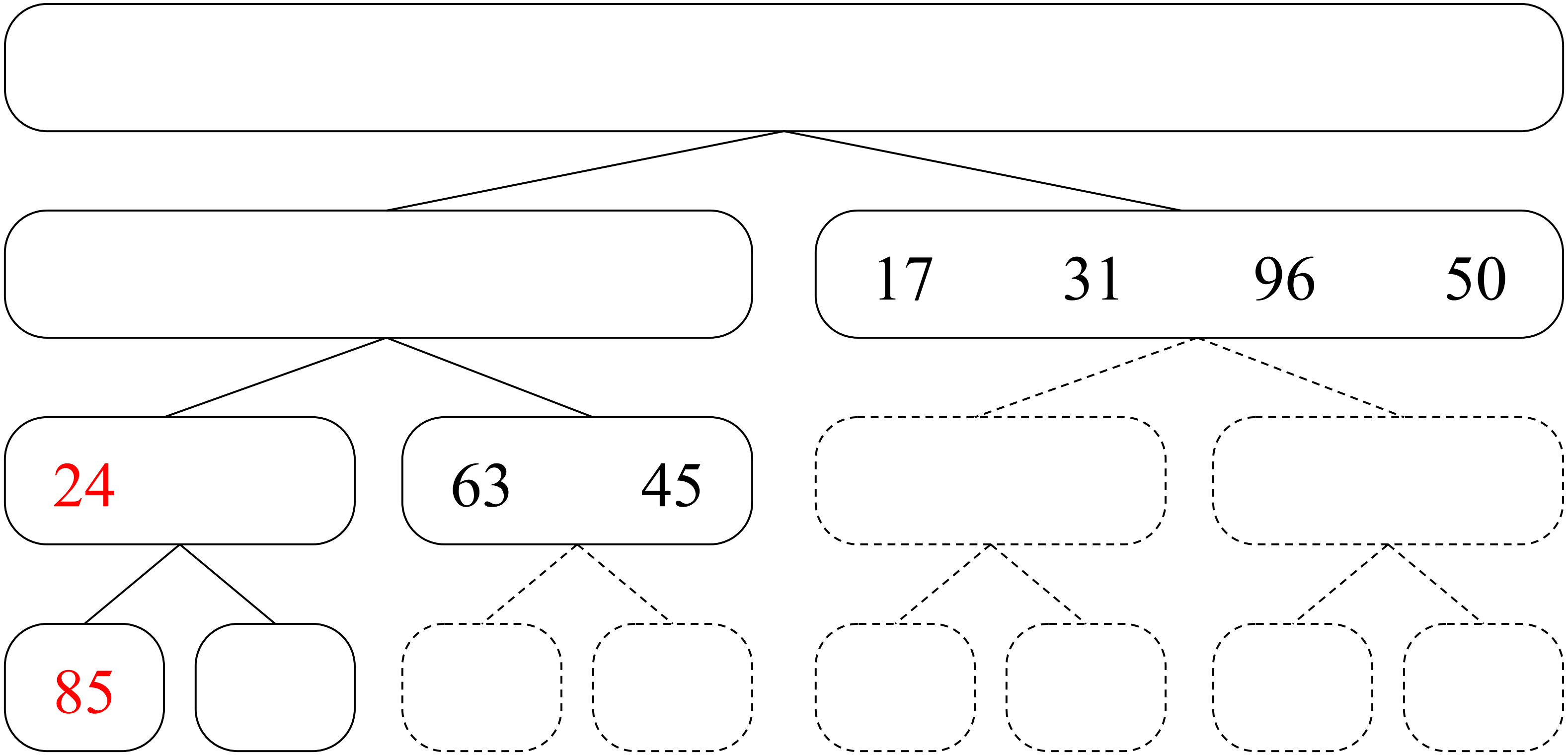
Mergesort: Illustration



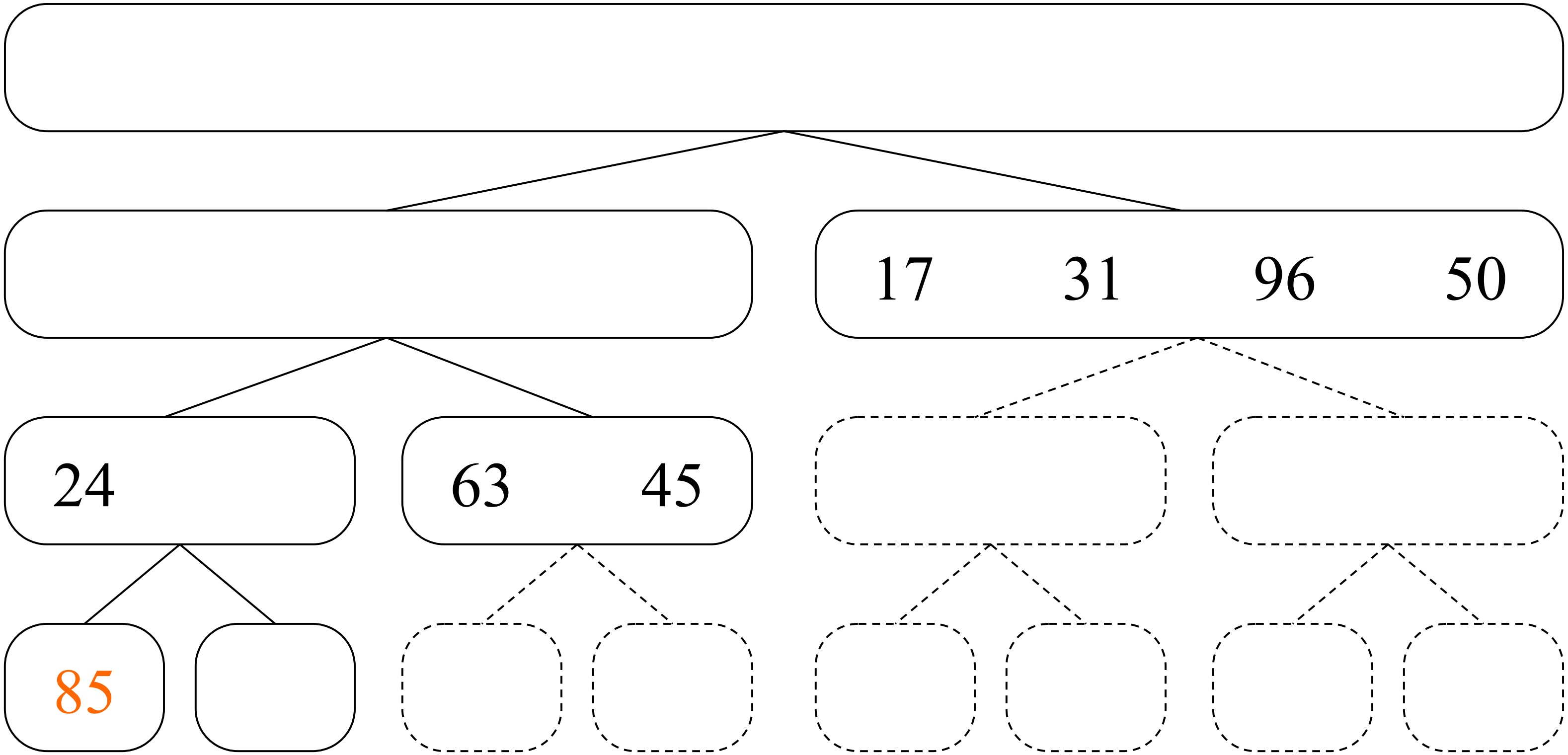
Mergesort: Illustration



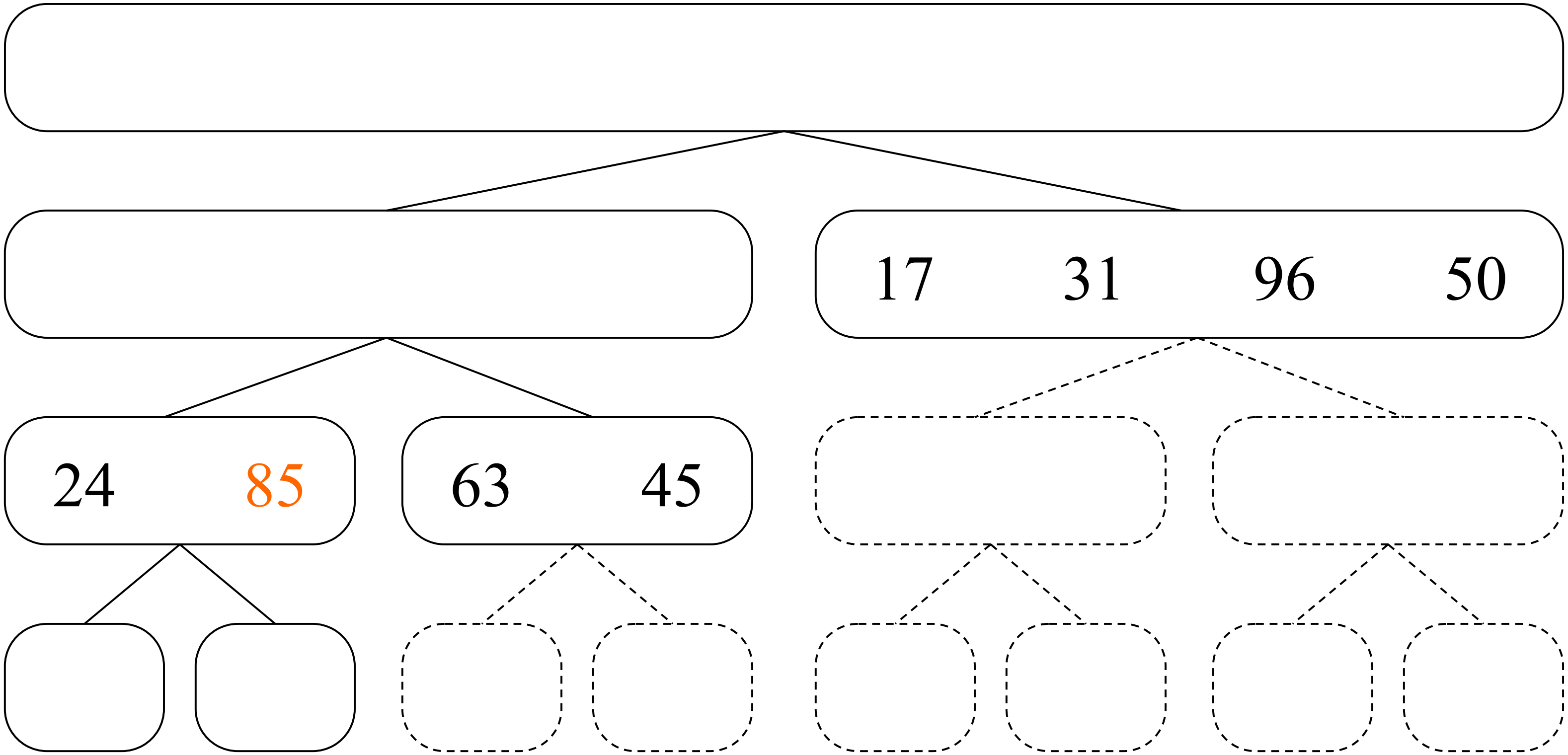
Mergesort: Illustration



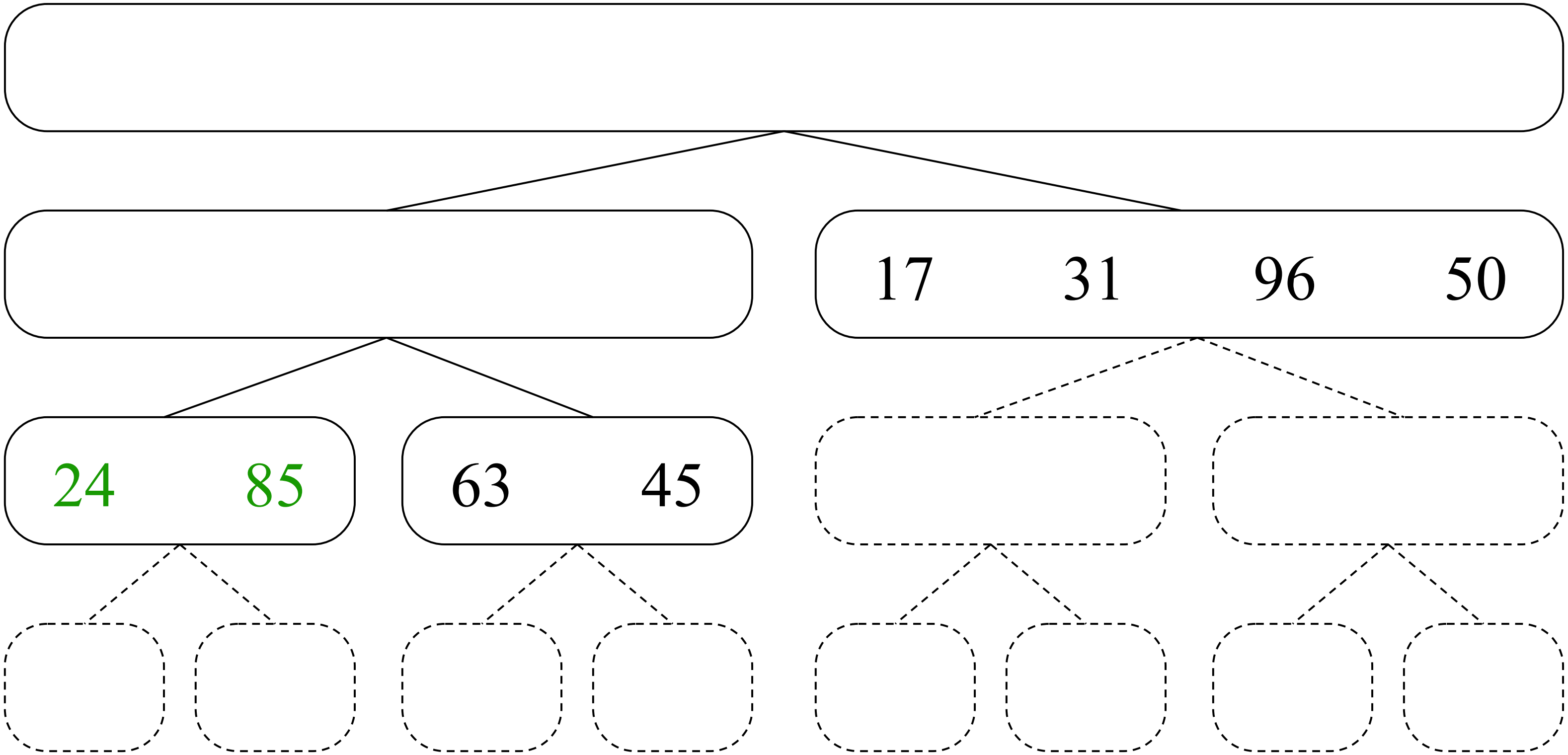
Mergesort: Illustration



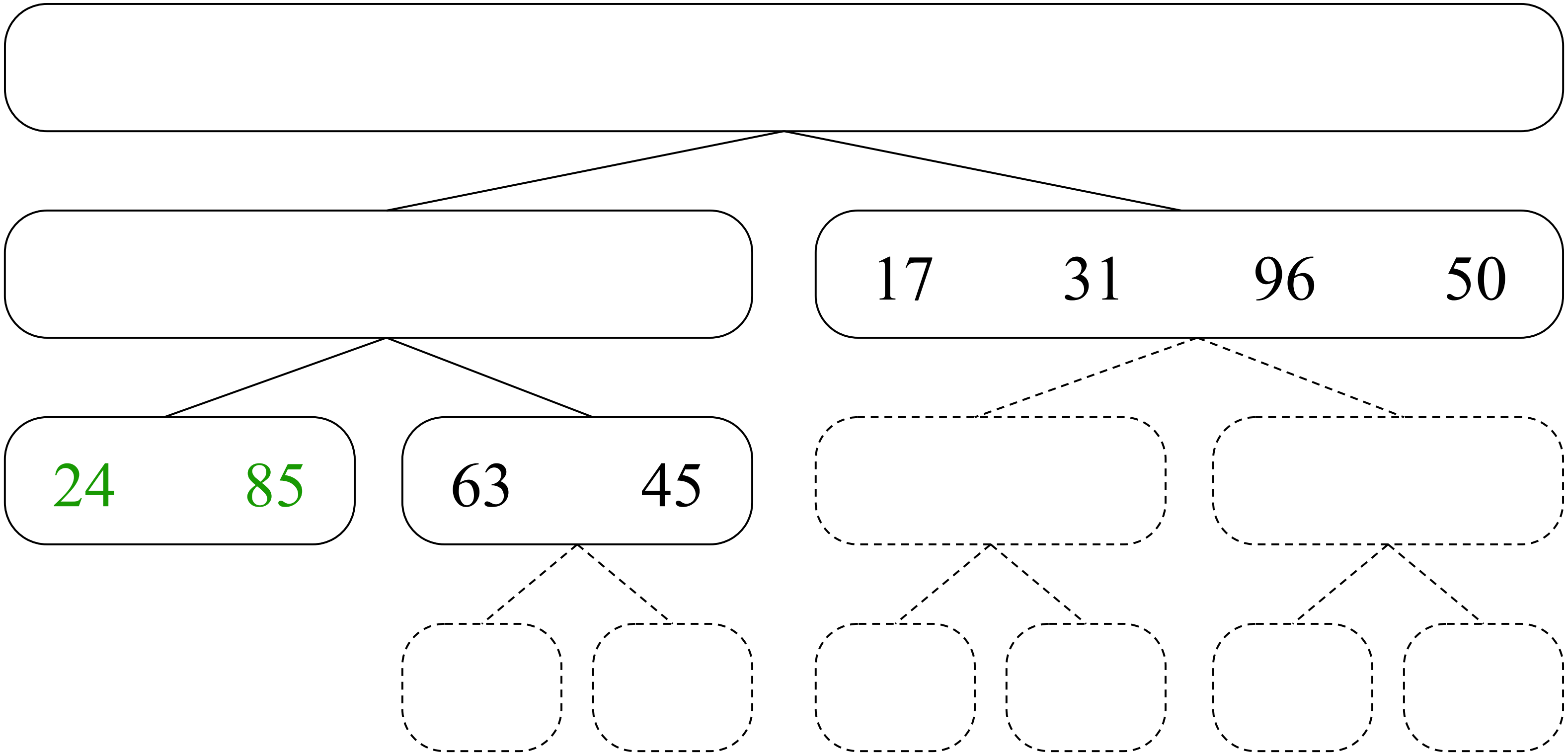
Mergesort: Illustration



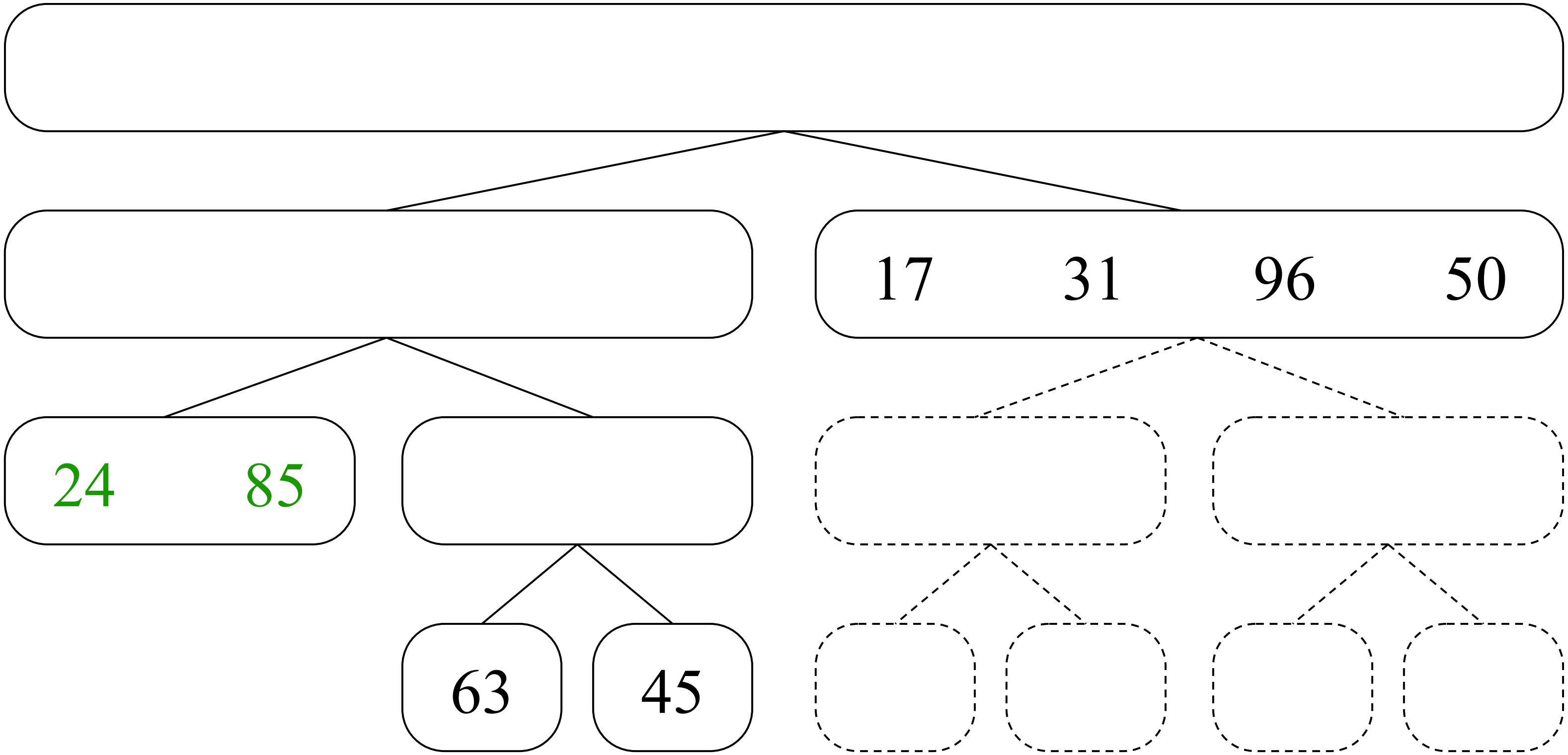
Mergesort: Illustration



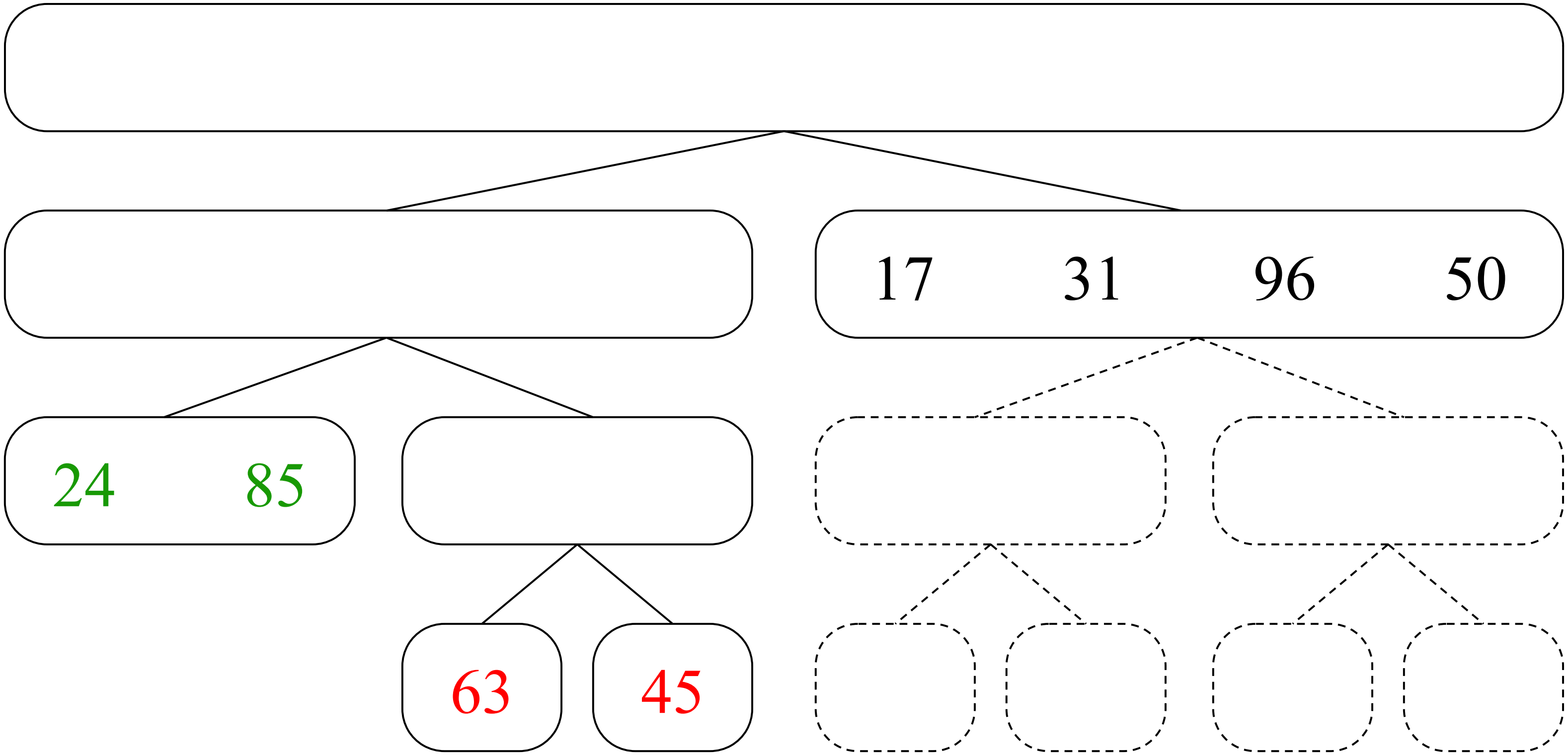
Mergesort: Illustration



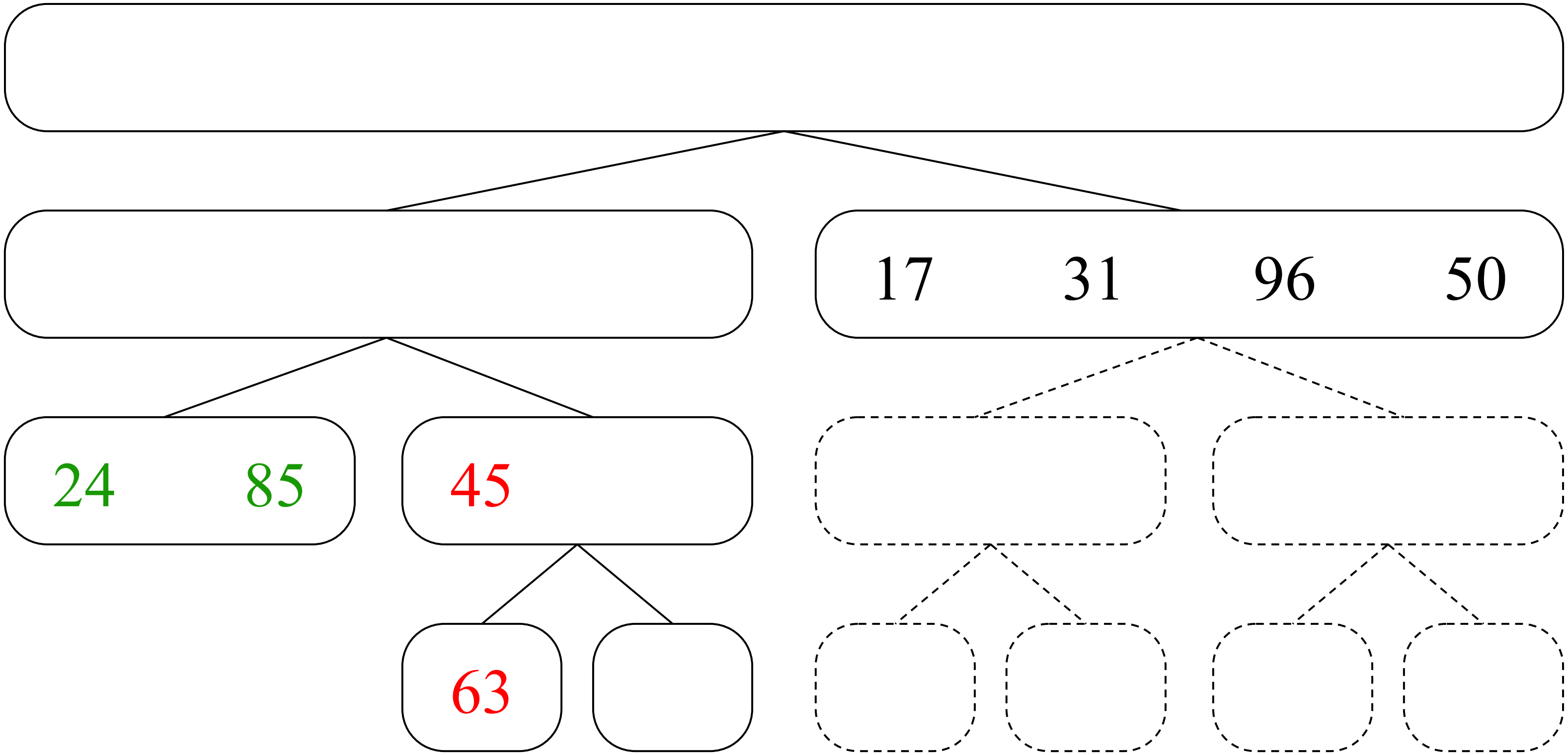
Mergesort: Illustration



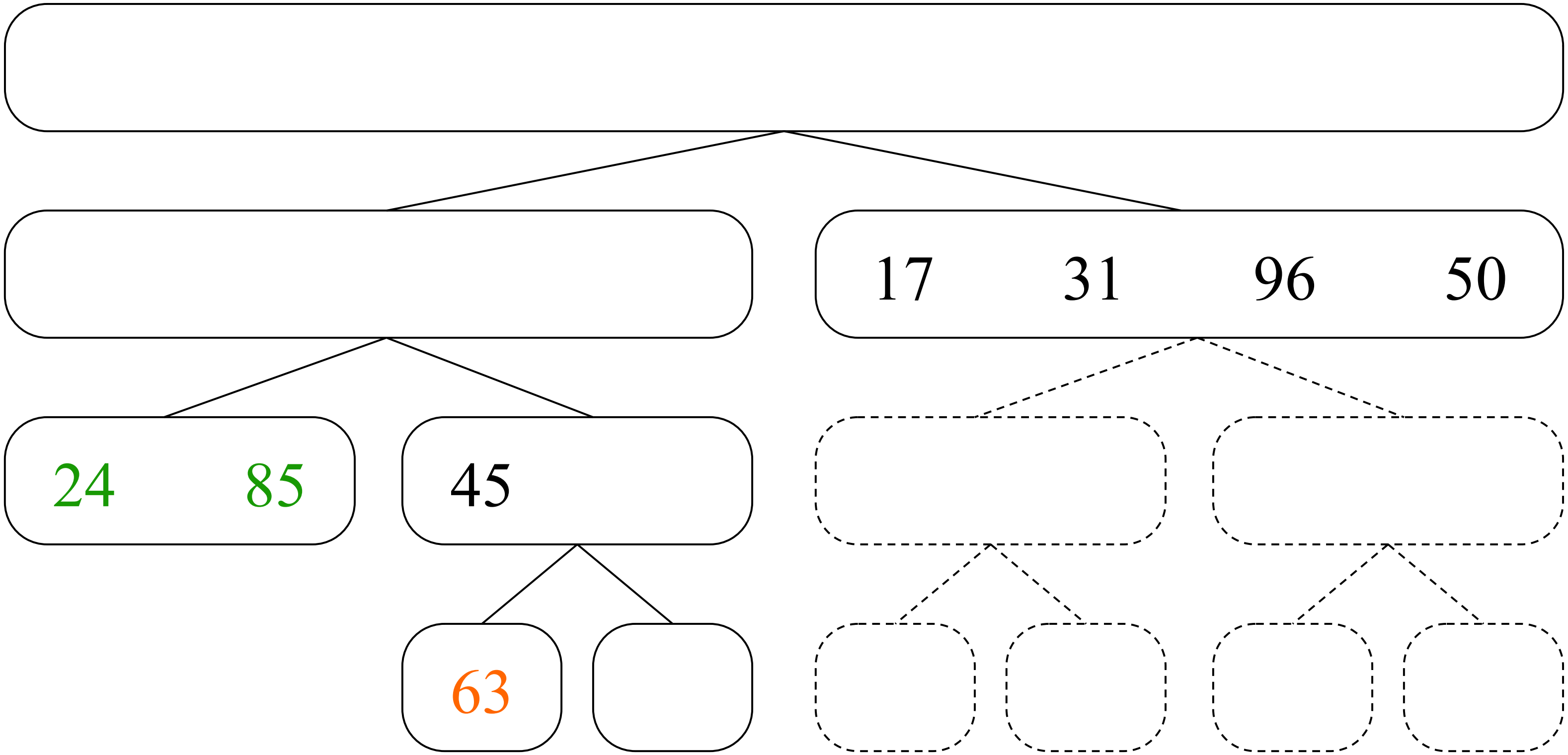
Mergesort: Illustration



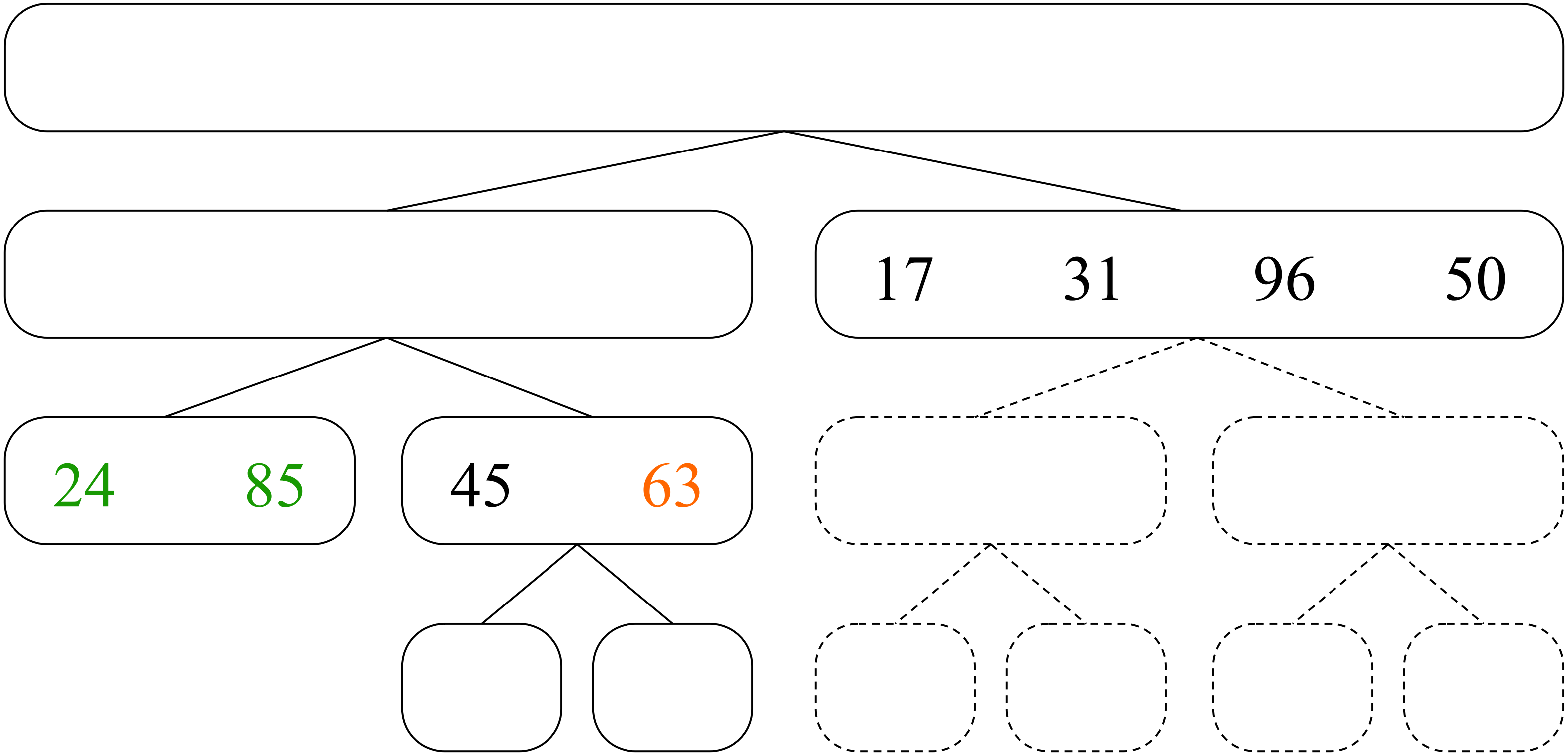
Mergesort: Illustration



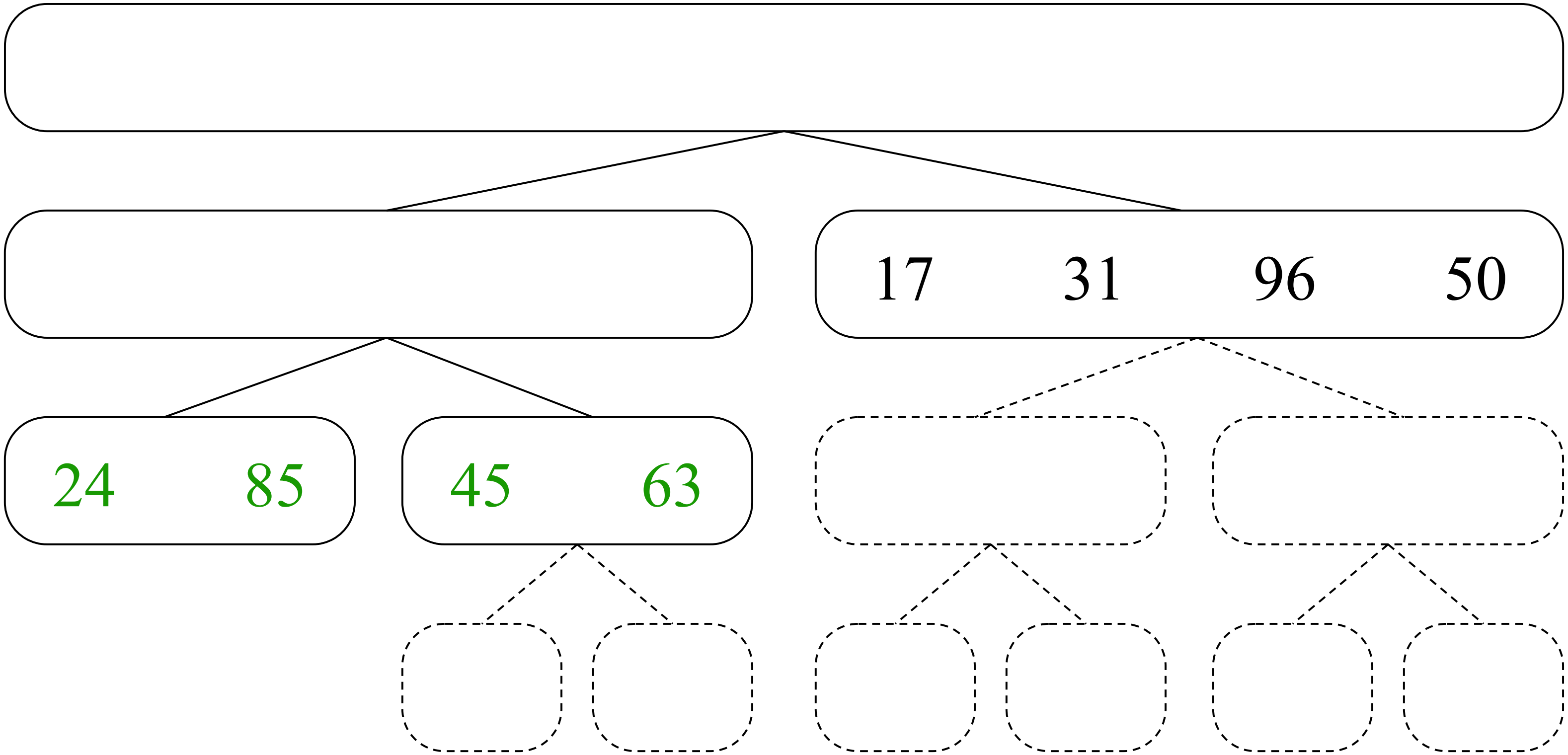
Mergesort: Illustration



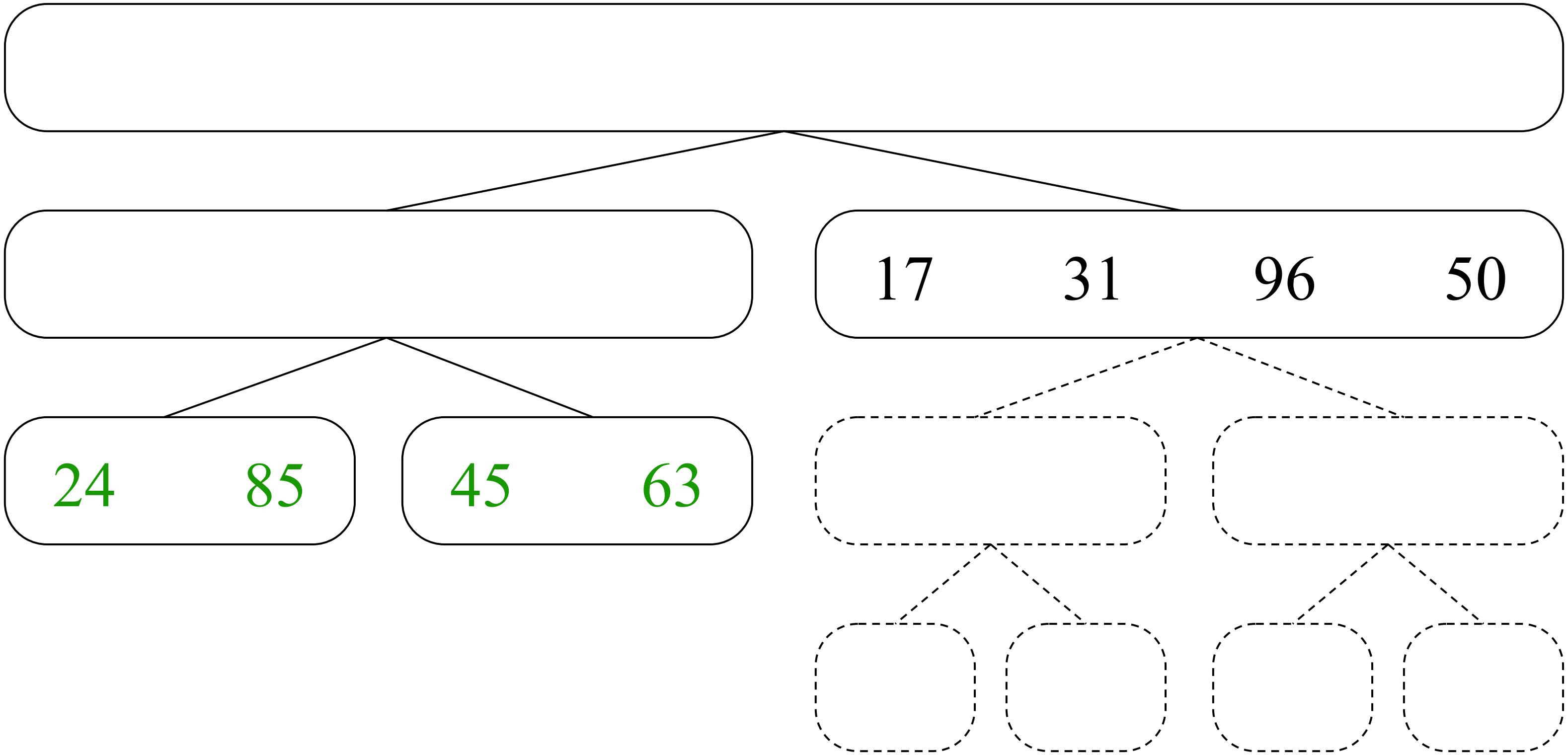
Mergesort: Illustration



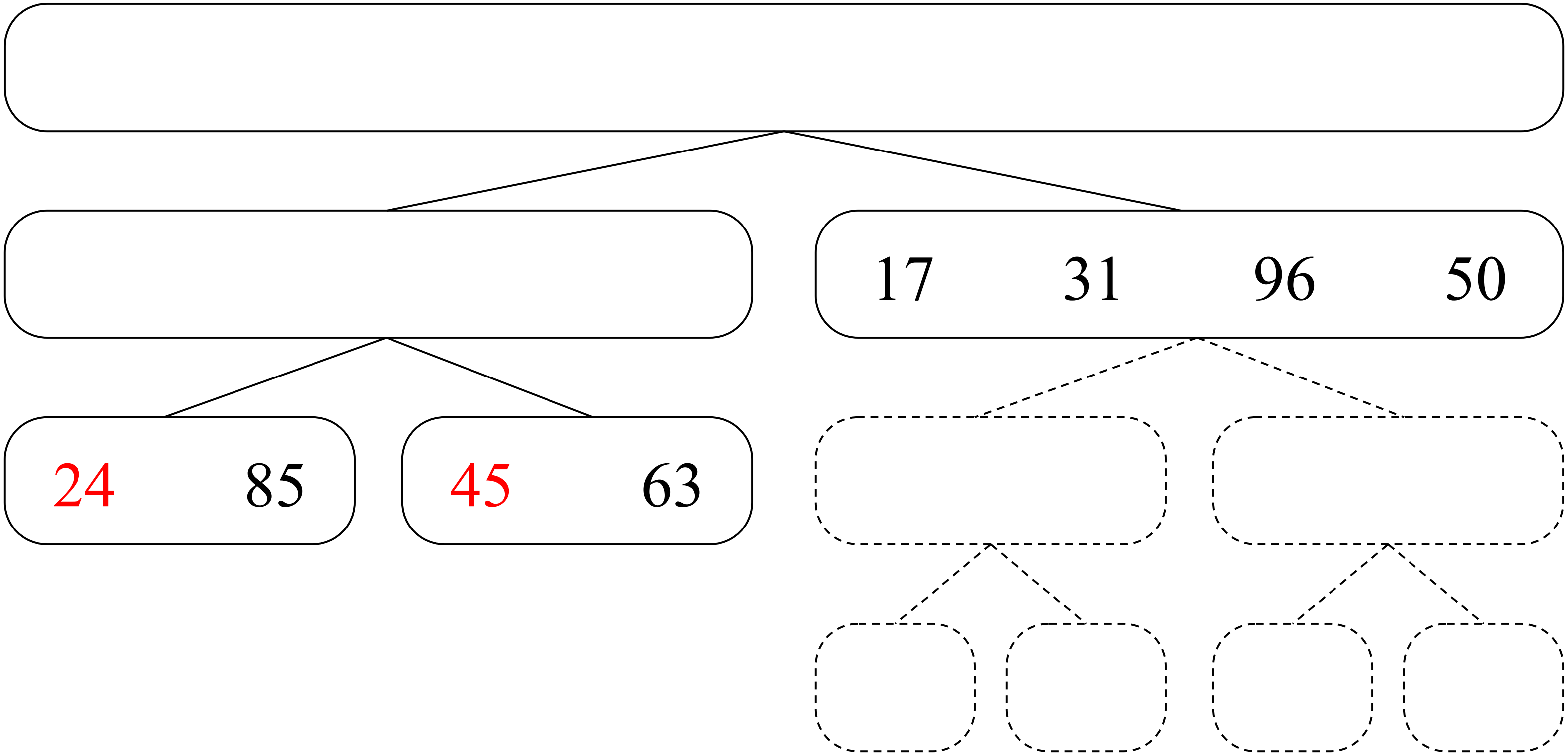
Mergesort: Illustration



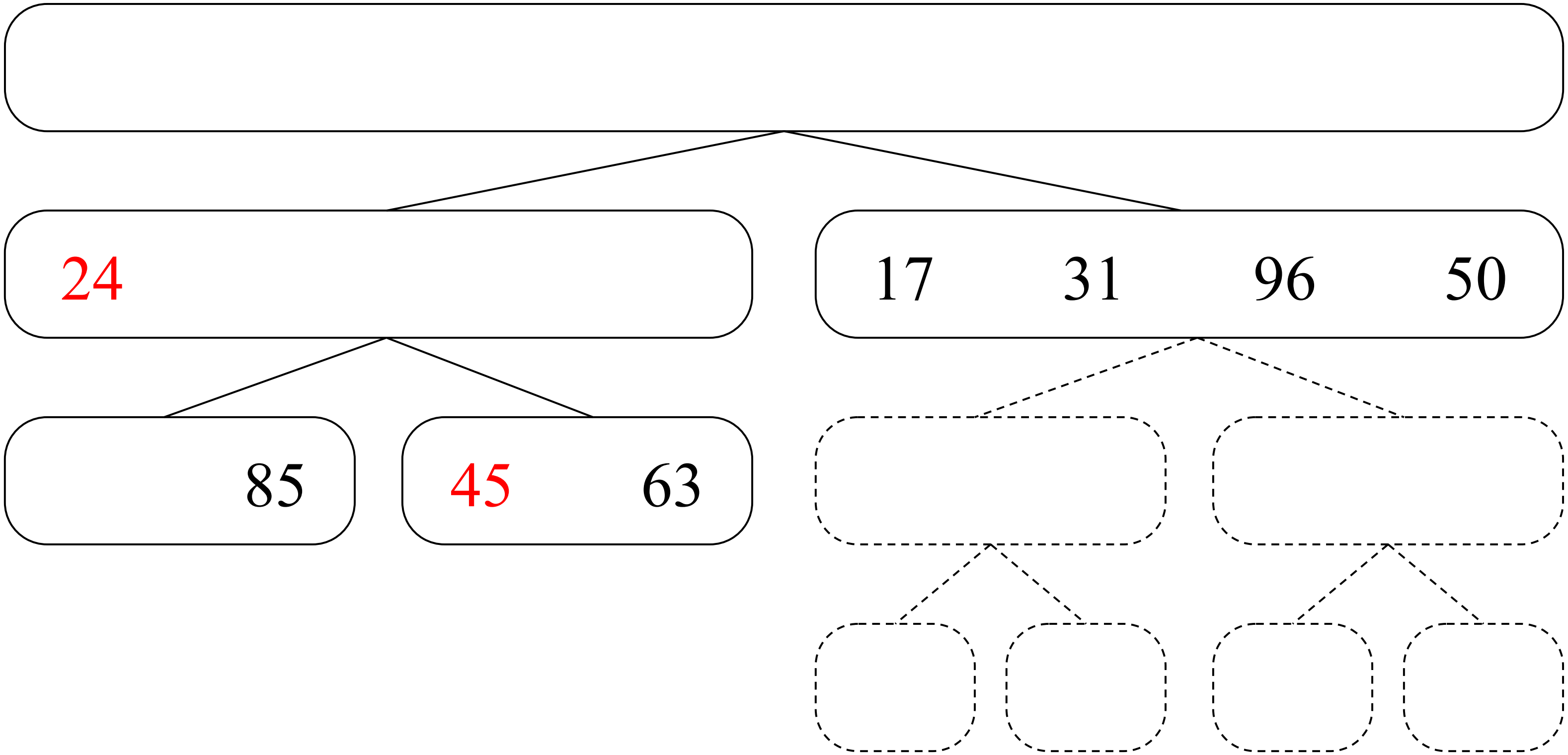
Mergesort: Illustration



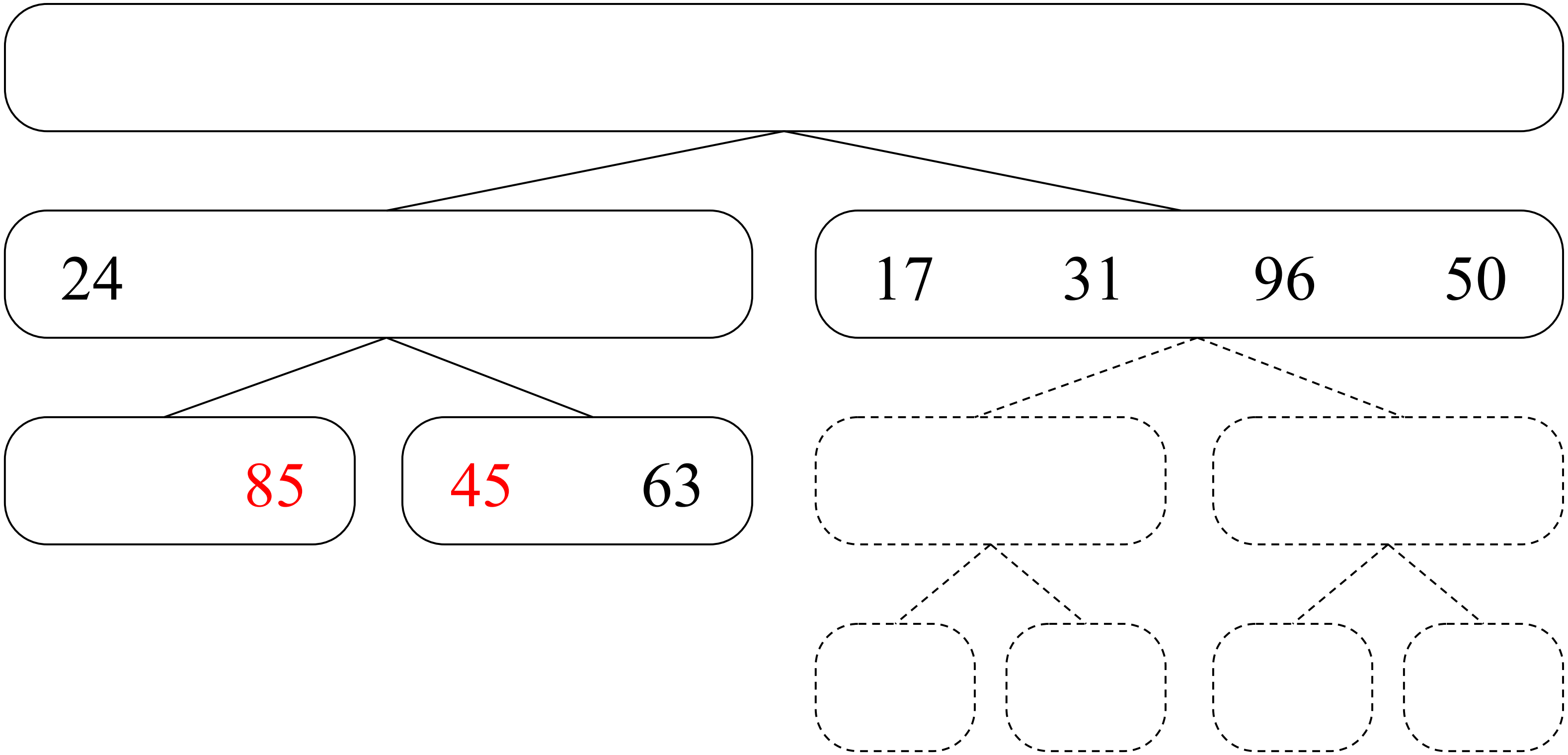
Mergesort: Illustration



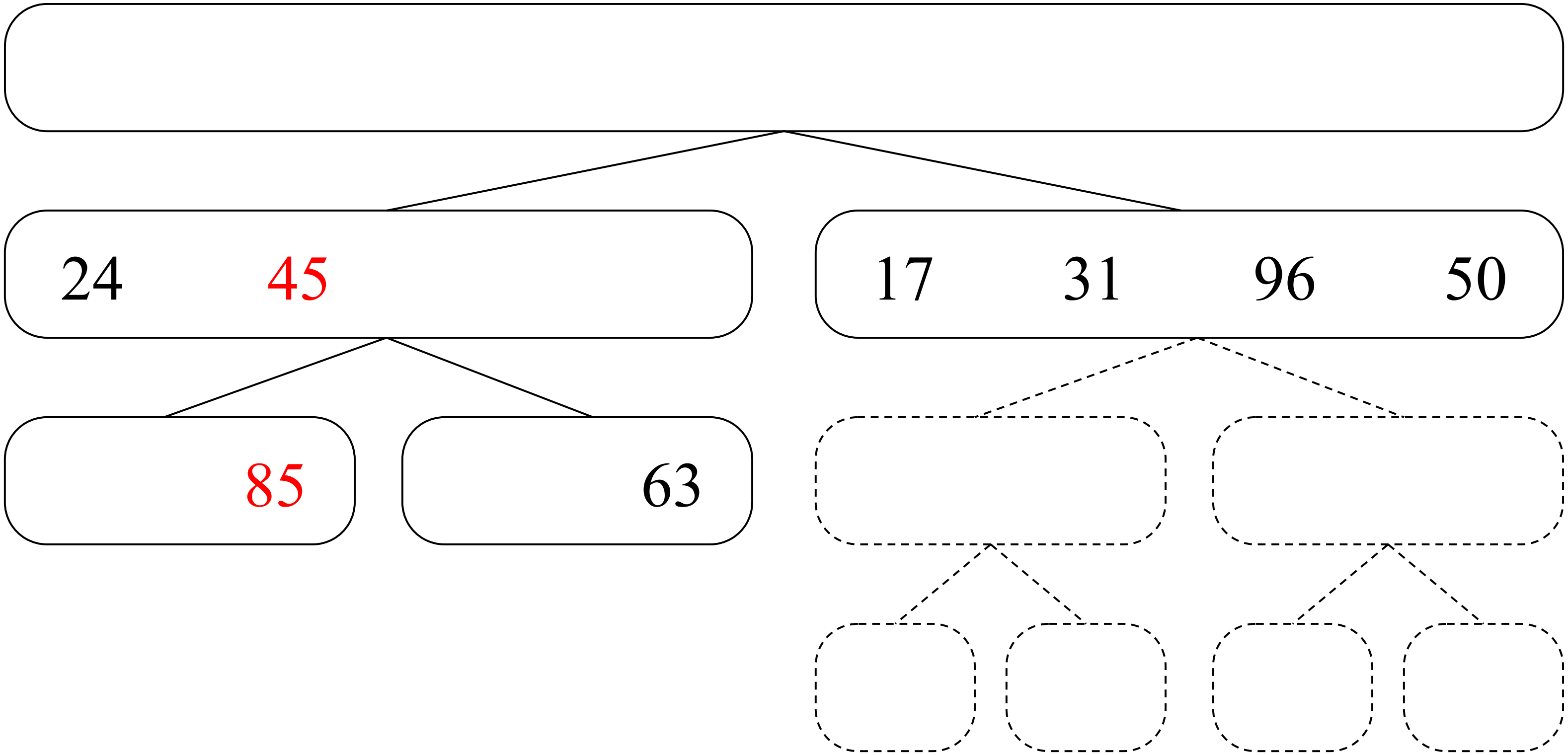
Mergesort: Illustration



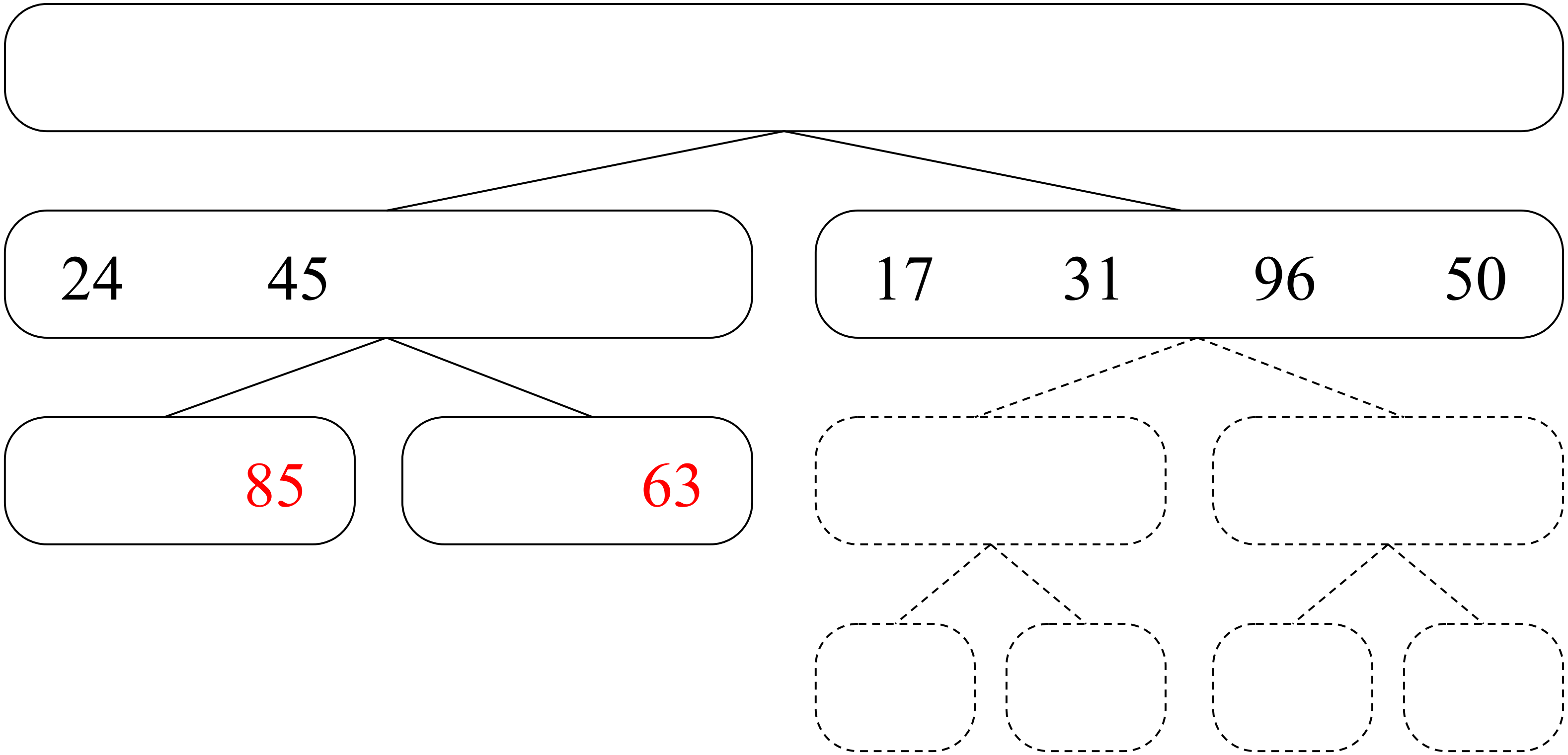
Mergesort: Illustration



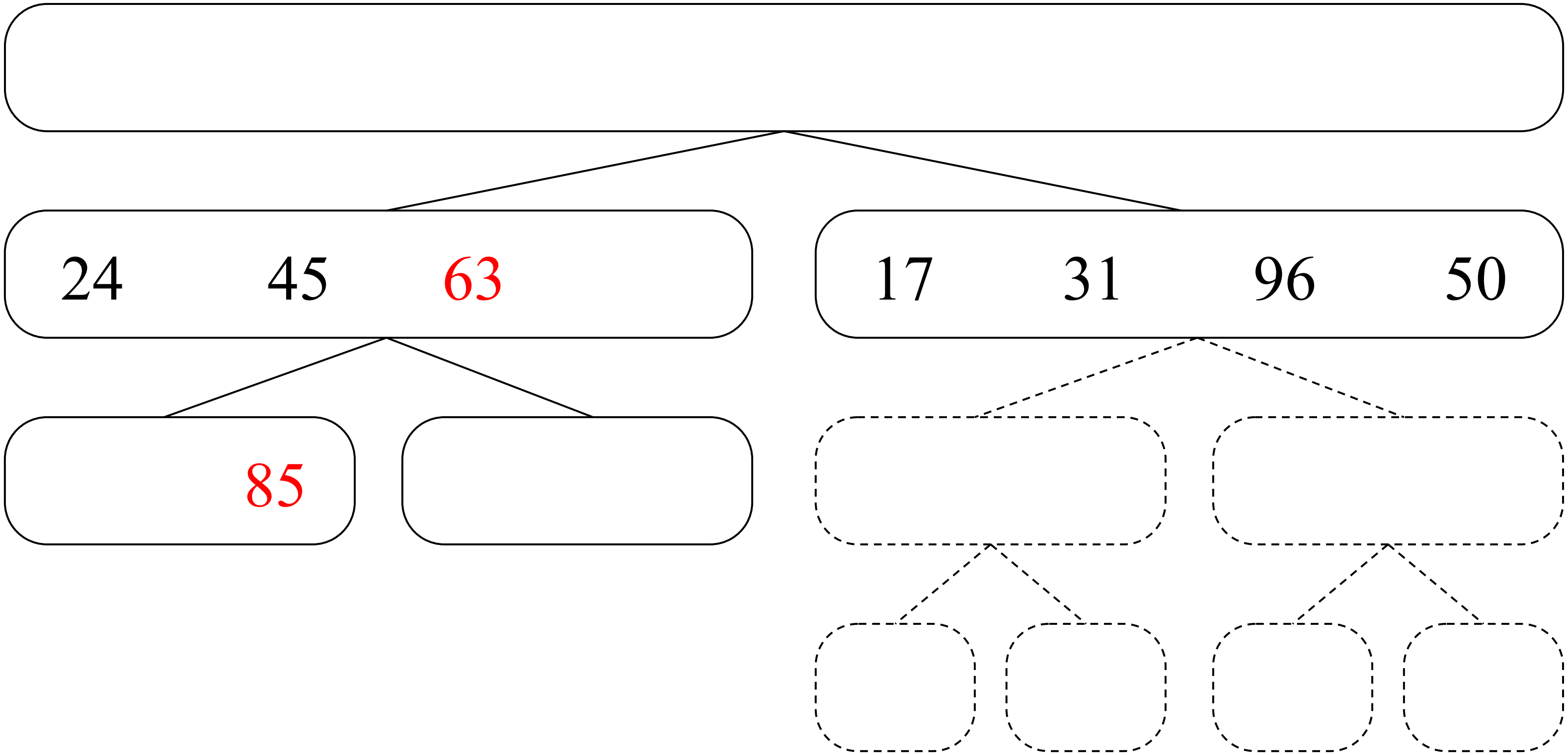
Mergesort: Illustration



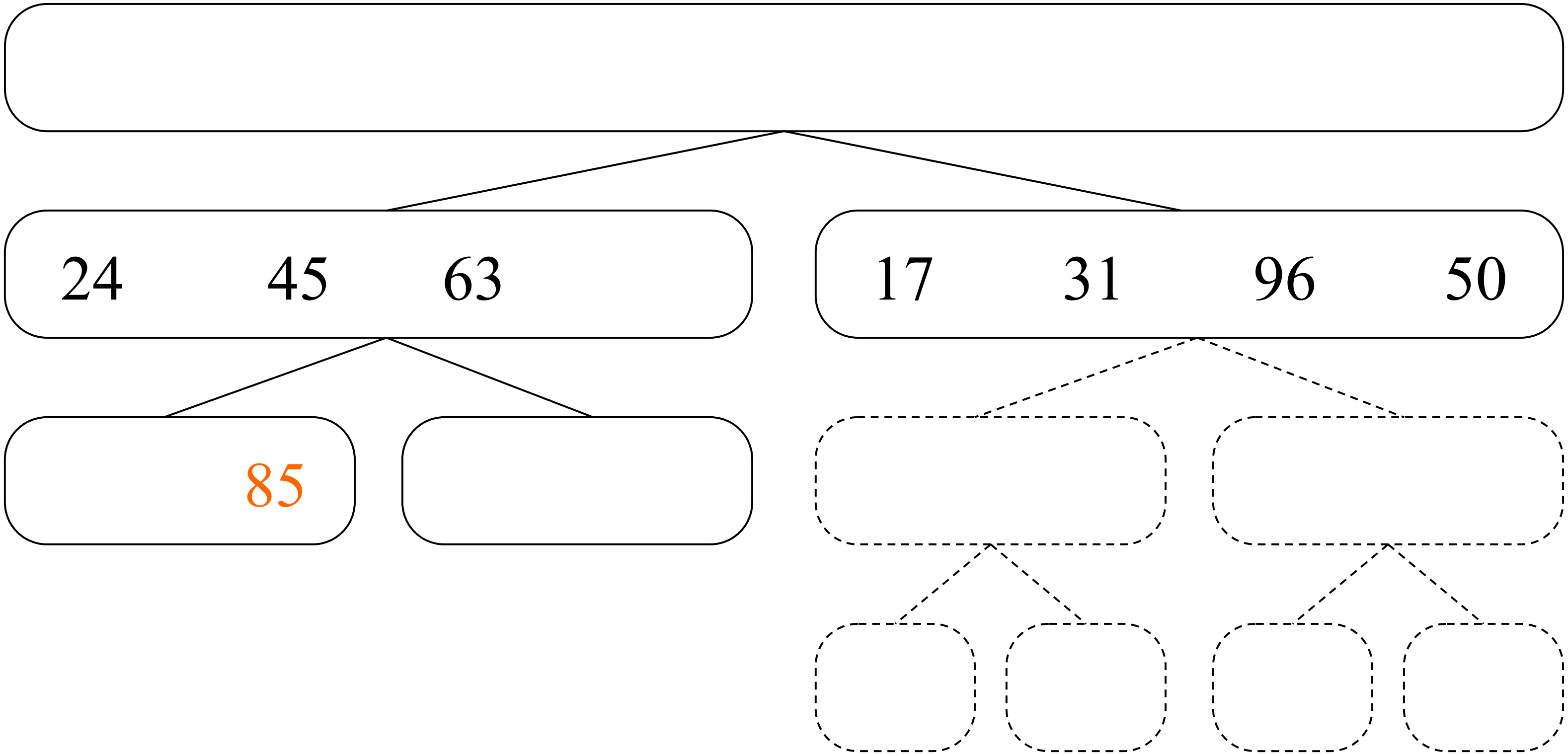
Mergesort: Illustration



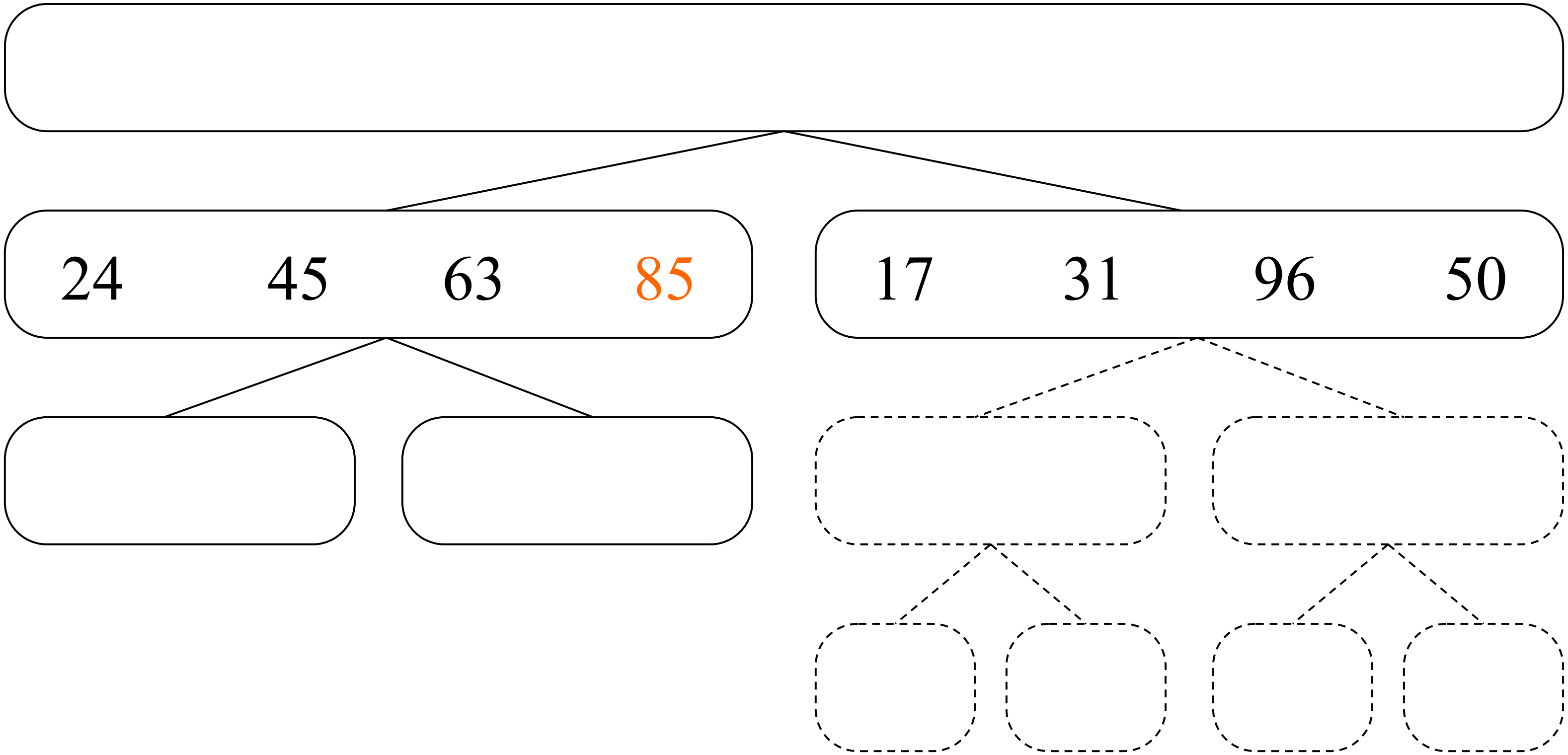
Mergesort: Illustration



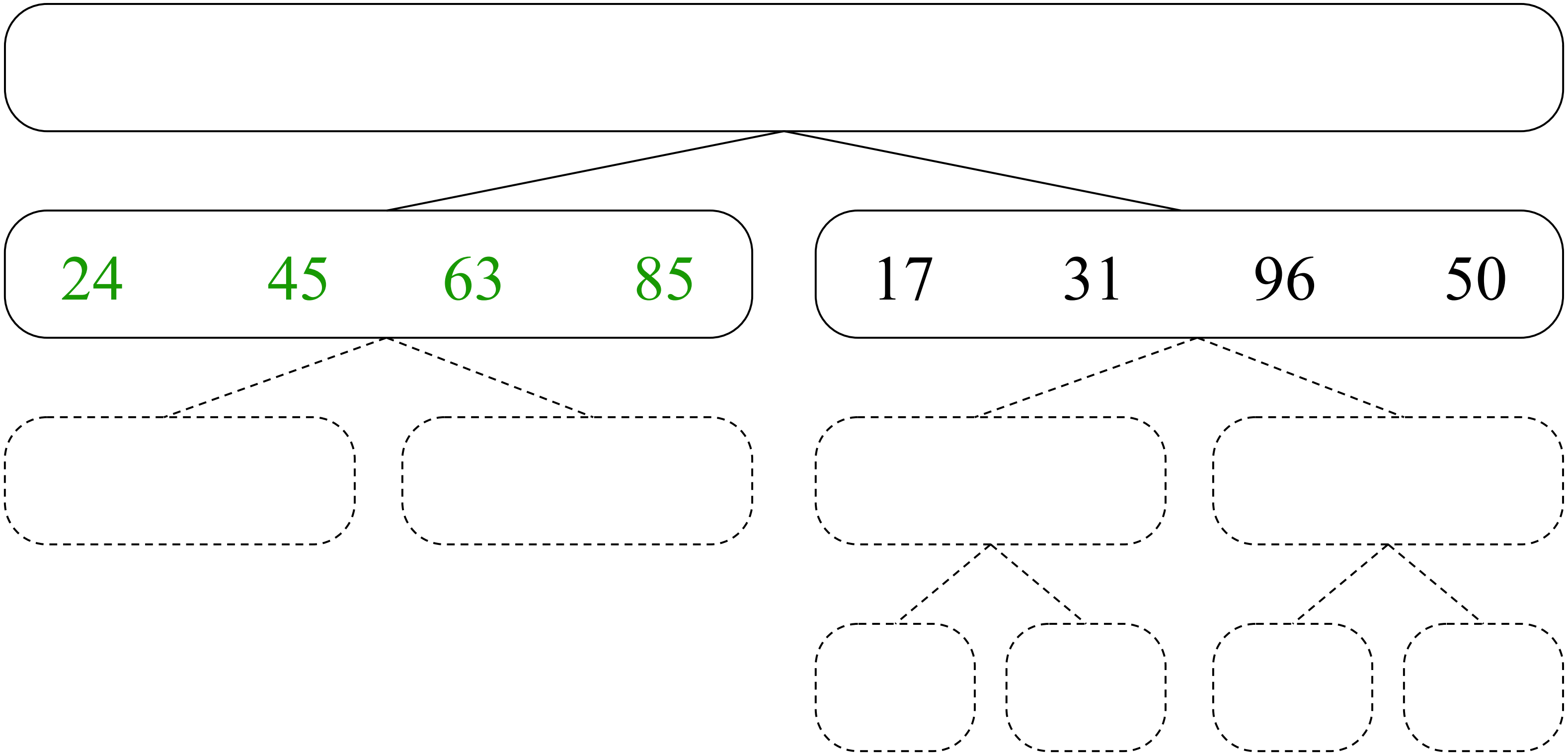
Mergesort: Illustration



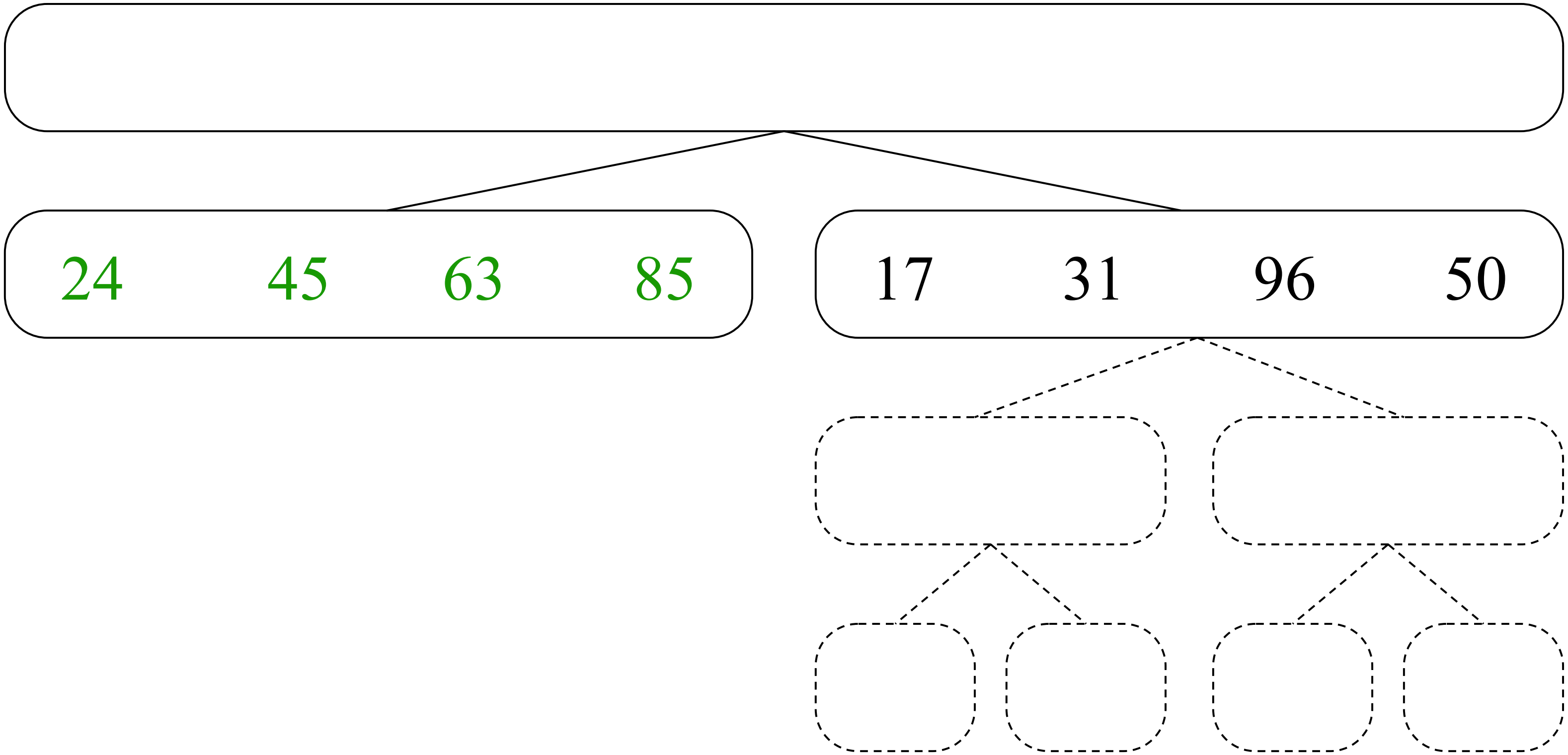
Mergesort: Illustration



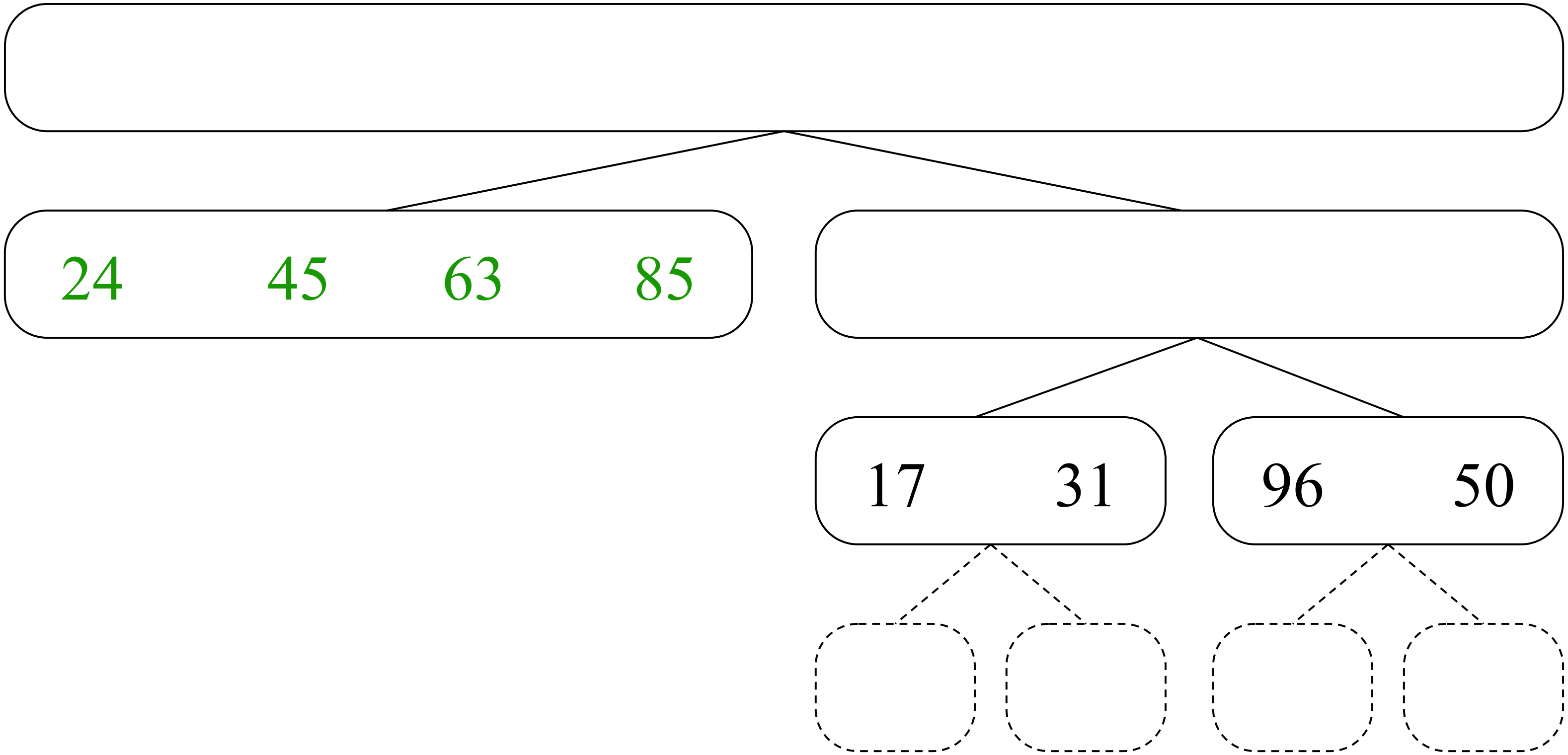
Mergesort: Illustration



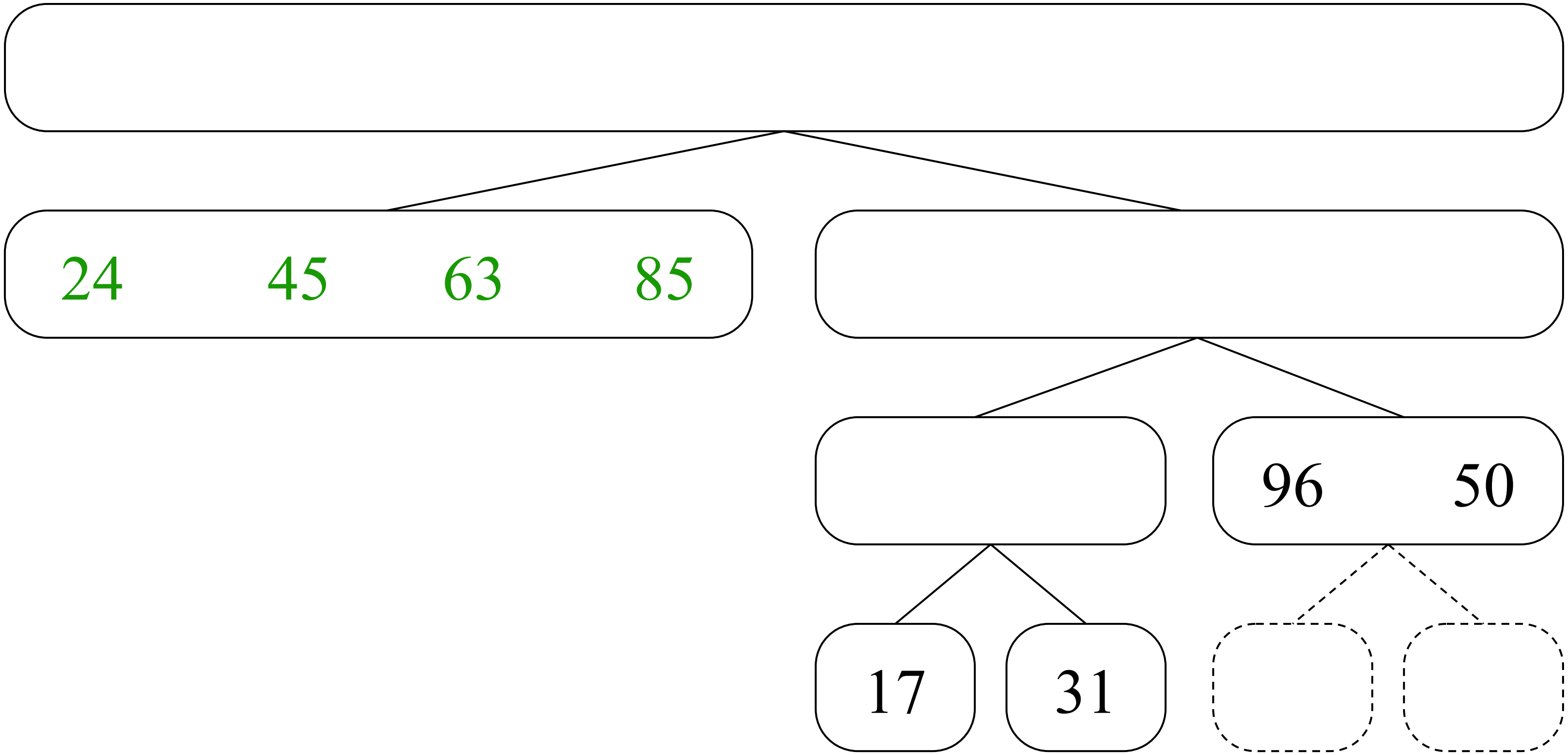
Mergesort: Illustration



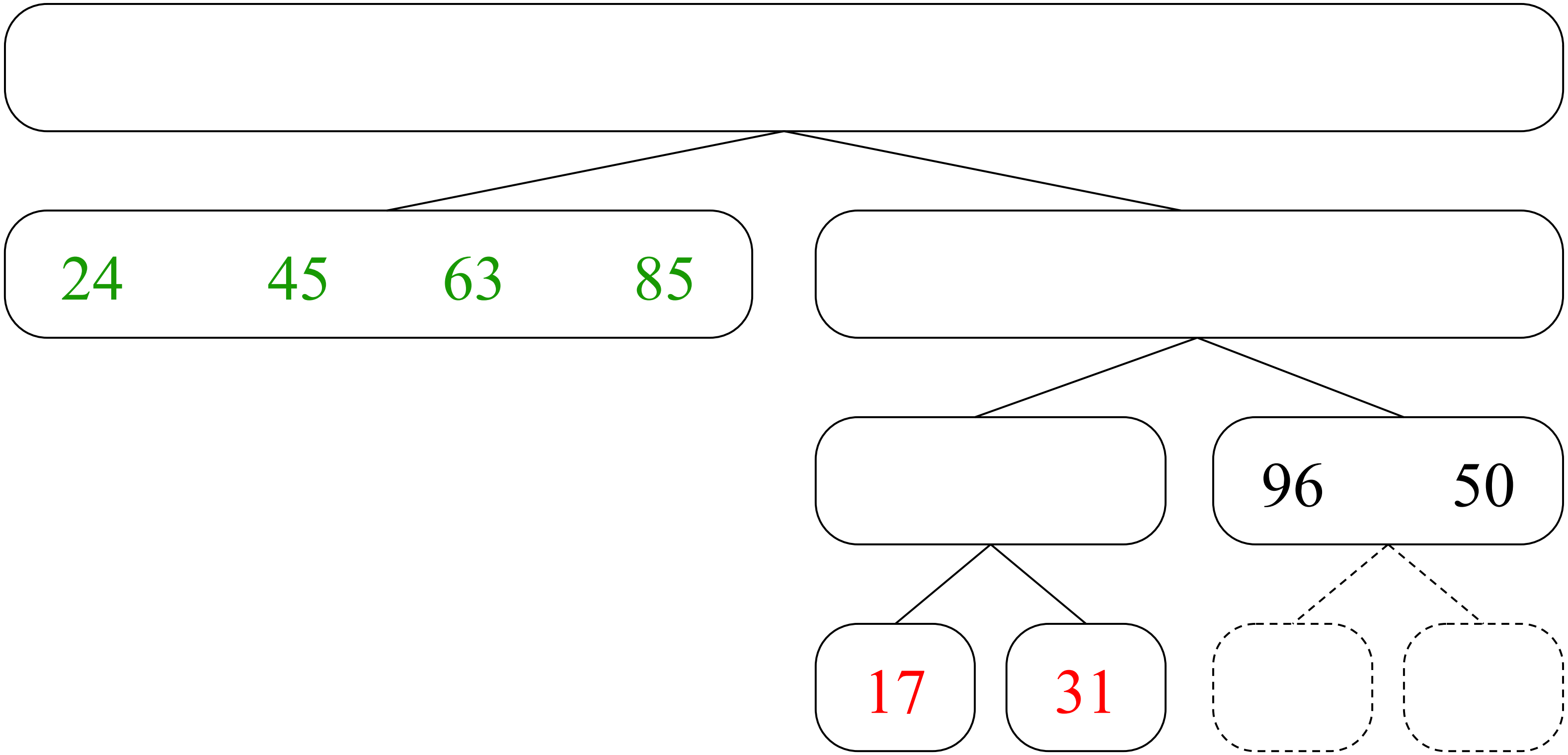
Mergesort: Illustration



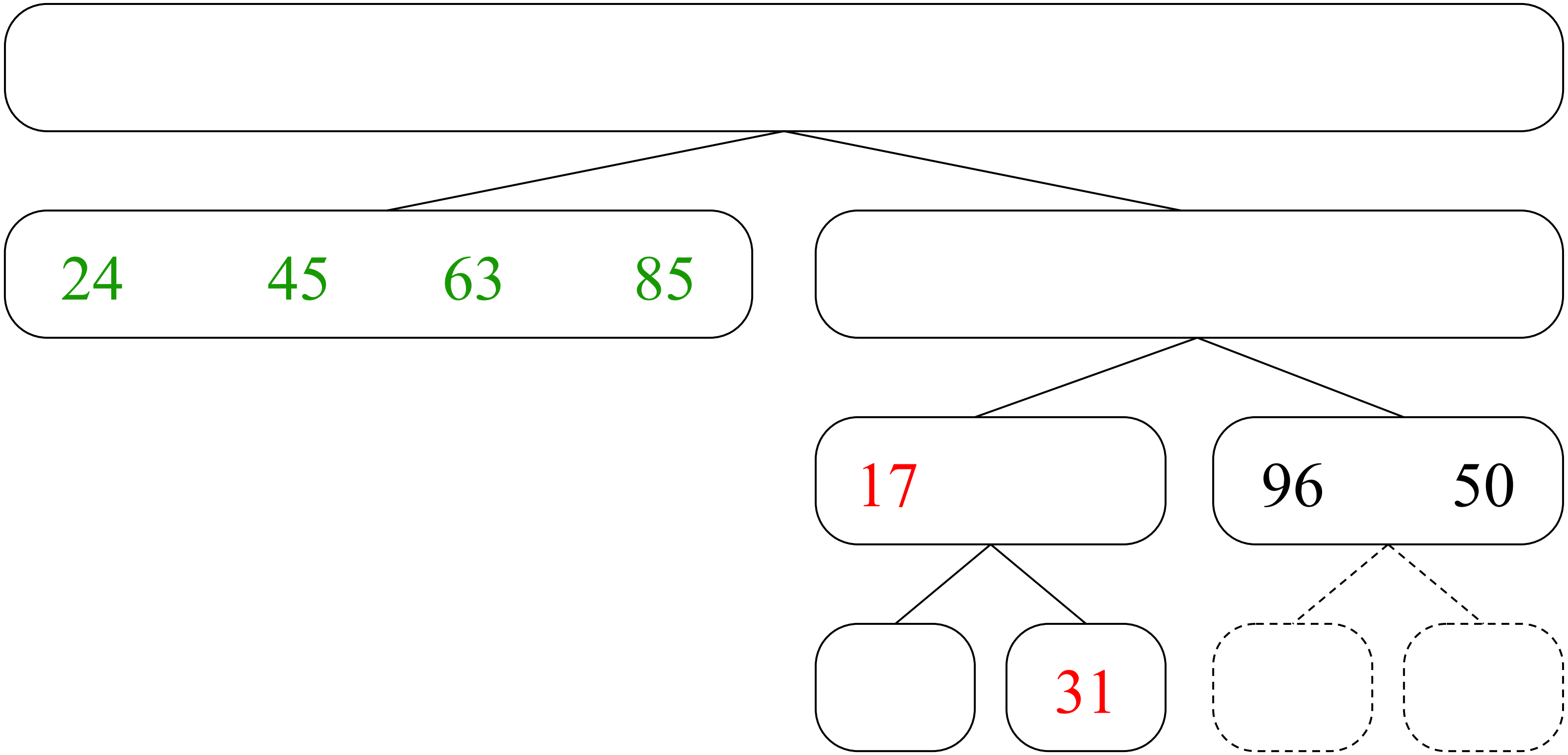
Mergesort: Illustration



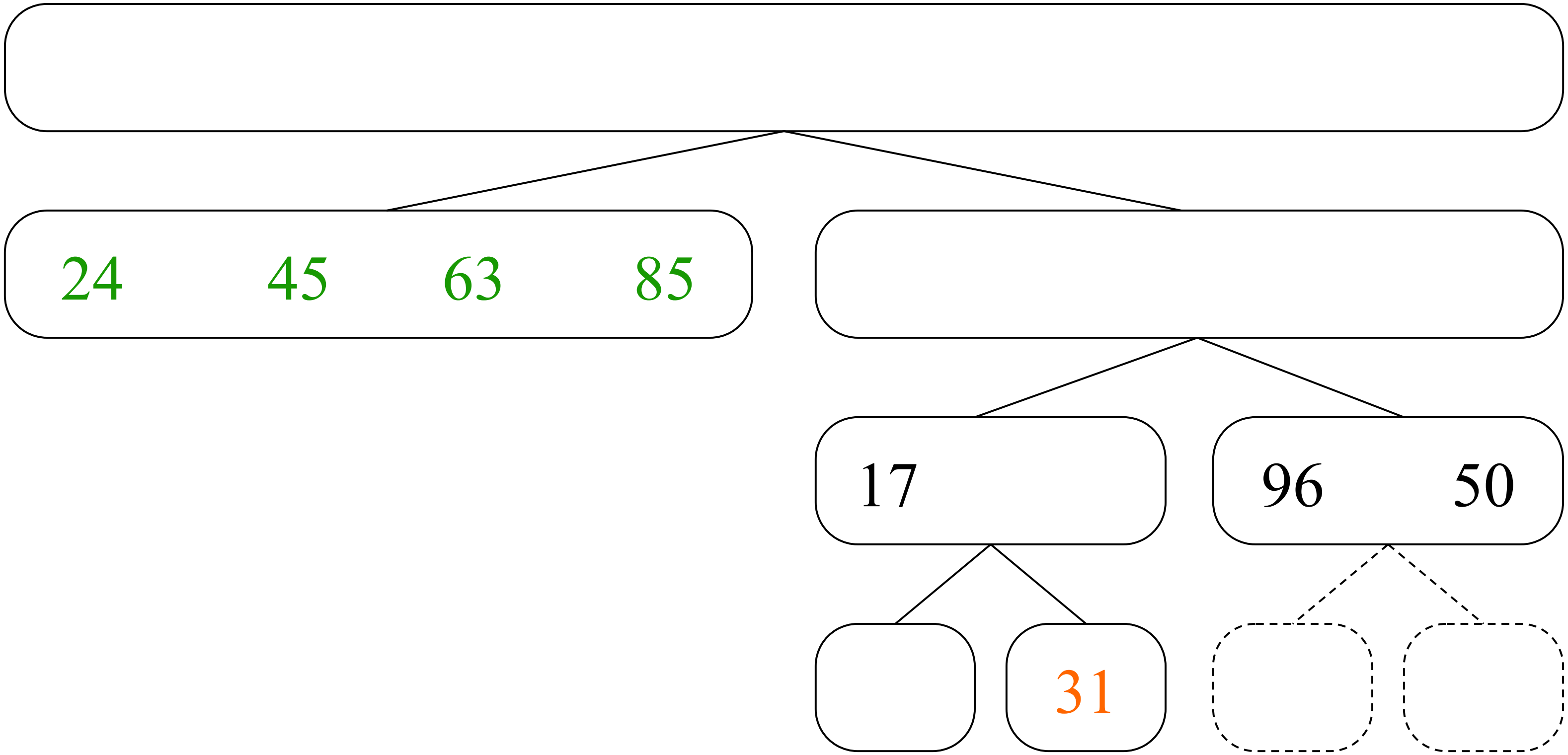
Mergesort: Illustration



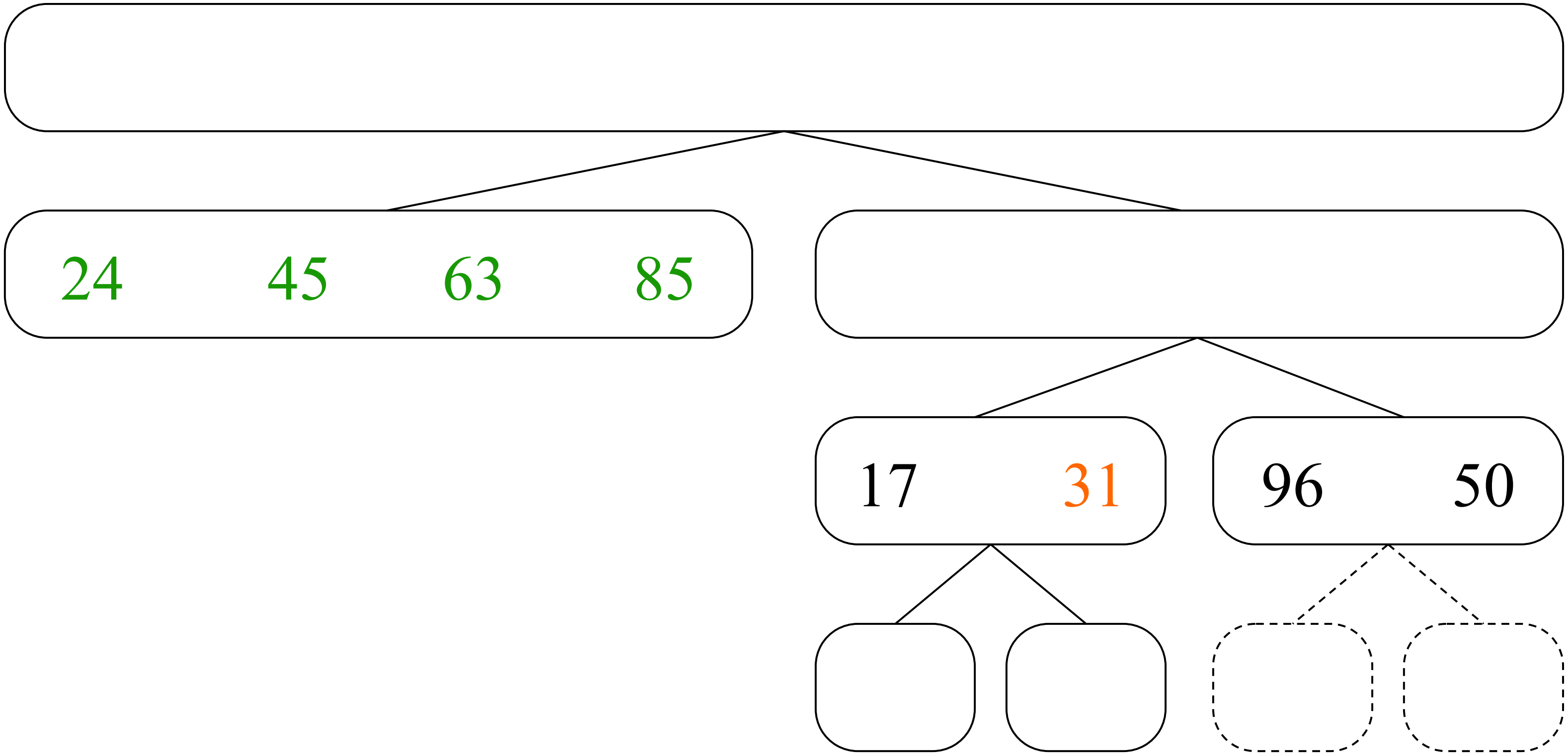
Mergesort: Illustration



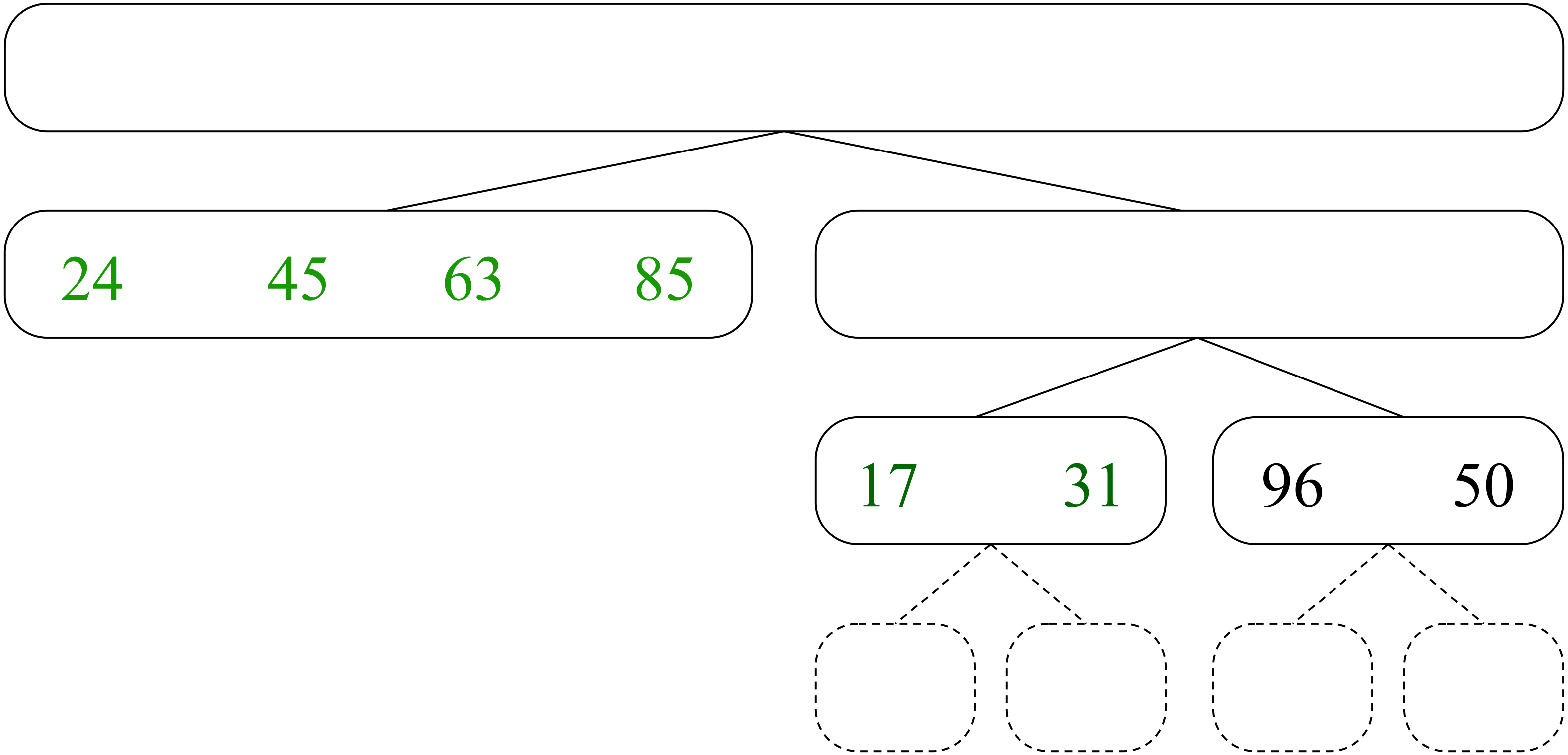
Mergesort: Illustration



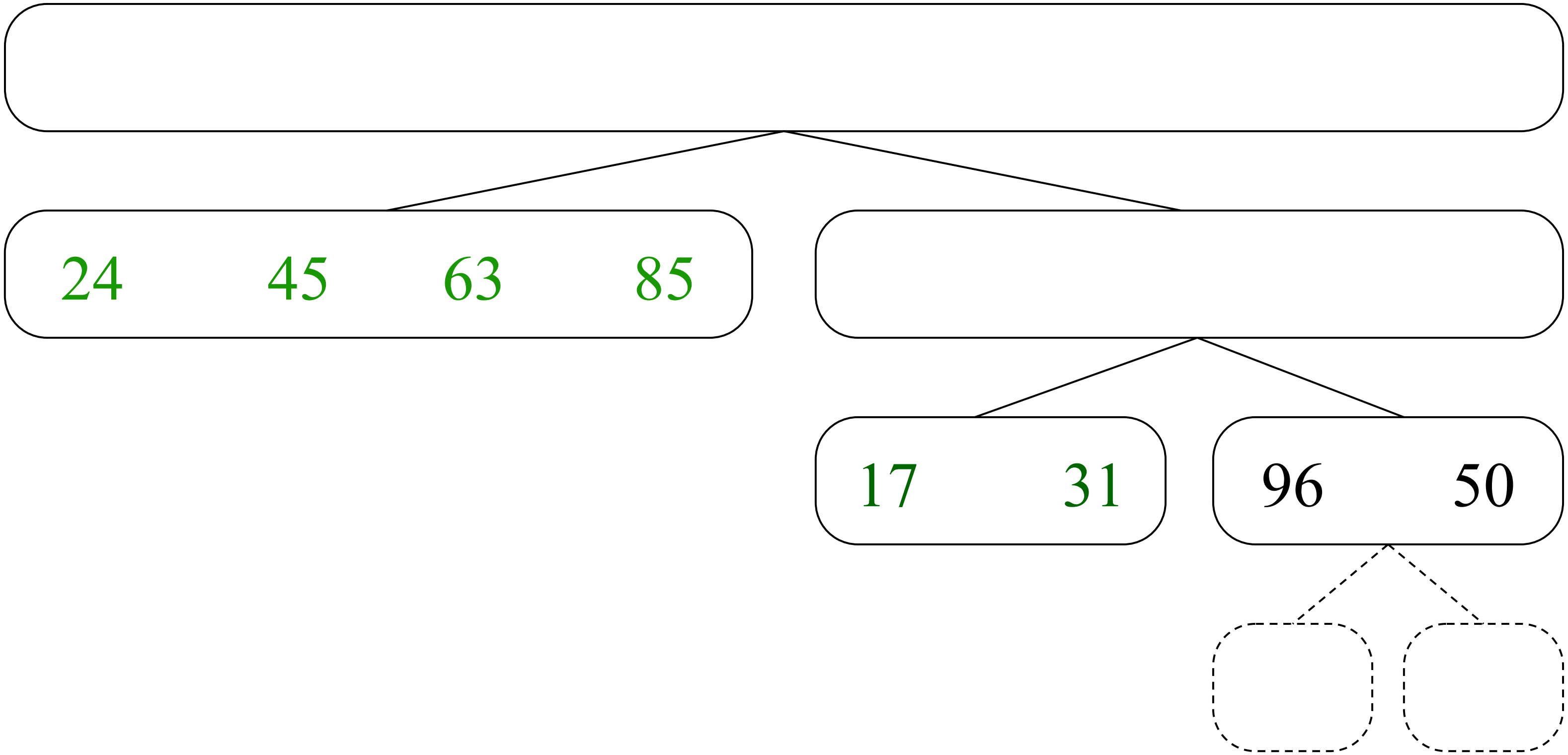
Mergesort: Illustration



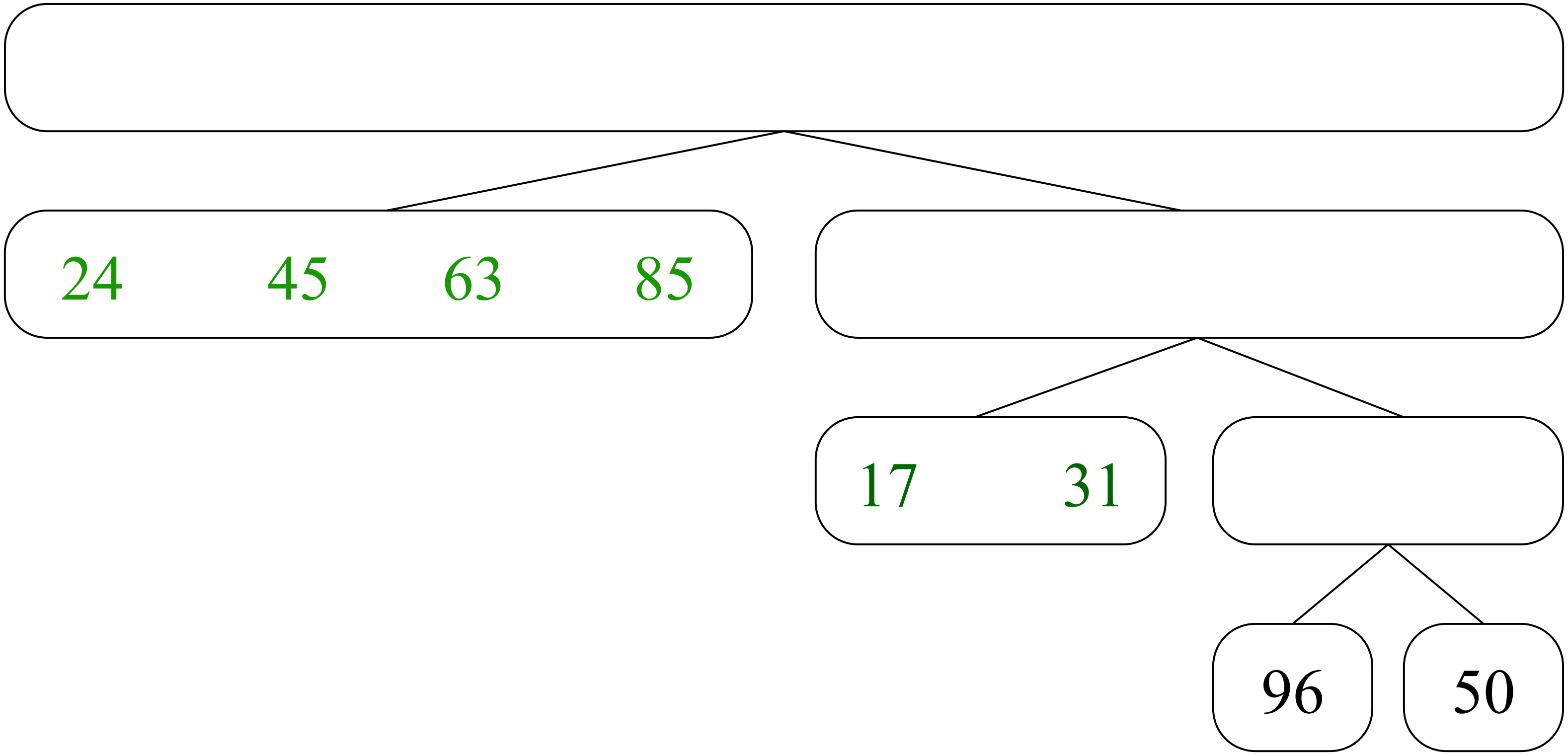
Mergesort: Illustration



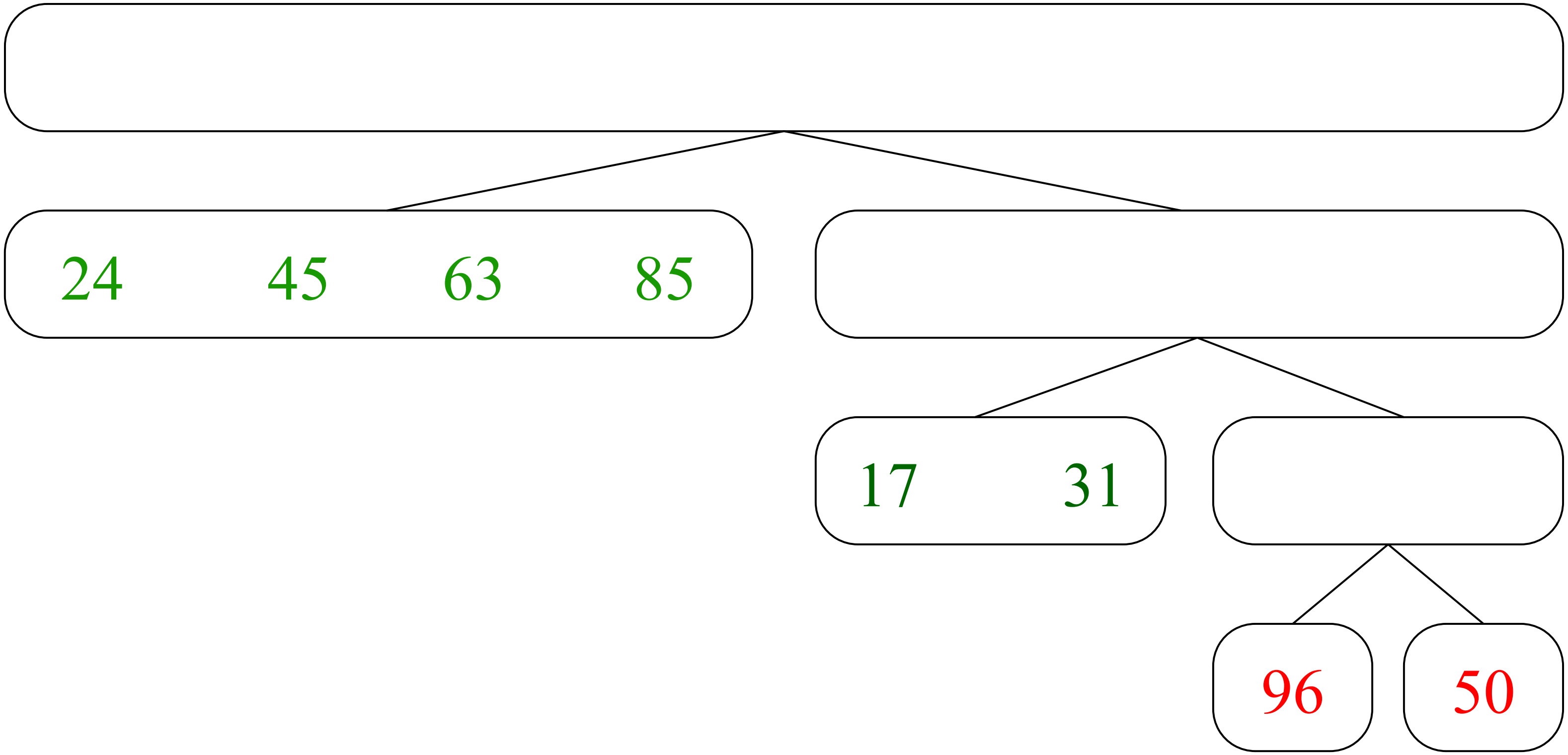
Mergesort: Illustration



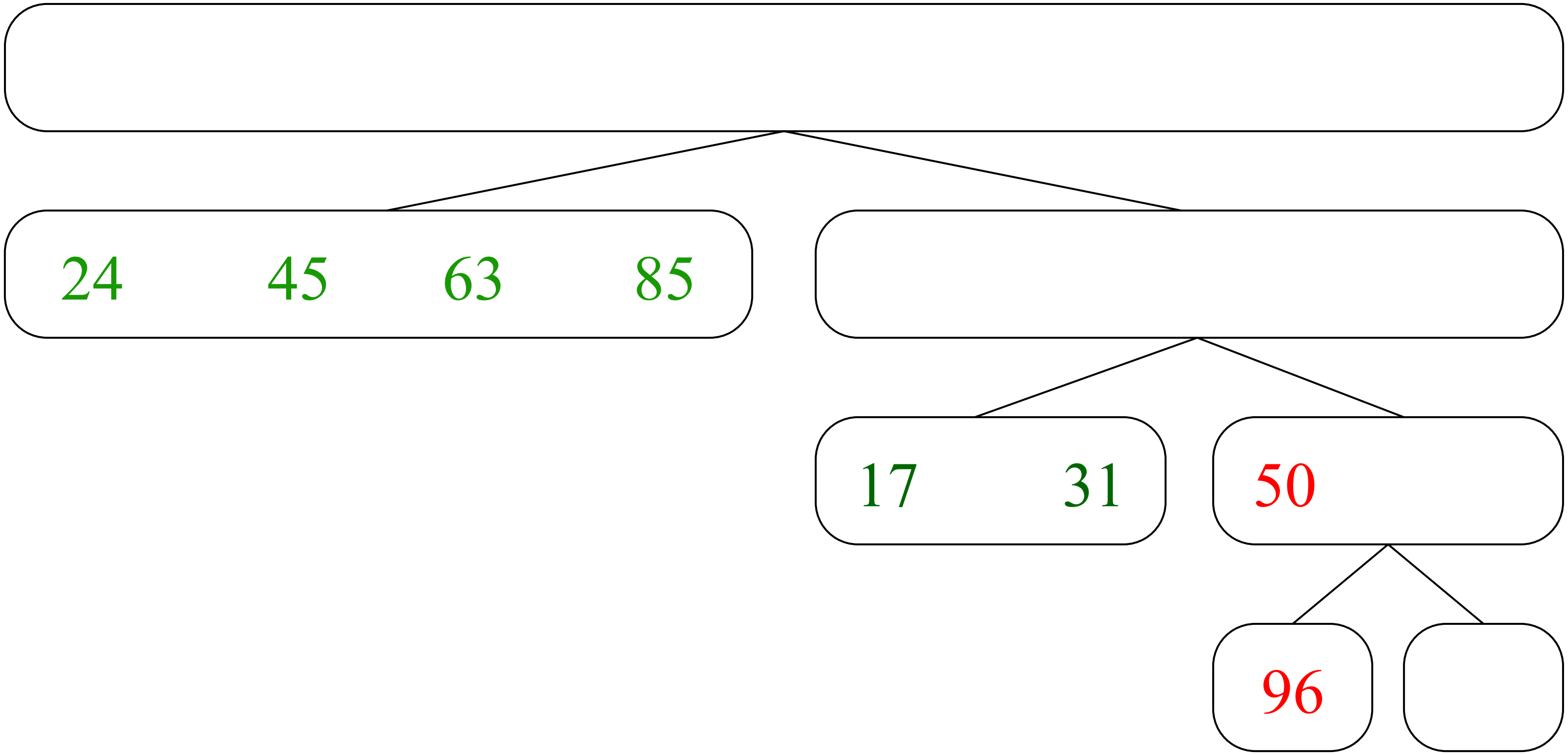
Mergesort: Illustration



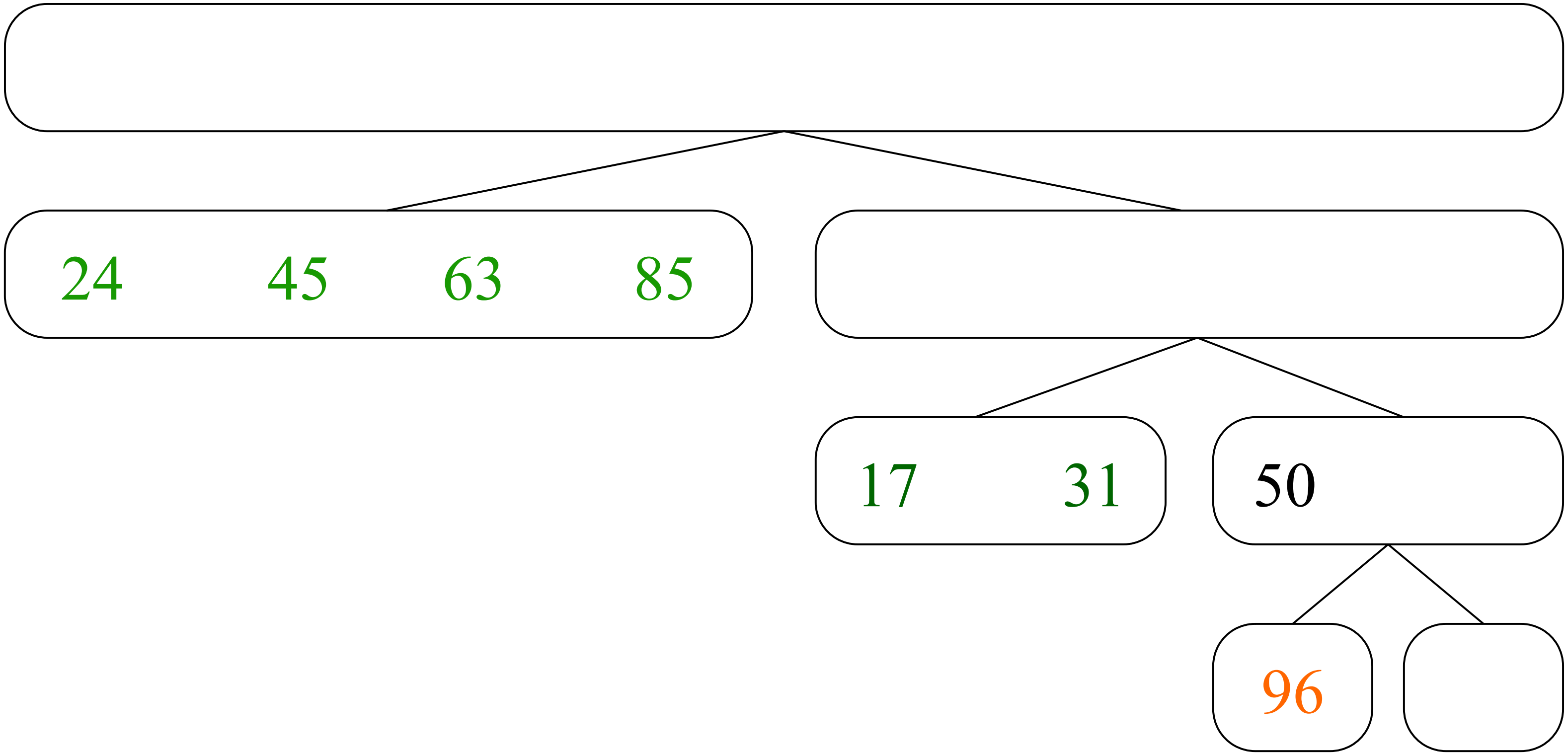
Mergesort: Illustration



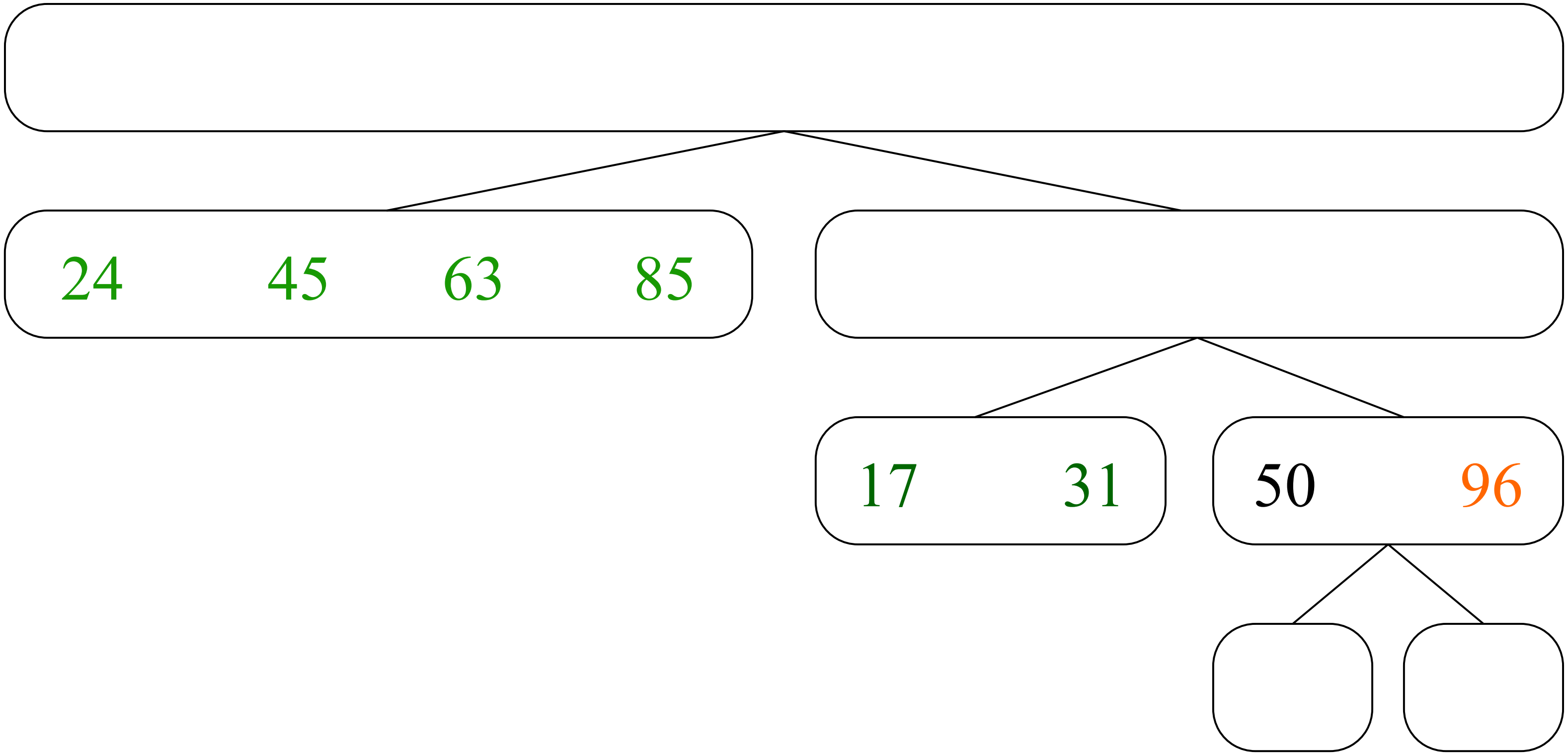
Mergesort: Illustration



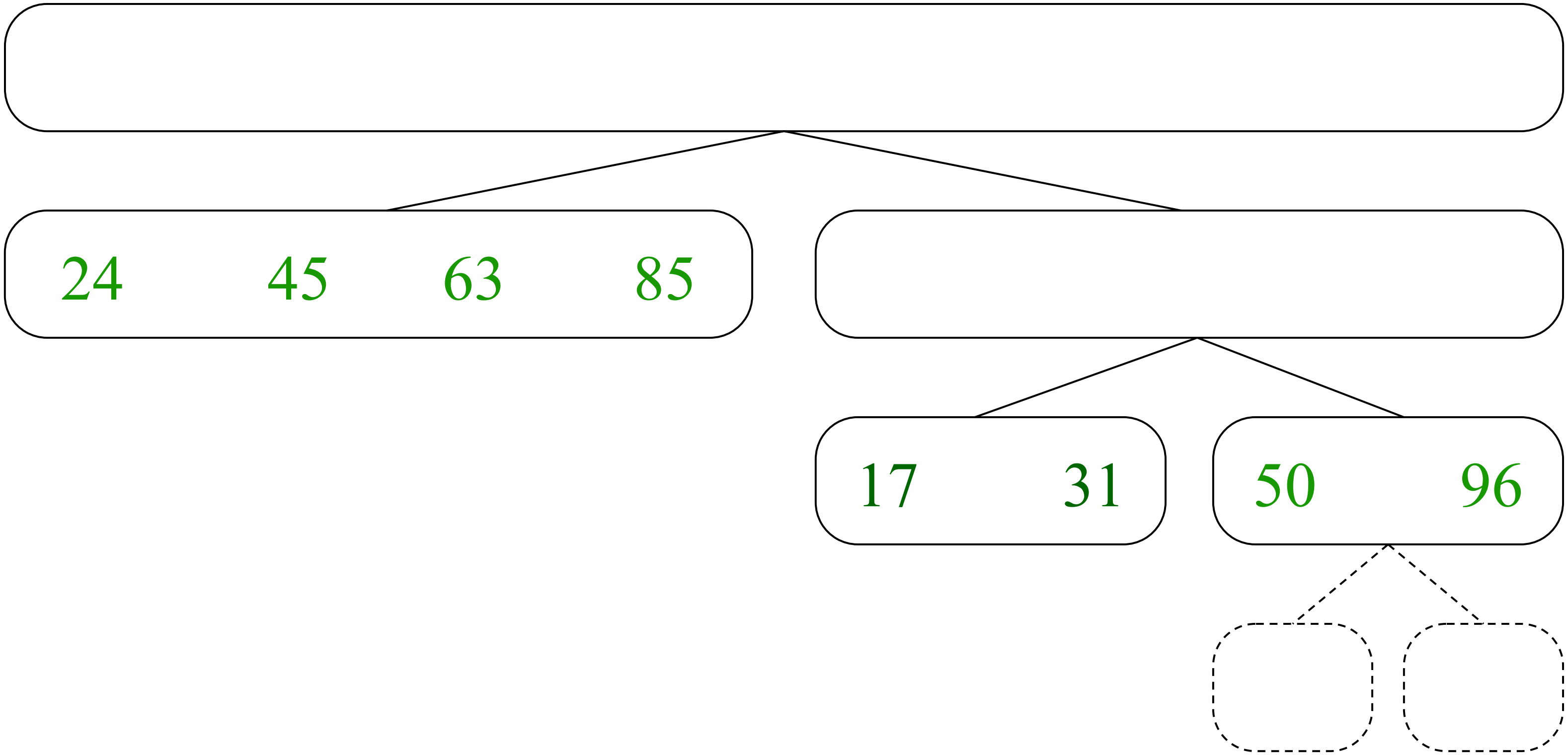
Mergesort: Illustration



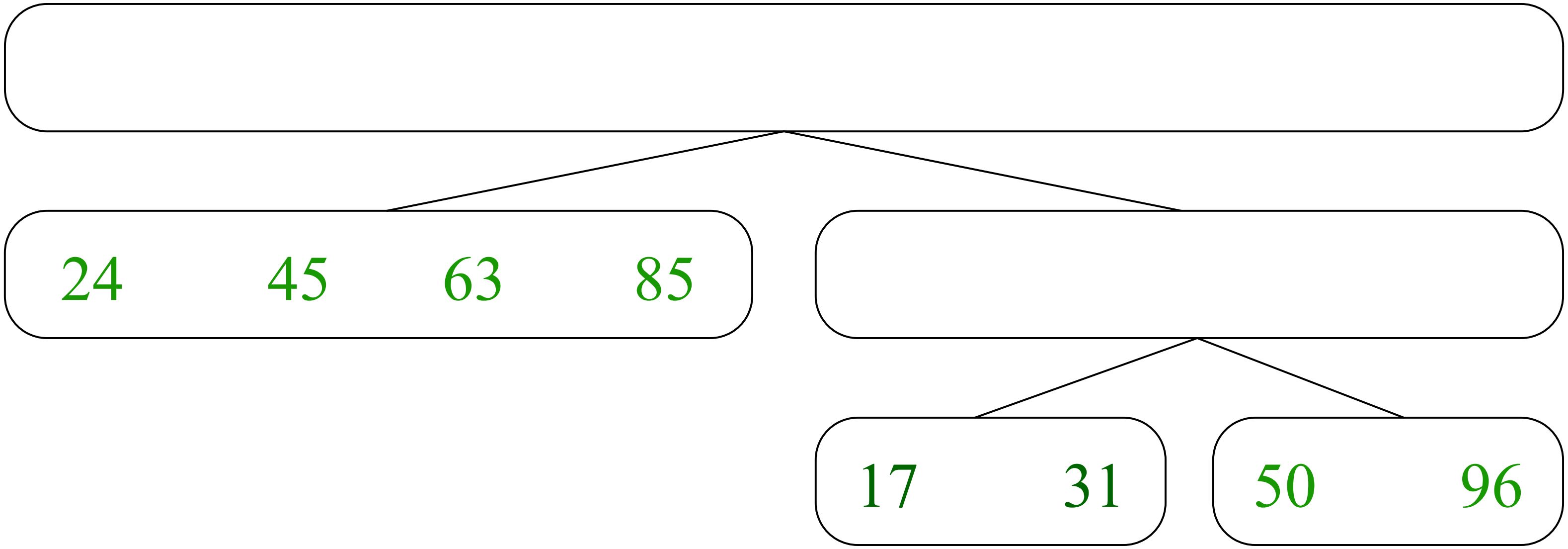
Mergesort: Illustration



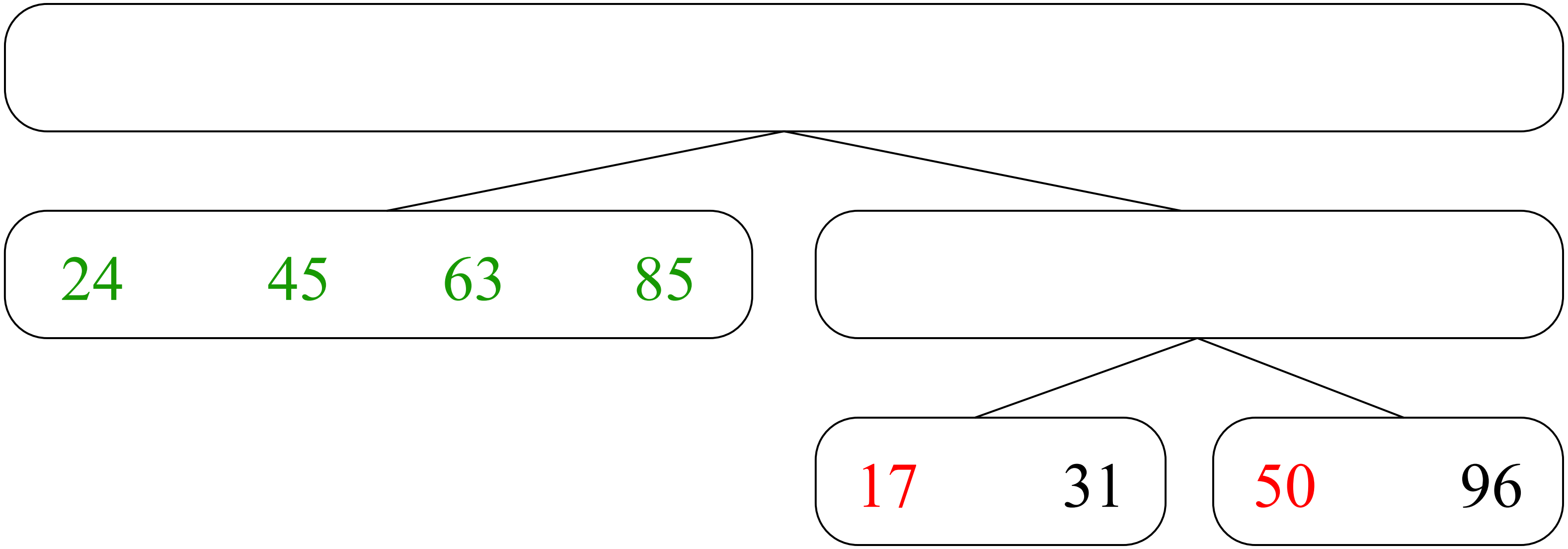
Mergesort: Illustration



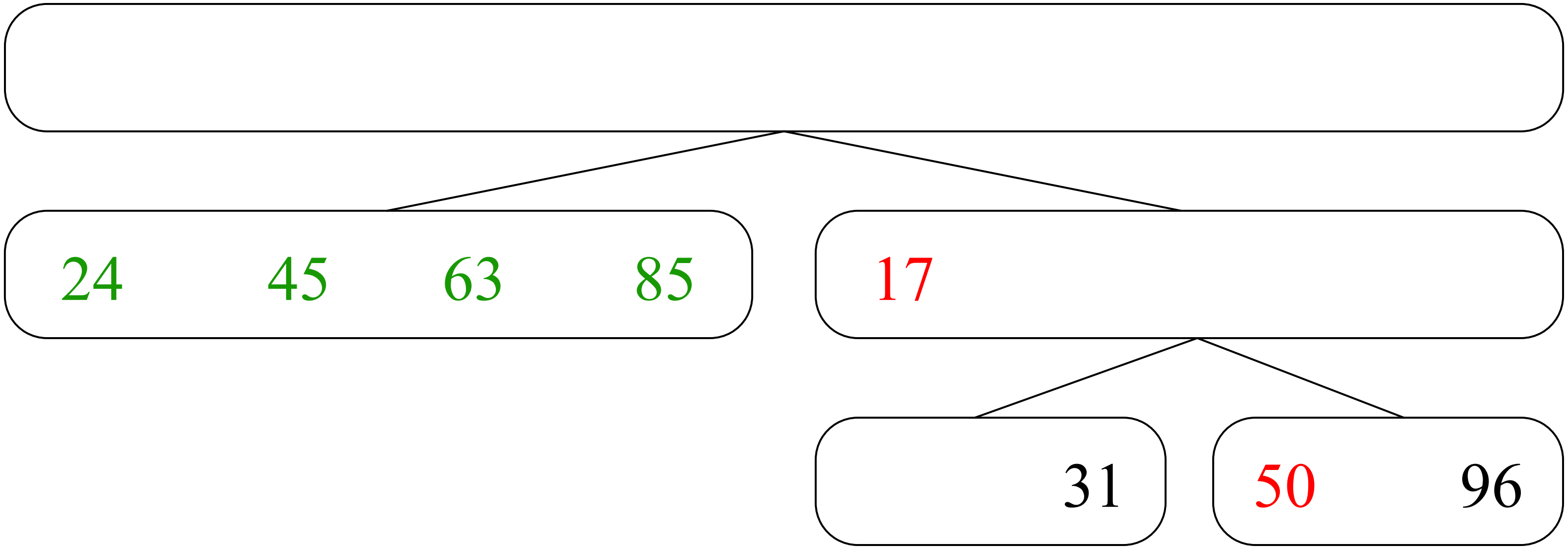
Mergesort: Illustration



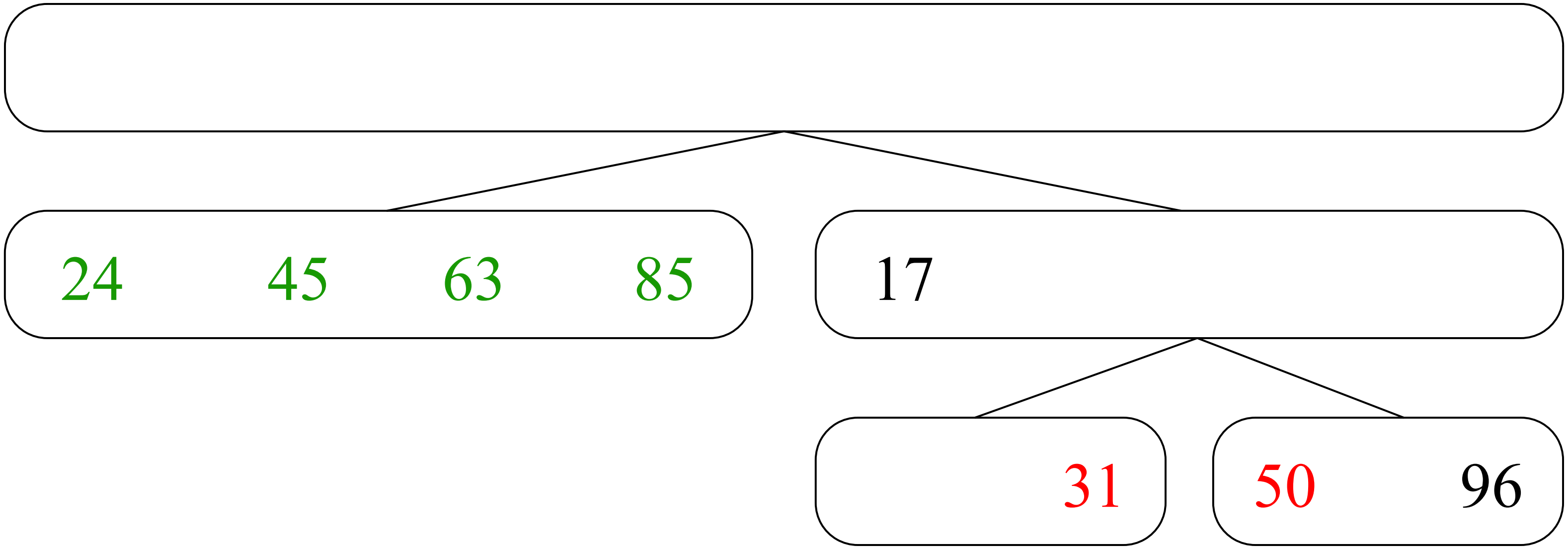
Mergesort: Illustration



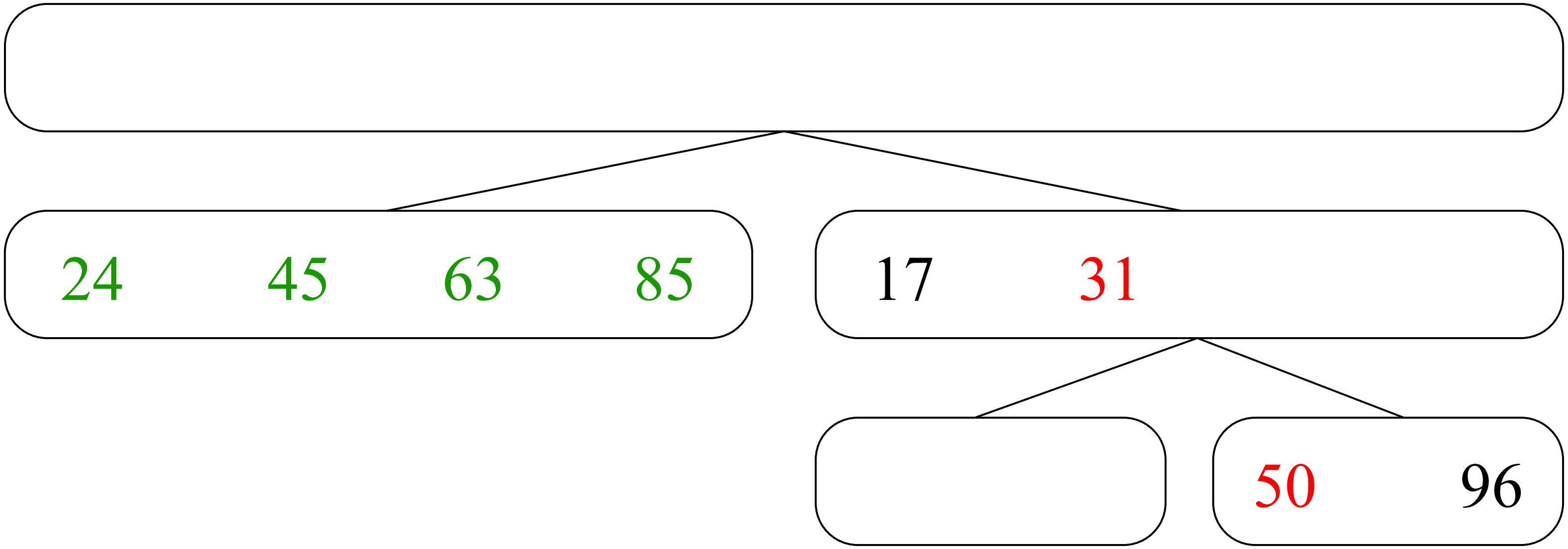
Mergesort: Illustration



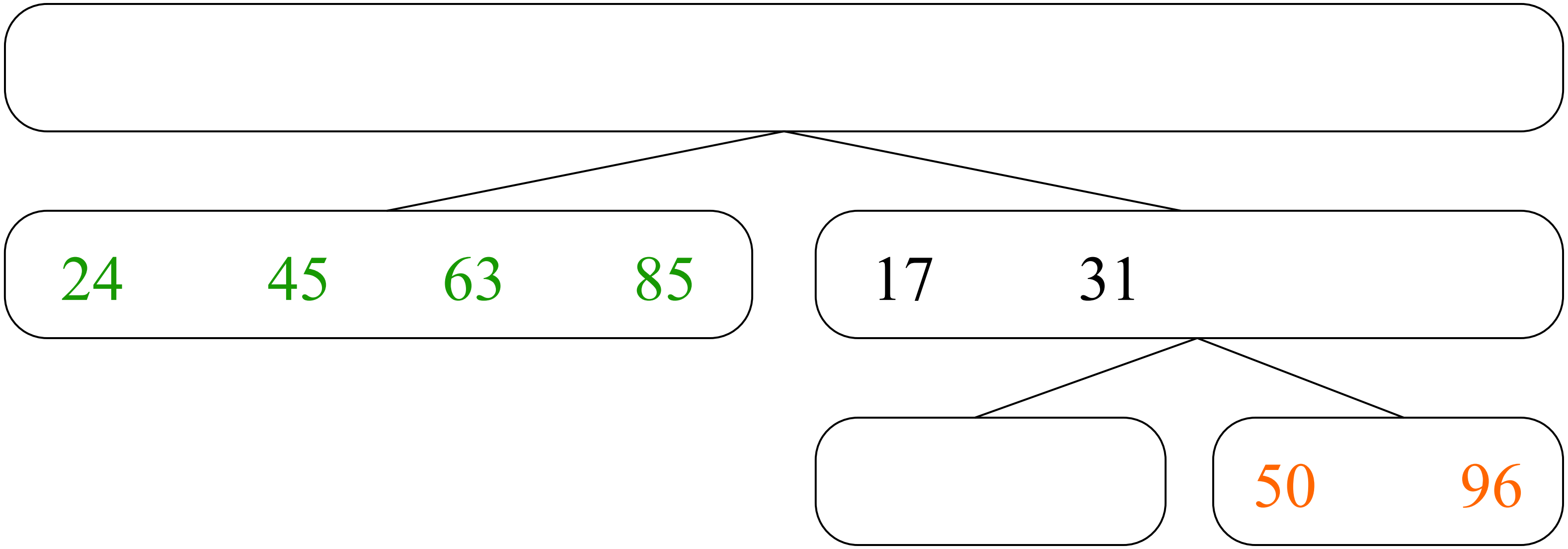
Mergesort: Illustration



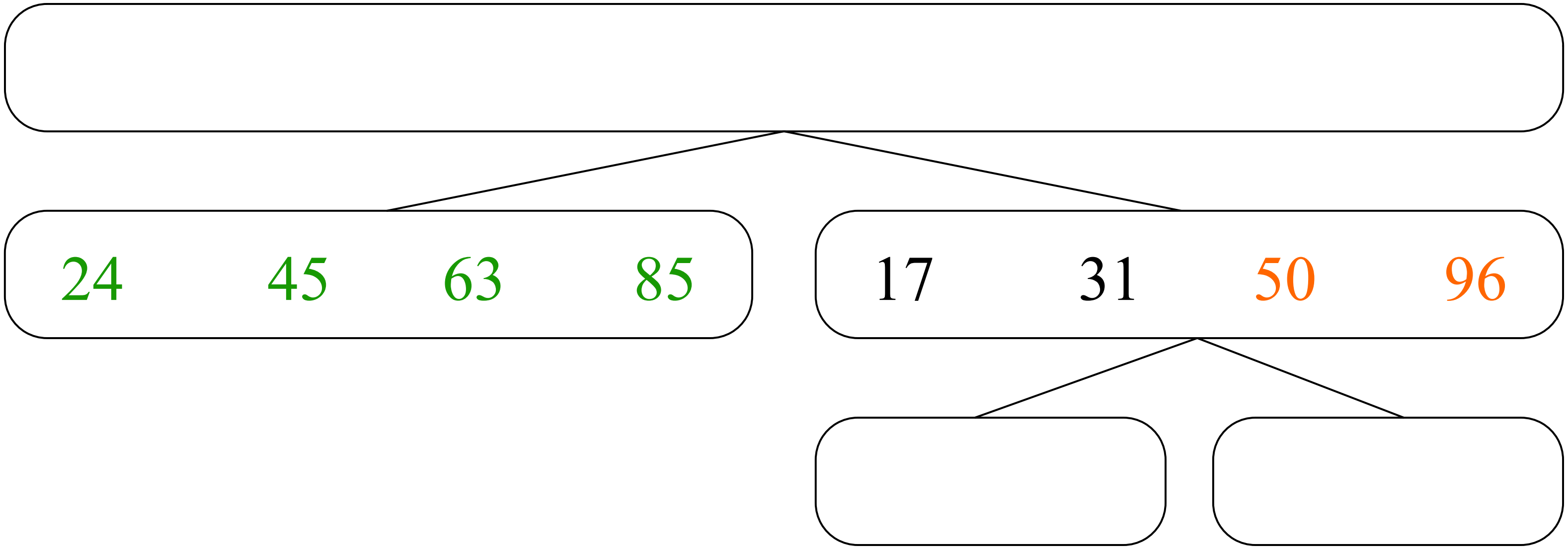
Mergesort: Illustration



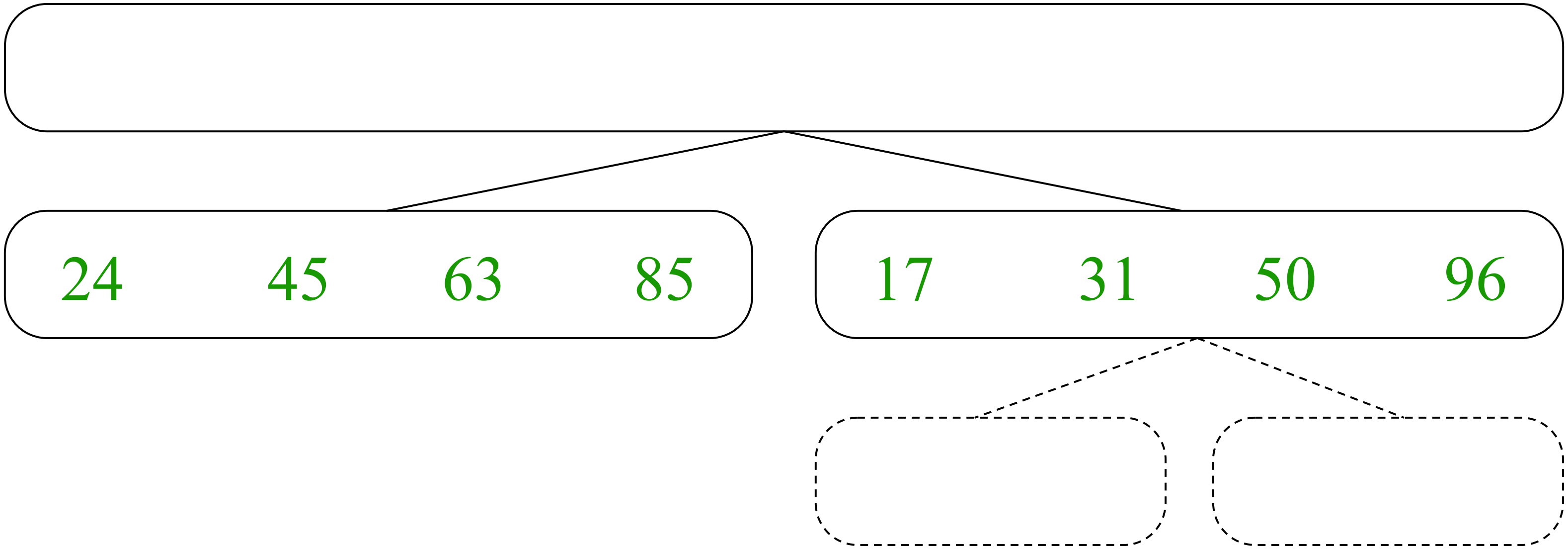
Mergesort: Illustration



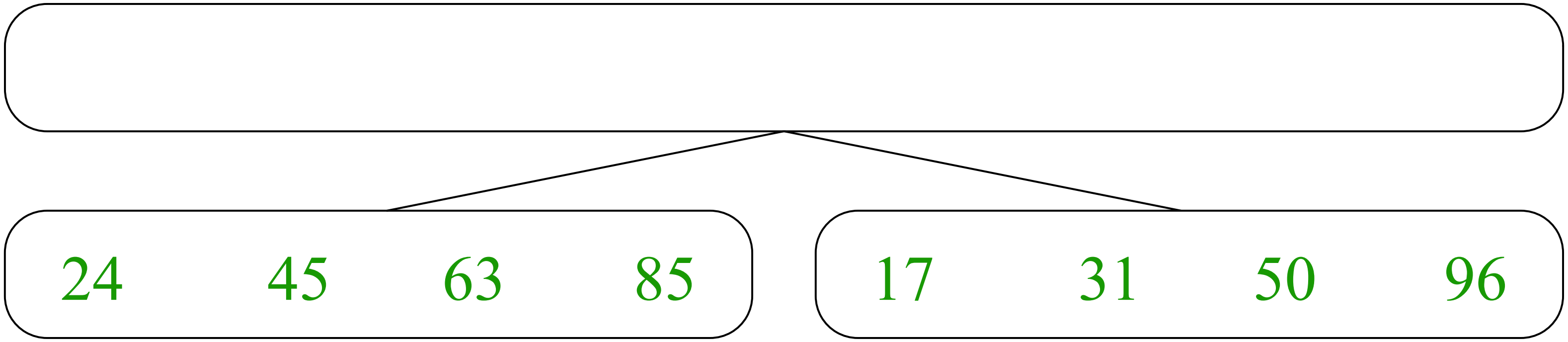
Mergesort: Illustration



Mergesort: Illustration



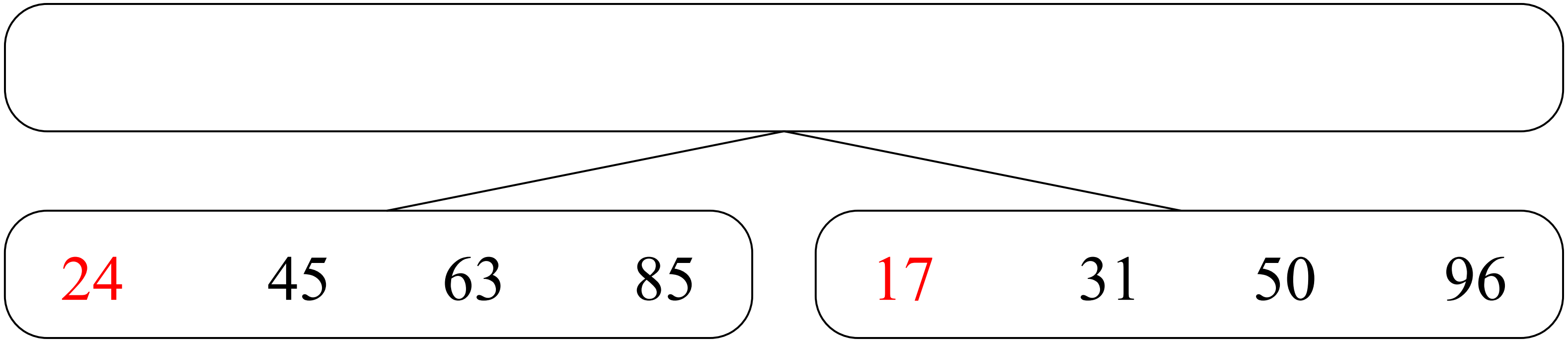
Mergesort: Illustration



24 45 63 85

17 31 50 96

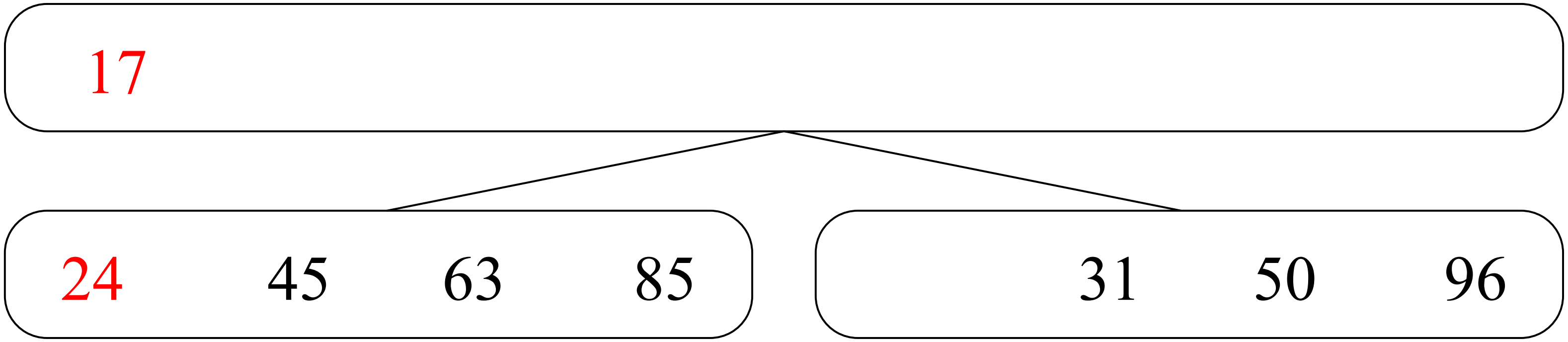
Mergesort: Illustration



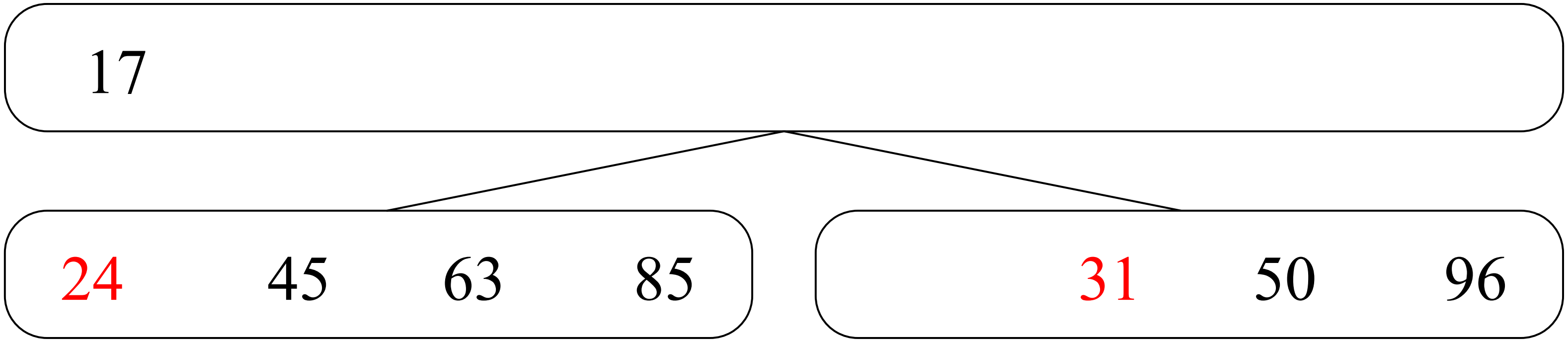
24 45 63 85

17 31 50 96

Mergesort: Illustration



Mergesort: Illustration

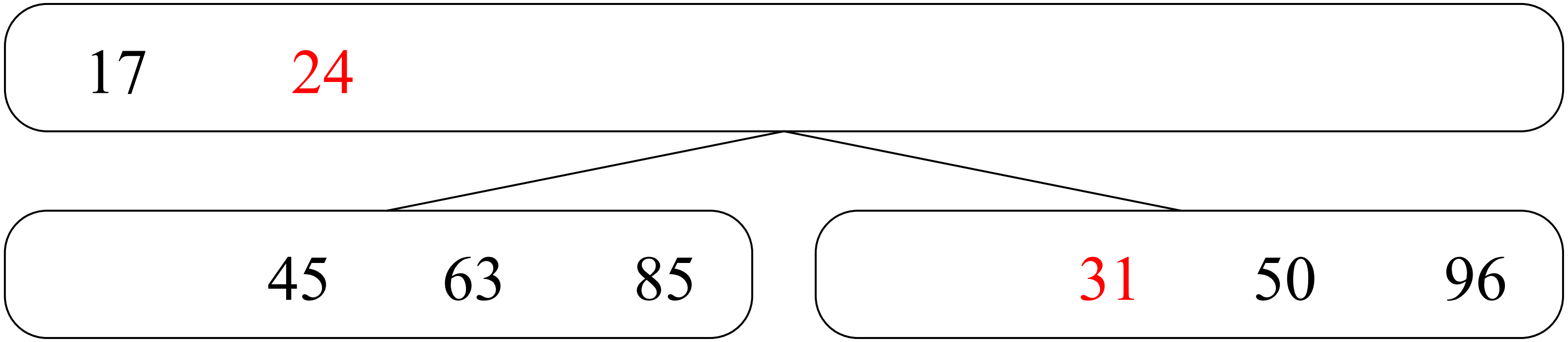


17

24 45 63 85

31 50 96

Mergesort: Illustration

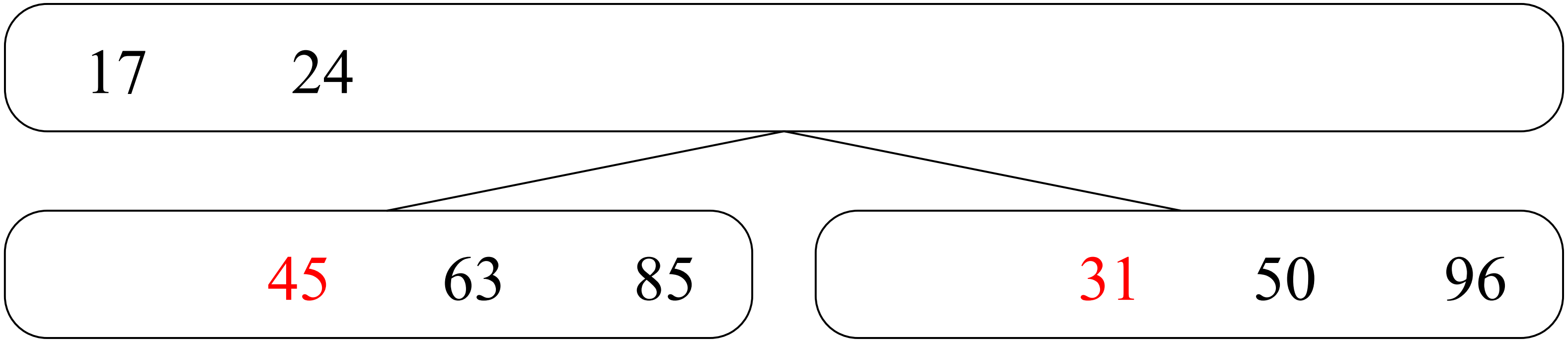


17 24

45 63 85

31 50 96

Mergesort: Illustration

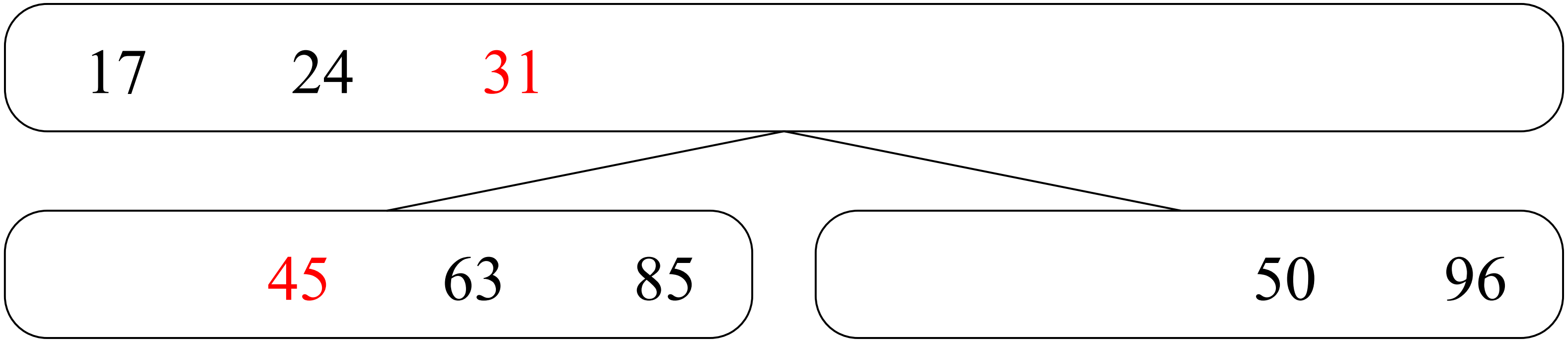


17 24

45 63 85

31 50 96

Mergesort: Illustration

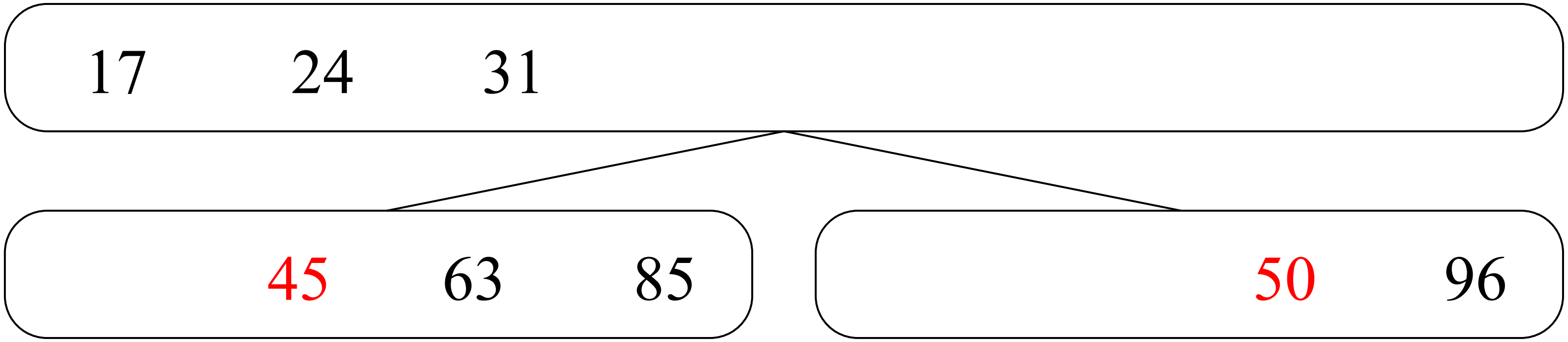


17 24 31

45 63 85

50 96

Mergesort: Illustration

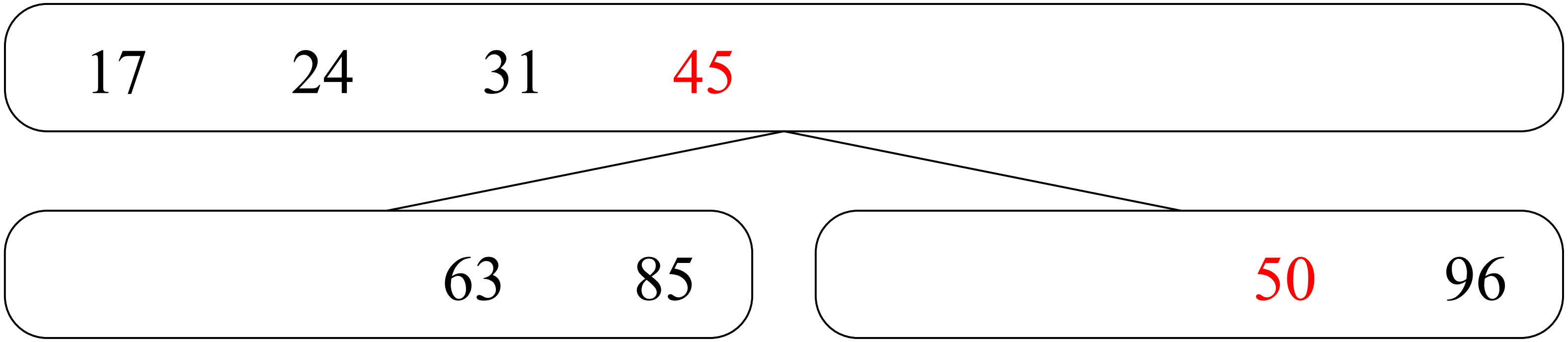


17 24 31

45 63 85

50 96

Mergesort: Illustration

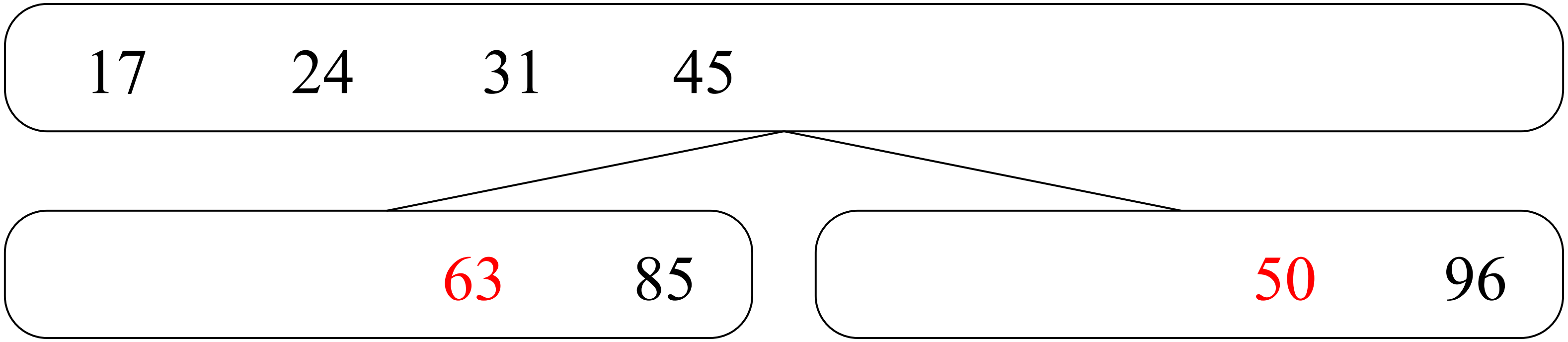


17 24 31 45

63 85

50 96

Mergesort: Illustration

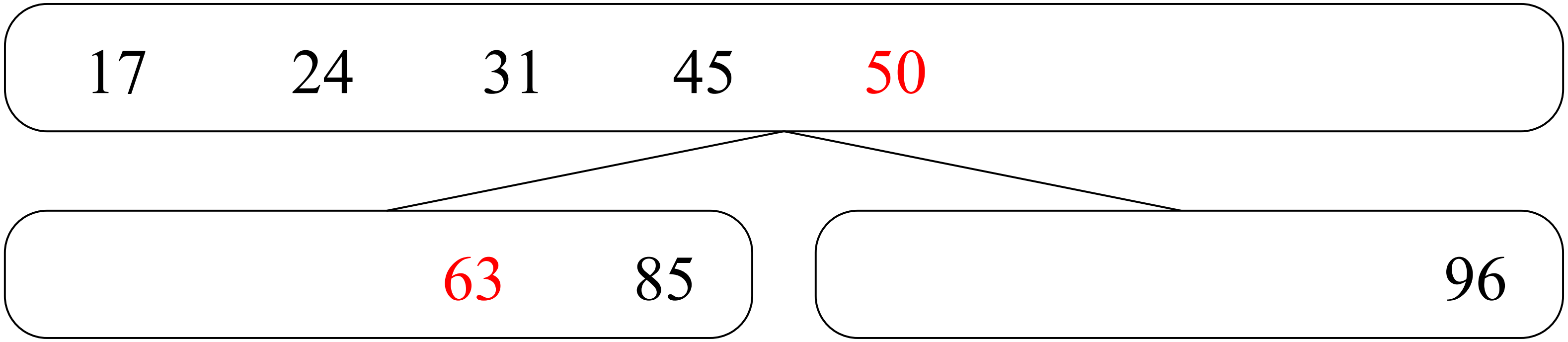


17 24 31 45

63 85

50 96

Mergesort: Illustration

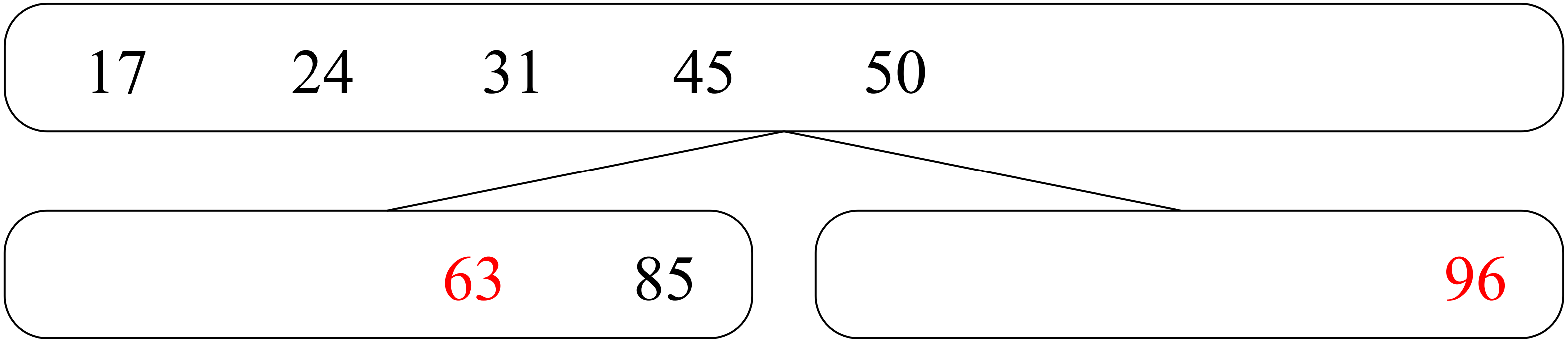


17 24 31 45 50

63 85

96

Mergesort: Illustration

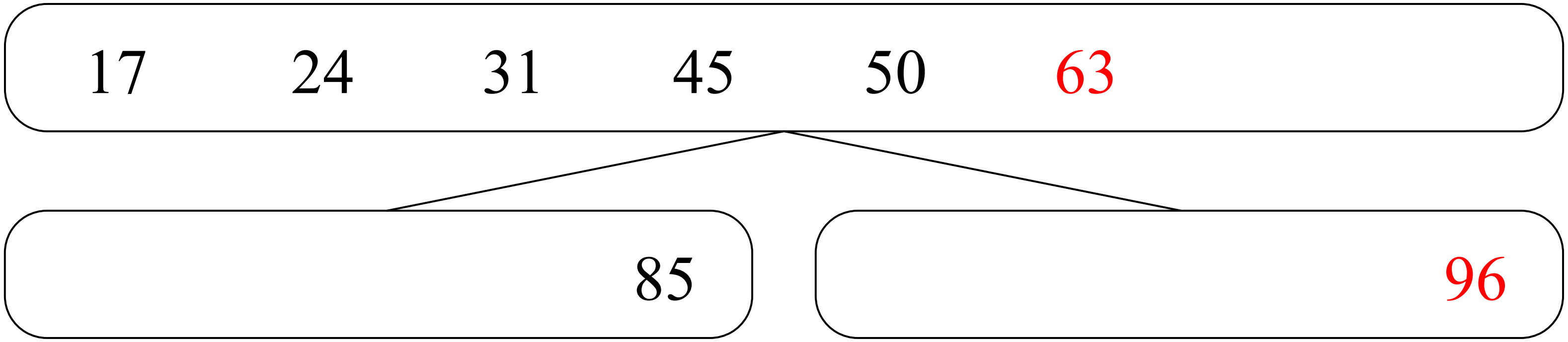


17 24 31 45 50

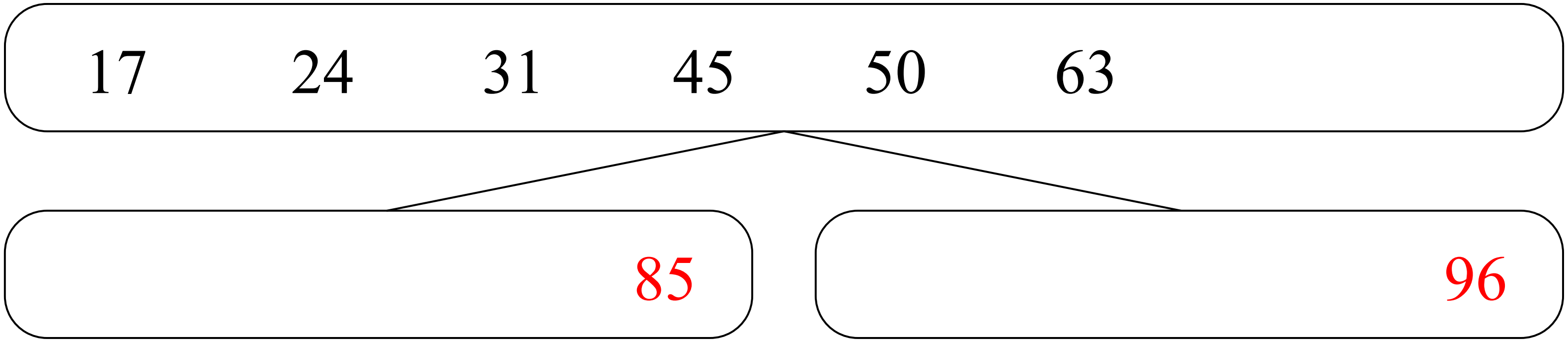
63 85

96

Mergesort: Illustration



Mergesort: Illustration

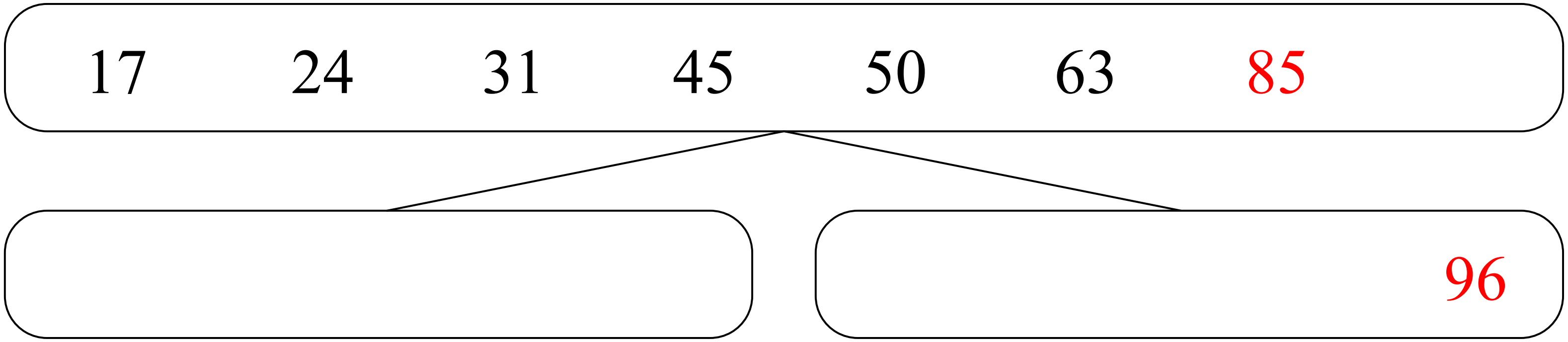


17 24 31 45 50 63

85

96

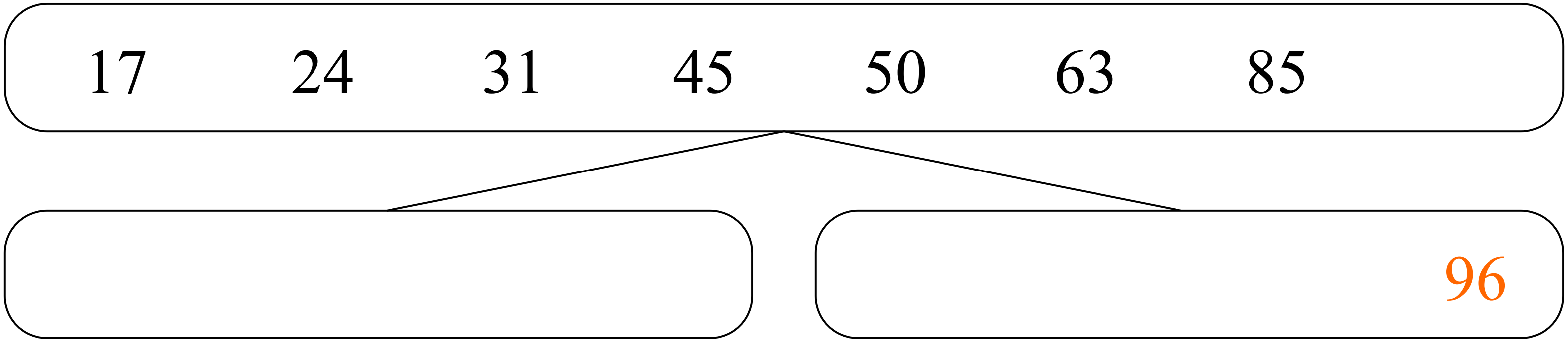
Mergesort: Illustration



17 24 31 45 50 63 85

96

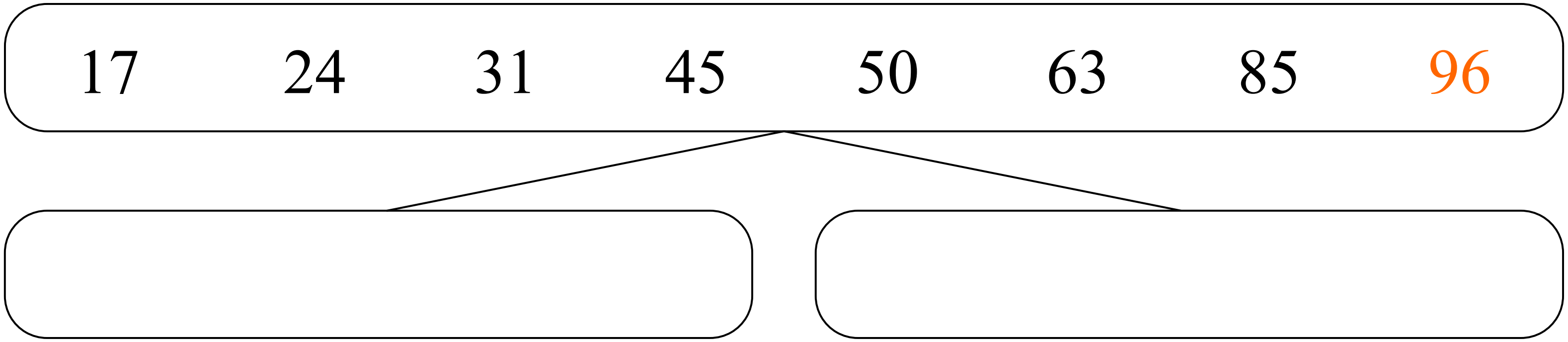
Mergesort: Illustration



17 24 31 45 50 63 85

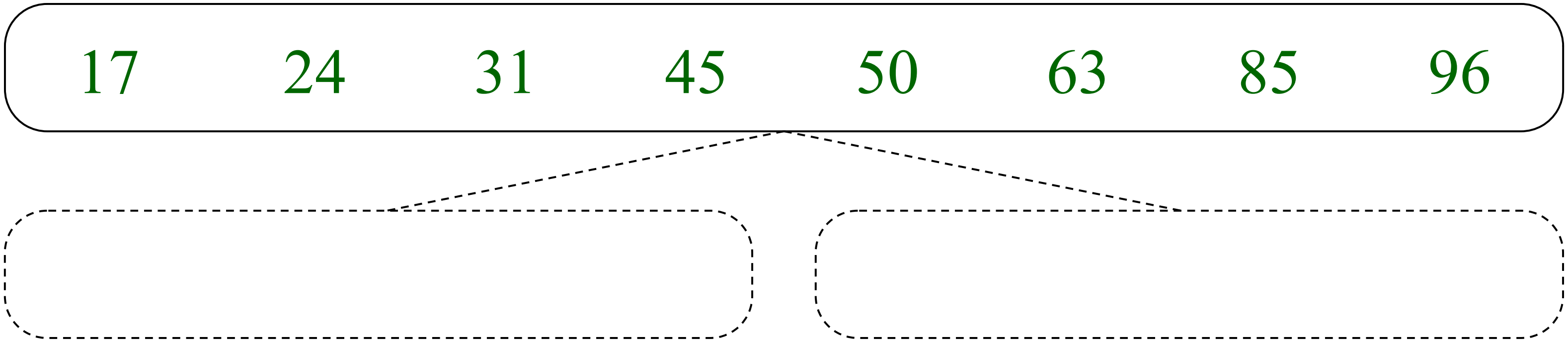
96

Mergesort: Illustration



17 24 31 45 50 63 85 96

Mergesort: Illustration



17 24 31 45 50 63 85 96

Mergesort: Illustration

17 24 31 45 50 63 85 96

Mergesort: Time complexity

- Best, worst, average-case
  - Each merge operation takes 0(k) time for 2 lists each k/2 elements long (merged into one list k elements long)
  - There will be $\log_2 n$ levels
    - 1st level: 2 *n/2 long lists* to be merged into 1 *n long list*
    - 2nd level: 4 *n/4 long lists* to be merged into 2 *n/2 long lists*
    - 3rd level: 8 *n/8 long lists* to be merged into 4 *n/4 long lists*
    - …
  - Time Complexity: $O(n\log_2 n)$

Complexity of MergeSort

Pass Number	Number of merges	Merge list length	# of comps / moves per merge
1	2^{k-1} or n/2	1 or $n/2^{k}$	$\leq 2^{1}$
2	2^{k-2} or n/4	2 or $n/2^{k-1}$	$\leq 2^{2}$
3	2^{k-3} or n/8	4 or $n/2^{k-2}$	$\leq 2^{3}$
.	.	.	.
.	.	.	.
.	.	.	.
k – 1	2^{1} or $n/2^{k-1}$	2^{k-2} or n/4	$\leq 2^{k-1}$
k	2^{0} or $n/2^{k}$	2^{k-1} or n/2	$\leq 2^{k}$

k = log n

Complexity of MergeSort

Multiplying **the number of merges** by the **maximum number of comparisons** per merge, we get:

$$(2^{k-1})2^1 = 2^k$$

$$(2^{k-2})2^2 = 2^k$$

$$\vdots$$

$$(2^1)2^{k-1} = 2^k$$

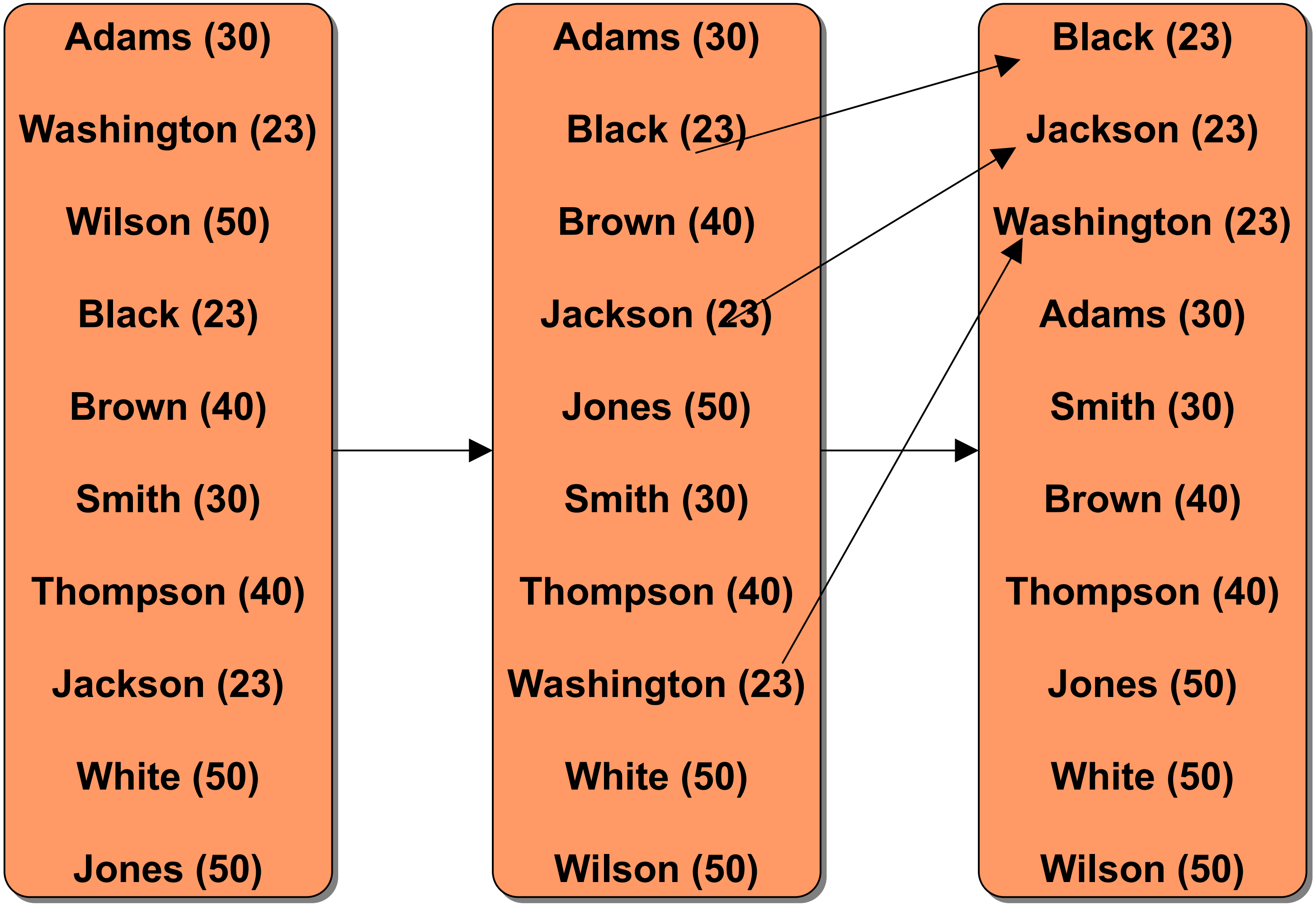
$$(2^0)2^k = 2^k$$

***k* passes each require *2^k* comparisons (and moves). But *k* = *lg n* and hence, we get lg(n) • n comparisons or O(n lgn)**

Stable vs. Non-Stable Sorts

- We frequently use sorting methods for items with multiple keys
- Sometimes we need to apply the sorting with different keys
- For instance we want to sort a list of people based on first name and than on age
- So Black age 30 should appear before Jones age 30
- If we sort a list based on the first key (name) and then apply a sort based on the second key (age) how can we guarantee that the list is still ordered based on the first key?
- Definition: A sorting method is said the be stable if it preserves the relative order of the items with duplicated keys on the list

Stable vs. Non-Stable Sorts



Adams (30)	Adams (30)	Black (23)
Washington (23)	Black (23)	Jackson (23)
Wilson (50)	Brown (40)	Washington (23)
Black (23)	Jackson (23)	Adams (30)
Brown (40)	Jones (50)	Smith (30)
Smith (30)	Smith (30)	Brown (40)
Thompson (40)	Thompson (40)	Thompson (40)
Jackson (23)	Washington (23)	Jones (50)
White (50)	White (50)	White (50)
Jones (50)	Wilson (50)	Wilson (50)

Stable vs. Non-Stable Sorts

- Mergesort is relatively easy to be made stable
  - Just make sure the merge function is stable
- Another algorithms that sort in O(n log n) is heapsort but it is not stable
- Quicksort is also not stable
- Selection Sort is Stable
- Bubble Sort is Stable