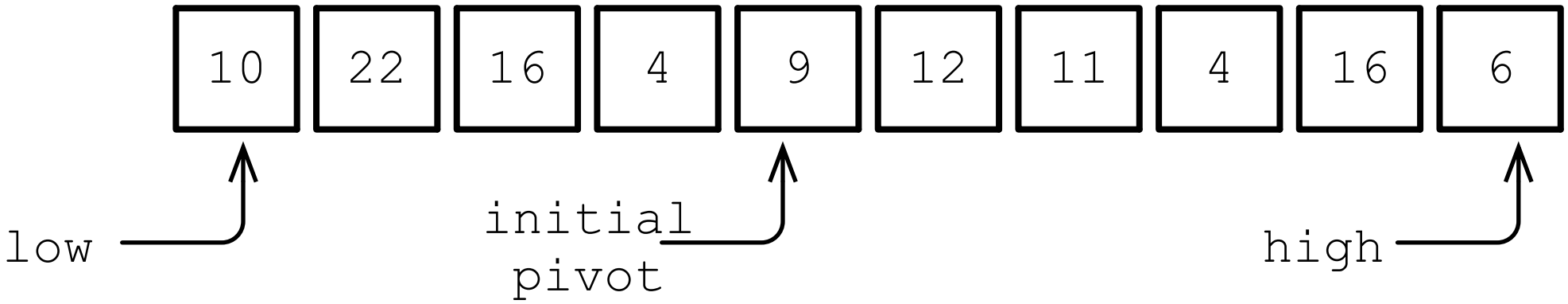
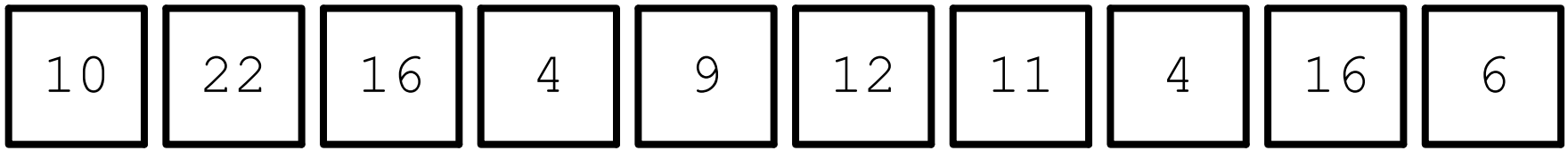
Quick Sort:

1. Takes a *low* and *high* index.
2. Pick a *pivot* index in range [low, high].
3. **Partition** data by moving everything less than data[pivot] to left side, all else to right side. The *pivot* index may move to the right.
4. **Recursively** do this in range [low, pivot) and (pivot, high].

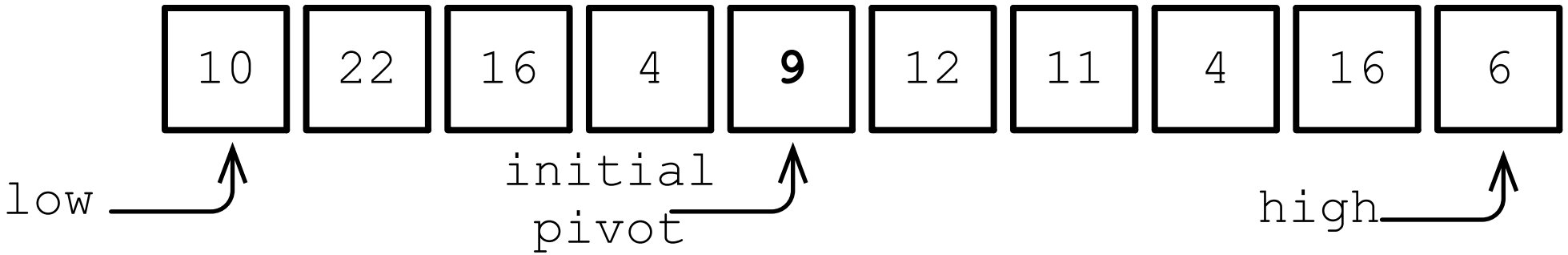


Most of the algorithm is spent in the **partition** function. Almost all the remaining slides cover that.

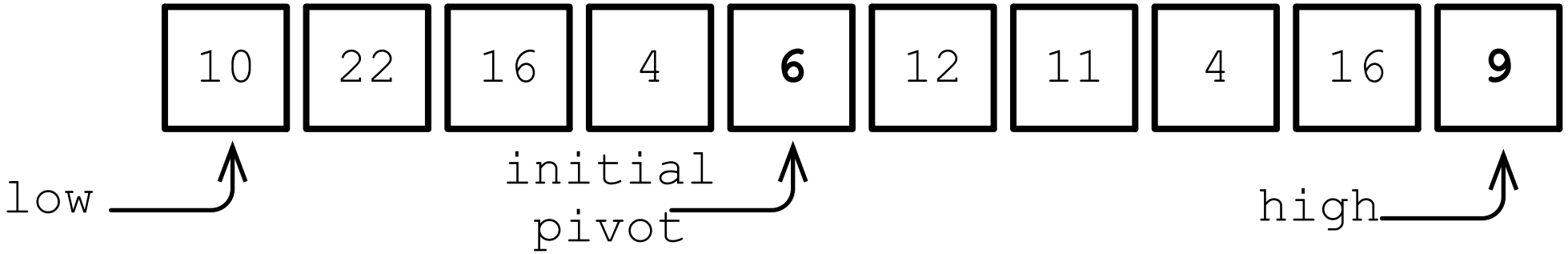
Say we start with the following:



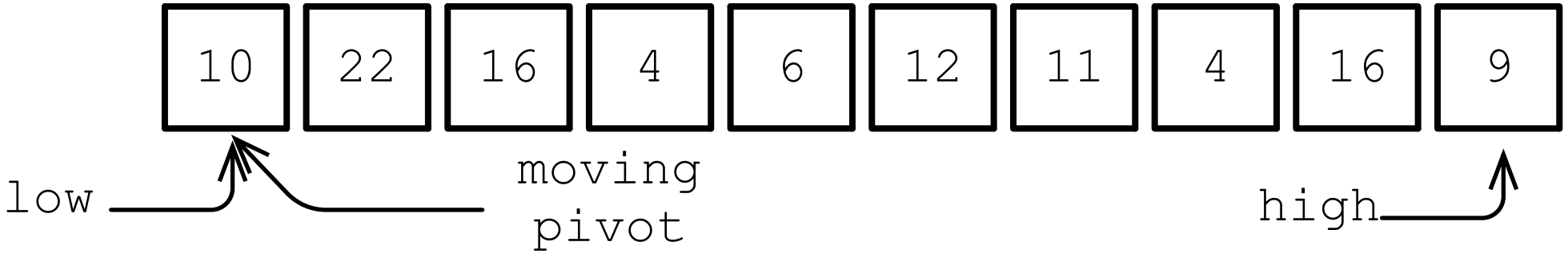
Value at initial pivot is **9**.



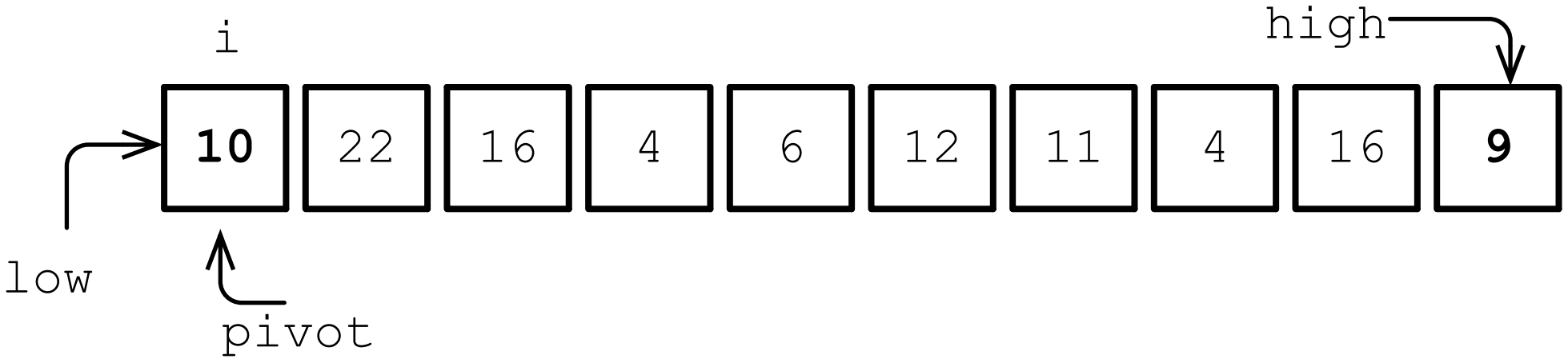
First, swap values at pivot and high.

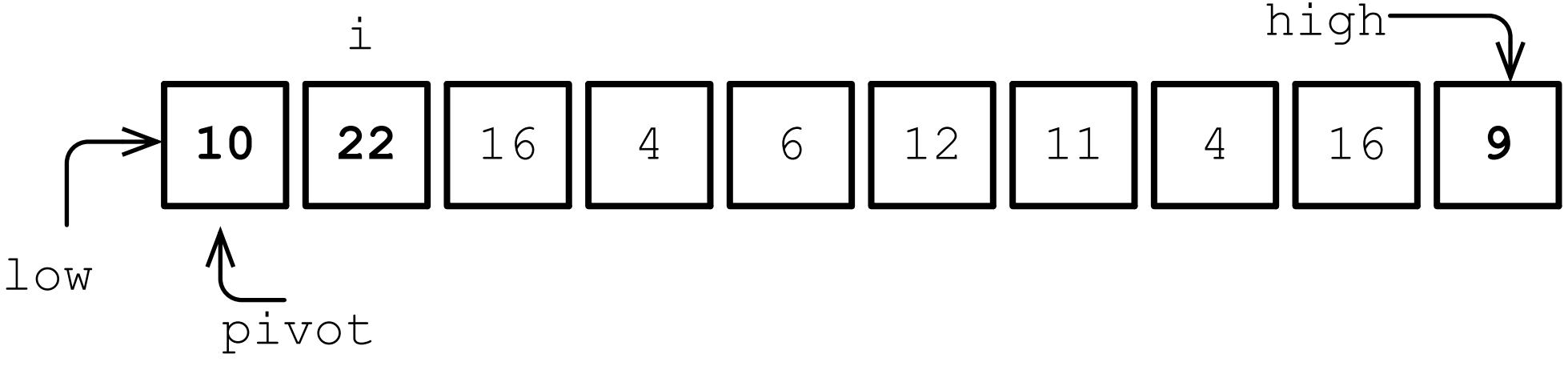


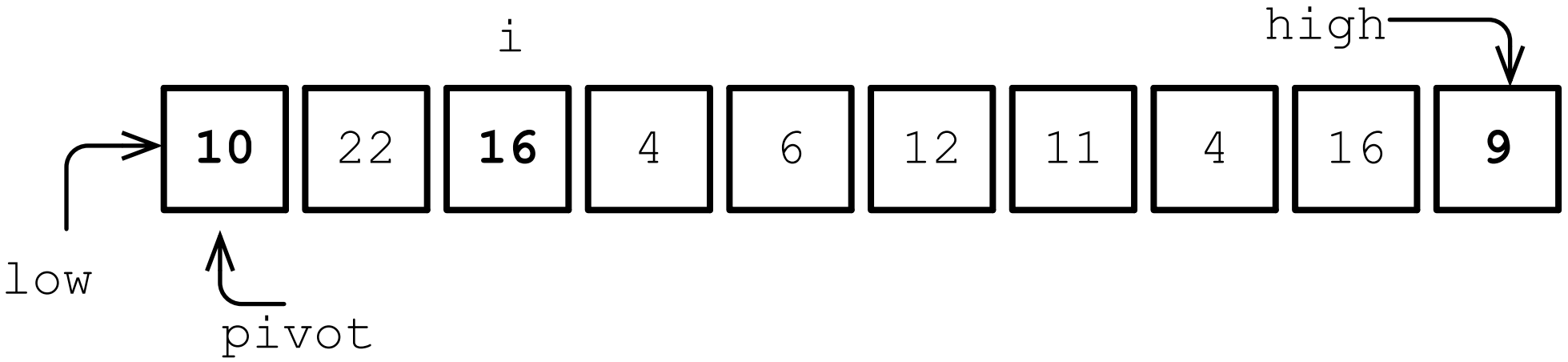
Set pivot = low, and begin the partition routine.

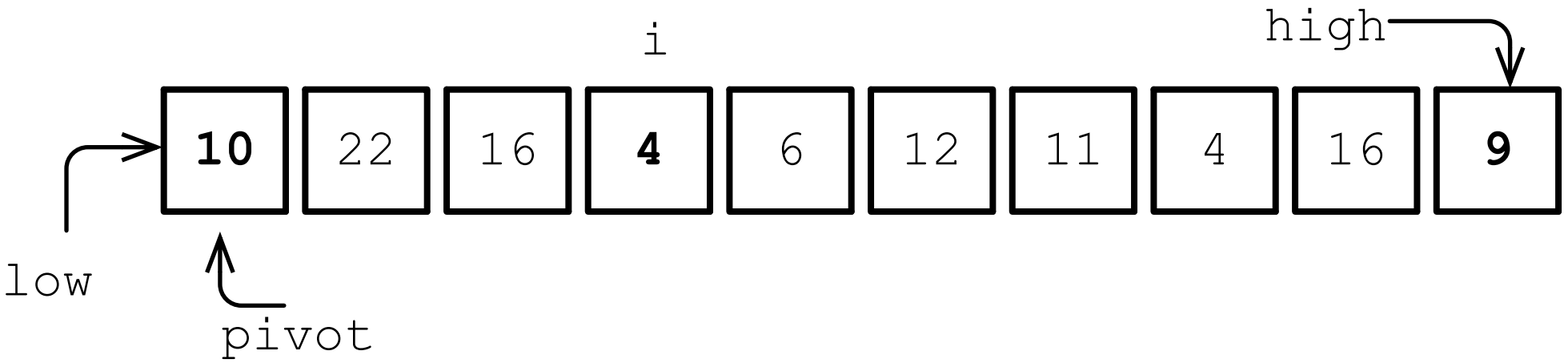


When we see data[i] < data[high],
swap data[i] and data[pivot], and increment pivot.

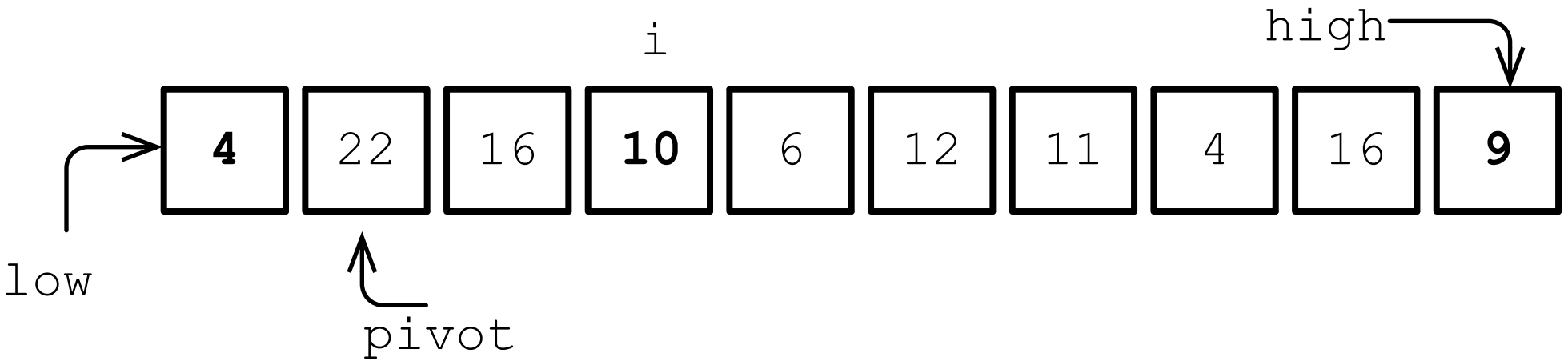


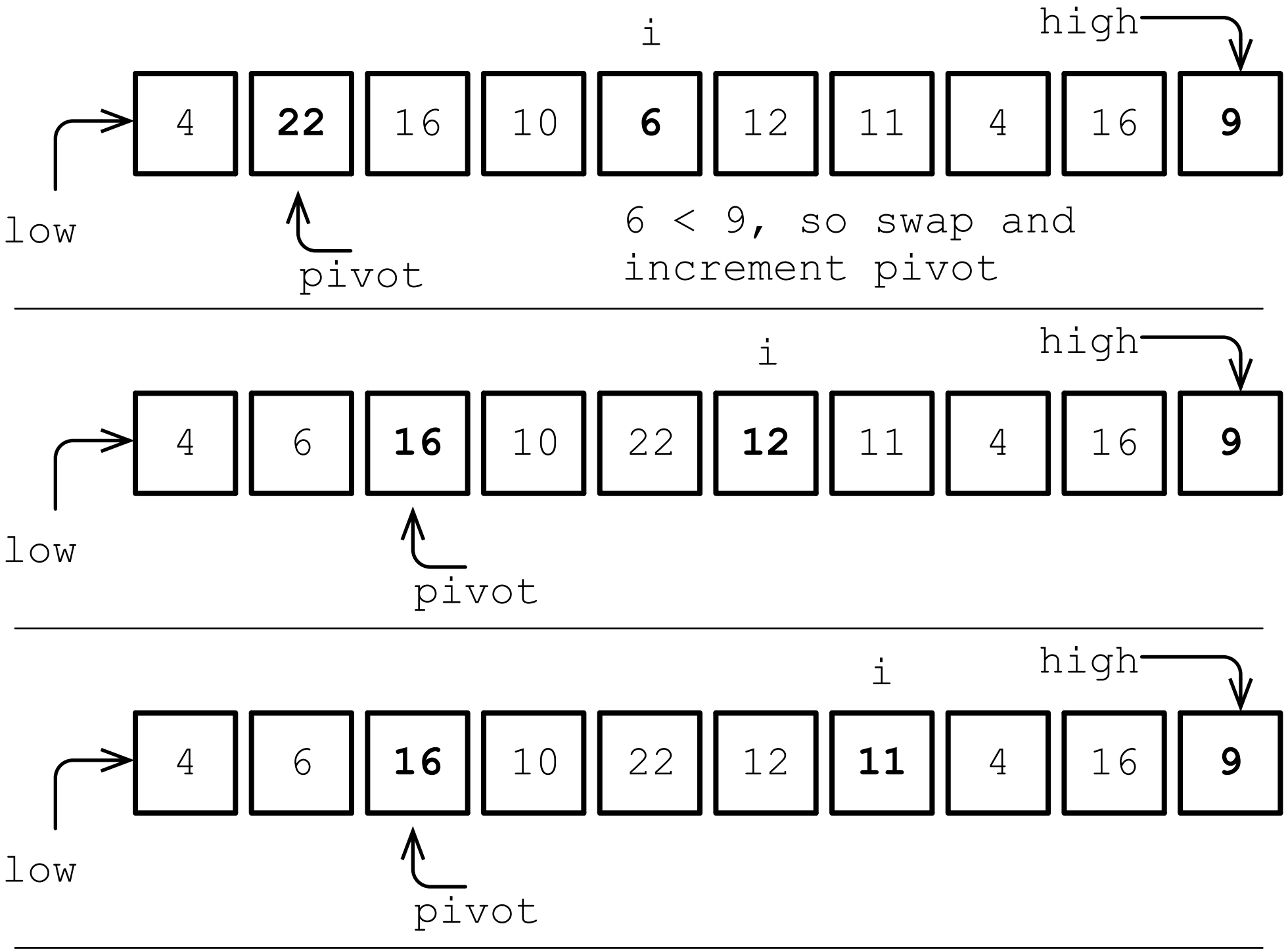


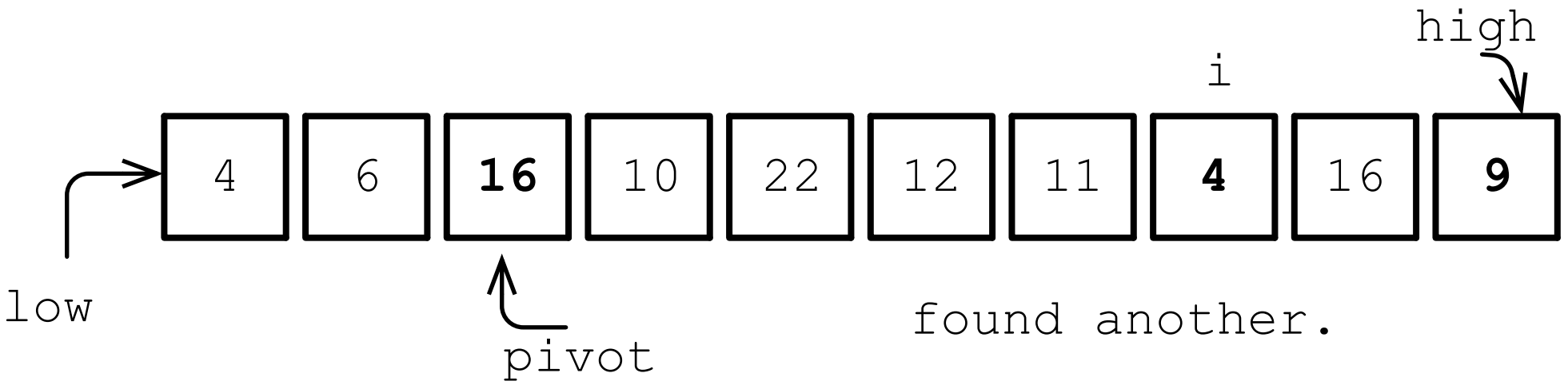


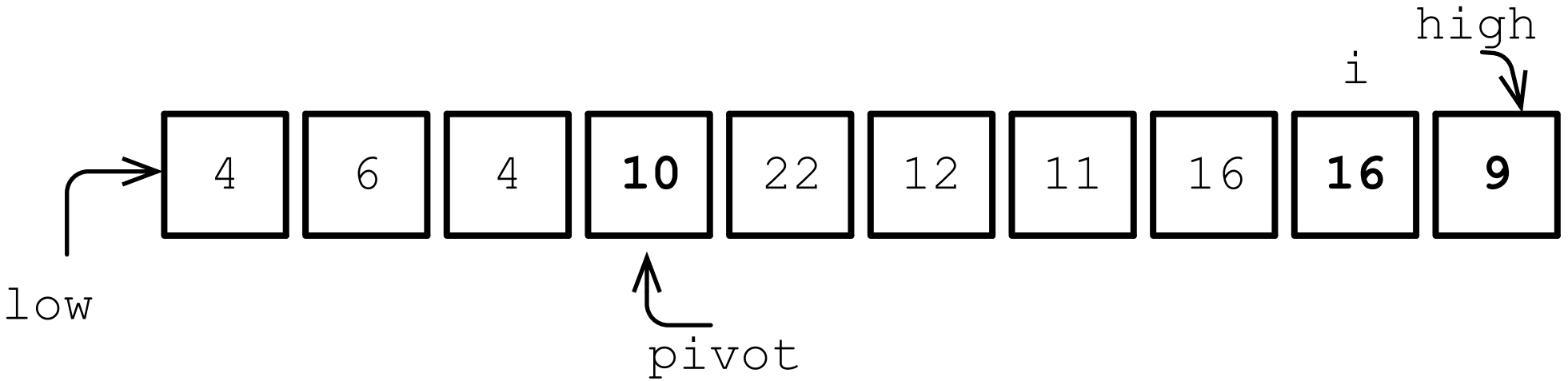


Found 4 < 9, so swap cells index
by pivot and i, and increment pivot.

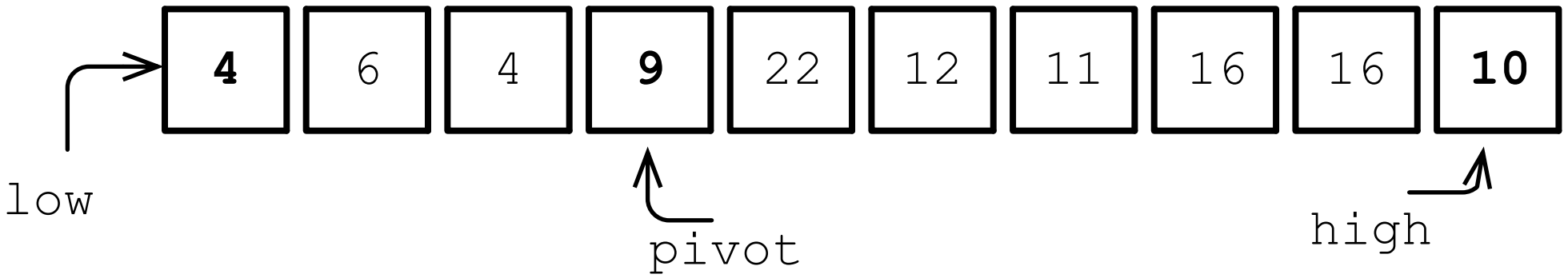

i
high
4
22
16
10
6
12
11
4
16
9
low
pivot


i
high
4
22
16
10
6
12
11
4
16
9
low
pivot
6 < 9, so swap and
increment pivot
i
high
4
6
16
10
22
12
11
4
16
9
low
pivot
i
high
4
6
16
10
22
12
11
4
16
9
low
pivot



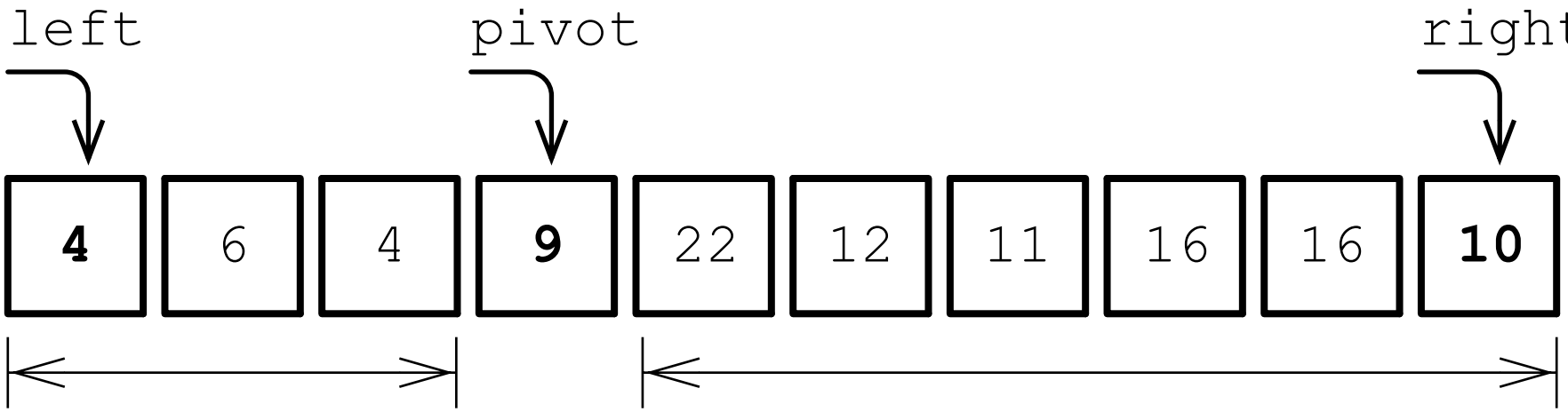


at this point we end the loop.
still have to clean up by swapping
data[pivot] and data[high].



After the final cleanup swap, our data is nicely partitioned with small values (less than 9) on the left, and larger values (greater than or equal to 9) on the right. All this is relative to the updated pivot.

Return the updated pivot index from the partition function.



Now all that is left is to recursively quick sort the left and right sides using the same algorithm but using the pivot to divide the process.

```
// recurse with left side
quicksort(data, left, pivot - 1)

// recurse with right side
quicksort(data, pivot + 1, right)
```