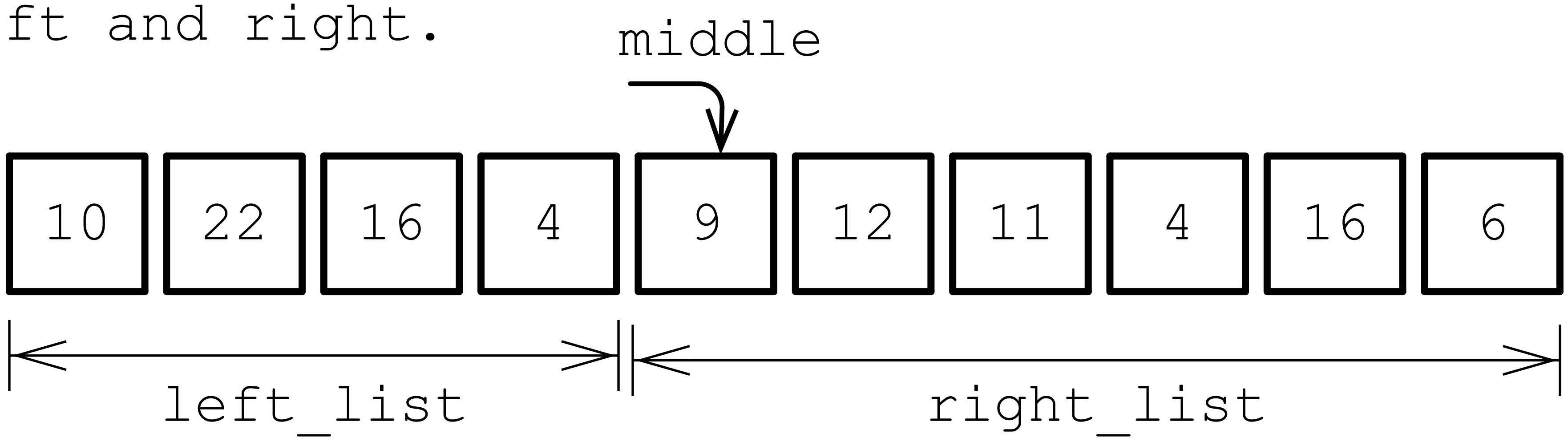
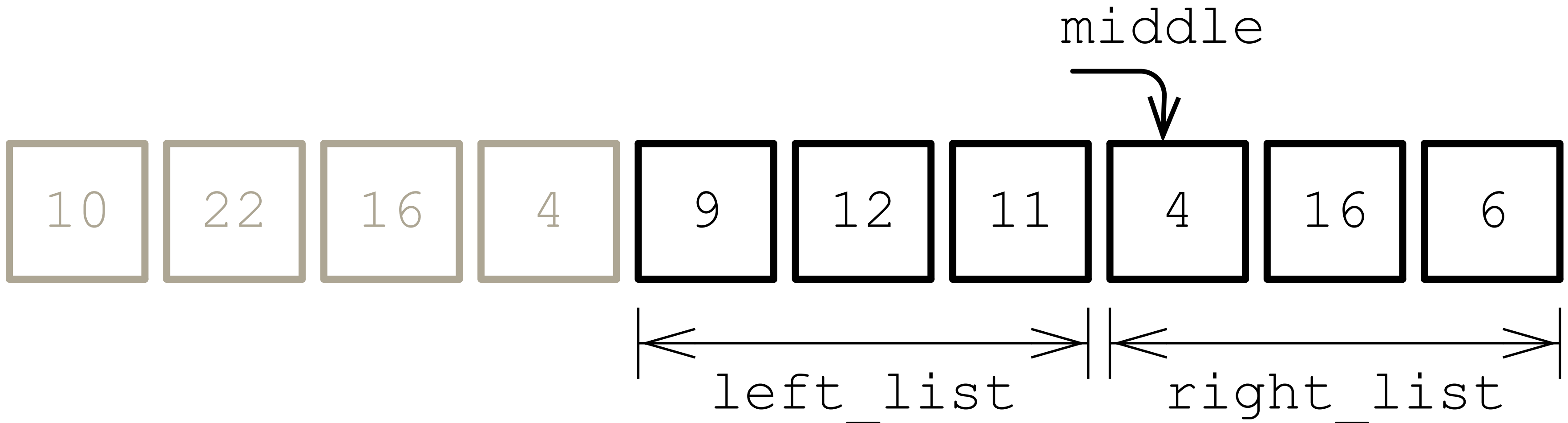
Mergesort is based on chopping a list into smaller lists and recursively merge sorting them. To chop a list, find the middle, and *create two new lists* based on what is to the left and right.



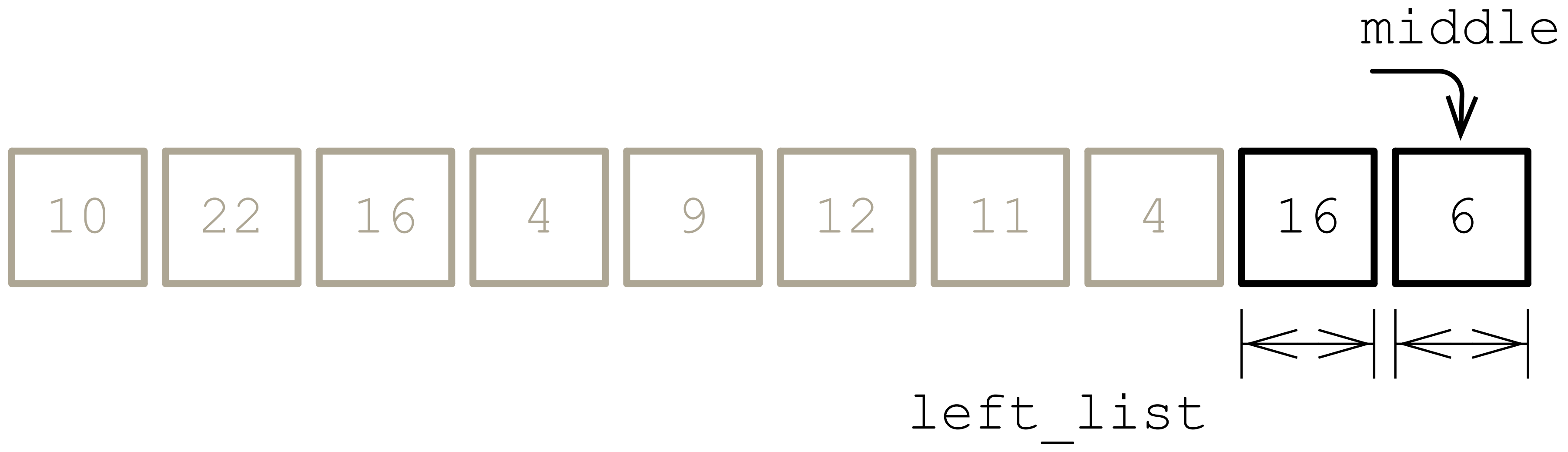
```
// then recursively mergesort these lists
left_list = mergesort(left_list)
right_list = mergesort(right_list)

// then merge the results
merge(left_list, right_list)
```

This illustrates how the 'right_list' of the initial list would be divided up further.

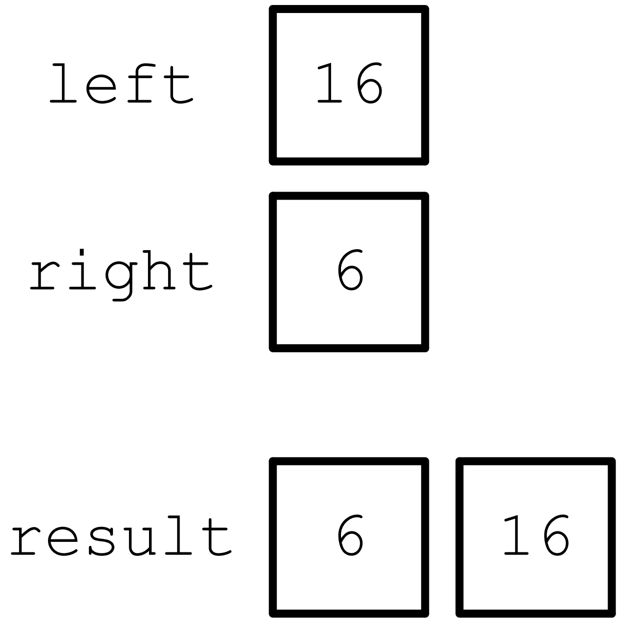


The recursion stopping condition is when the list is zero or one elements long. When the stopping condition is reached, the input list is returned (with zero or one elements).

This is one such stopping condition. The lists are both one element long, so they can't be divided any more.



Since we're dividing lists first, before merging, this means the first lists we merge are the zero- and one-element lists. This guarantees we merge small lists first, then larger ones.

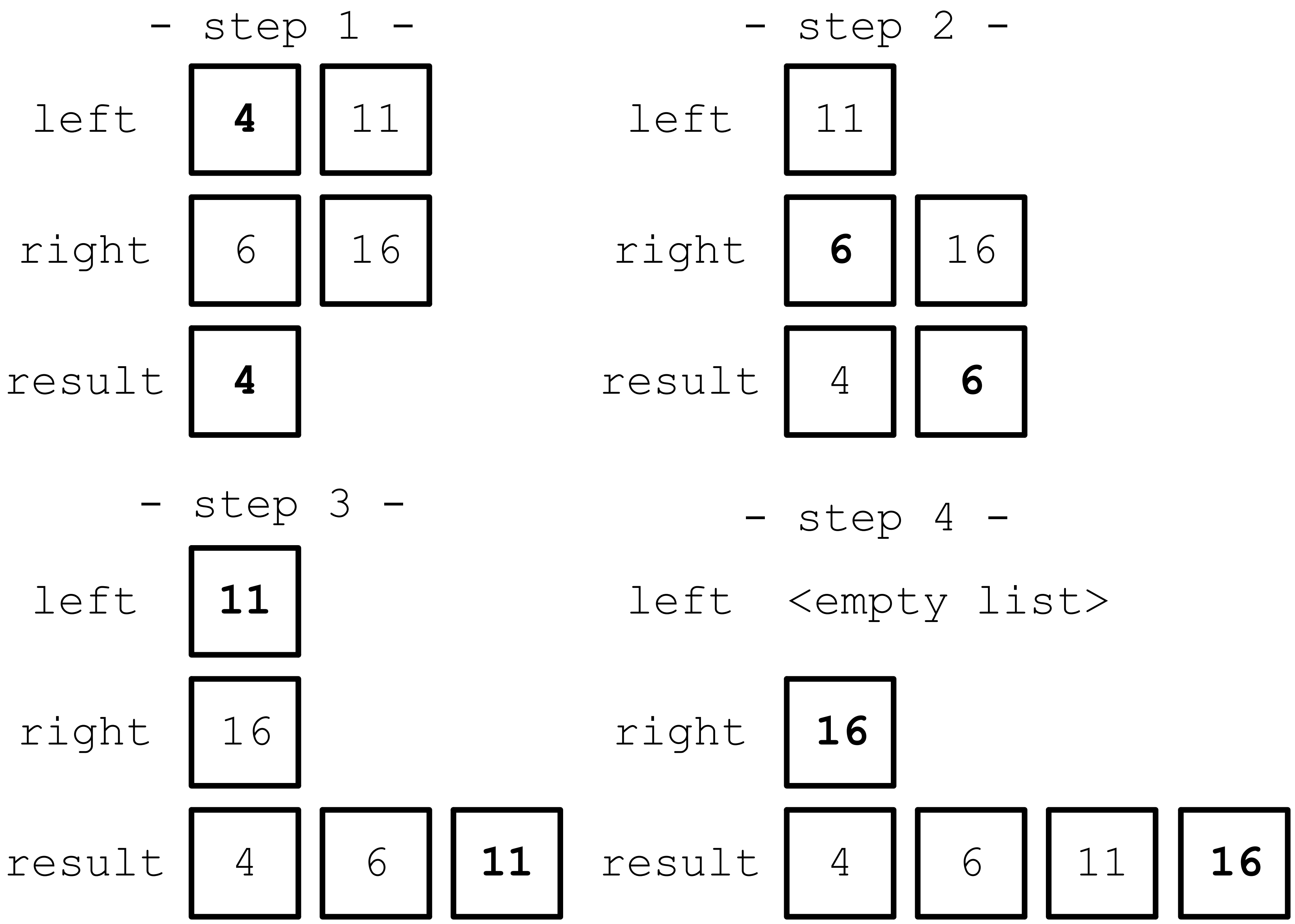


Merging two lists: examine the first one in each list, and *remove* the smaller of the two, and *append* it to a result list. (If there's a tie, it doesn't matter which we choose.)

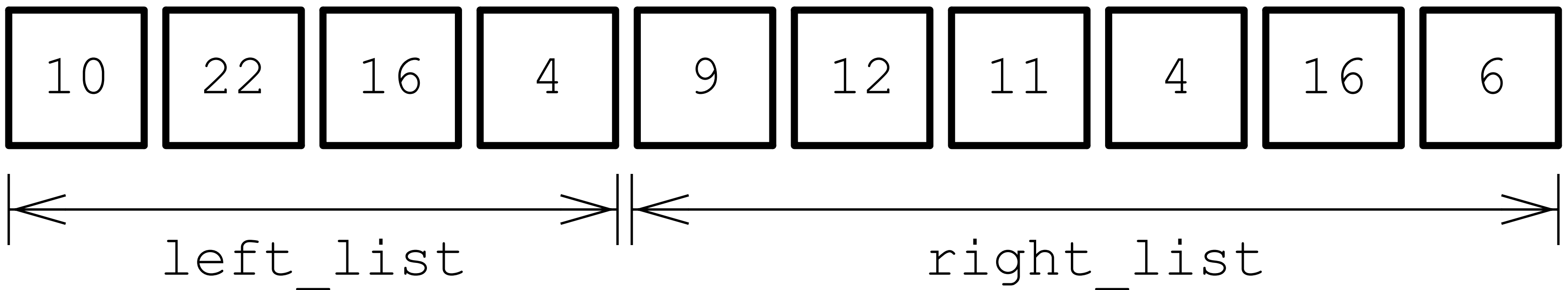
Notice that the resulting list is going to be in order, since we're choosing to append the smaller value first.

The first merge is not very interesting because it is only merging two single-item lists. But remember it is necessary to start with zero- or one-item lists to ensure sort order is intact.

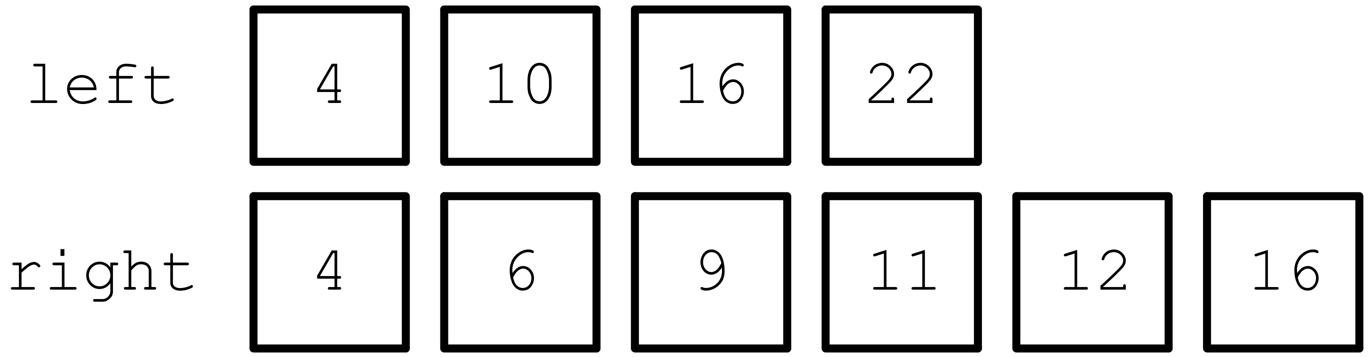
Merging Lists



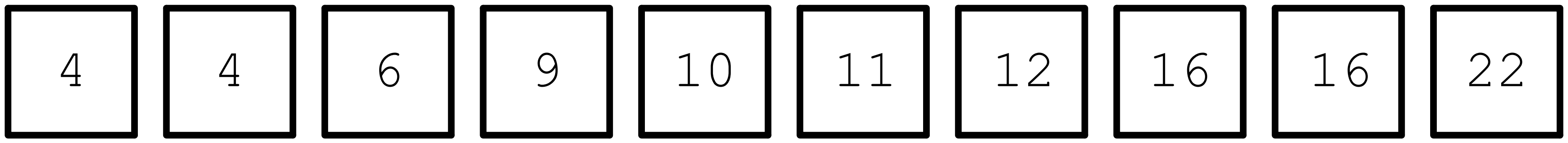
So given the initial list subdivision:



When it's recursive calls return, it will merge the following sorted lists:



result is in the correct sort order.


4 4 6 9 10 11 12 16 16 22