## Explanation of Heapsort

The *outer loop invariant* is that the rightmost portion of the array (colored red) consists of items in the **final** positions. The rest of the array (colored black consists of items in *max-heap order*. The outer loop invariant first holds after the initial heapification algorithm is executed. Whenever the outer loop invariant holds, the heap is restored, and the leftmost element is excchanged with the item to the left of the sorted items and colored red. The new leftmost item is then colored blue if it needs to bubbledown, and remains blue until it bubbles down to a position where the heap is restored.

## Explanation of Quicksort

The *outer loop invariant* is that the array consists of 0 or more items (colored red) which are in their **final** positions, and 0 or more unsorted subarrays (colored black) such that no black item is less than any red item to its left nor greater than any red item to its right.

The *inner loop invariant* is that the same, except that leftmost item of the unsorted subarray is called the *pivot* item (colored dark magenta), and the unsorted subarray consists of four parts: the key item, items (colored brown) which are known to be no larger than than the key, unsorted items (colored black), and items (colored green) known to be no smaller than the key.

When the outer loop invariant holds, the smallest black block is sorted first. If the number of items is 1, the block is already sorted, and its item is then colored red. If the number of items is 2, one comparison is made, and the items are possibly switched, and then colored red. first. If the number of items is 3, bubblesort or selection sort is used, since those are the most efficient methods for n = 3. (No examples of this are shown in Figure 2. If the number of items is greater than 4, a pivot item is chosen and swapped to the front of the subarray, and the inner loop invariant now holds.

7 5 8 1 3 4 2 7 9 6

7 5 8 1 6 4 2 7 9 3

7 5 8 9 6 4 2 7 1 3

7 9 8 5 6 4 2 7 1 3

7 9 8 7 6 4 2 5 1 3

9 7 8 7 6 4 2 5 1 3

initial heapification complete

3 7 8 7 6 4 2 5 1 9

8 7 3 7 6 4 2 5 1 9

8 7 4 7 6 3 2 5 1 9

heap restored

1 7 4 7 6 3 2 5 8 9

7 1 4 7 6 3 2 5 8 9

7 7 4 1 6 3 2 5 8 9

7 7 4 5 6 3 2 1 8 9

heap restored

1 7 4 5 6 3 2 7 8 9

7 1 4 5 6 3 2 7 8 9

7 6 4 5 1 3 2 7 8 9

2 6 4 5 1 3 7 7 8 9

6 2 4 5 1 3 7 7 8 9

6 5 4 2 1 3 7 7 8 9

3 5 4 2 1 6 7 7 8 9

5 3 4 2 1 6 7 7 8 9

heap restored

1 3 4 2 5 6 7 7 8 9

4 3 1 2 5 6 7 7 8 9

heap restored

2 3 1 4 5 6 7 7 8 9

3 2 1 4 5 6 7 7 8 9

heap restored

1 2 3 4 5 6 7 7 8 9

2 1 3 4 5 6 7 7 8 9

heap restored

1 2 3 4 5 6 7 7 8 9

1 2 3 4 5 6 7 7 8 9

Figure 1: Heapsort Steps

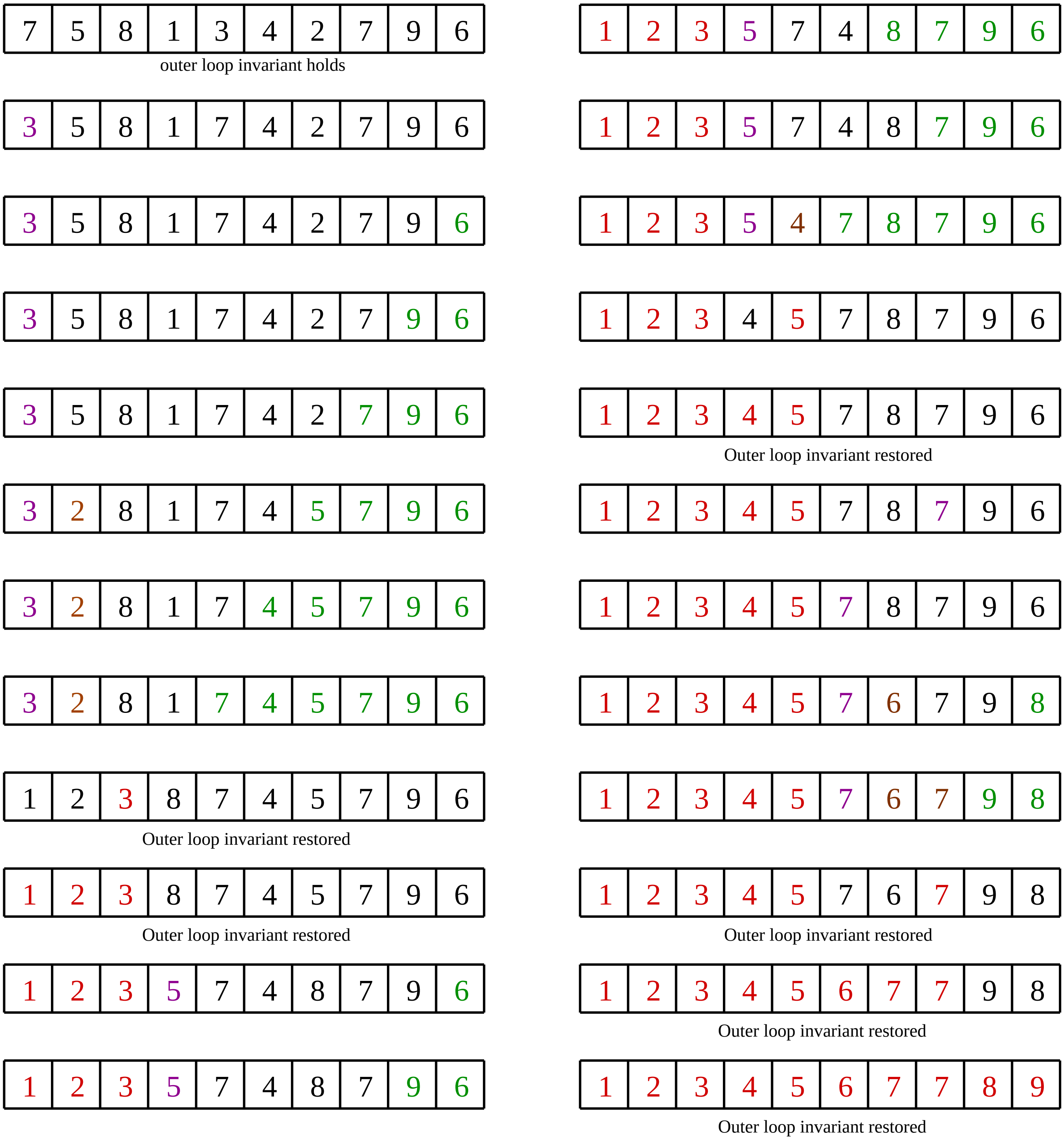


Figure 2: Quicksort Steps