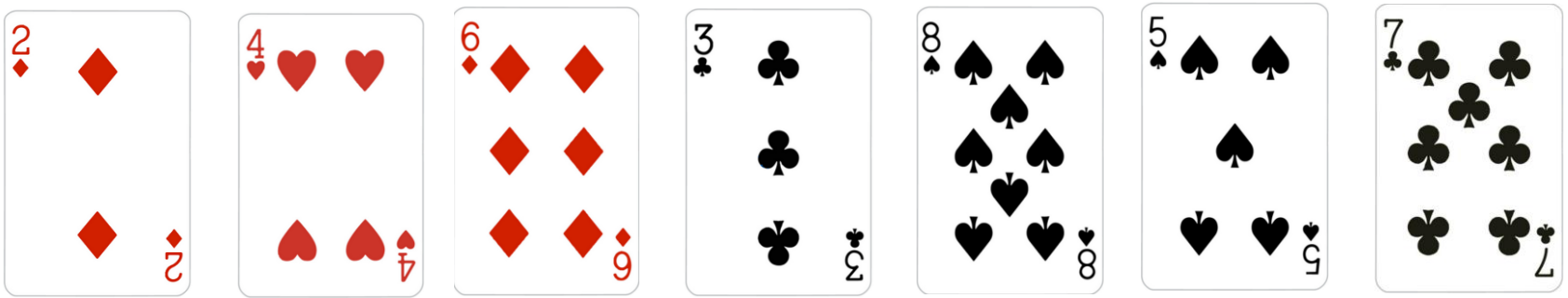
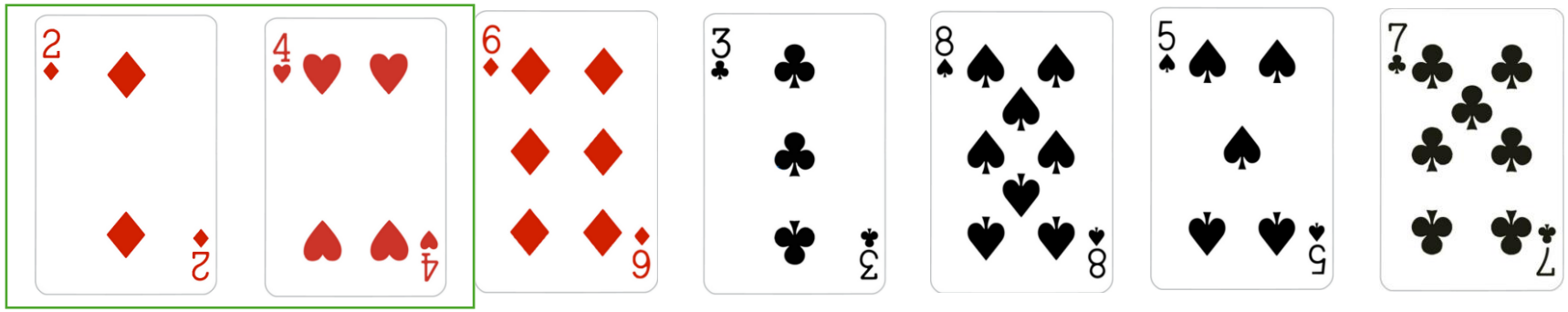
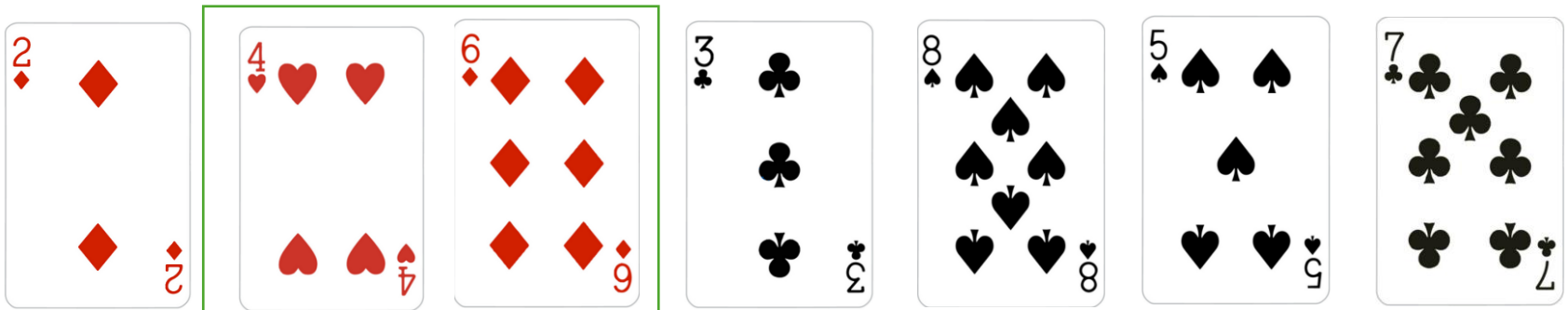
Diagrams of Sorting Algorithms – Roy Van

1. Bubble sort

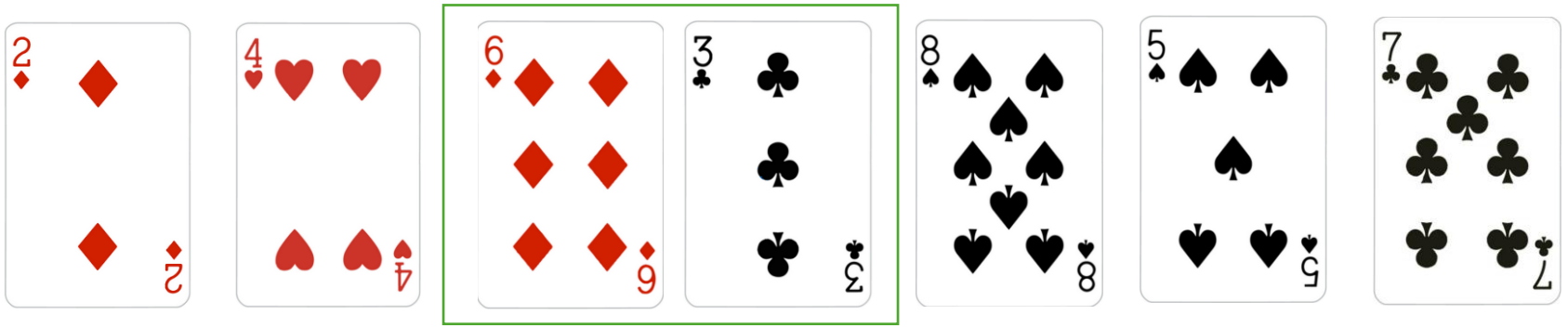




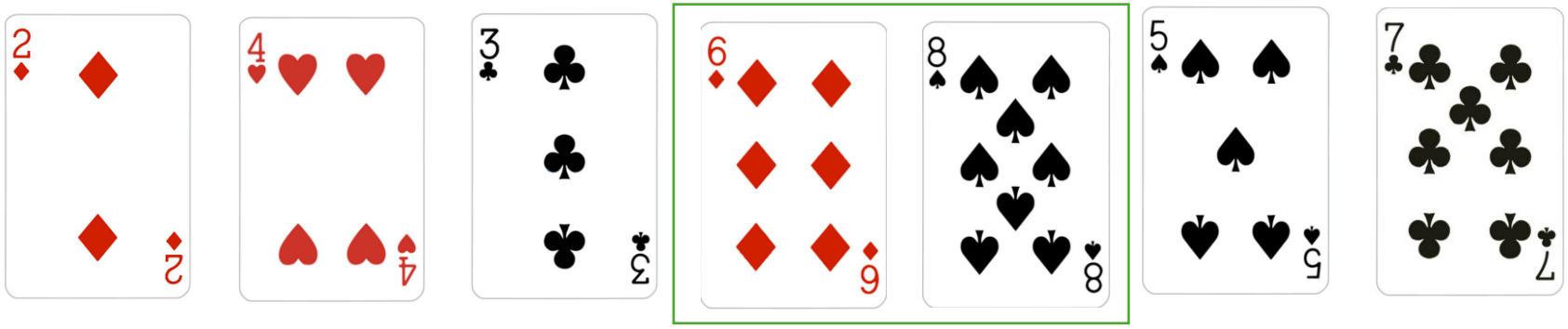
Compare: 2>4, not swap.



Compare: 4<6, not swap.

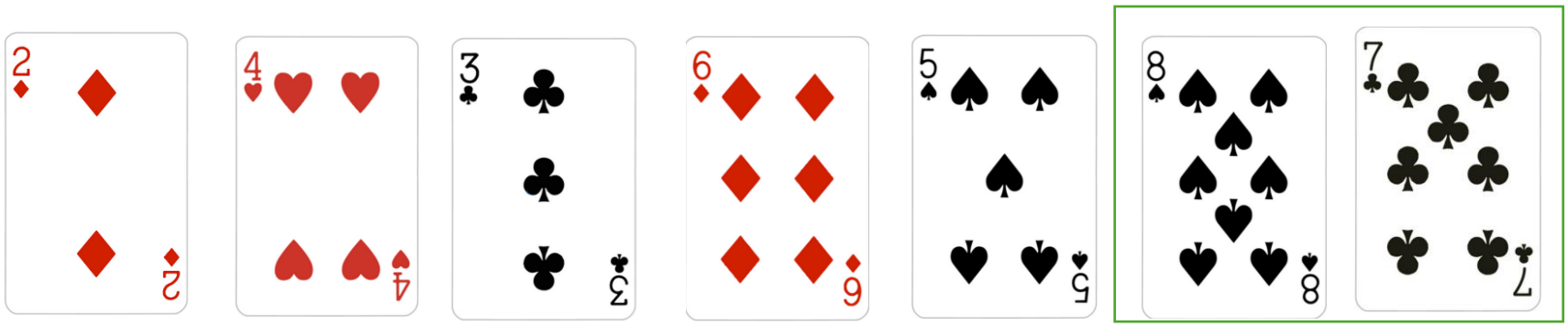


Compare: 6>3, swap.

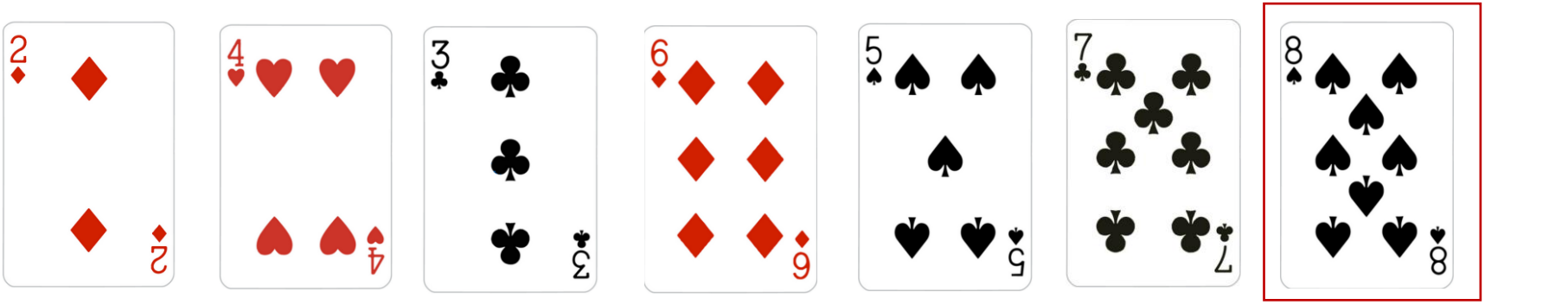


Compare: 6<8, not swap.

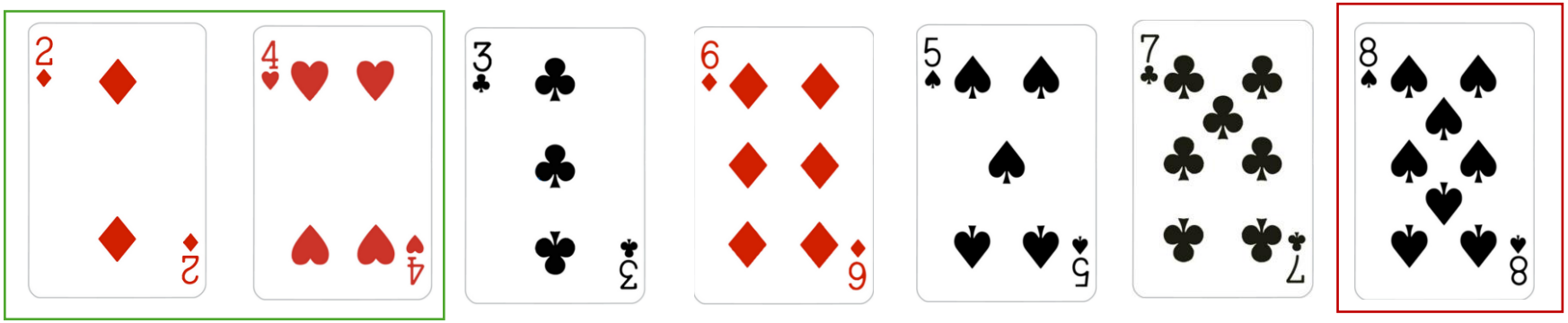
Compare: 8>5, swap.



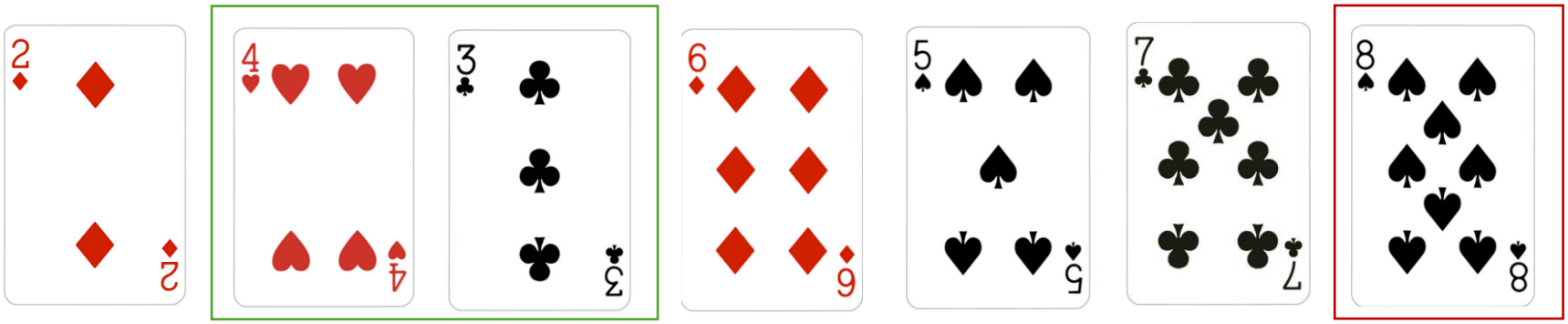
Compare: 8>7, swap.



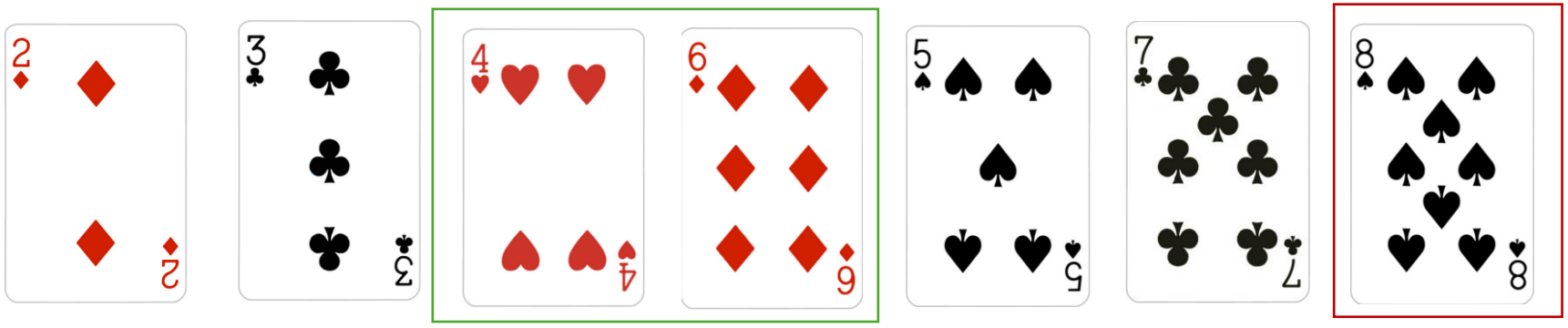
8 is already in the correct position



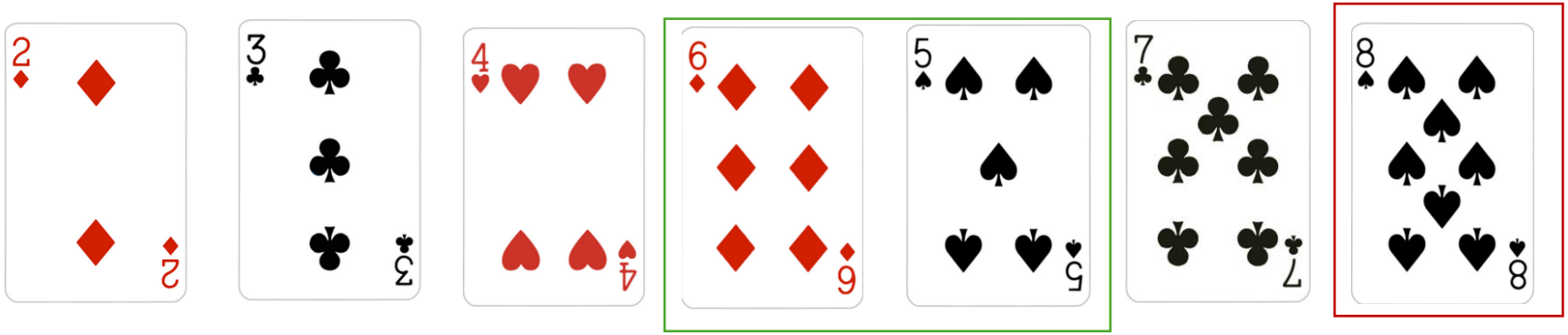
Compare: 2<4, not swap.



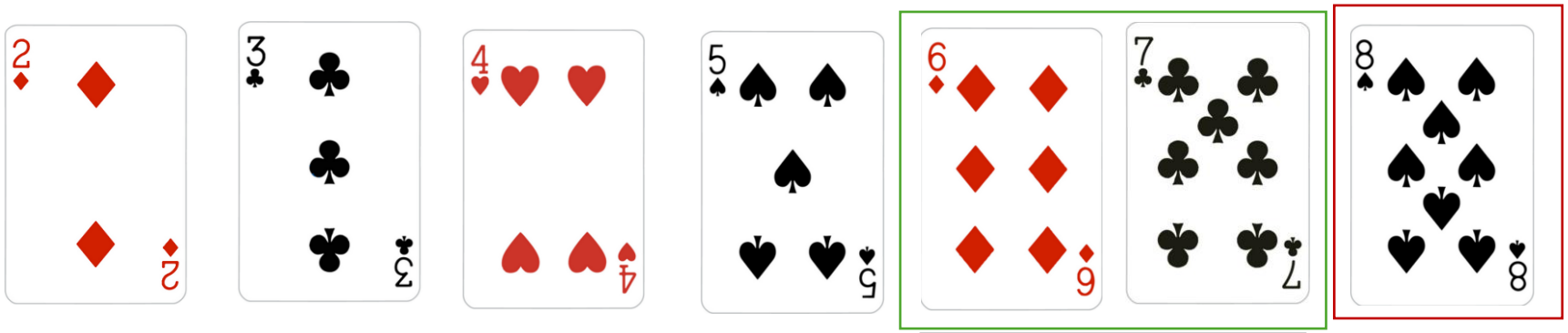
Compare: 4>3, swap.



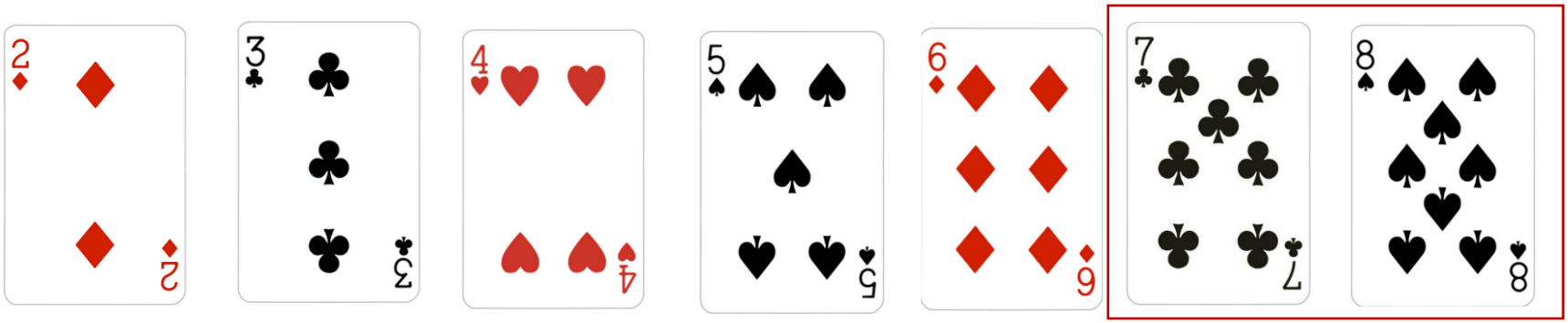
Compare: 4<6, not swap.



Compare: 6>5, swap.

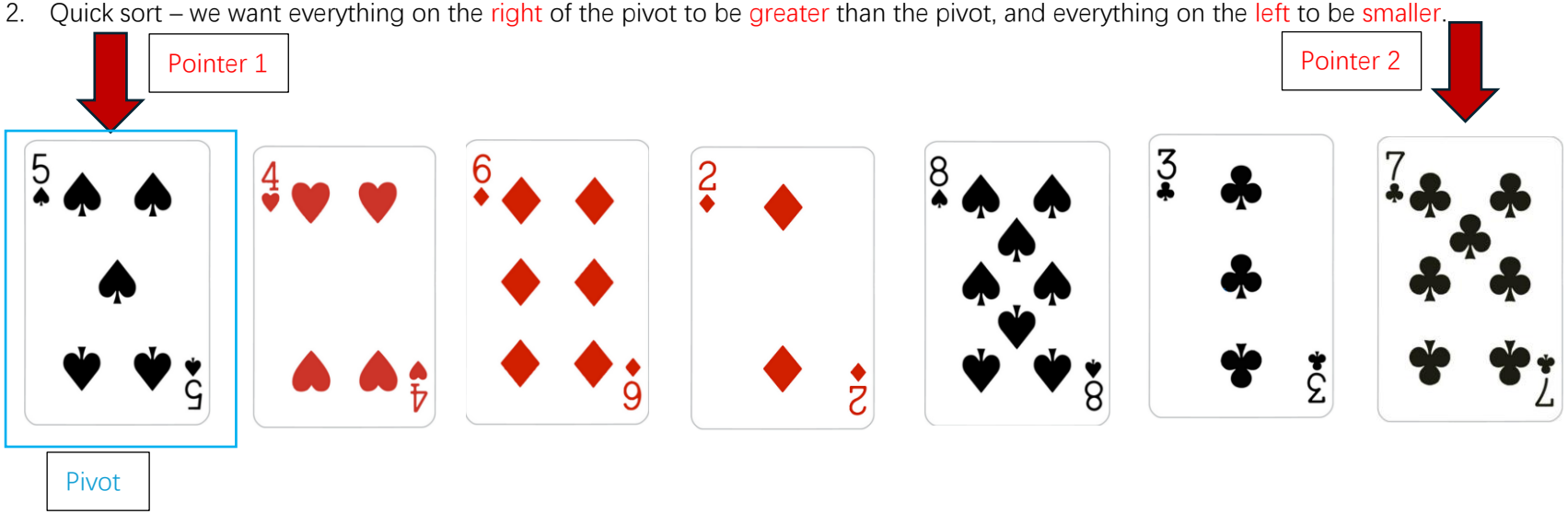


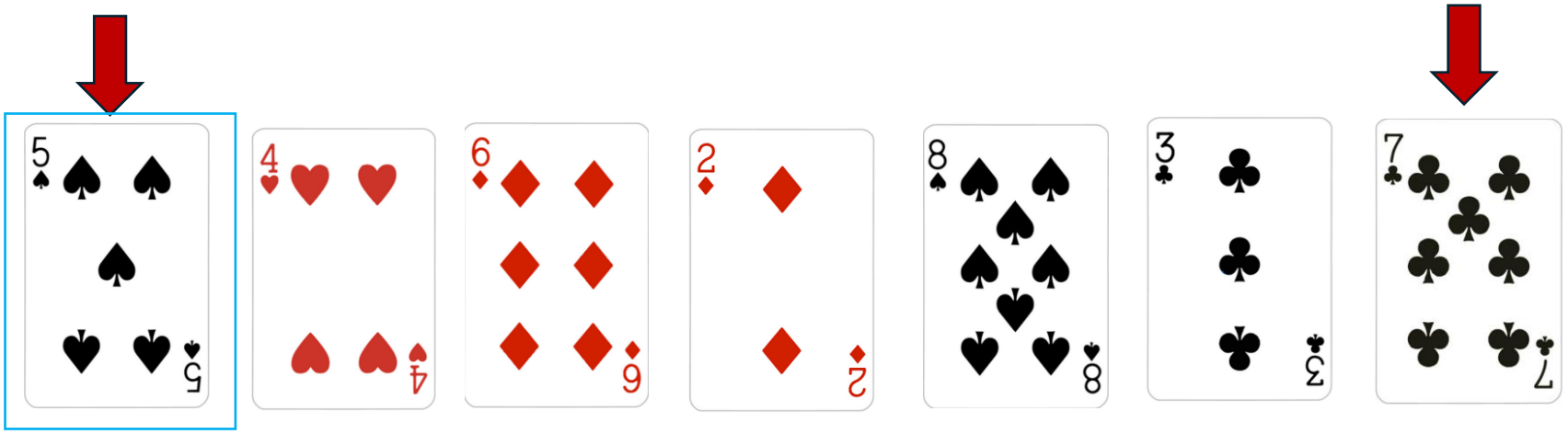
Compare: 6<7, not swap.



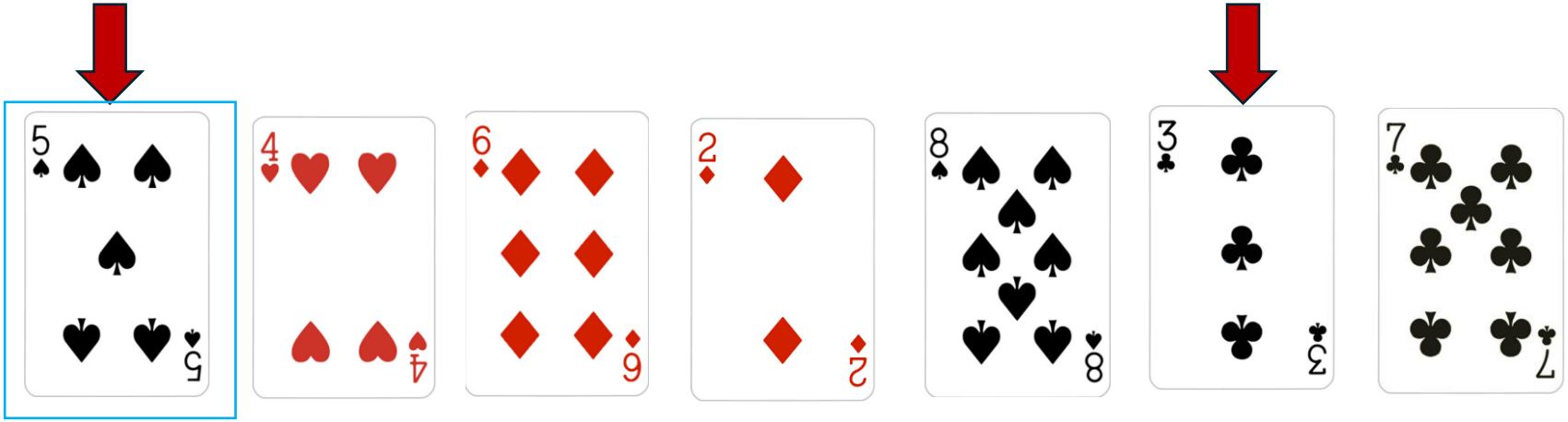
This array is already sorted, but the computer will go through the entire array again until all the cards are within the “red box”

2. Quick sort – we want everything on the right of the pivot to be greater than the pivot, and everything on the left to be smaller.

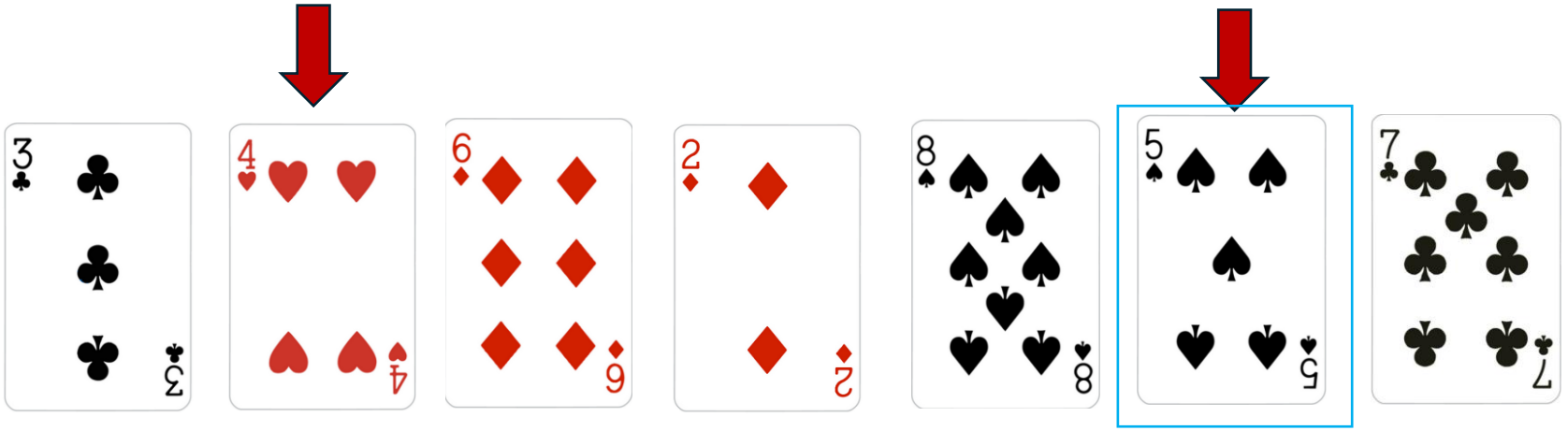




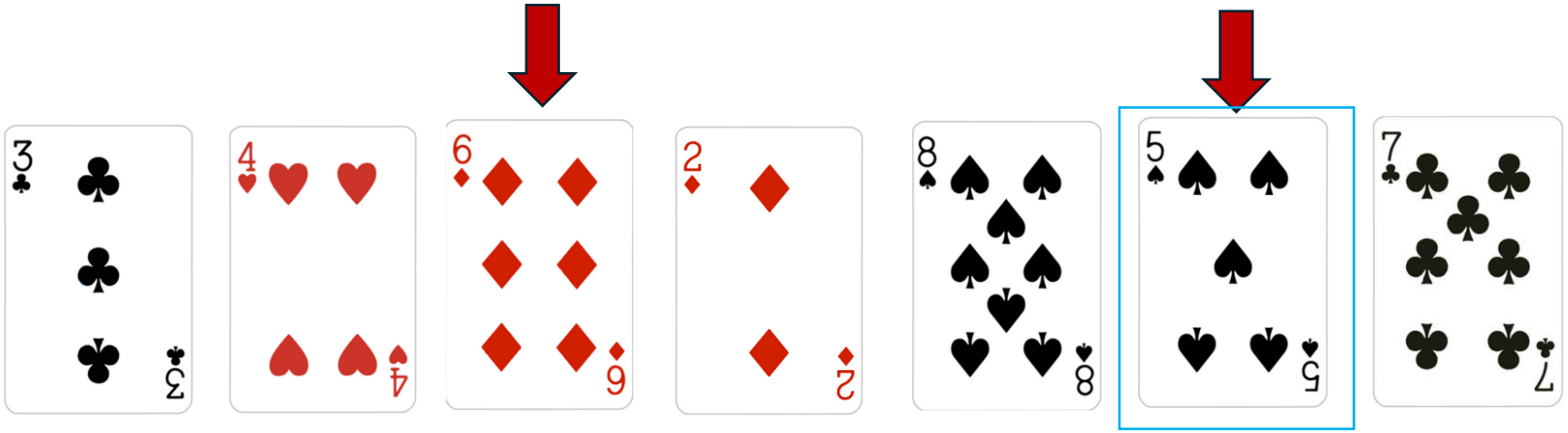
Compare: 5 < 7 (pivot vs. another pointer), not swap.
Then: shift the pointer.



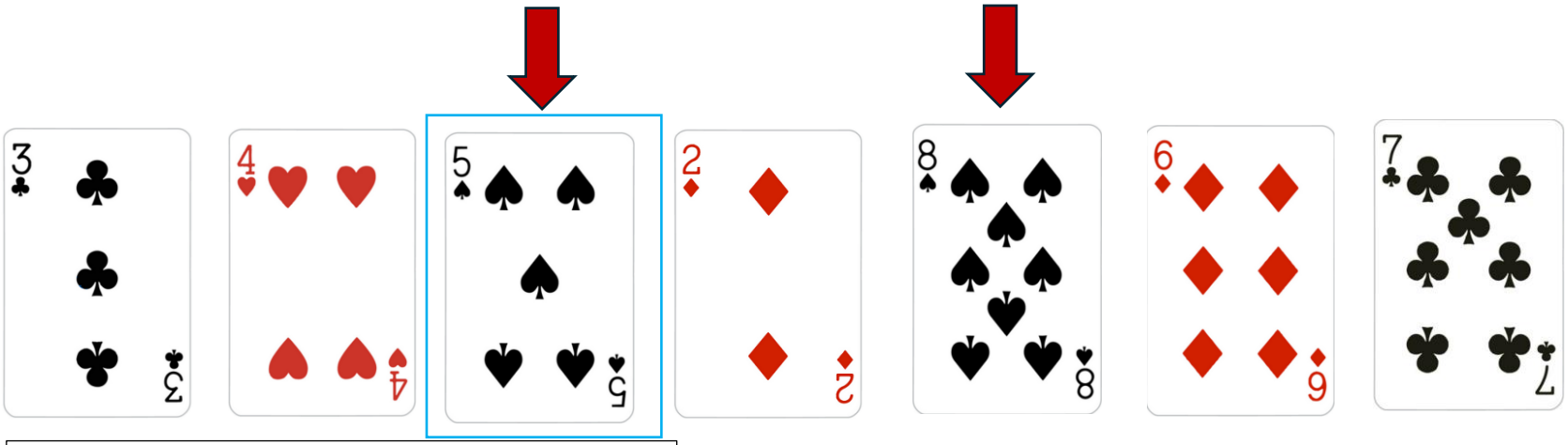
Compare: 5 > 3 (pivot vs. another pointer), swap.
Then: shift the pointer.



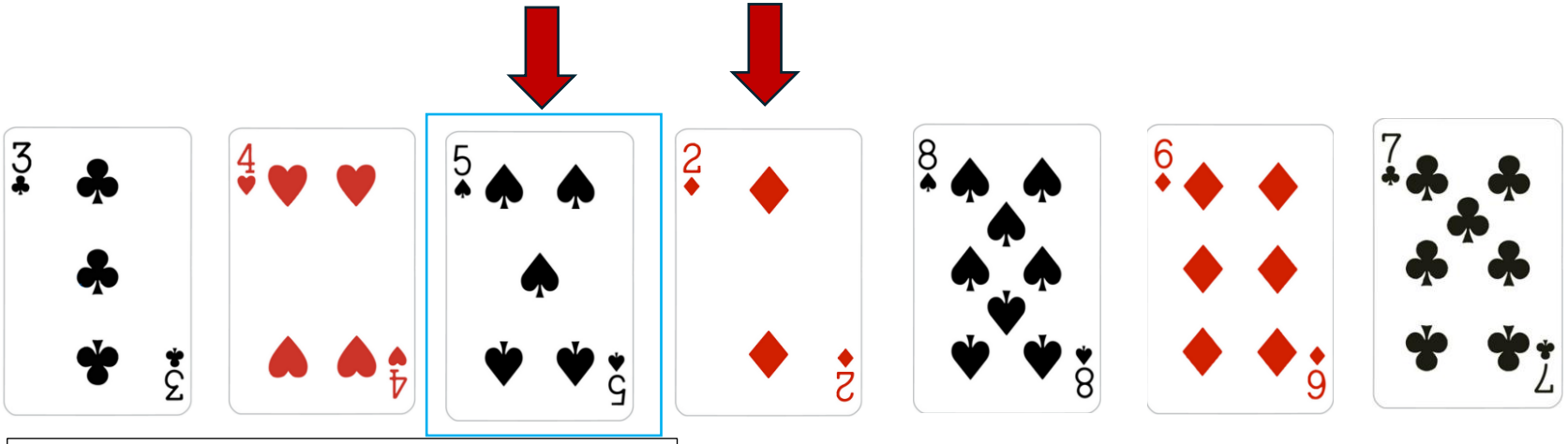
Compare: 5 > 4 (pivot vs. another pointer), not swap.
Then: shift the pointer.



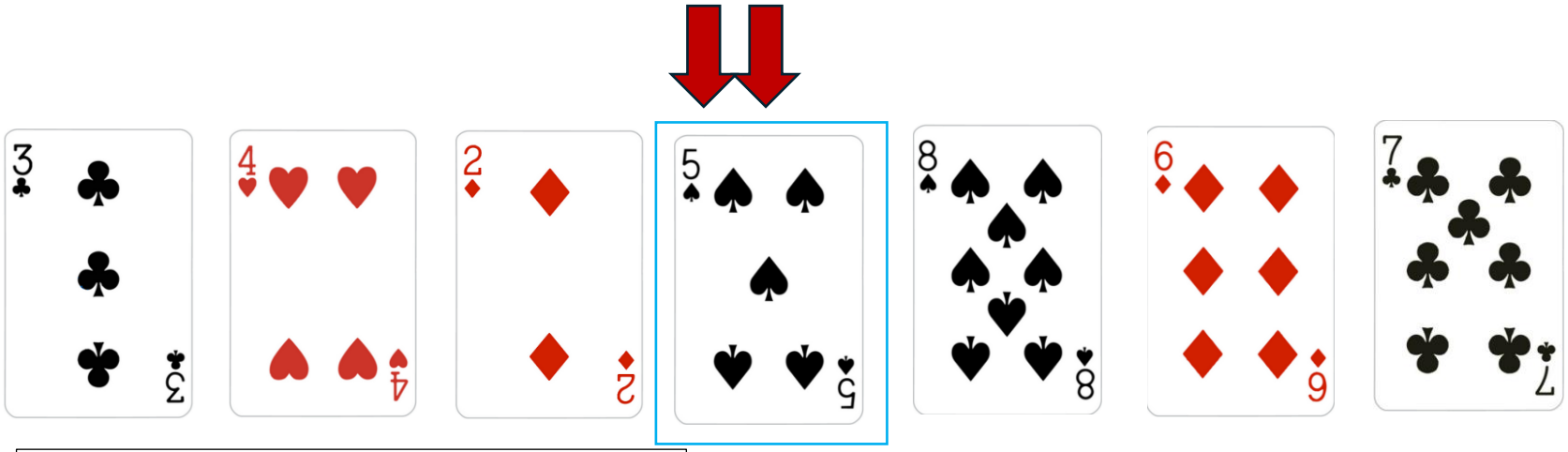
Compare: 5 < 6 (pivot vs. another pointer), swap.
Then: shift the pointer.



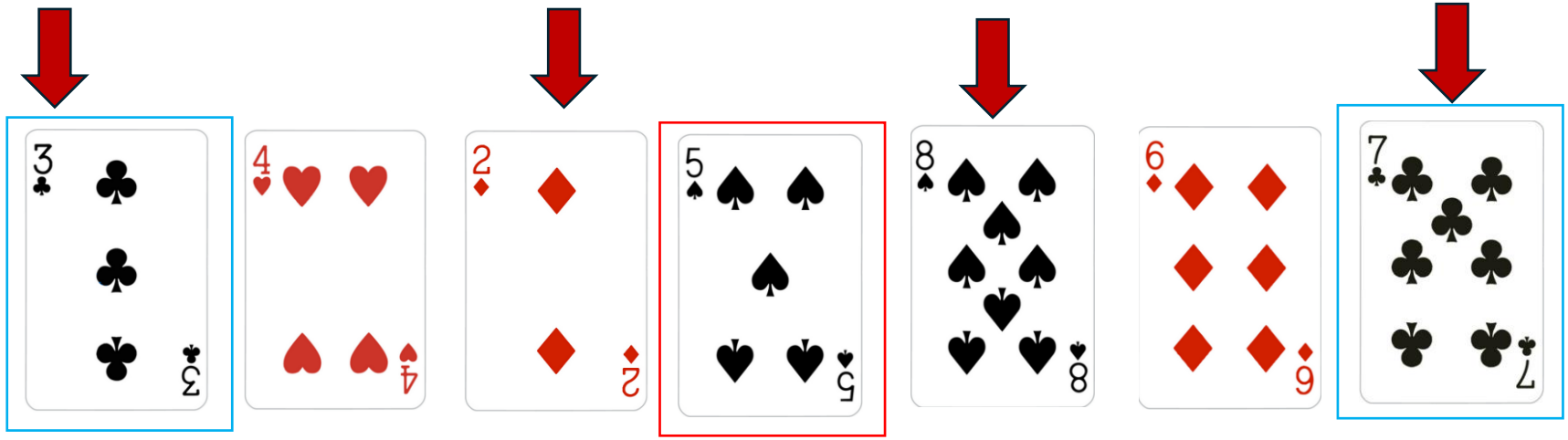
Compare: 5 < 8 (pivot vs. another pointer), not swap.
Then: shift the pointer.



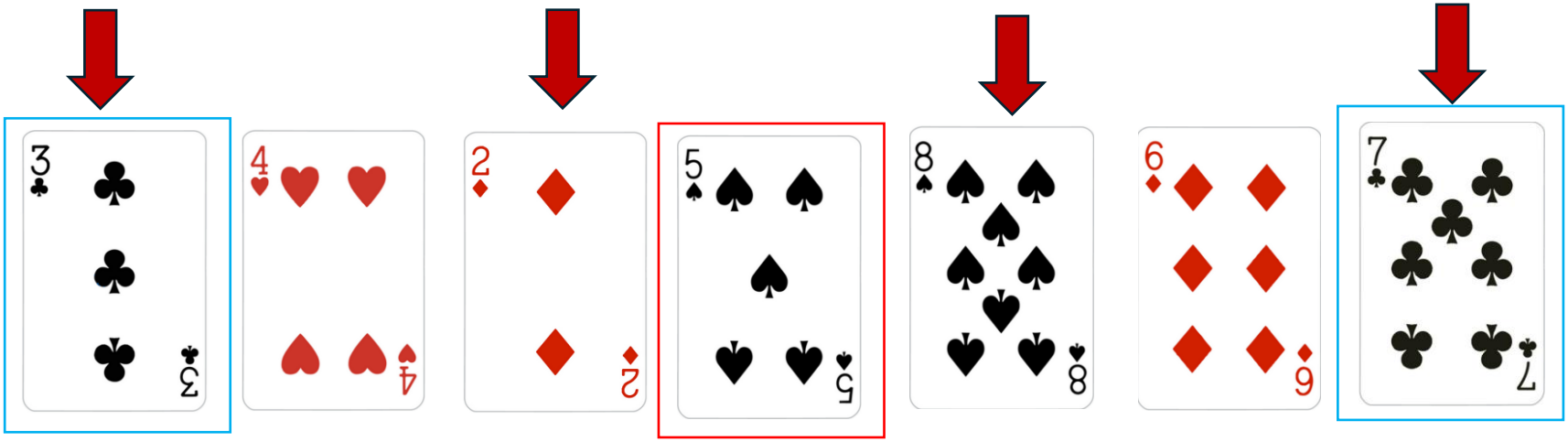
Compare: 5 > 2 (pivot vs. another pointer), swap.
Then: shift the pointer.



Two pivots meet each other, first round done.
Now 5 is in the correct position.

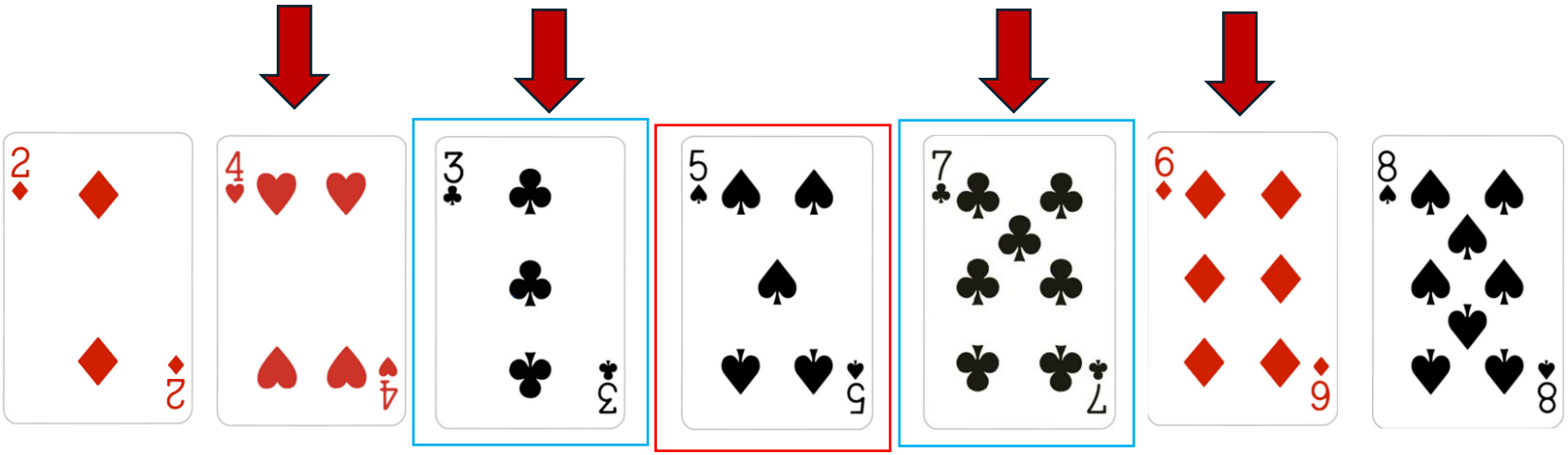


We could repeat the same algorithm on both sides of the "position confirmed card" 5, now we try to do them simultaneously.



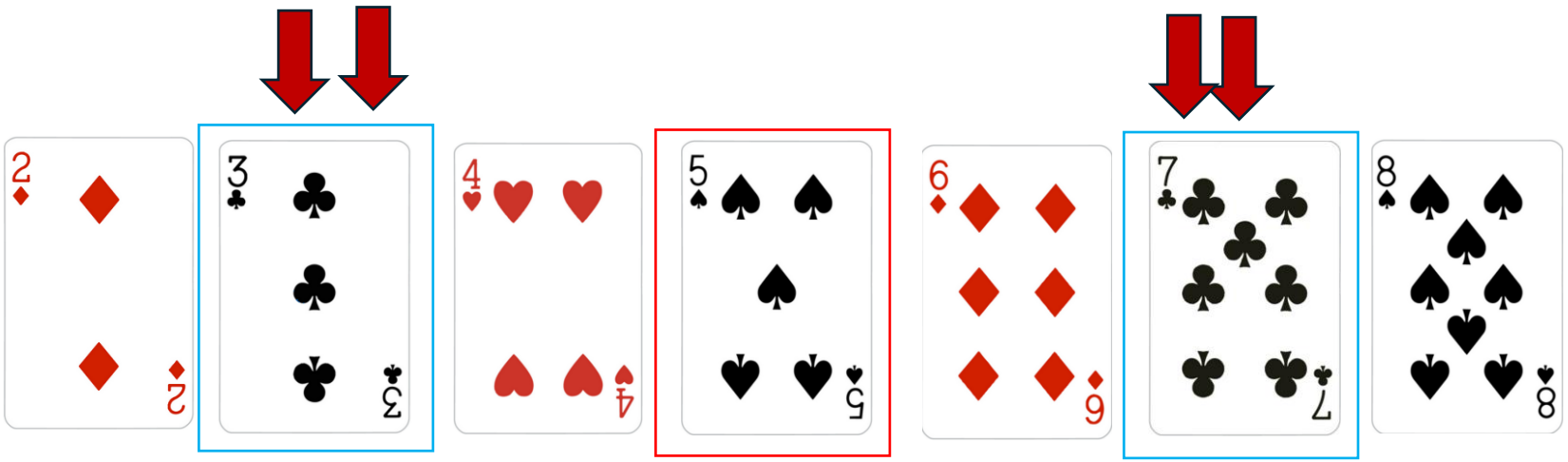
Compare: 3 > 2 (pivot vs. another pointer), swap.
Then: shift the pointer.

Compare: 8 > 7 (pivot vs. another pointer), swap.
Then: shift the pointer.

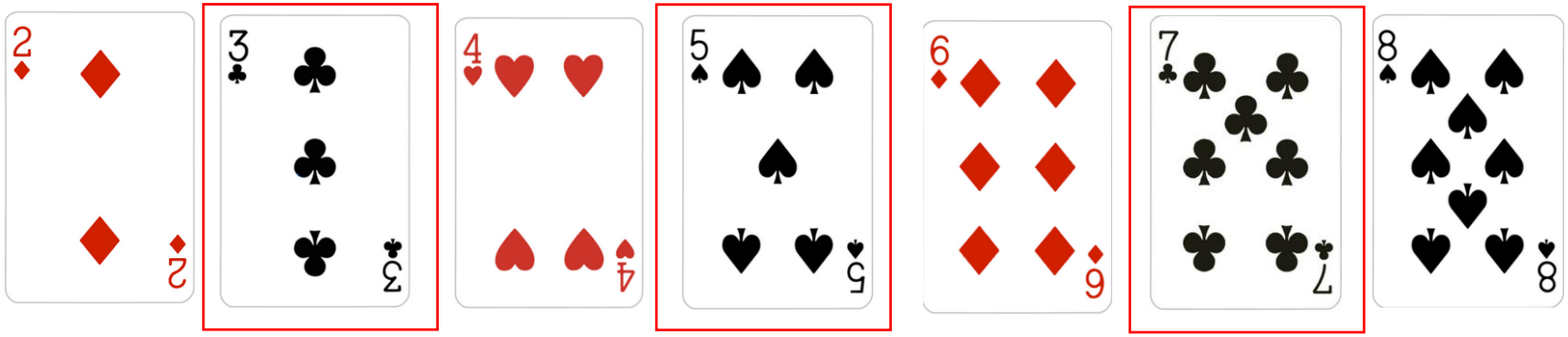


Compare: 3 < 4 (pivot vs. another pointer), swap.
Then: shift the pointer.

Compare: 7 > 6 (pivot vs. another pointer), swap.
Then: shift the pointer.

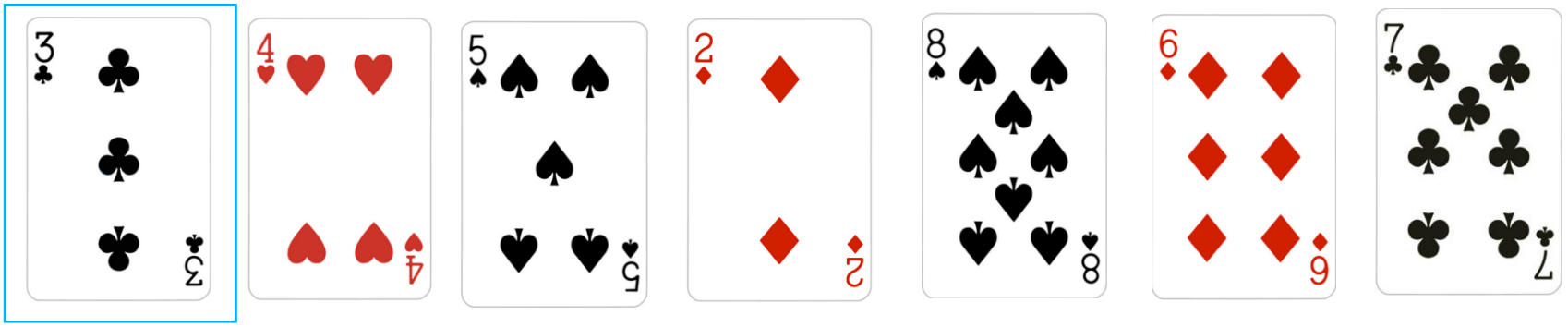


Two pivots meet each other, another two rounds end, now we have three cards sorted.

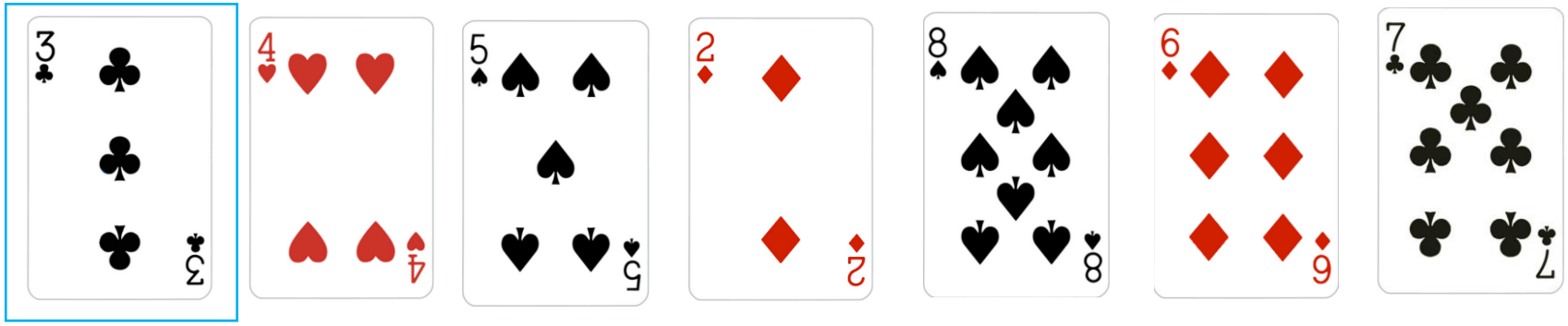


This array is already sorted.

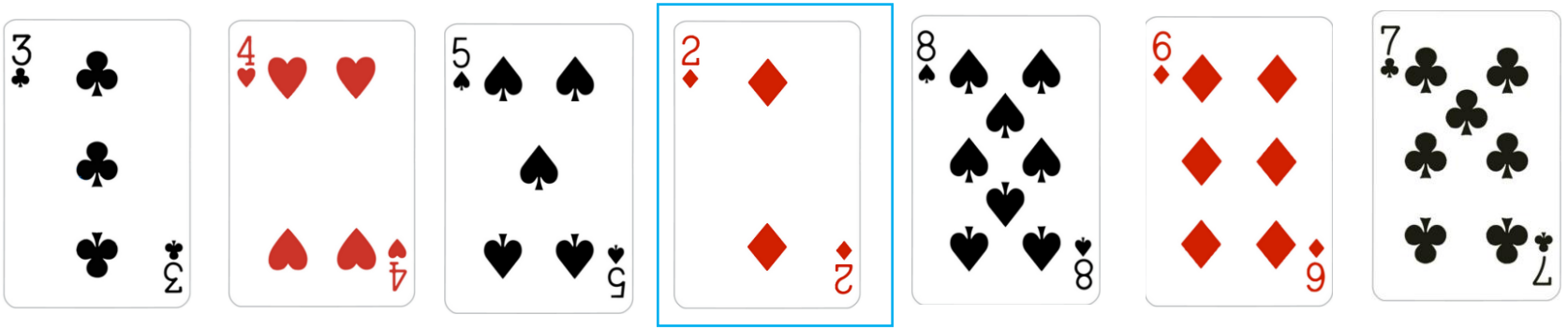
3. Selection sort



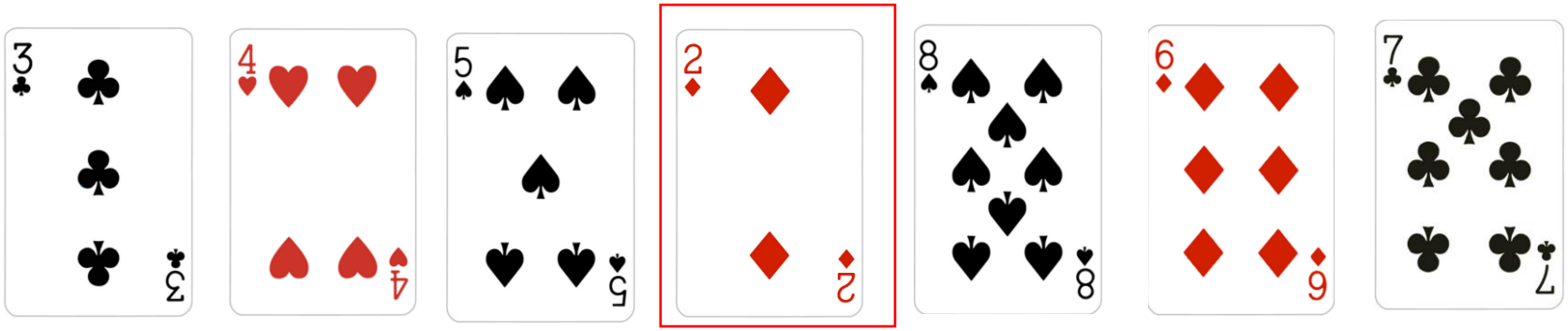
Find the smallest in the array. So let the first card be the “temporary smallest”



Find the smallest in the array. So let the first card be the “temporary smallest”.



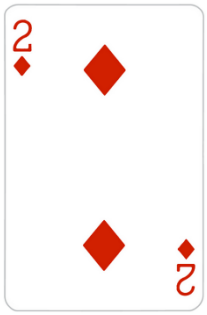
Then we go through a traversal.
Is 4 less than 3? No.
Is 5 less than 3? No.
Is 2 less than 3? Yes, so 2 is the new “temporary smallest”.
Is 8 less than 2? No.
Is 6 less than 2? No.
Is 7 less than 2? No.
Therefore, 2 is the smallest number in this array. We will use the same method to find the smallest number.



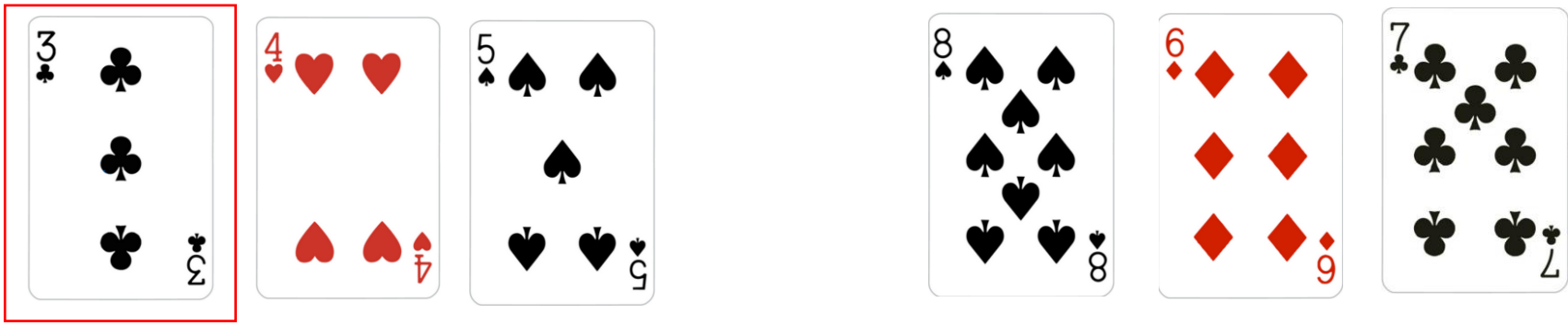
Then create a new, empty array, in order to store the result. We put the smallest found into the new array.

Original Array

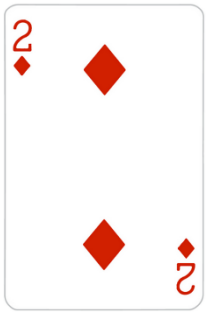
New Array



Original Array

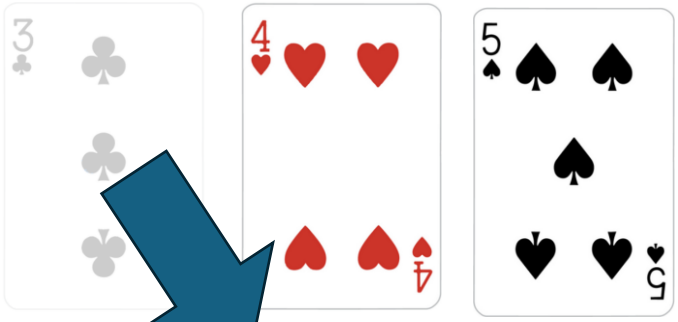


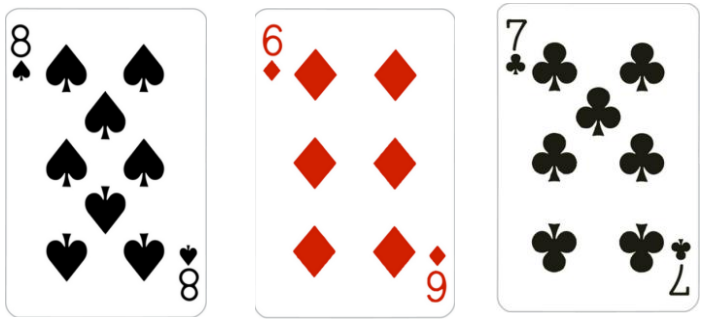
New Array



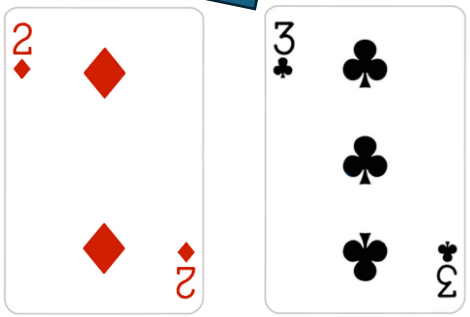
Repeat finding the smallest number from Original Array, and put it at the end of New Array

Original Array

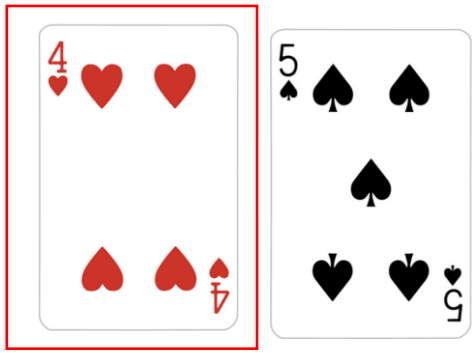


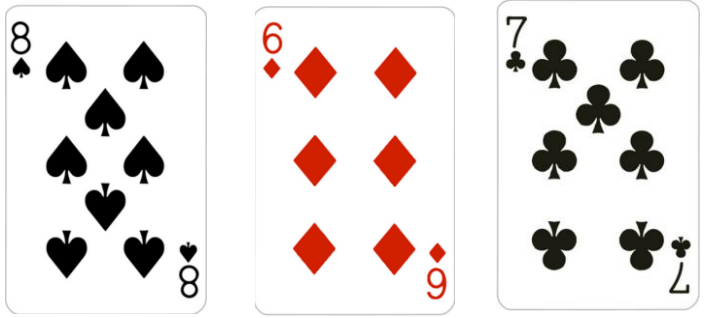


New Array

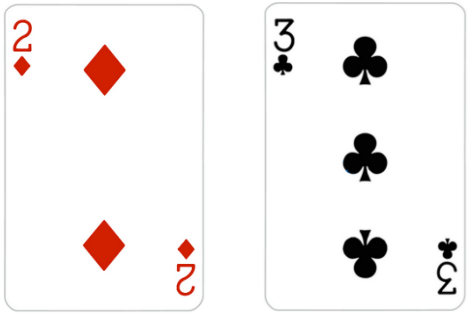


Original Array

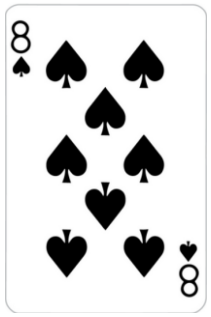


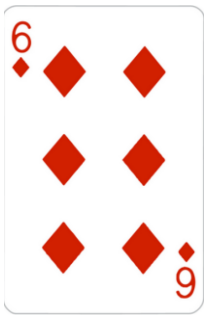


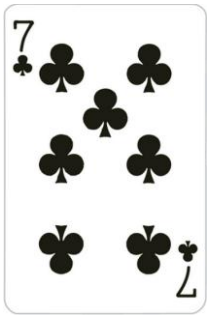
New Array



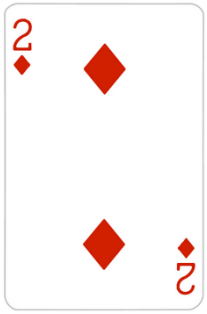
Original Array

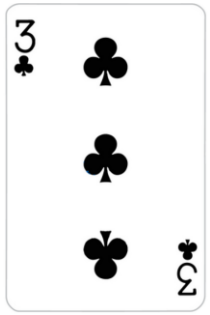


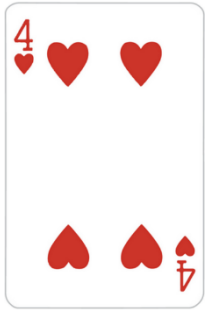




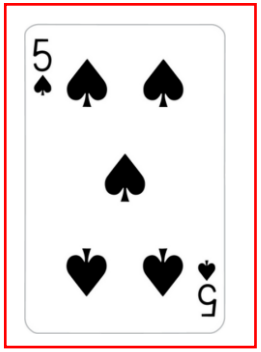
New Array

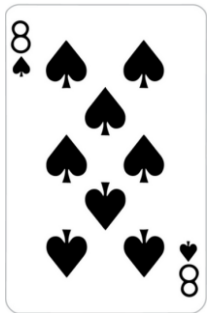


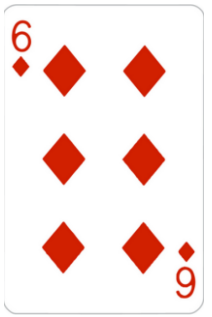


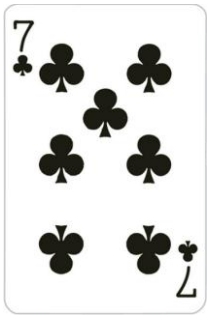


Original Array

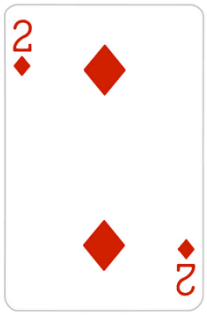


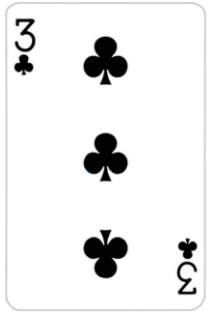


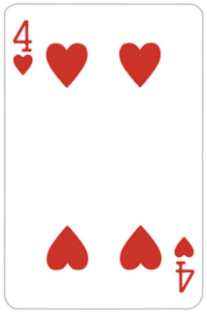




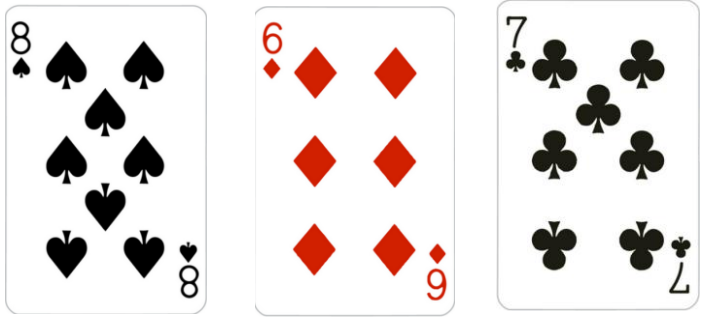
New Array



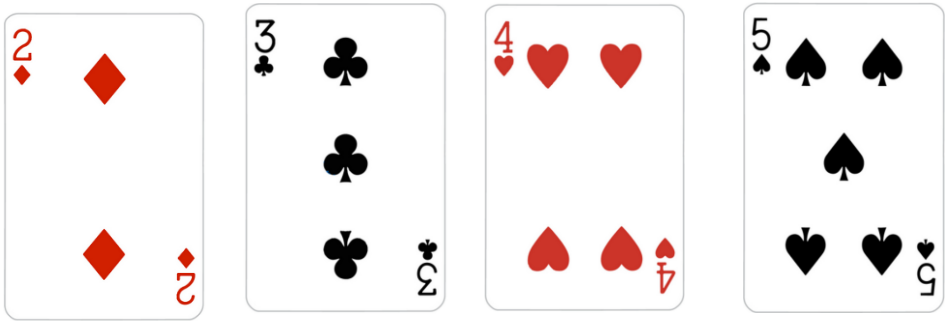




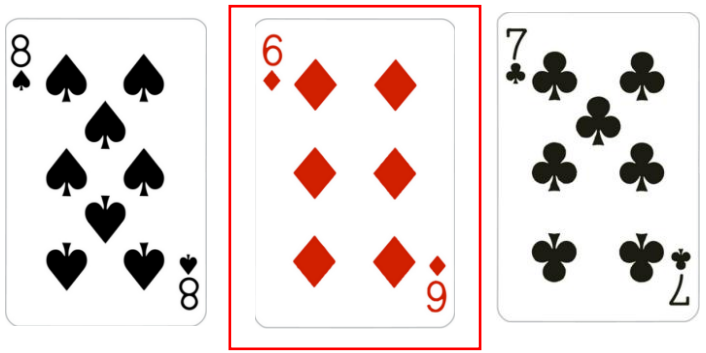
Original Array



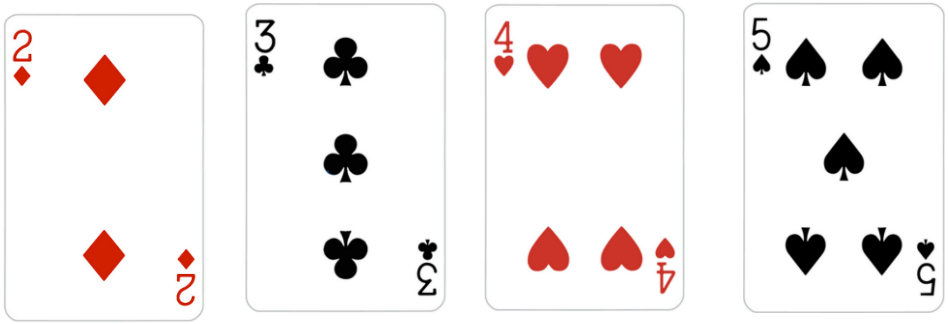
New Array



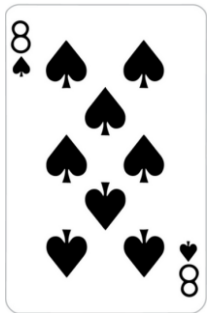
Original Array

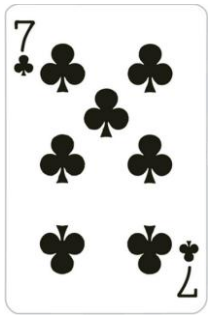


New Array

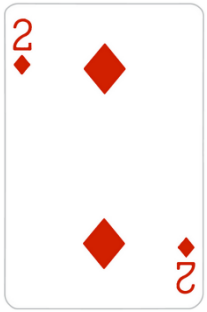


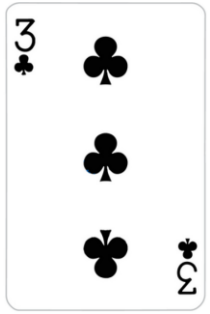
Original Array

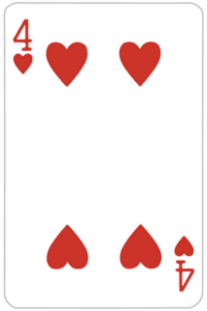


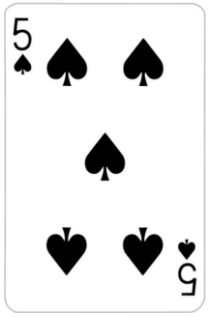


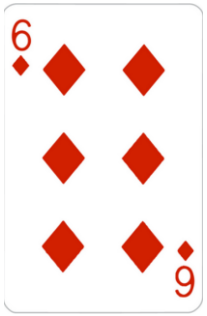
New Array



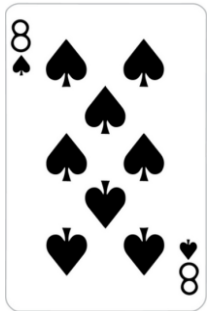


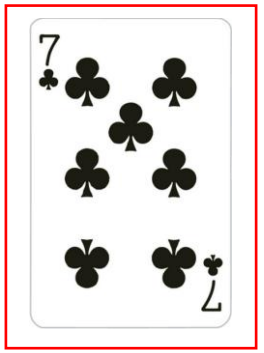






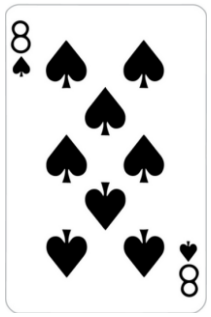
Original Array

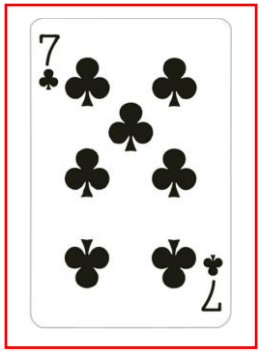




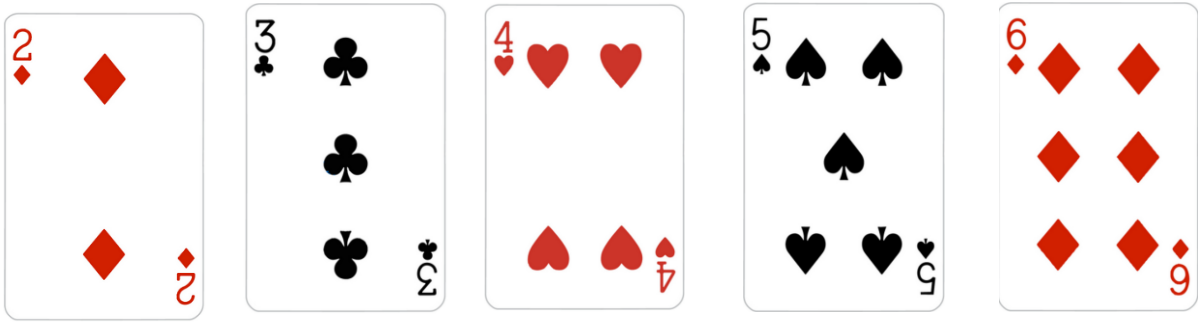
New Array

Original Array

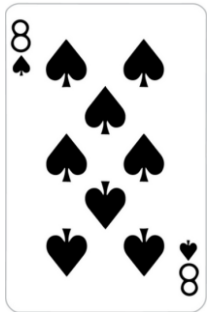




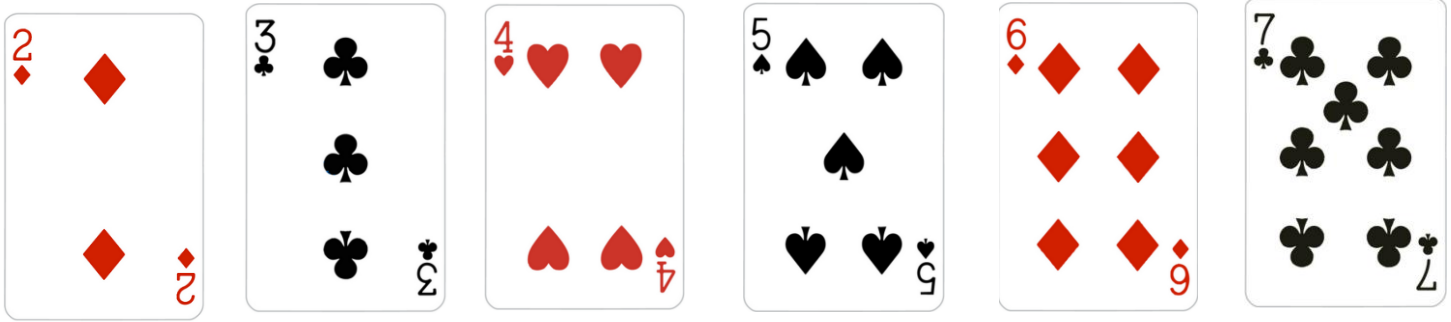
New Array



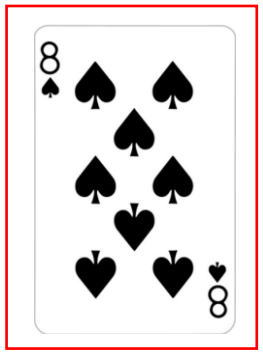
Original Array



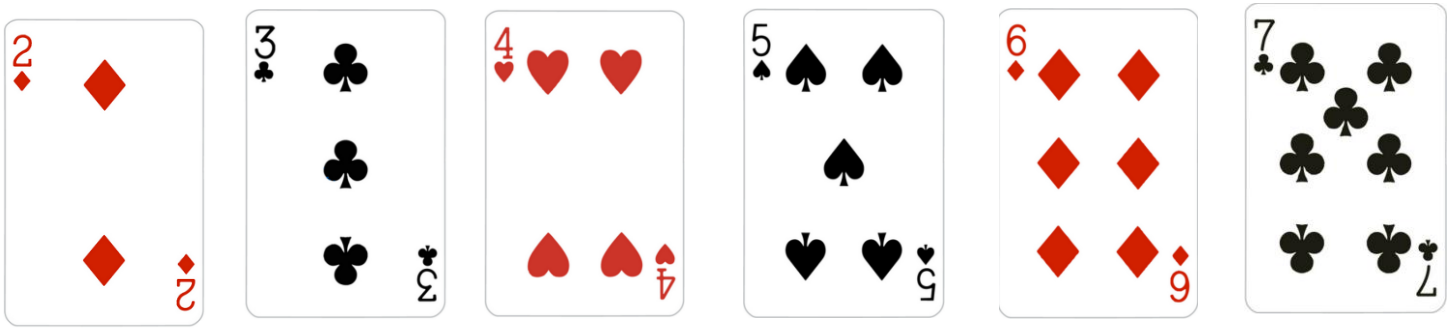
New Array



Original Array



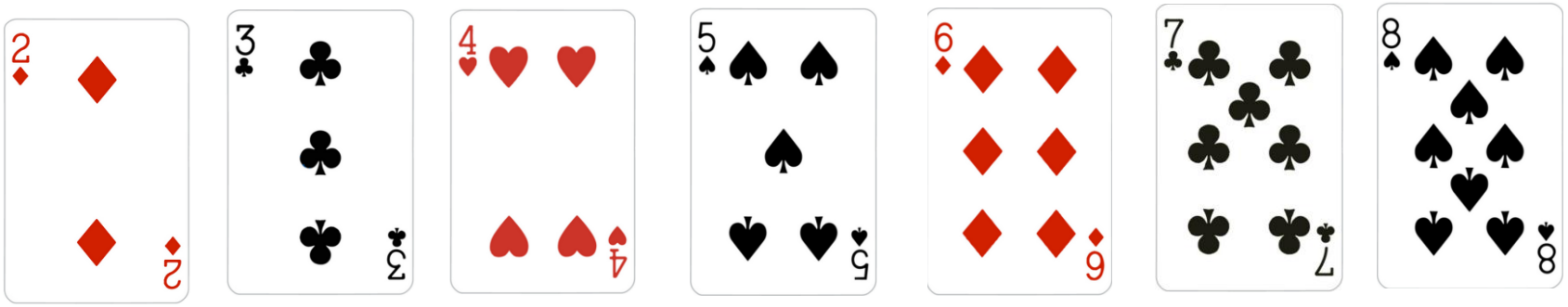
New Array



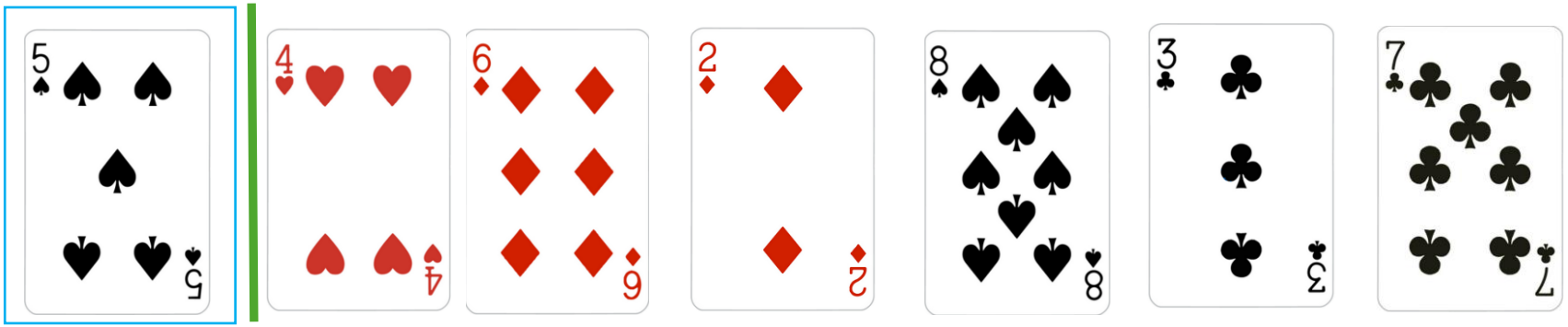
Original Array

When there is no element in the Original Array, selection sort is done.

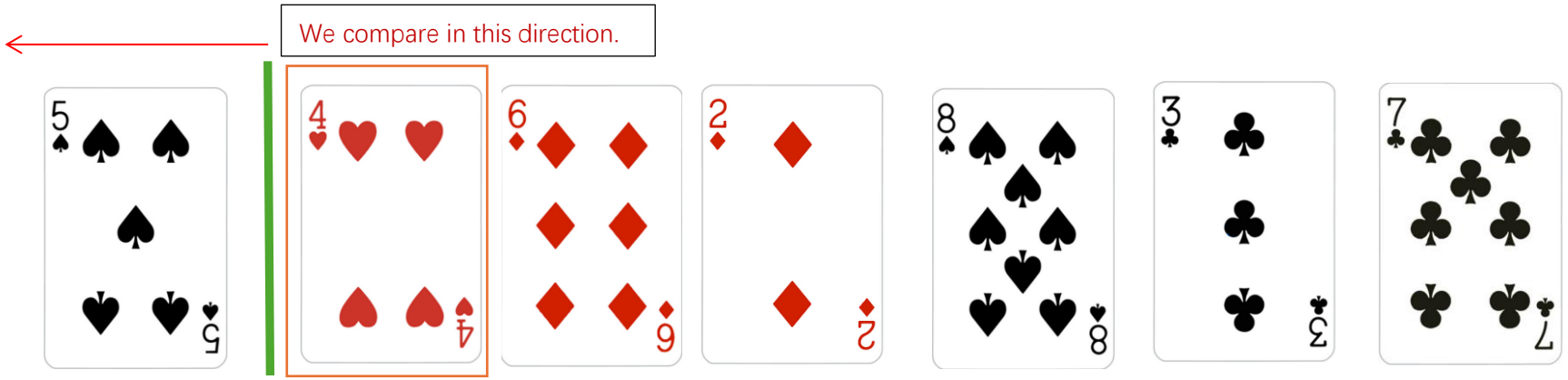
New Array



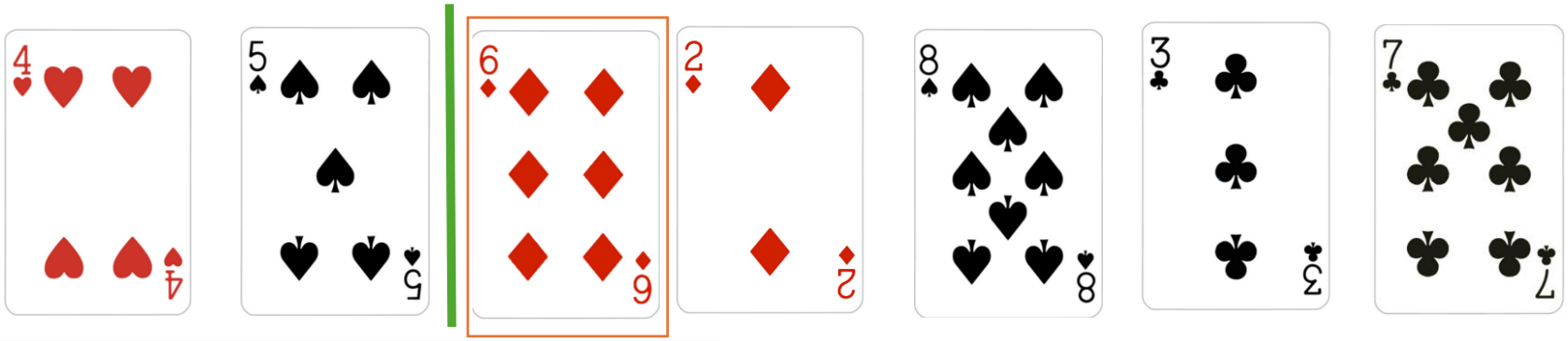
4. Insertion sort



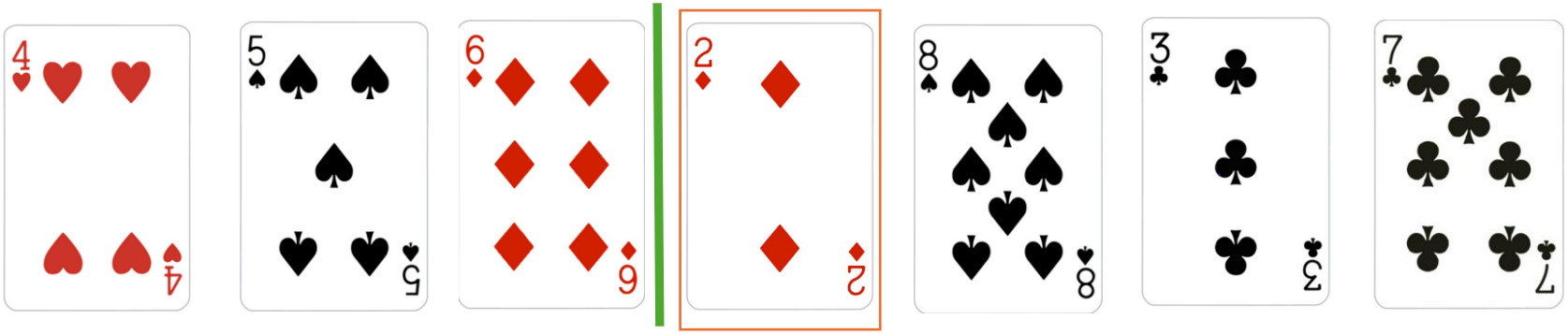
The first element in insertion sort is already sorted, and we want to create a vertical line at the right side of the first element, if there is no element on the right of the line, we finish sorting.



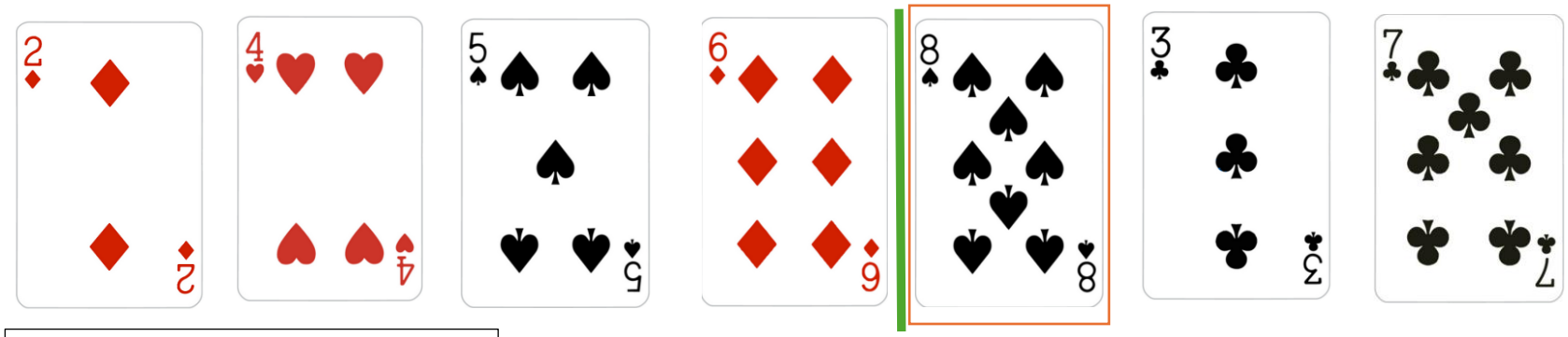
Starting on the first element on the right of the line (for our convenience, we call it key), we compare key to all the element on the left of the line. Starting from the element closest to the line, when key is greater than an element, we place key on the right of that element; OR when key is less than all the elements on the left of the line, then put it at the beginning of the array.



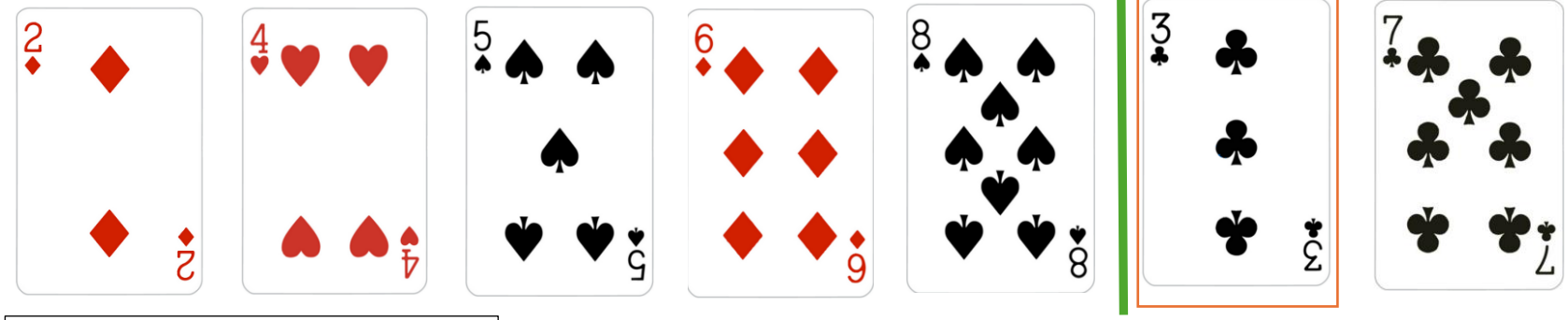
6 > 5, so we put it on the right of 5



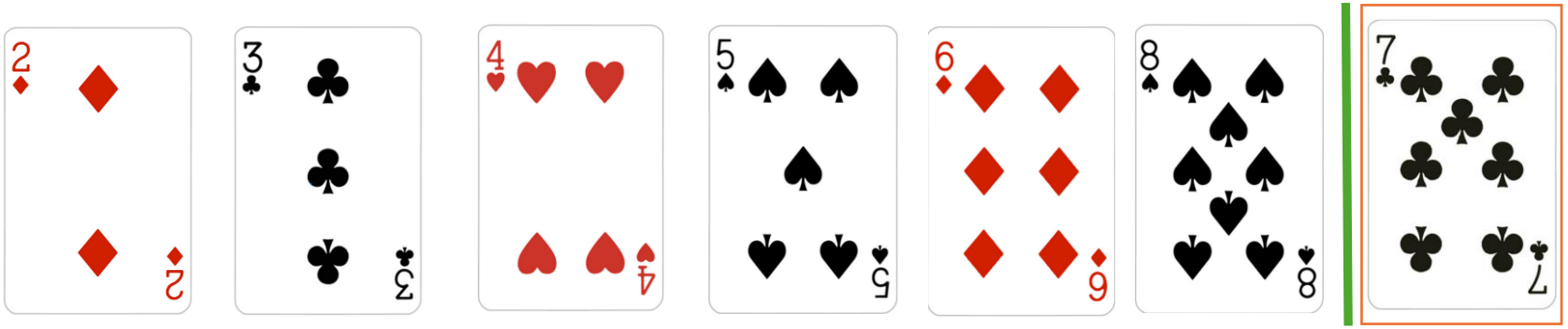
2 is less than all the elements on the left of the line, so we put it at the beginning of the array.



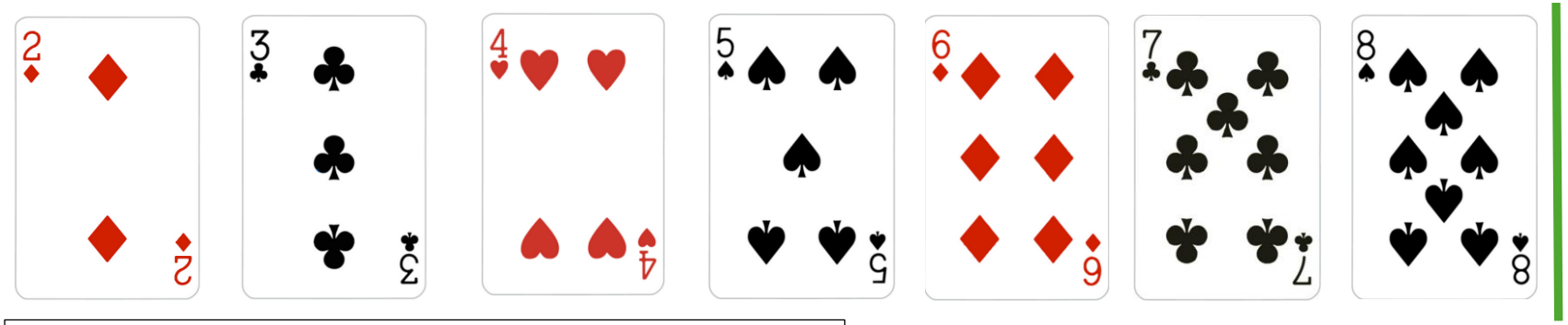
8 > 6, so we put it on the right of 6



3 > 2, so we put it on the right of 2



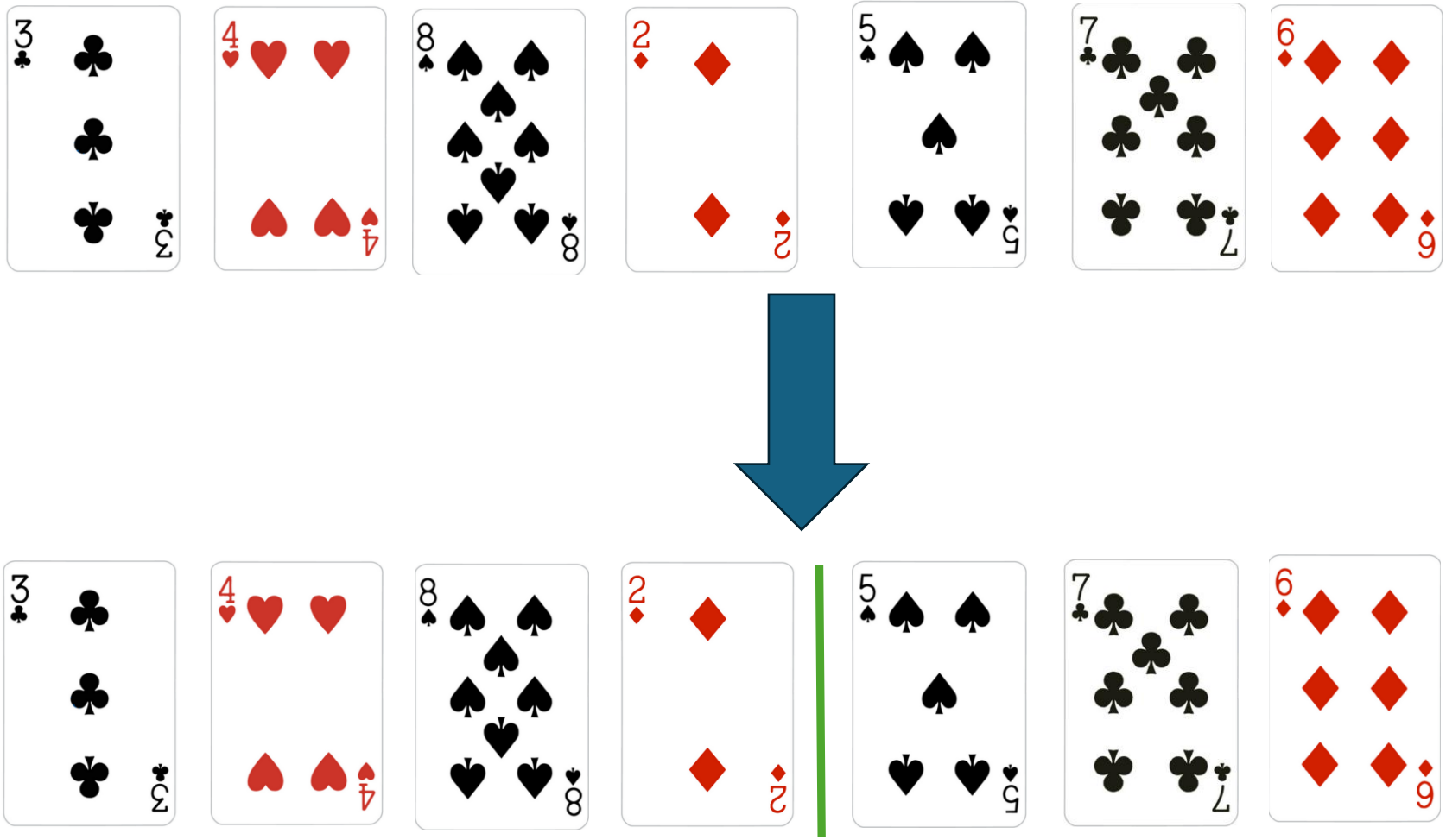
7 > 6, so we put it on the right of 6

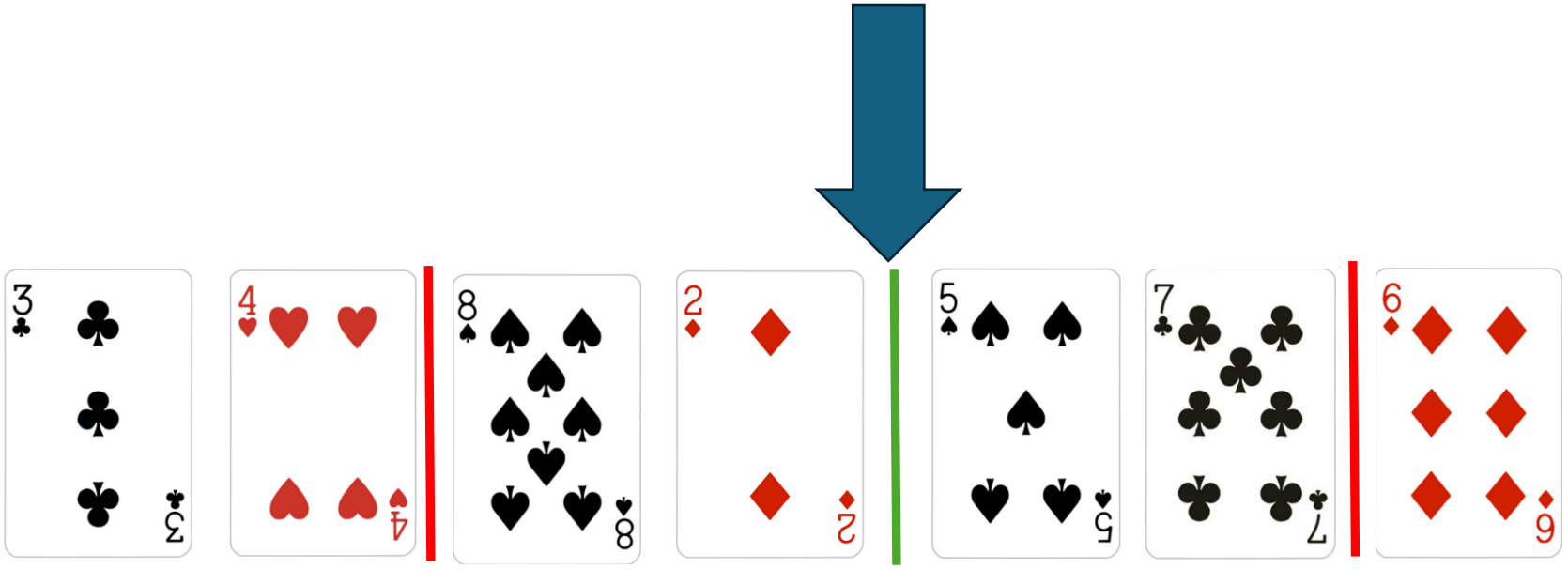


There is no element on the right of the line, we have done insertion sort.

5. Merge sort

In merge sort, our first goal is to separate all elements. (first index + end index) / 2 = (0 + 6) / 2 = 3, thus we separate the cards after the element at index 3. We repeat the same method, until we separated all the elements.



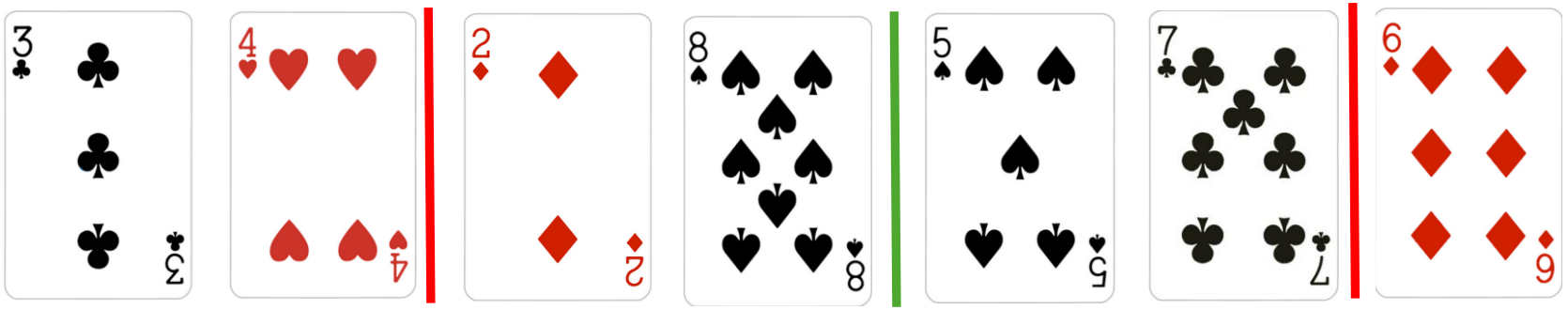


Separate 3 and 4, and merge them, but when we are merging them, order them from the smallest to the greatest.
Compare: 3 < 4, therefore not swap.

Separate 8 and 2, and merge them.
Compare: 8 > 2, therefore swap.

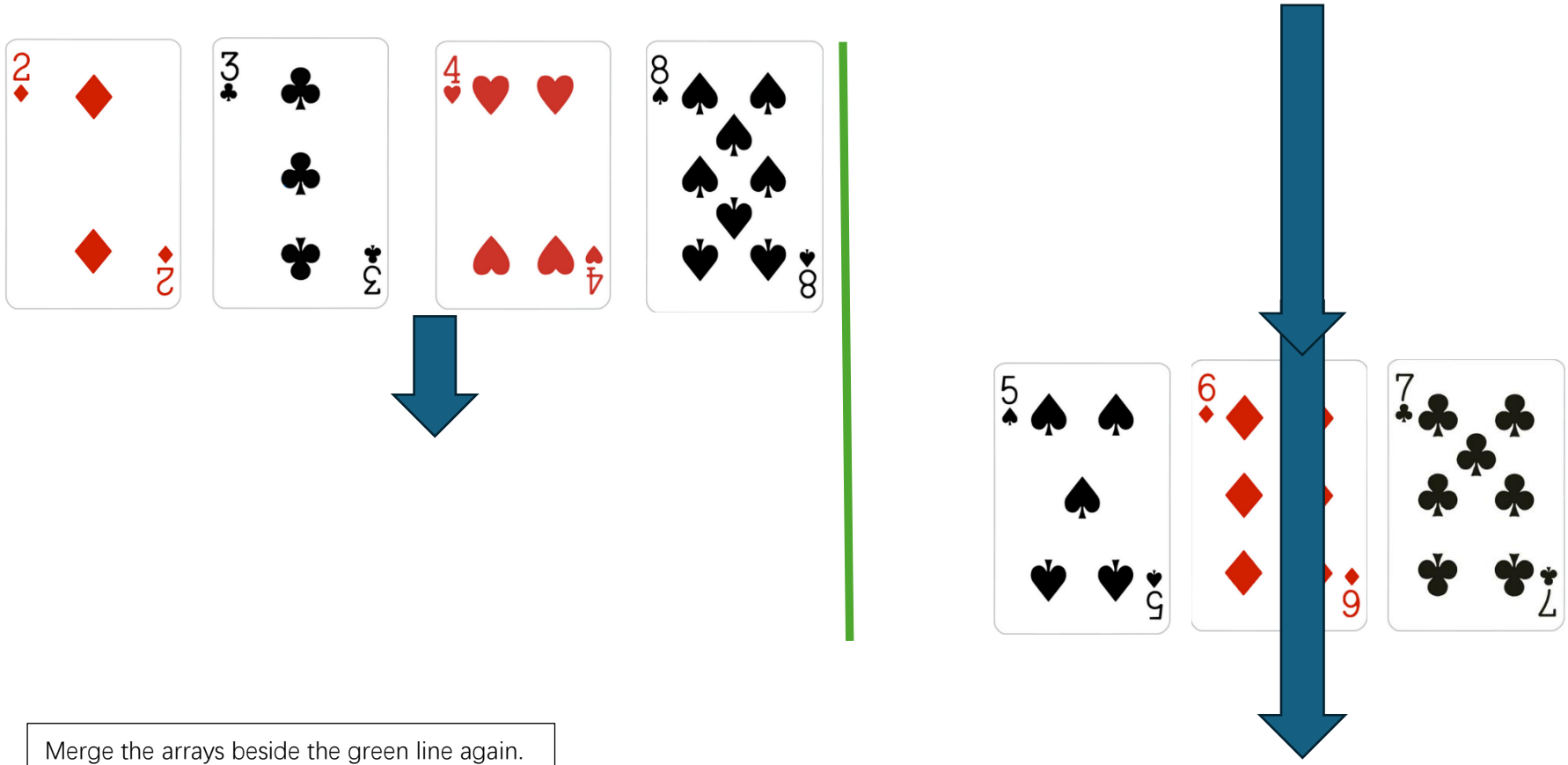
Separate 5 and 7, and merge them.
Compare: 5 < 7, therefore not swap.

Remains the same.

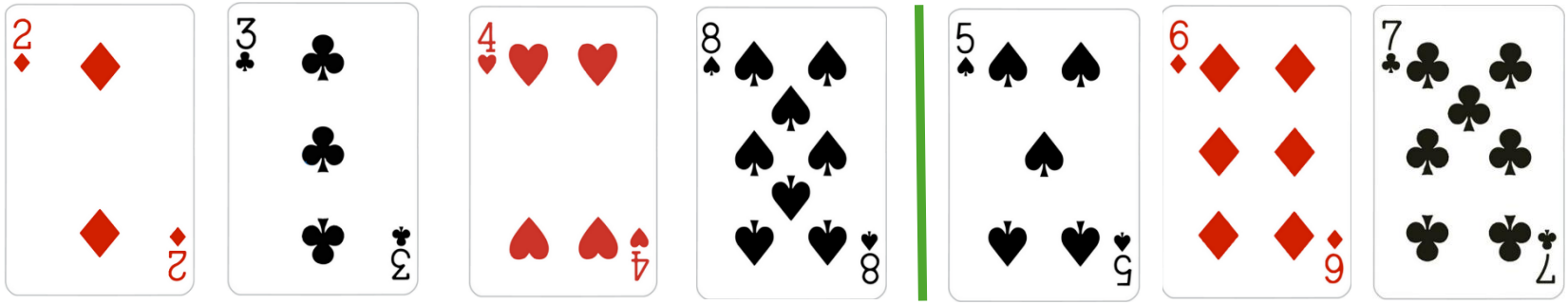


We create a new array below, and merge two arrays beside the red line. Compare 3 and 2, 2 is less than 3, so we put the smaller element 2 into the new array. Compare 3 and 8, 3 is less than 8, so we put 3 into the new array. Compare 4 and 8, 4 is less than 8, so we put 4 into the new array. 8 is left over so put 8 into the new array.

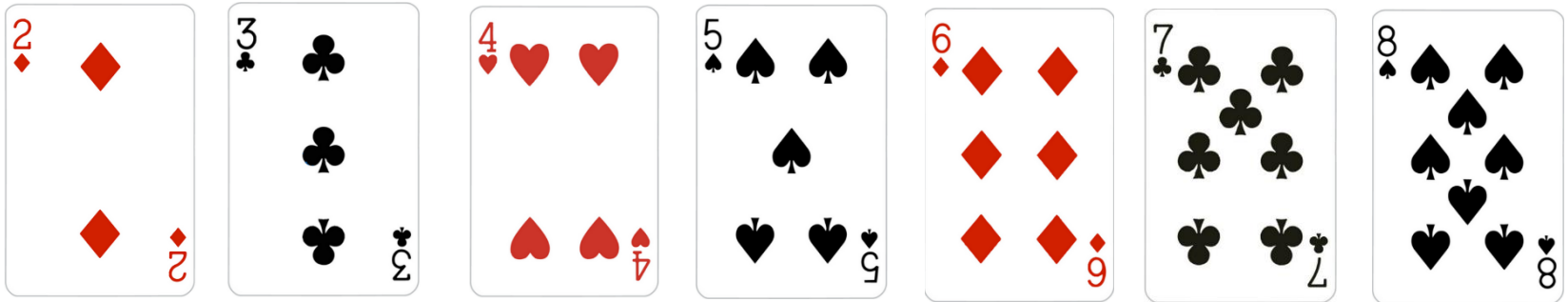
We create another new array below, and merge two arrays beside the red line. Compare 5 and 6, 5 is smaller so we move 5 to the new array. Compare 7 and 6, 6 is less than 7 so we put 6 into the new array. 7 is left over so put 7 into the new array.



Merge the arrays beside the green line again.



Merge the arrays beside the green line again.
2 is less than 5, so put 2 into the new array.
3 is less than 5, so put 3 into the new array.
4 is less than 5, so put 4 into the new array.
8 is greater than 5, so put 5 into the new array.
8 is less than 6, so put 6 into the new array.
8 is less than 7, so put 7 into the new array.
8 is left over so put 8 into the new array.



We have finished merge sort.