Selection Sort Pseudocode

There are many different ways to sort the cards. Here's a simple one, called selection sort, possibly similar to how you sorted the cards above:

Find the smallest card. Swap it with the first card.

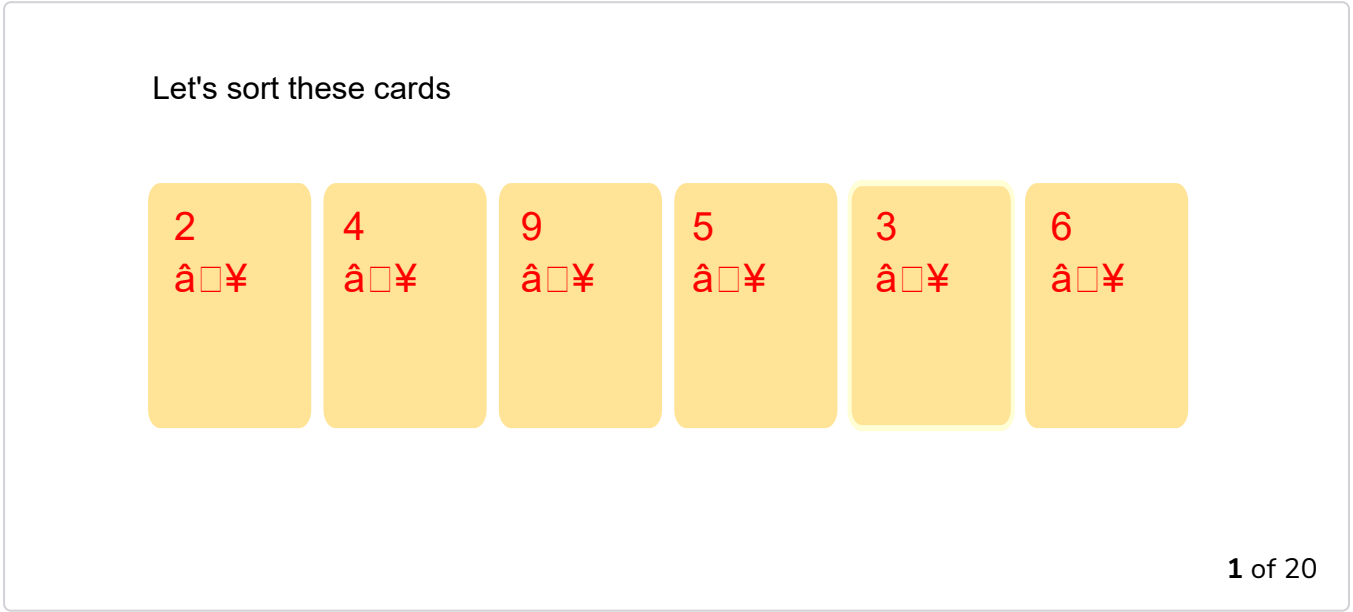
Find the second-smallest card. Swap it with the second card.

Find the third-smallest card. Swap it with the third card.

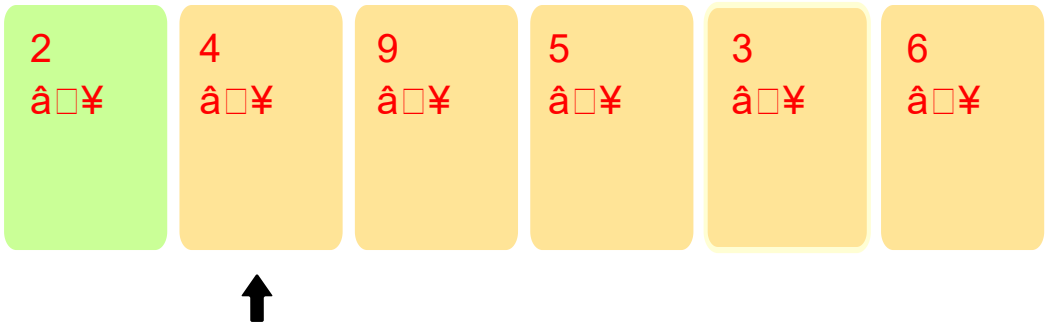
Repeat finding the next-smallest card, and swapping it into the correct position until the array is sorted.

This algorithm is called selection sort because it repeatedly selects the next-smallest element and swaps it into place.

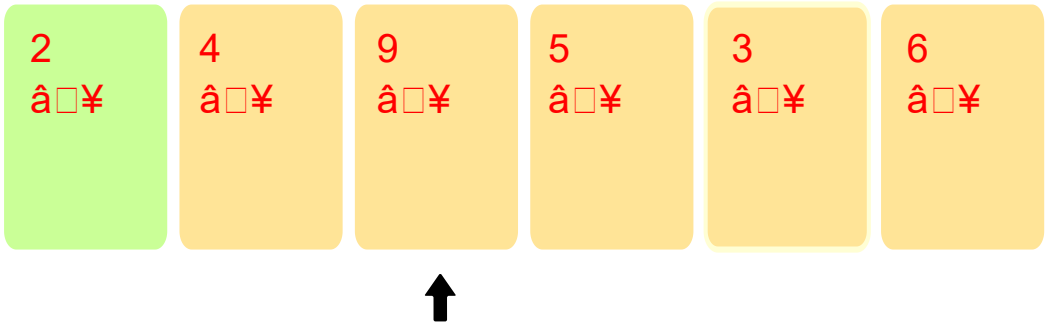
You can see the algorithm for yourself below. Start by using "Step" to see each step of the algorithm, and then try "Automatic" once you understand it to see the steps all the way through.



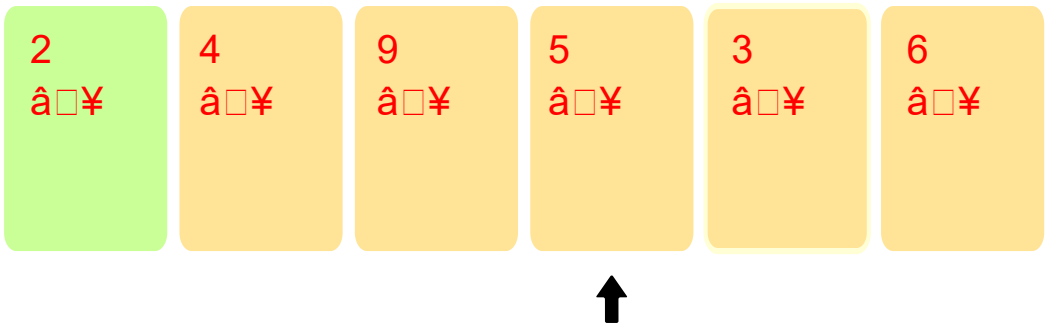
Finding next smallest



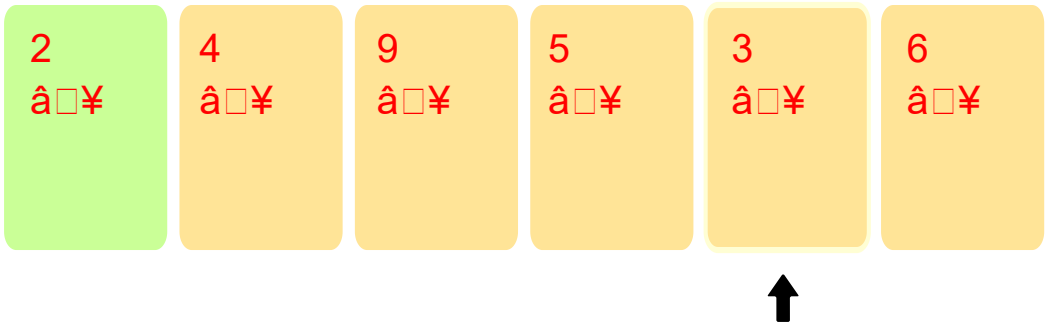
Finding next smallest



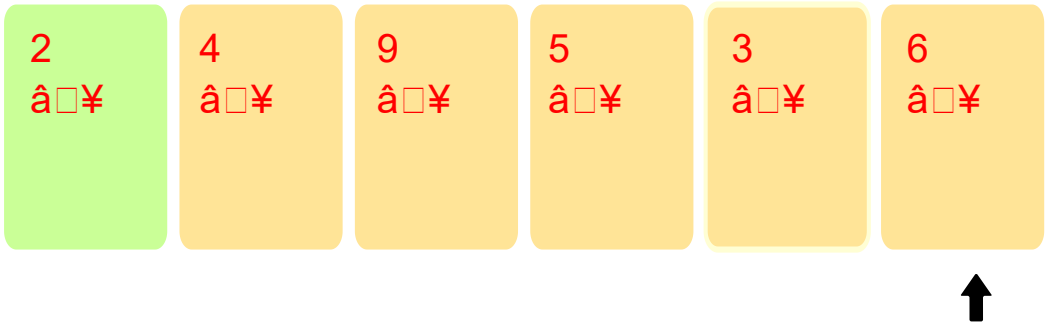
Finding next smallest



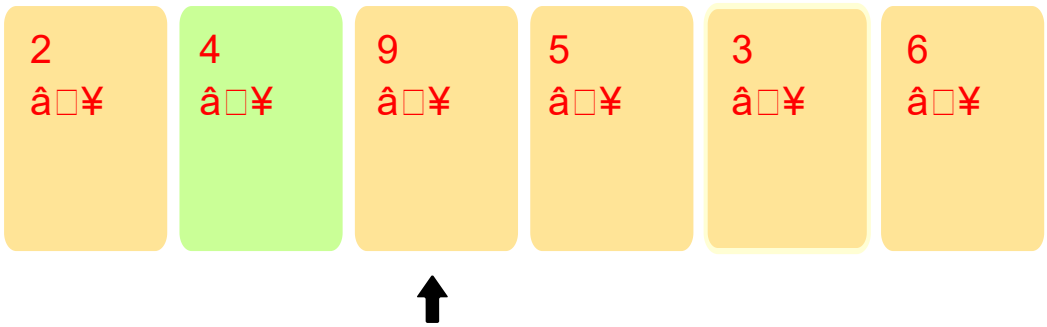
Finding next smallest



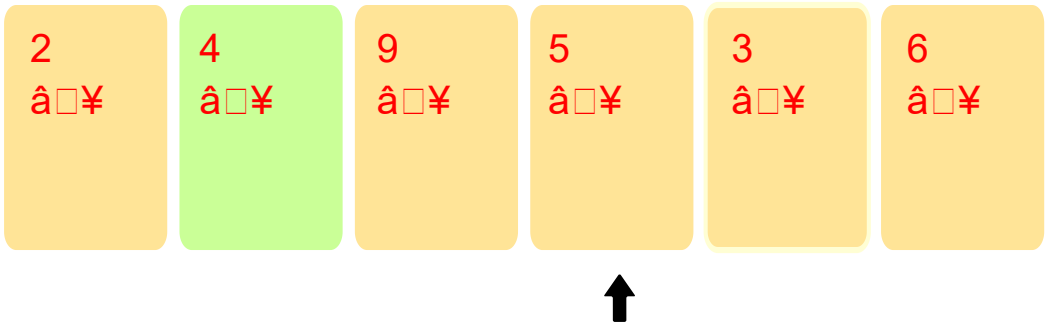
No card smaller than 2 so it stays where it is.



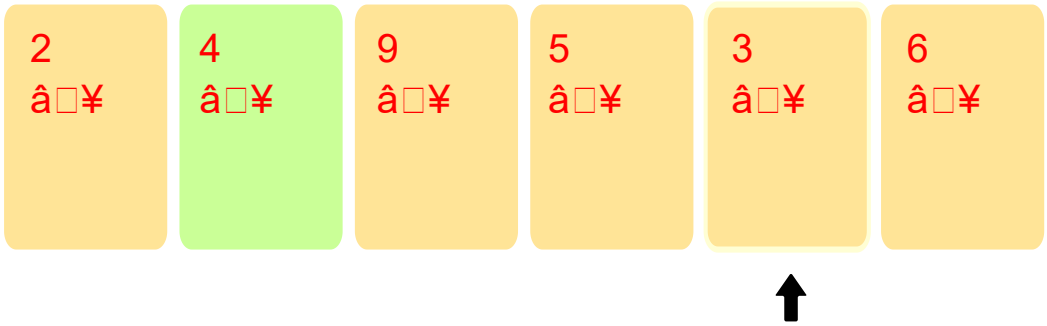
Finding next smallest



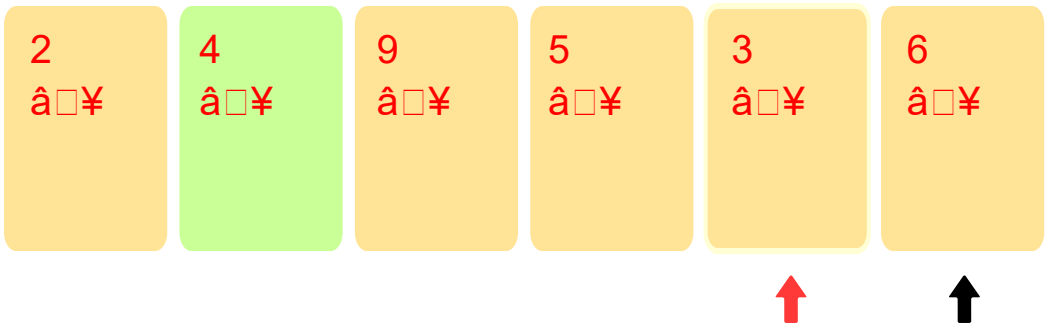
Finding next smallest



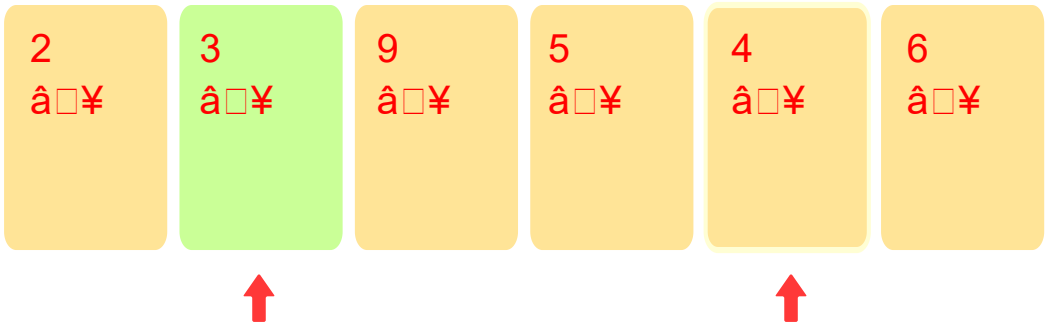
Finding next smallest



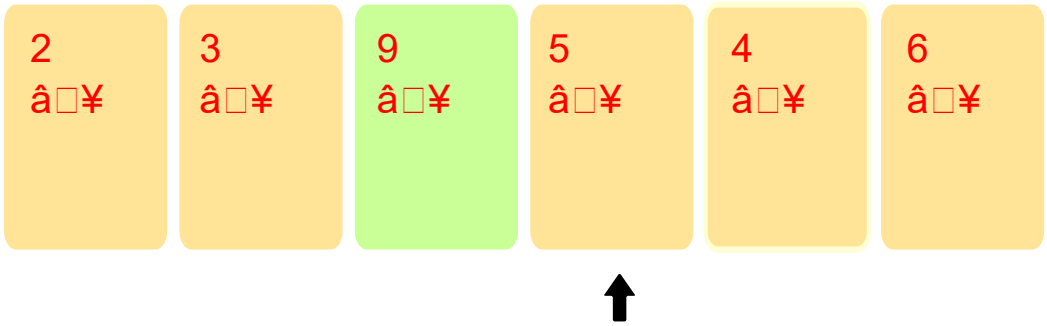
Remember 3 as it's the smallest we have seen and proceed



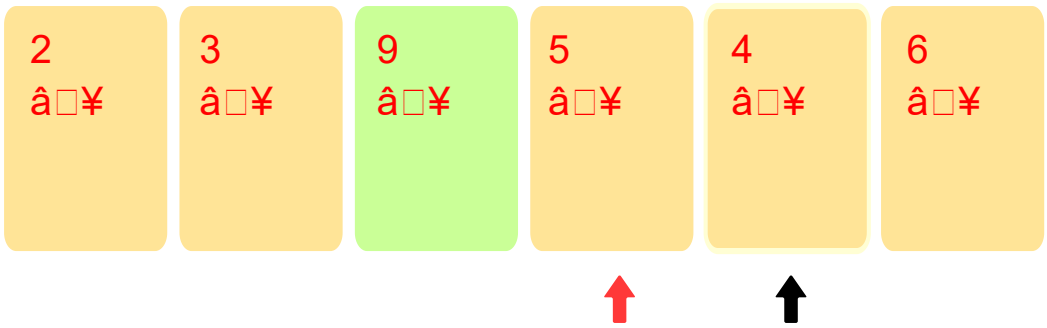
Swap 3 and 4



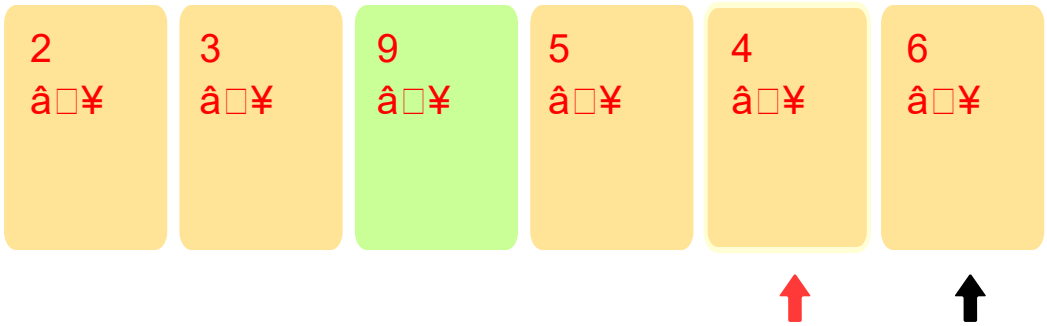
Finding next smallest



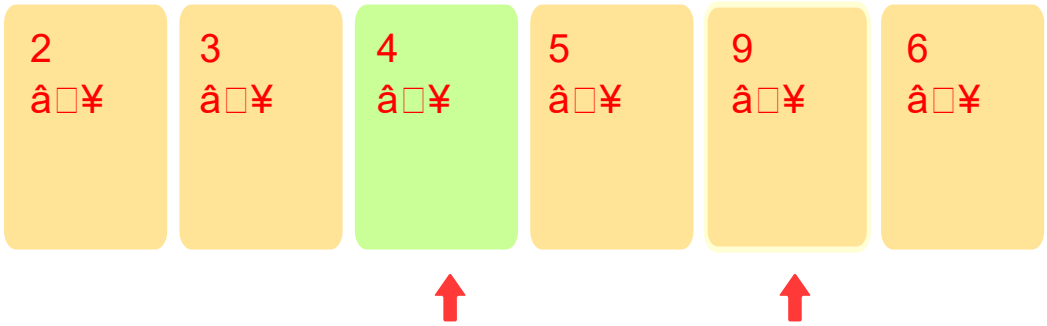
Remembering 5 as the next smallest



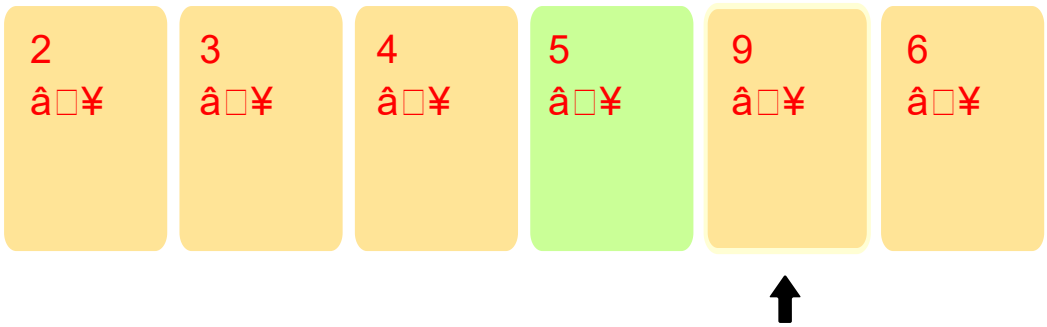
Remembering 4 as the next smallest



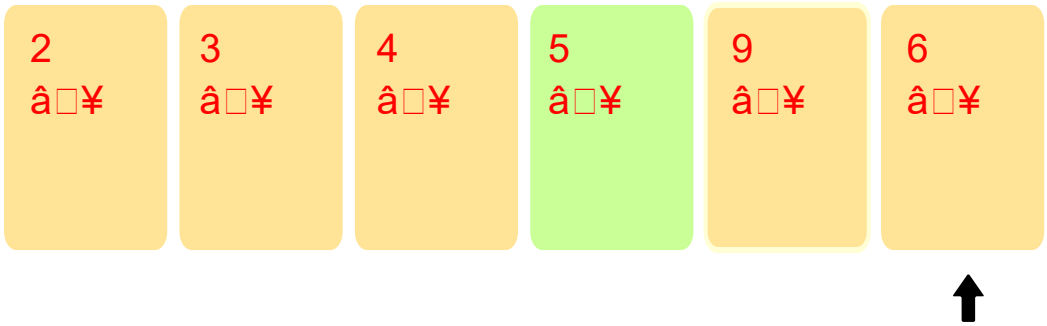
Swap 9 and 4



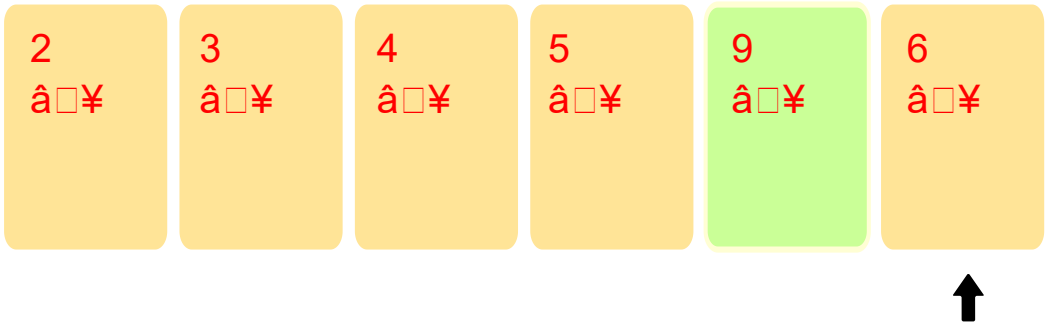
Finding next smallest



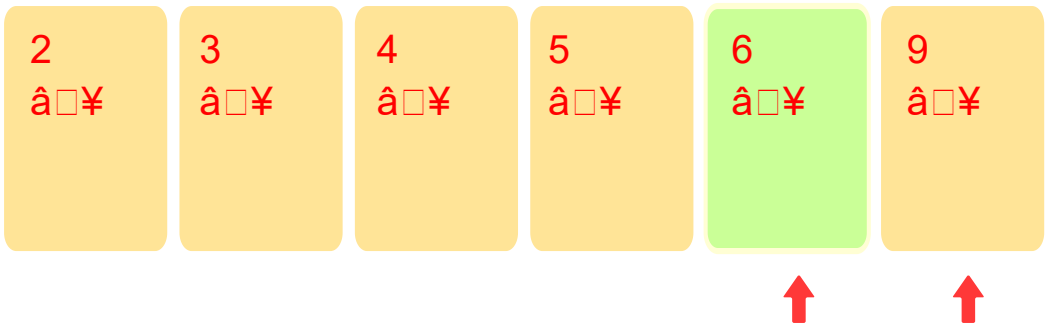
Finding next smallest

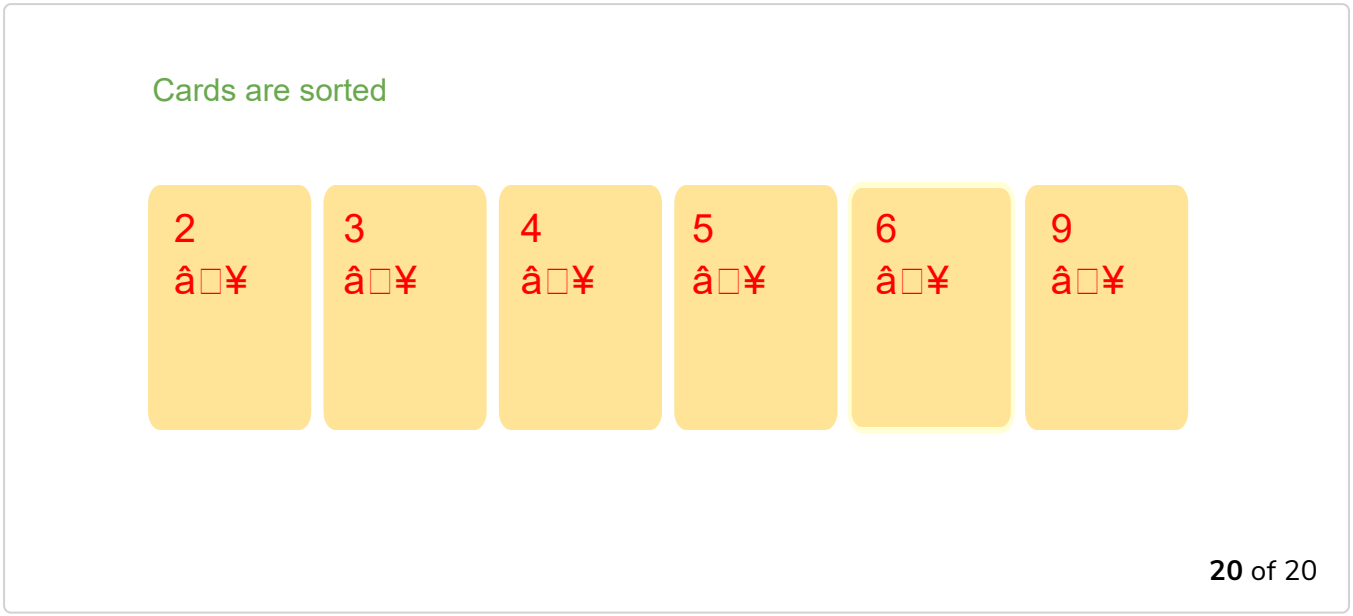


5 stays where it is. Finding next smallest.



Swapping 6 and 9.





After seeing it for yourself, what do you think aout this algorithm? What parts of it seem to take the longest? How do you think it would perform on big arrays? Keep those questions in mind as you go through and implement this algorithm.

Finding the index of the minimum element in a subarray

One of the steps in selection sort is to find the next-smallest card to put into its correct location. For example, if the array initially has values [13, 19, 18, 4, 10], we first need to find the index of the smallest value in the array. Since 4 is the smallest value, the index of the smallest value is 3.

Selection sort would swap the value at index 3 with the value at index 0, giving [4, 19, 18, 13, 10]. Now we need to find the index of the second-smallest value to swap into index 1.

It might be tricky to write code that found the index of the second-smallest value in an array. I'm sure you could do it, but there's a better way. Notice that since the smallest value has already been swapped into index 0, what we really want is to find the smallest value in the part of the array that starts at index 1. We call a section of an array a **subarray**, so that in this case, we want the index of the smallest value in the subarray that starts at index 1. For our example, if the full array is [4, 19, 18, 13, 10], then the smallest value in the

subarray starting at index 1 is 10, and it has index 4 in the original array. So index 4 is the location of the second-smallest element of the full array.

Try out that strategy in the next challenge, and then you'll have most of what you need to implement the whole selection sort algorithm.