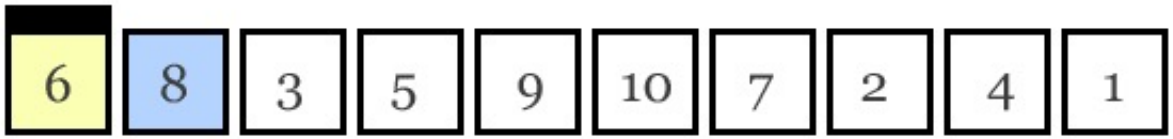
Selection Sort

Starting at the first slot, **look** through all the values for the smallest and then **swap** it into place. Then move on to the following slots consecutively. $O(n^2)$

Selection Sort



Yellow is smallest number found
Blue is current item
Green is sorted list

http://sonny.io/2015/12/21/selection-sort/

The Selection Sort Algorithm

- Of the many sorting algorithms, the easiest one to describe is selection sort, which is implemented by the following code:

```
private void sort(int[] array) {
   for (int lh = 0; lh < array.length; lh++) {
      int rh = findSmallest(array, lh, array.length);
      swapElements(array, lh, rh);
   }
}
```

  The variables lh and rh indicate the positions of the left and right hands if you were to carry out this process manually. The left hand points to each position in turn; the right hand points to the smallest value in the rest of the array.

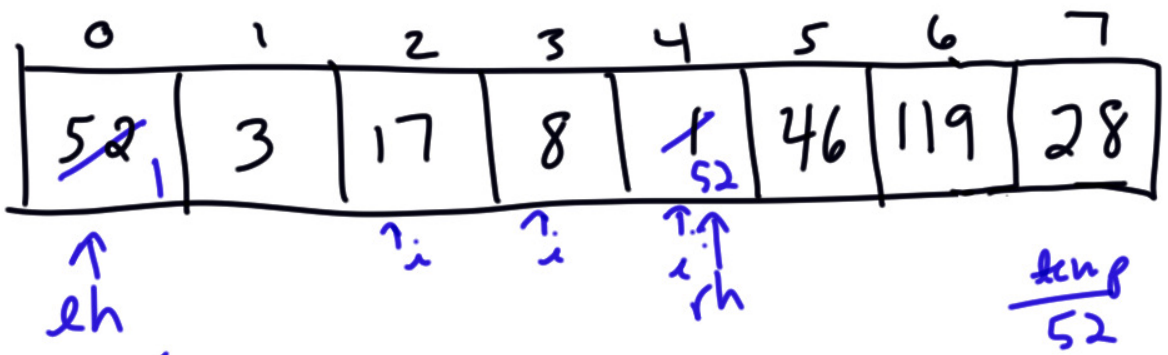
- The method findSmallest(array, p_1, p_2) returns the index of the smallest value in the array from position p_1 up to but not including p_2. The method swapElements(array, p_1, p_2) exchanges the elements at the specified positions.

Selection Sort on array of ints

```
 private void selectionSort(int[] array) {
      for (int lh = 0; lh < array.length; lh++) {
         int rh = findSmallest(array, lh, array.length);
         swapElements(array, lh, rh);
      }
   }

private int findSmallest(int[] array, int p1, int p2) {
      int smallestIndex = p1;
      for (int i = p1 + 1; i < p2; i++) {
         if (array[i] < array[smallestIndex]) smallestIndex = i;
      }
      return smallestIndex;
   }

private void swapElements(int[]array, int p1, int p2) {
      int temp = array[p1];
      array[p1] = array[p2];
      array[p2] = temp;
   }
```



1 3 17 8 52 46 119 28

smallestIndex
1

lh

8 17

1 3 ~~17~~ ~~8~~ 52 46 119 28

smallestIndex
~~2~~
3

1

lh

1 3 8 17 52 46 119 28

Efficiency of Selection Sort

- Selection sort is $O(n^2)$ regardless of the initial order of the entries.
  - Requires $O(n^2)$ comparisons
    - Nest loop of comparisons
  - Does only O(n) swaps
    - One potential swap per element