Sorting Algorithms

Bubble Sort, Selection Sort, Insertion Sort

Sorting

- Motivation
  - Generally, to arrange a list of elements in some order
- List of numbers
  - 10 20 50 30 40 60 25 (Unsorted list)
  - 10 20 25 30 40 50 60 (Sorted list, ascending)
  - 60 50 40 30 25 20 10 (Sorted list, descending)
- List of alphabets
  - P A K I S T A N (Unsorted list)
  - A A I K N P S T (Sorted list, ascending)
  - T S P N K I A A (Sorted list, descending)

Sorting Algorithms

1. Bubble Sort
2. Selection Sort
3. Insertion Sort
4. Quick Sort
5. Merge Sort
6. Heap sort

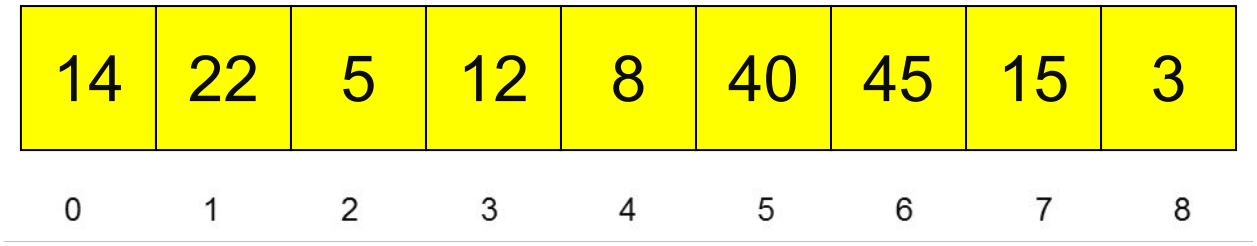
There are more other algorithms but we will focus on the first three!

Bubble Sort Algorithm: Informal (1)

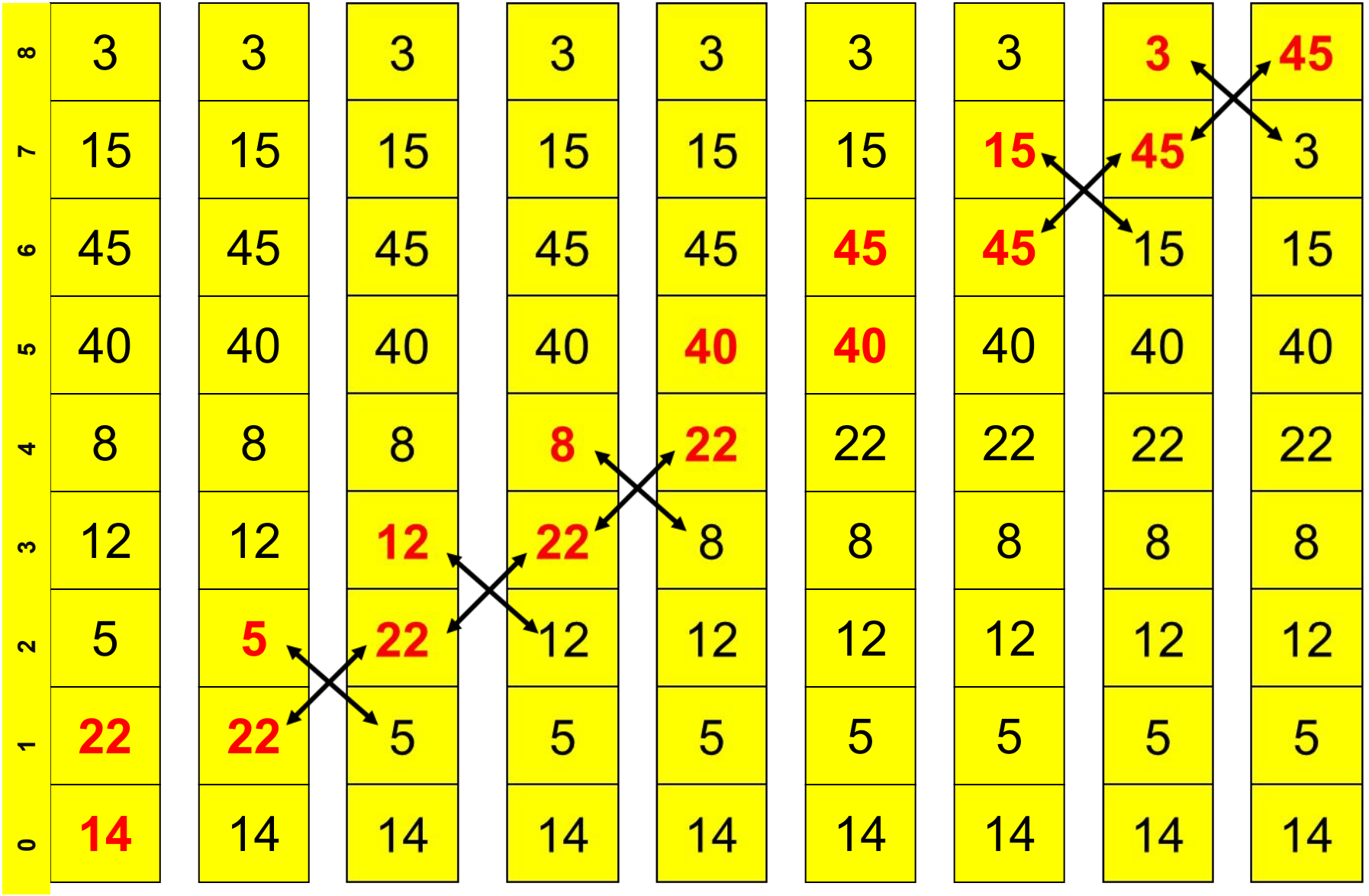
- Repeatedly compare the elements at consecutive locations in a given list, and do the following until the elements are in required order:
  - If elements are not in the required order, swap them (change their position)
  - Otherwise do nothing

Bubble Sort Algorithm: phases (2)

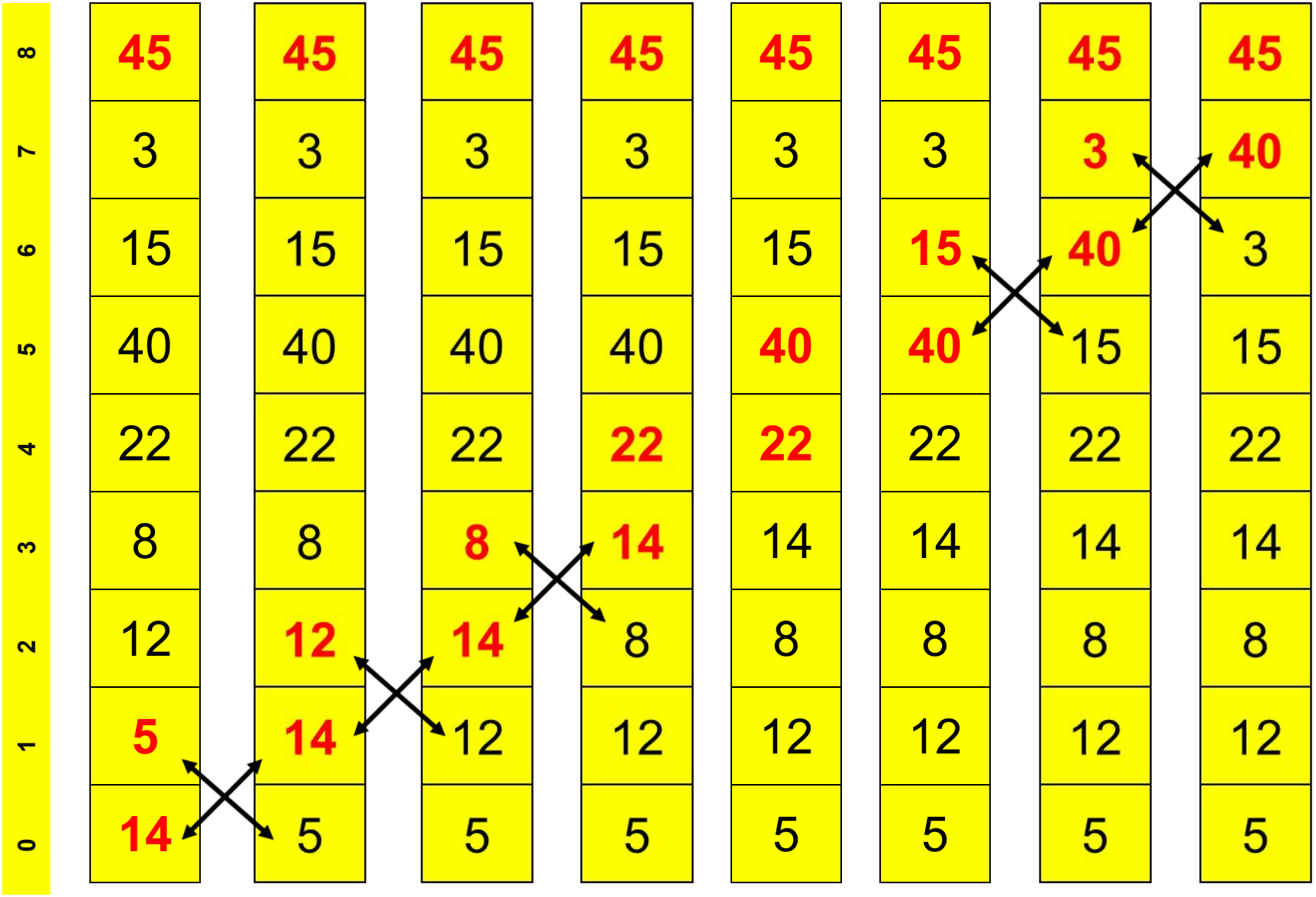
- In each phase (main step) of the bubble sort we perform the following action:
  - Start from the beginning of the list comparing adjacent items to move the current largest toward the end (top) of the list
- For example,
  - in the first phase we compare the adjacent items to take the largest to the end of the list (45 to position 8)
  - In the end phase we compare the adjacent items to take the second largest to the one location before the end of the list (40 to position 7)



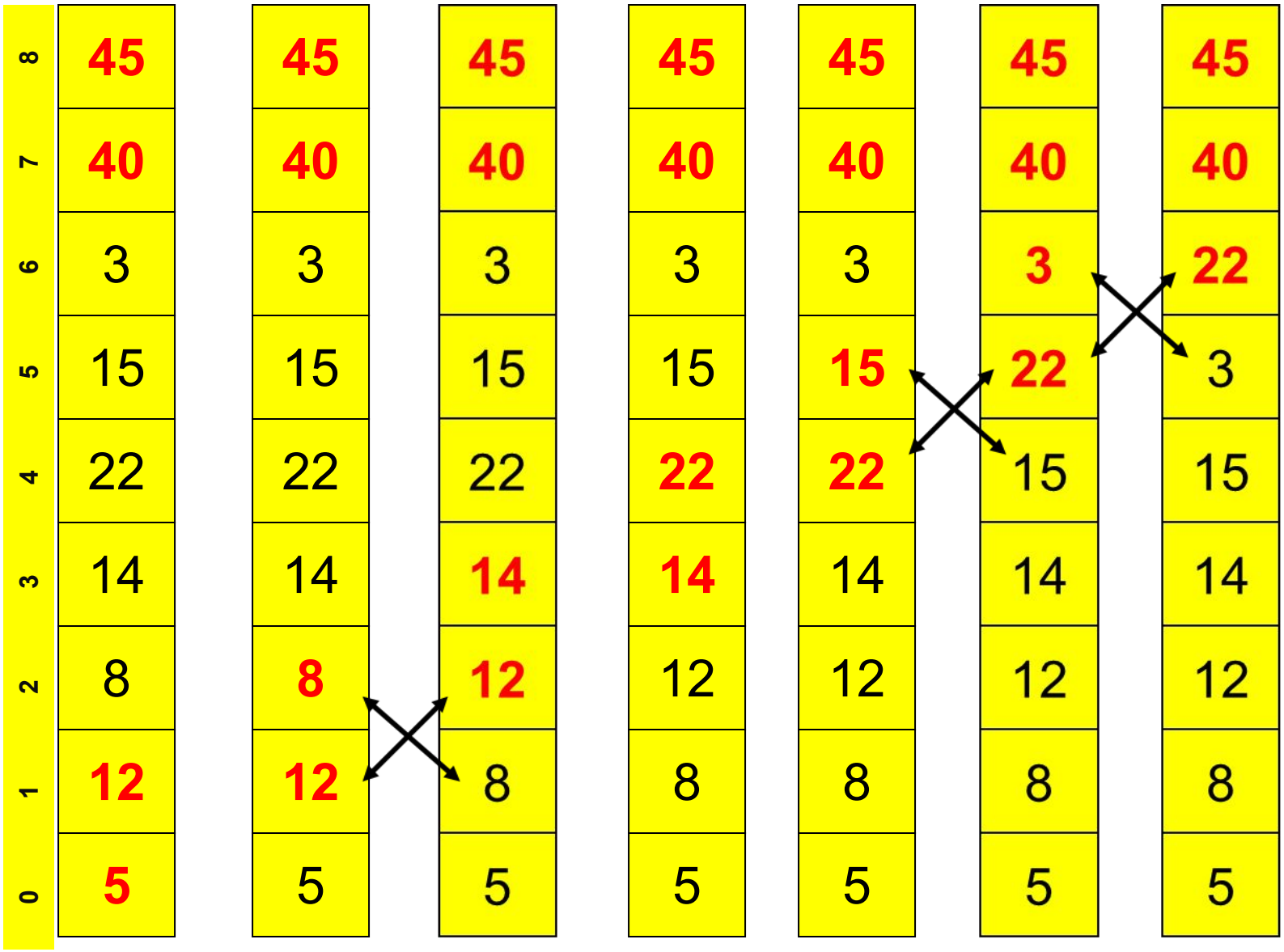
Bubble Sort in Action: Phase 1 (3)



Bubble Sort in Action: Phase 2 (4)

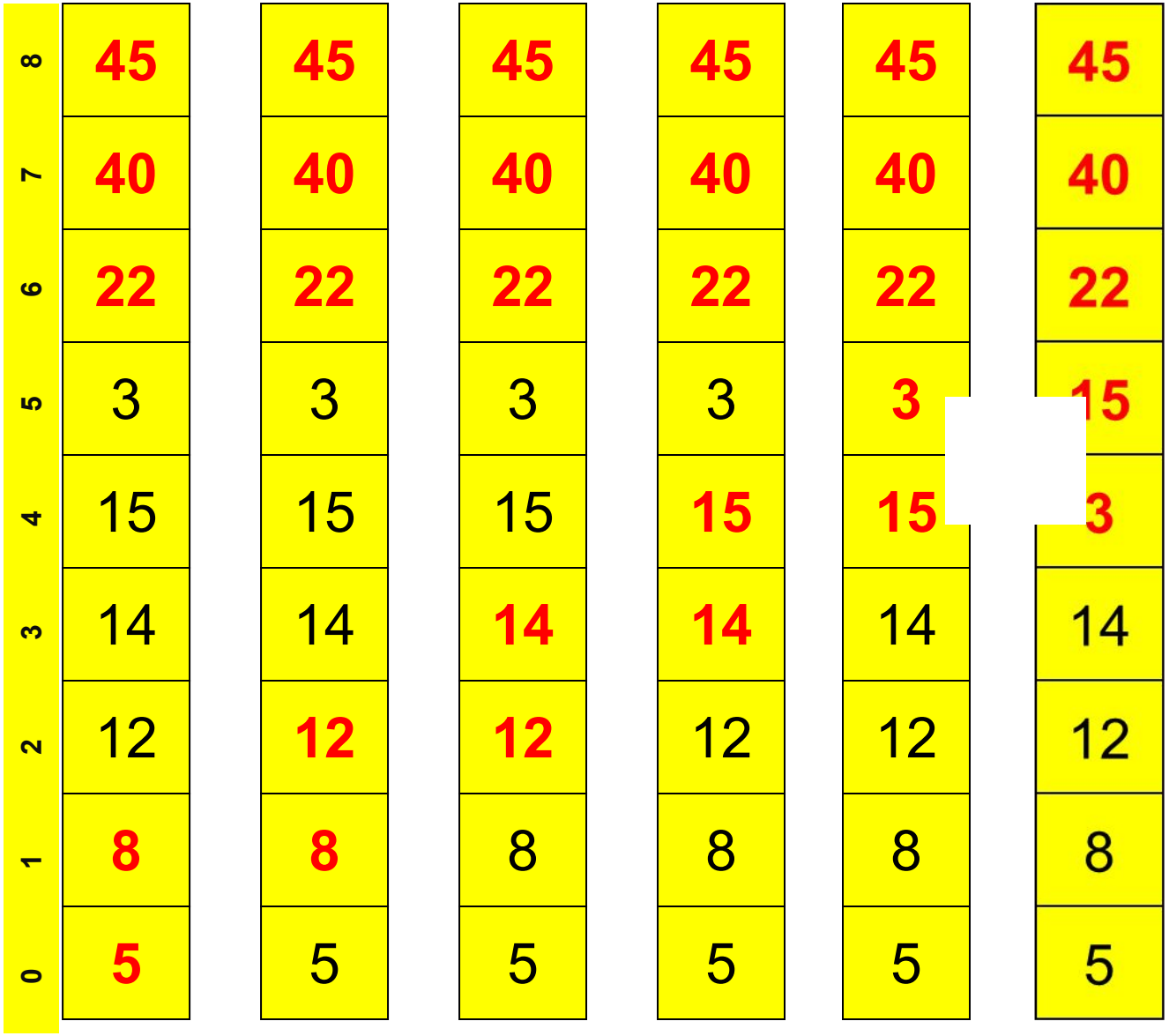


Bubble Sort in Action: Phase 3 (5)

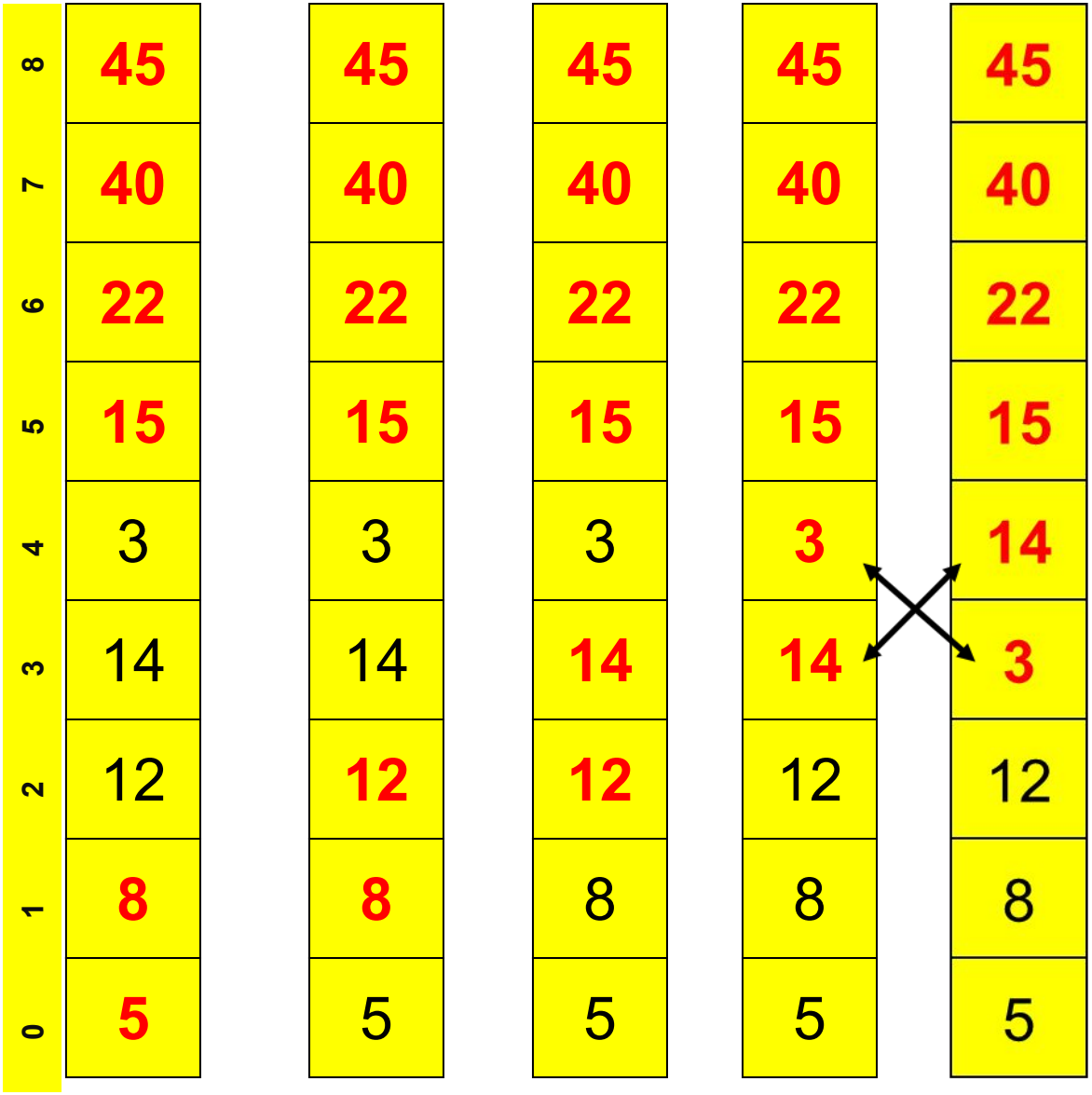


6 comparisons

Bubble Sort in Action: Phase 4 (6)

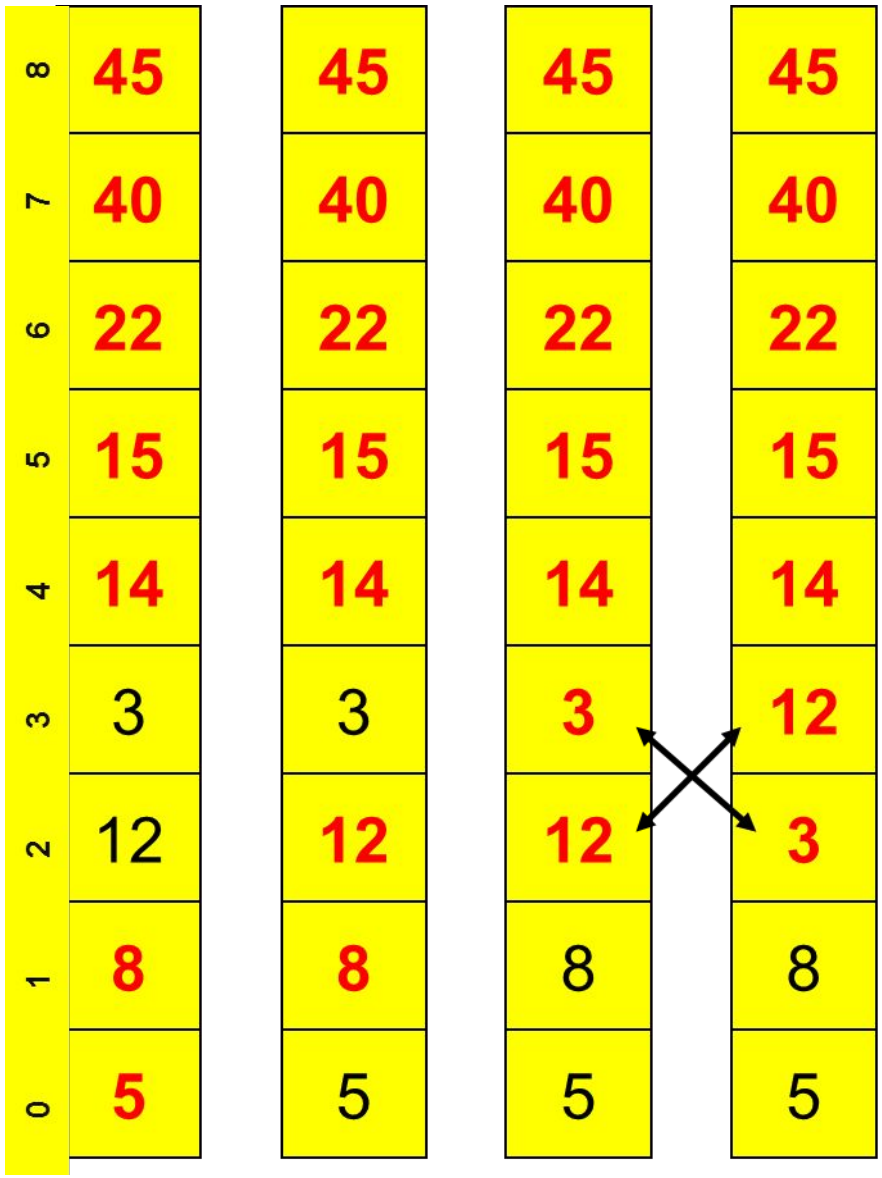


Bubble Sort in Action: Phase 5 (7)



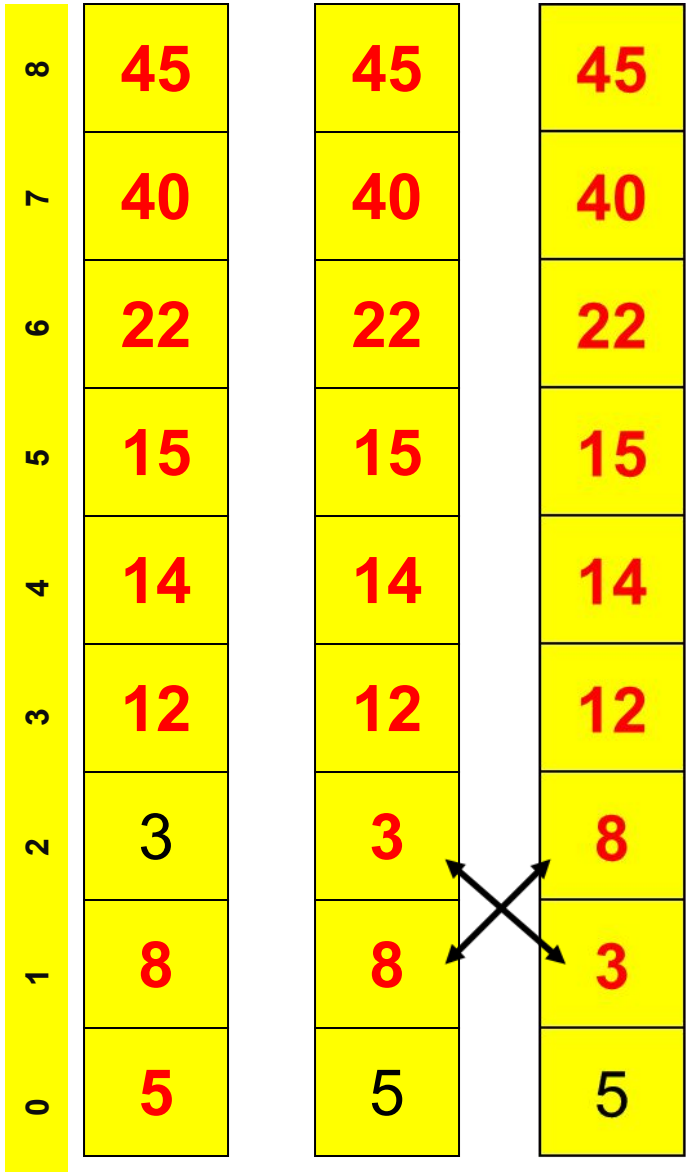
4 comparisons

Bubble Sort in Action: Phase 6 (8)



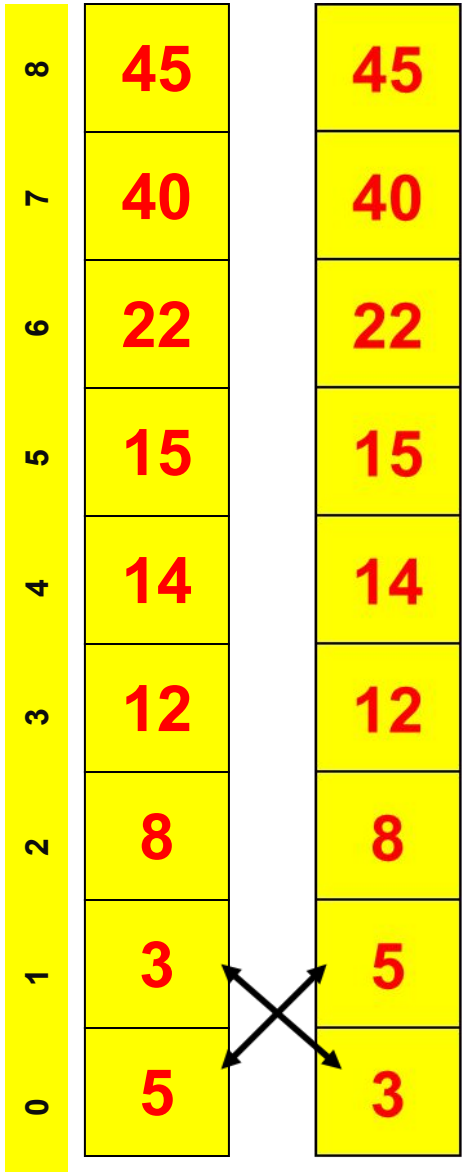
3 comparisons

Bubble Sort in Action: Phase 7 (9)



2 comparisons

Bubble Sort in Action: Phase 8 (10)



1 comparison

Bubble sort examples (11)

- Try
- http://math.hws.edu/eck/jsdemo/sortlab.html

Bubble Sort Algorithm (12)

Algorithm: bubble Sort
Input: List with size n
Output: List (sorted)

```
for i = 0; i < n - 1; i++
   for j = 0; j < n - (i + 1); j++
       if List[j] >  List[j+1]  then
        List[j] ↔ List[j+1]  //swap
       end if
   end for
end for
```

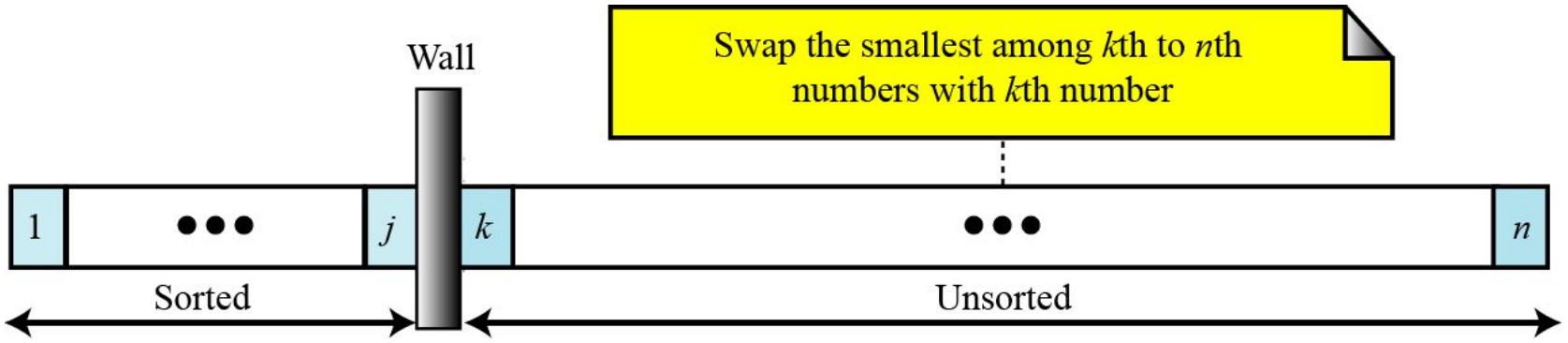
Bubble Sort Algorithm in Java (13)

```
void bubbleSort(int List[])
 {   int temp;
     int size = List.length;
     for (i = 0; i < size - 1; i++)
     {
       for (j = 0; j < size - (i + 1); j++)
       {
         if (List[j] > List[j+1])
         {    //swap
           temp = List[j];
           List[j] = List[j+1];
           List[j+1] = temp;
         }
       }
     }
 }
```

☞Time complexity of the Bubble Sort algorithm is $O(n^2)$. Think why?

Selection Sort: Informal (1)

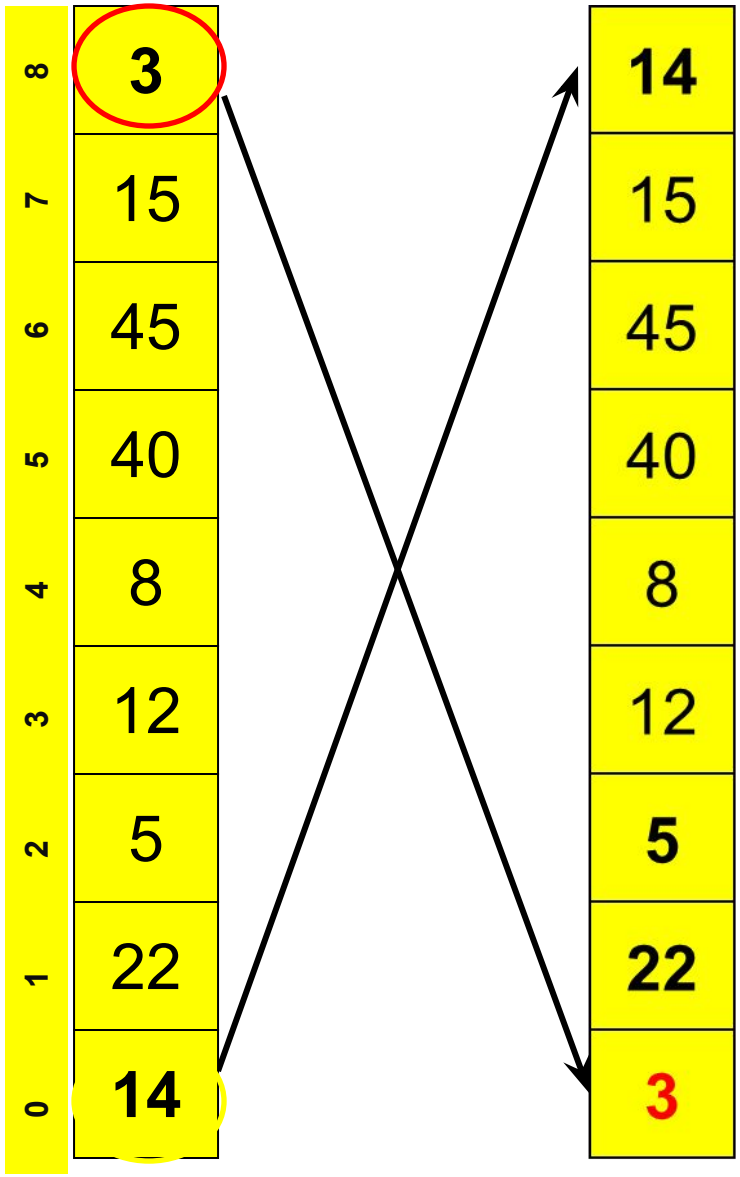
- Suppose we want to sort an array in ascending order :
  - Locate the smallest element in the array; swap it with element at index 0
  - Then, locate the next smallest element in the array; swap it with element at index 1.
  - Continue until all elements are arranged in order



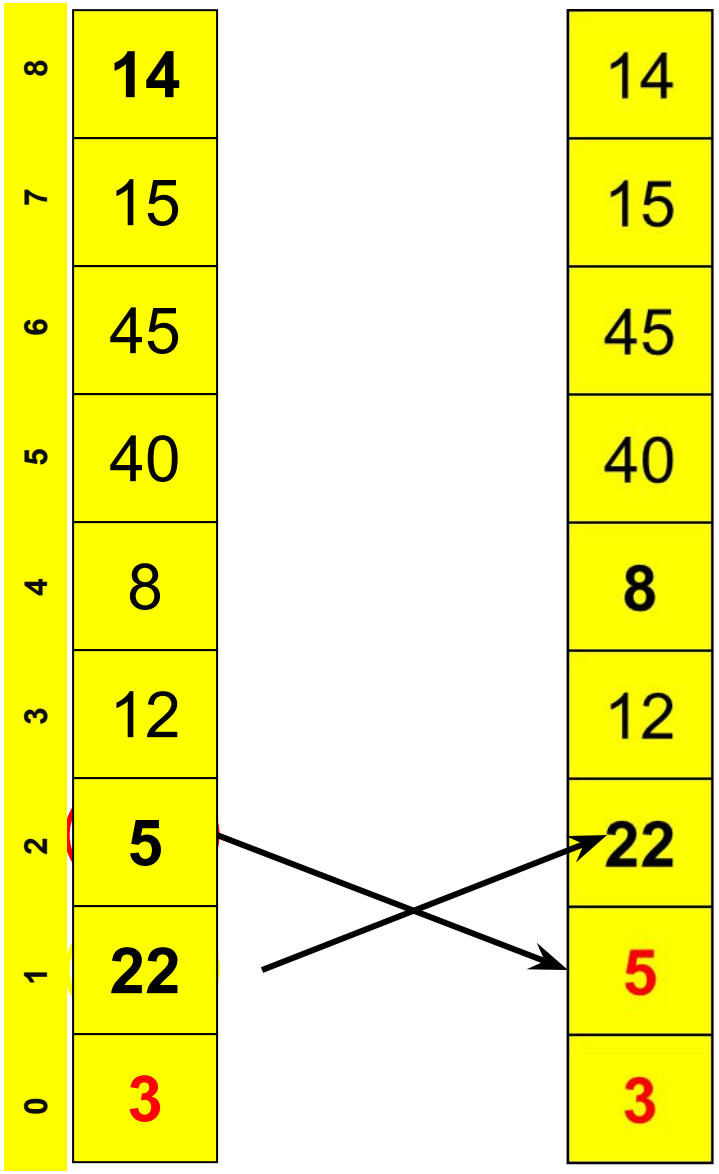
Selection Sort: Informal (2)

- Same thing can be done using the largest element:
  - Locate the largest element in the array; swap it with element at index n-1
  - Then, locate the next largest element in the array; swap it with element at index n-2.
  - Continue until all elements are arranged in order

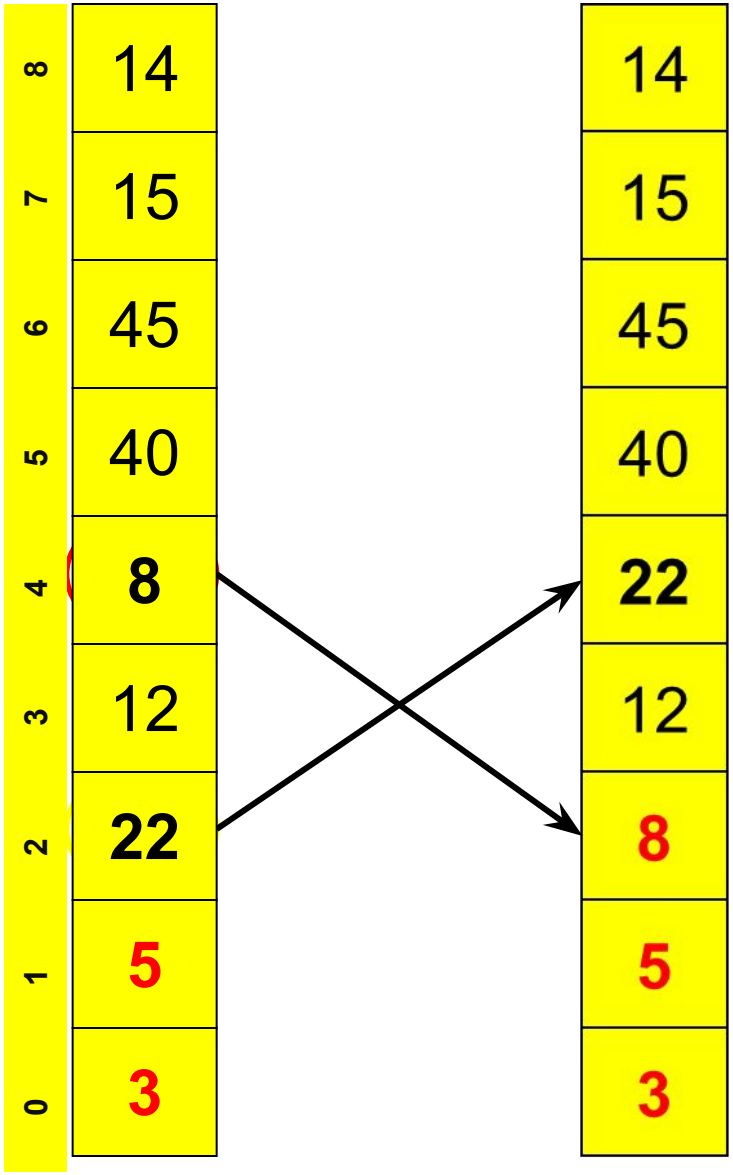
Selection Sort in Action: step 1 (3)



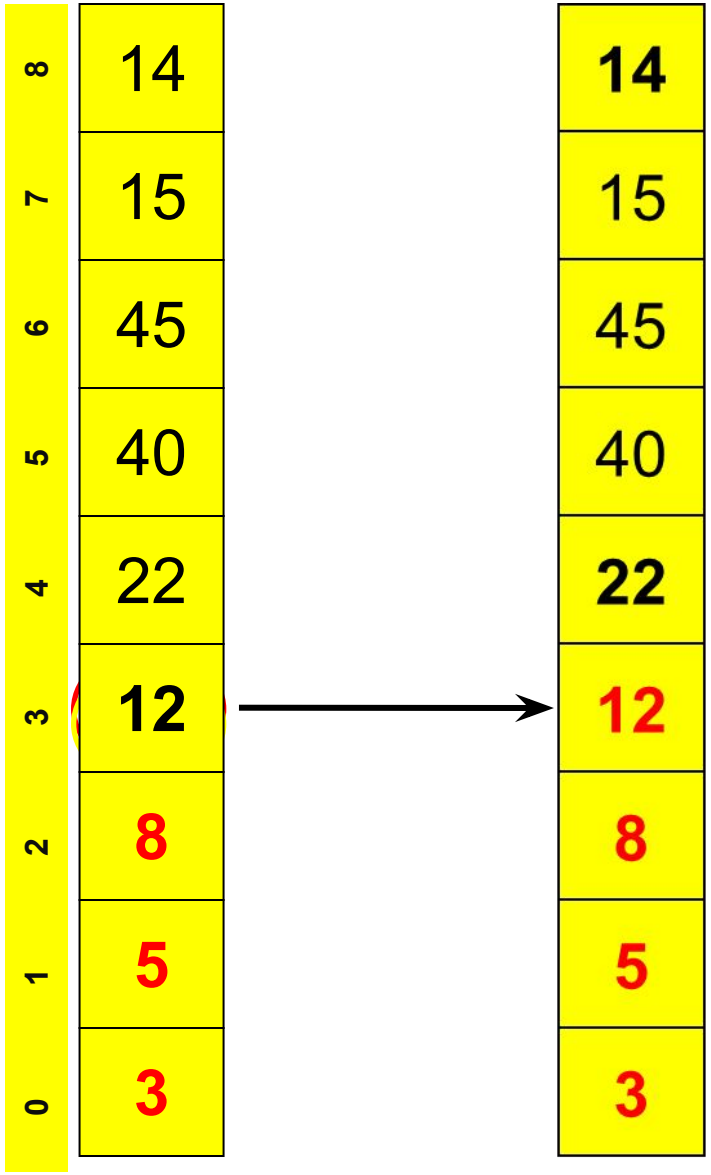
Selection Sort in Action: step 2 (4)



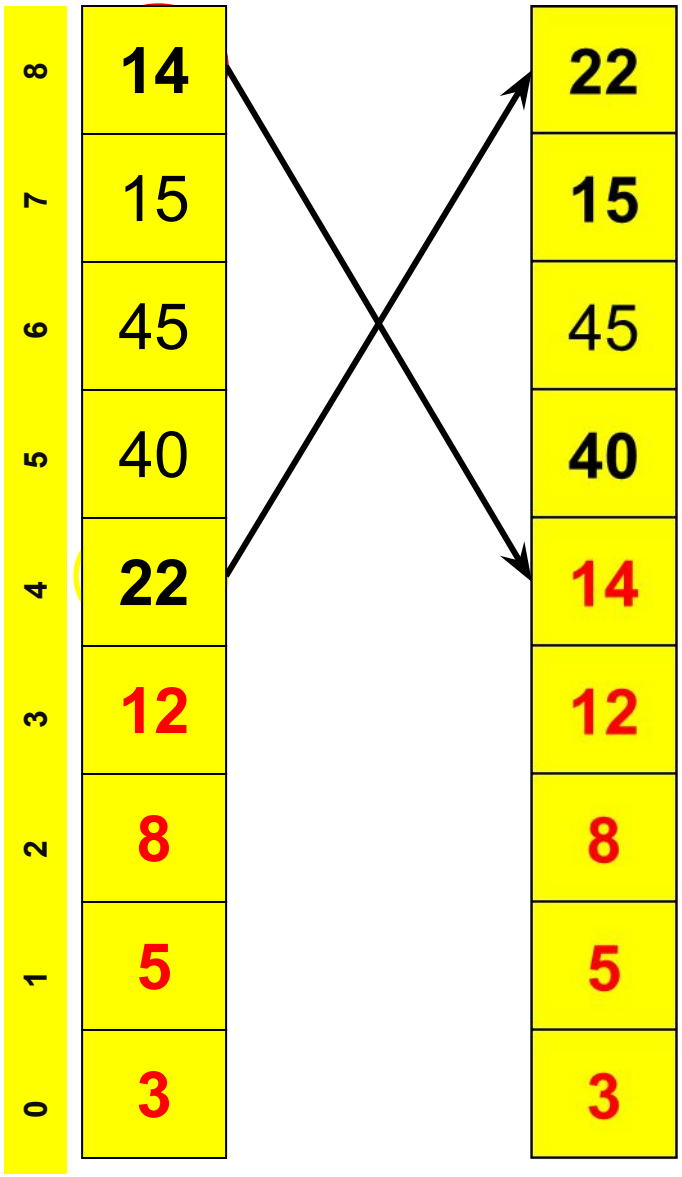
Selection Sort in Action: step 3 (5)



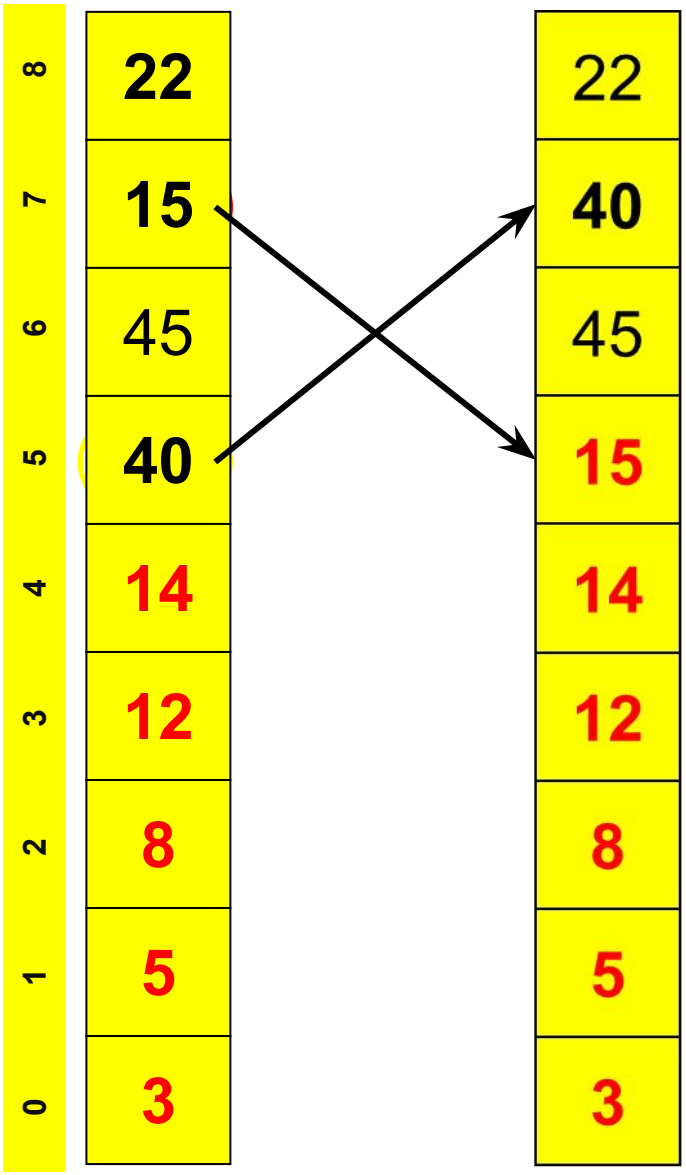
Selection Sort in Action: step 4 (6)



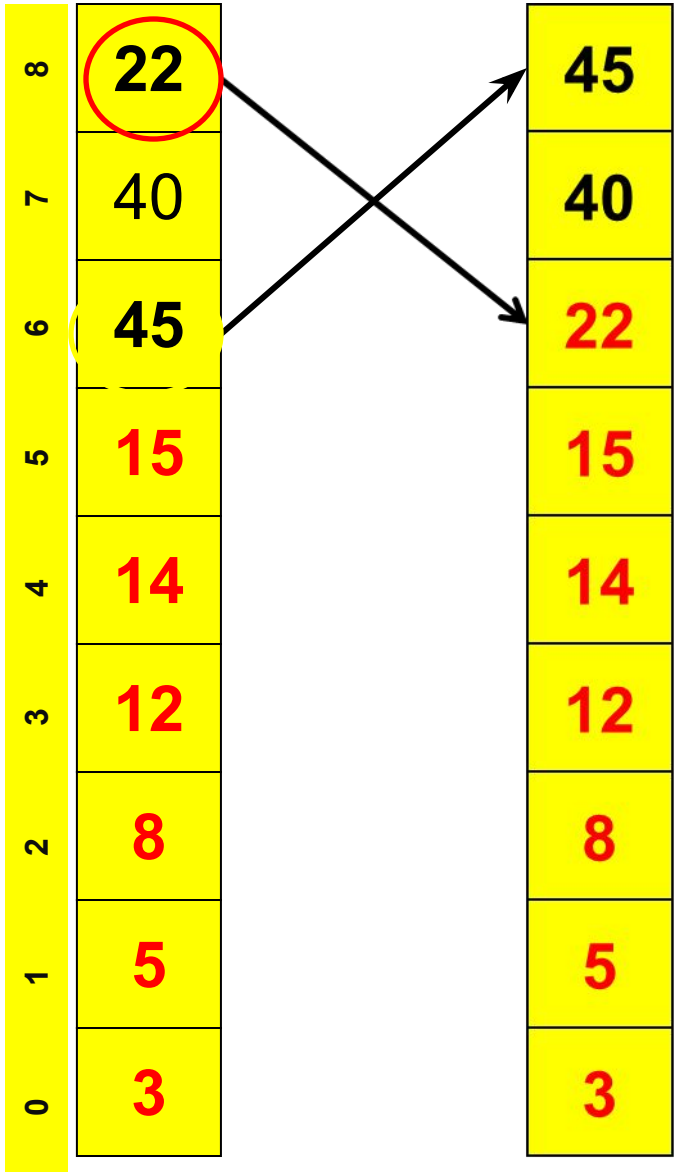
Selection Sort in Action: step 5 (7)



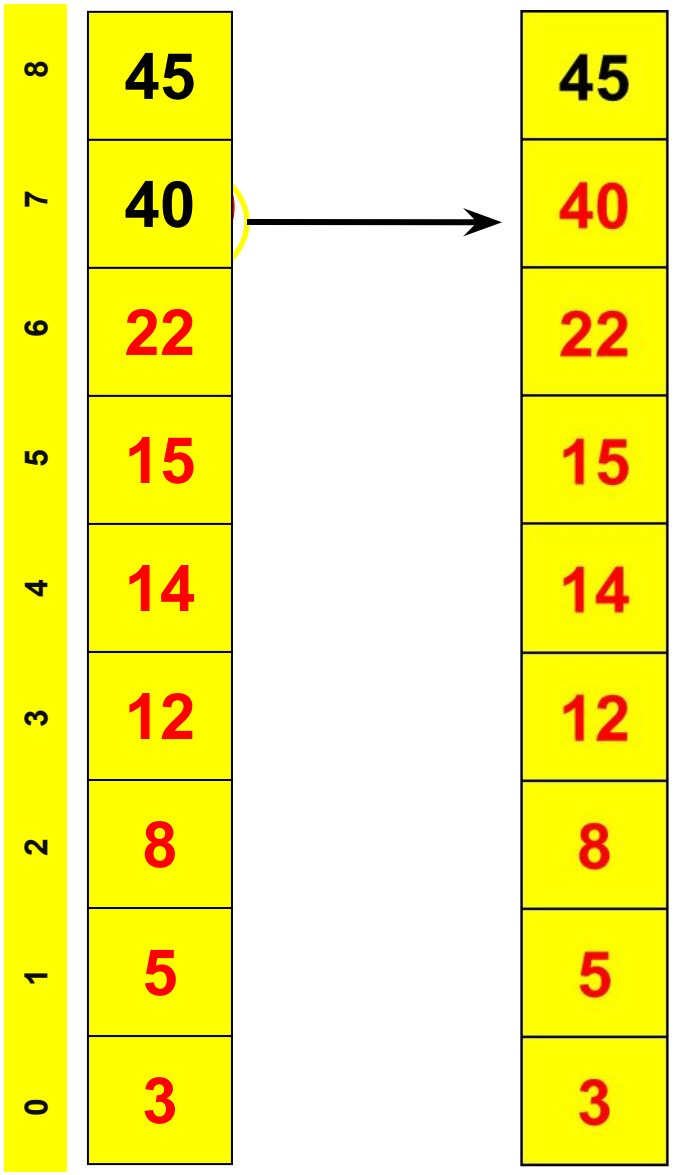
Selection Sort in Action: step 6 (8)



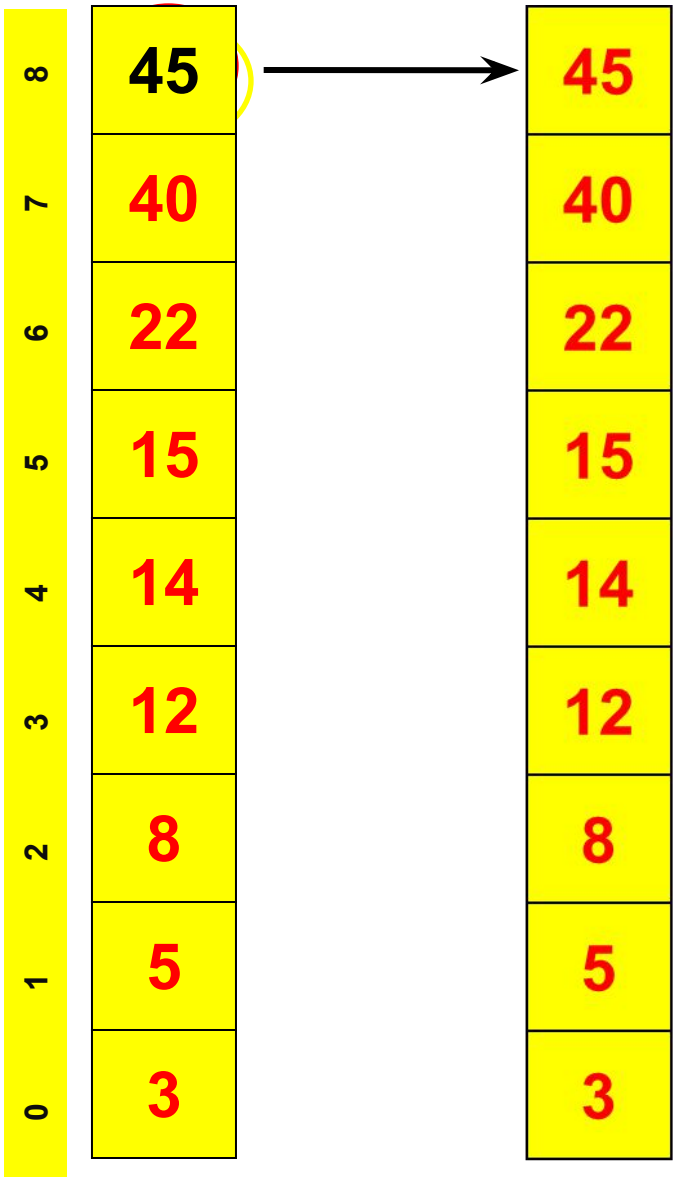
Selection Sort in Action: step 7 (9)



Selection Sort in Action: step 8 (10)



Selection Sort in Action: step 9 (11)



Selection Sort Algorithm (12)

Algorithm: Selection Sort
Input: List with size n
Output: List (sorted)

```
for i = 0; i < size; i++
    min = i
    for j = i + 1; j < size; j++
      if List[j] < List[min] then
        min = j
      end if
    end for

    List[min] ↔ List[i]
End for
```

Selection Sort Algorithm in Java (13)

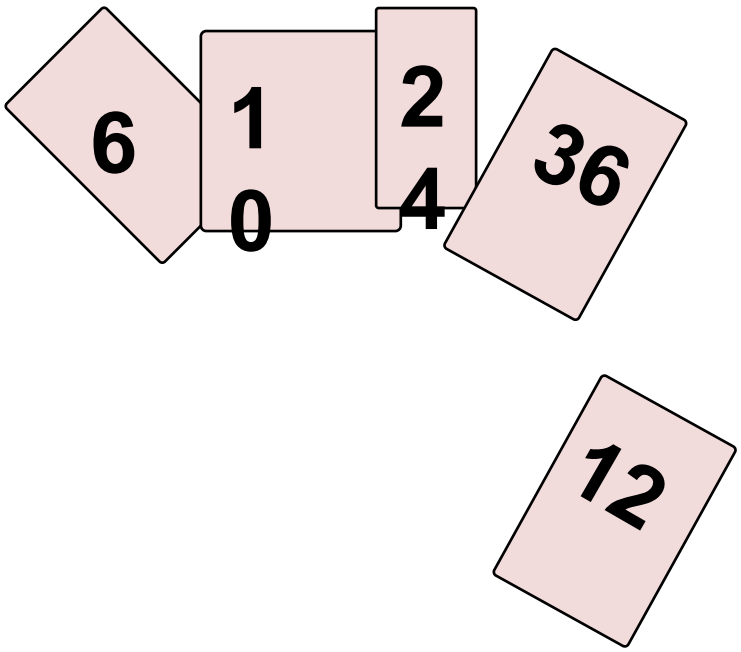
```
void selectionSort(int List[])
{
   int temp, min;
   int size = List.length;
   for (int i = 0; i < size; i++){
      min = i;
      for (j = i + 1; j < size; j++)
      {
         if (List[j] < List[min])
          {min = j;   }
      }
      temp = List[min];
      List[min] = List[i];    //swap
      List[i] = temp;
   }
}
```

☞**Time complexity of the Selection Sort algorithm is $O(n^2)$. Think why?**

Bubble Sort vs. Selection Sort

- Selection Sort is more efficient than Bubble Sort, because of fewer exchanges in the former
- Both Bubble Sort and Selection Sort belong to the same (quadratic) complexity class ($O(n^2)$)
- Bubble Sort may be easy to understand as compared to Selection Sort – What do you think?

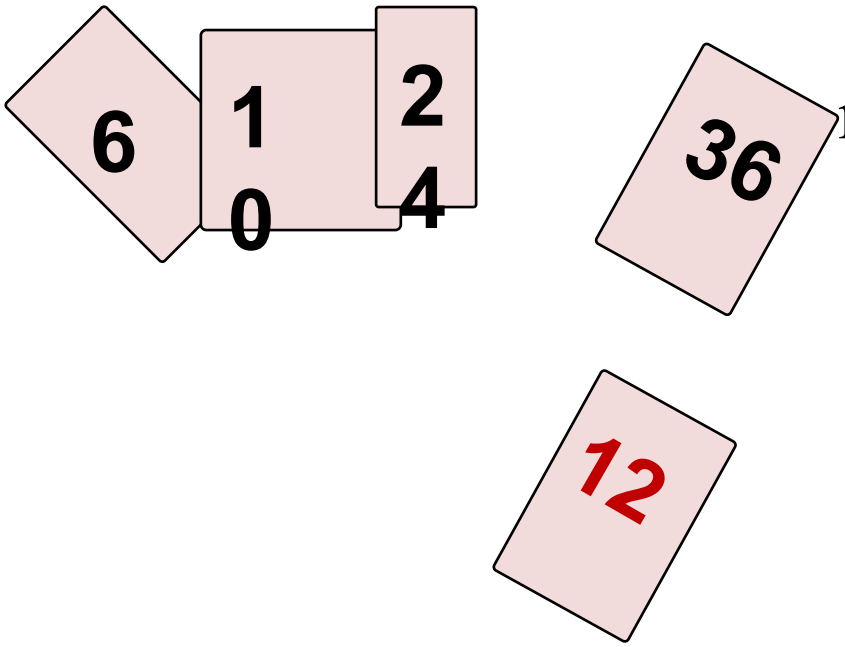
Insertion Sort (1)



Works like someone who inserts one more card at a time into a hand of cards that are already sorted

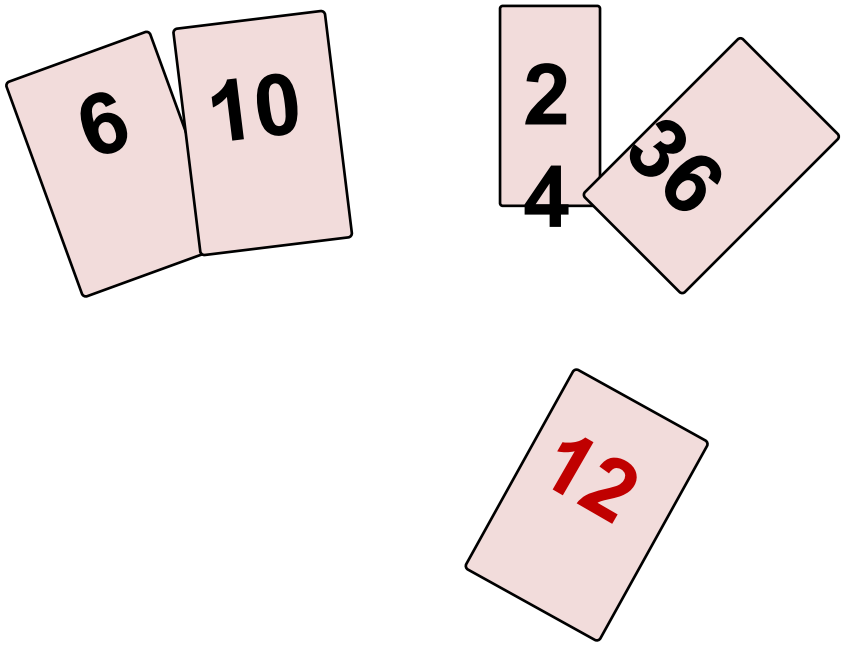
To insert 12, we need to make room for it …

Insertion Sort (2)



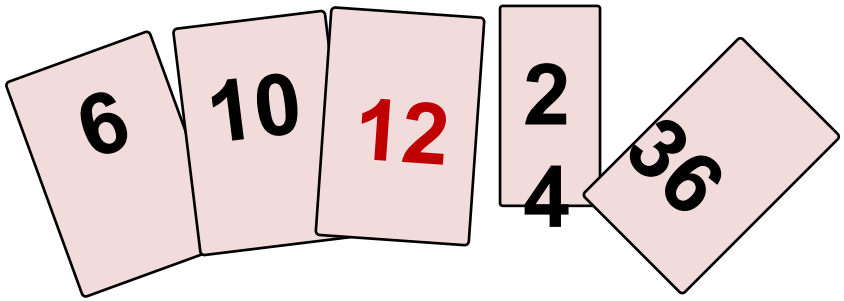
… by shifting first 36 towards right… 36

Insertion Sort (3)

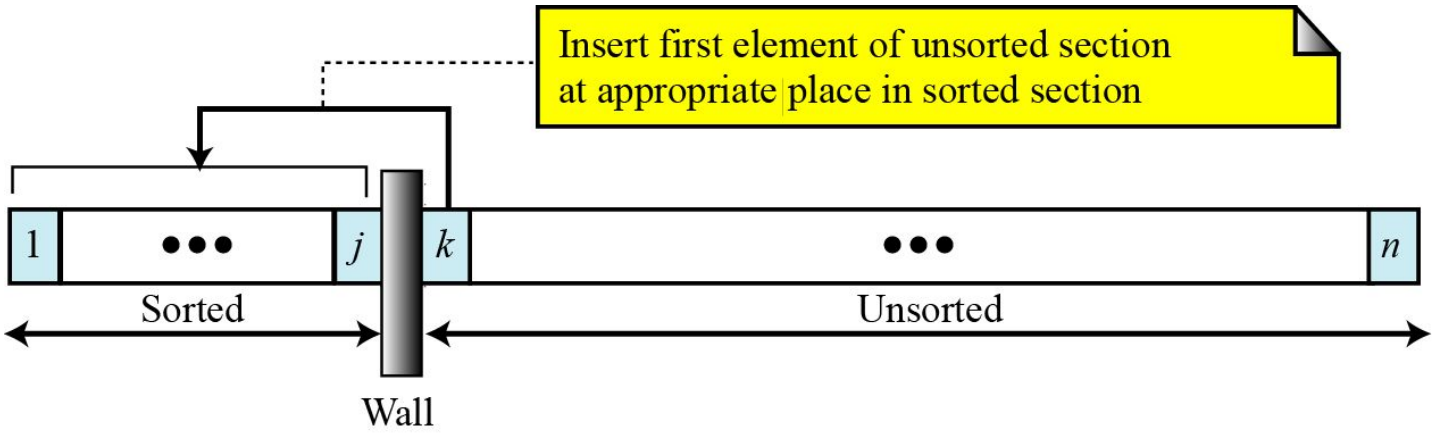


… and then shifting 24 towards right

Insertion Sort (4)



Once room is available, insert the element (12 in this case)

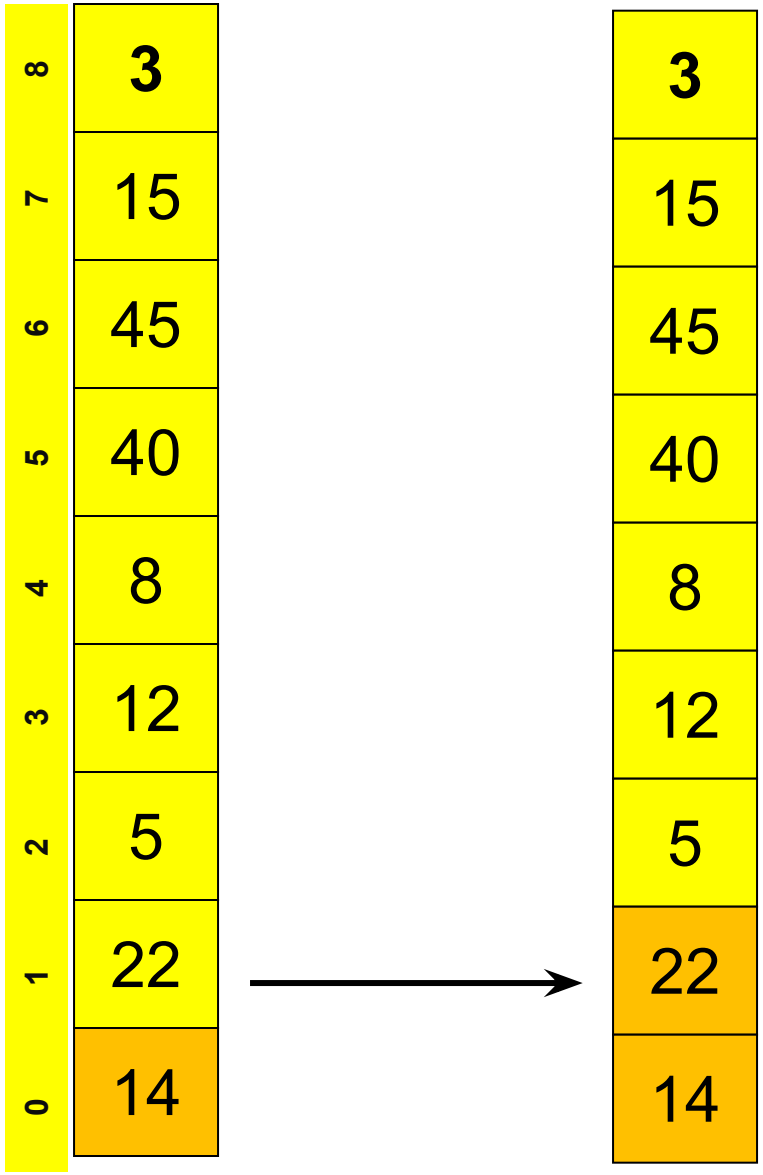


Insertion Sort: Informal (5)

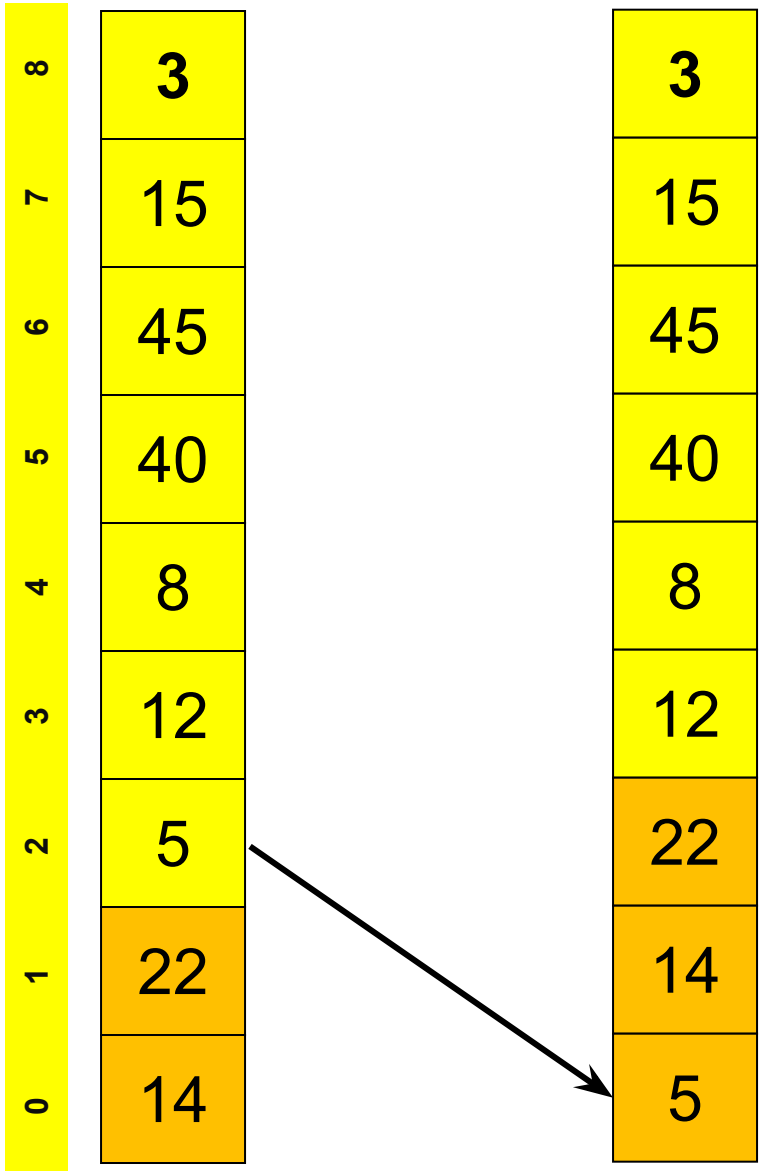
- We divide the list into two parts: Sorted and Unsorted parts
  - Initially
    - the sorted part contains the first element (at index 0)
    - the unsorted part contains the elements from index 1 to N-1
  - Then, we move element from index 1 to an appropriate position in the sorted part, keeping order intact
  - Then, we move element from index 2 to an appropriate position in the sorted part, keeping order intact

    ...
  - Finally, we move element from index N-1 to an appropriate position in the sorted part, keeping order intact

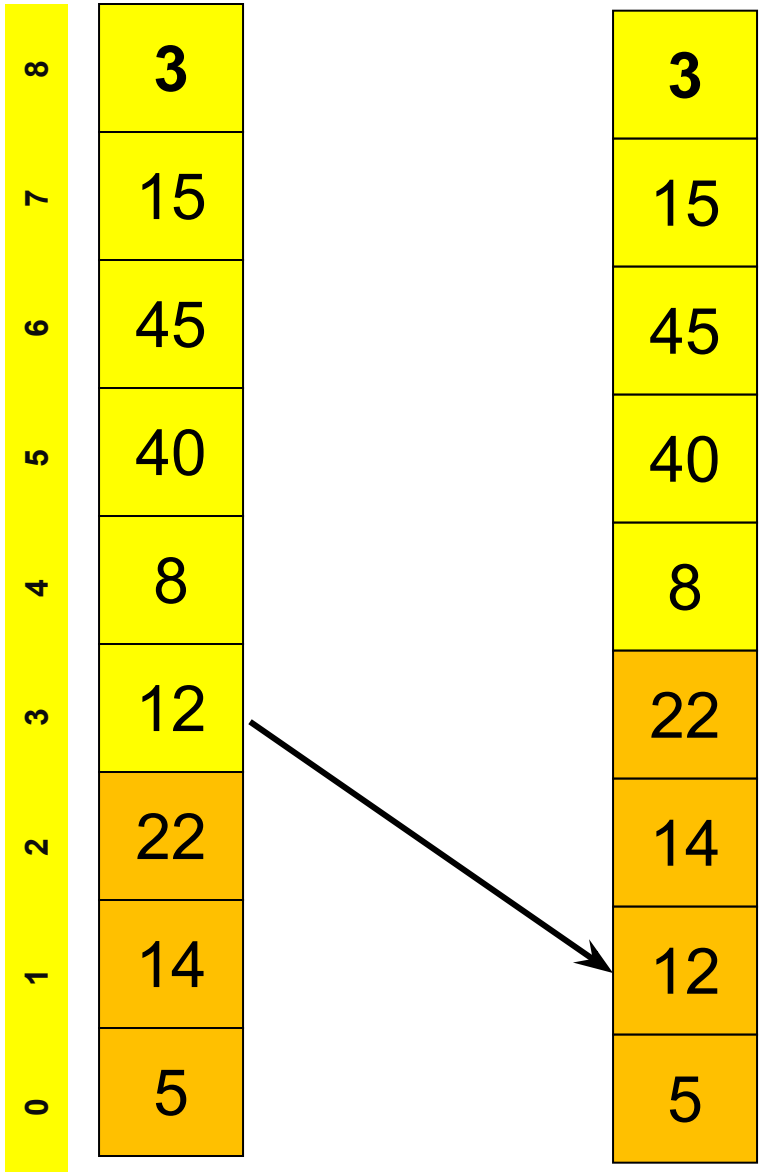
Insertion Sort example: step 1 (6)



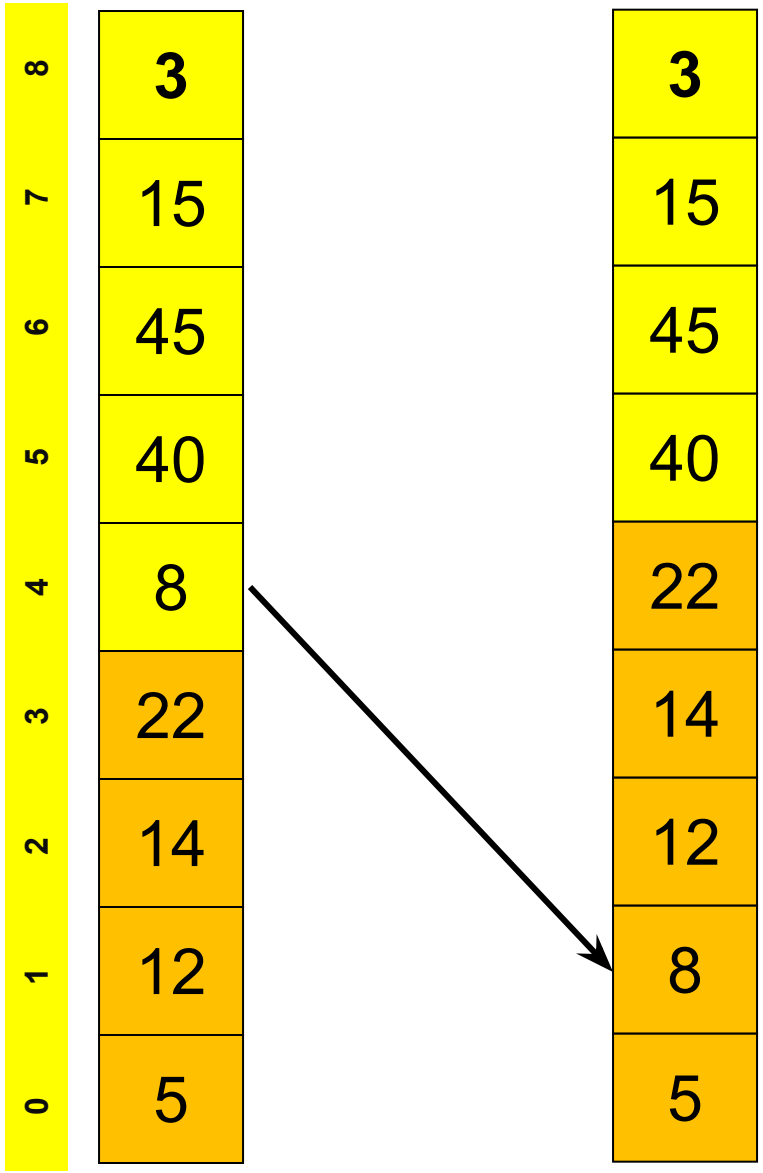
Insertion Sort example: step 2 (7)



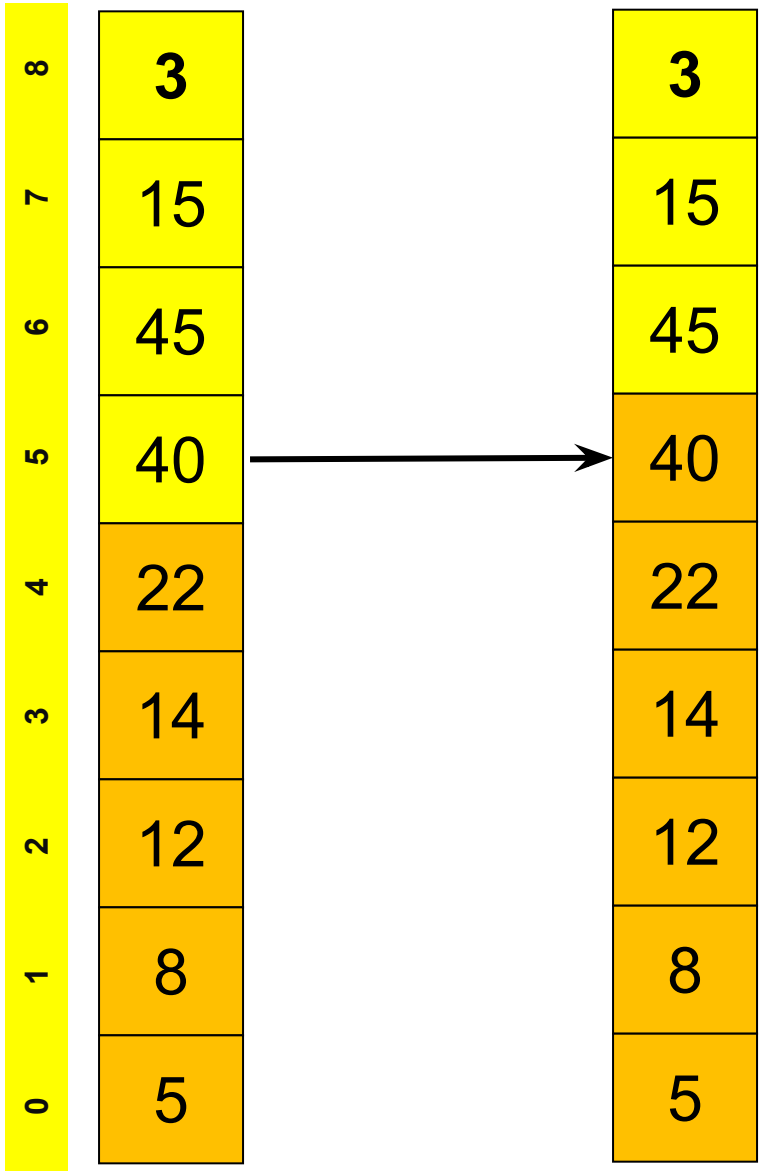
Insertion Sort example: step 3 (8)



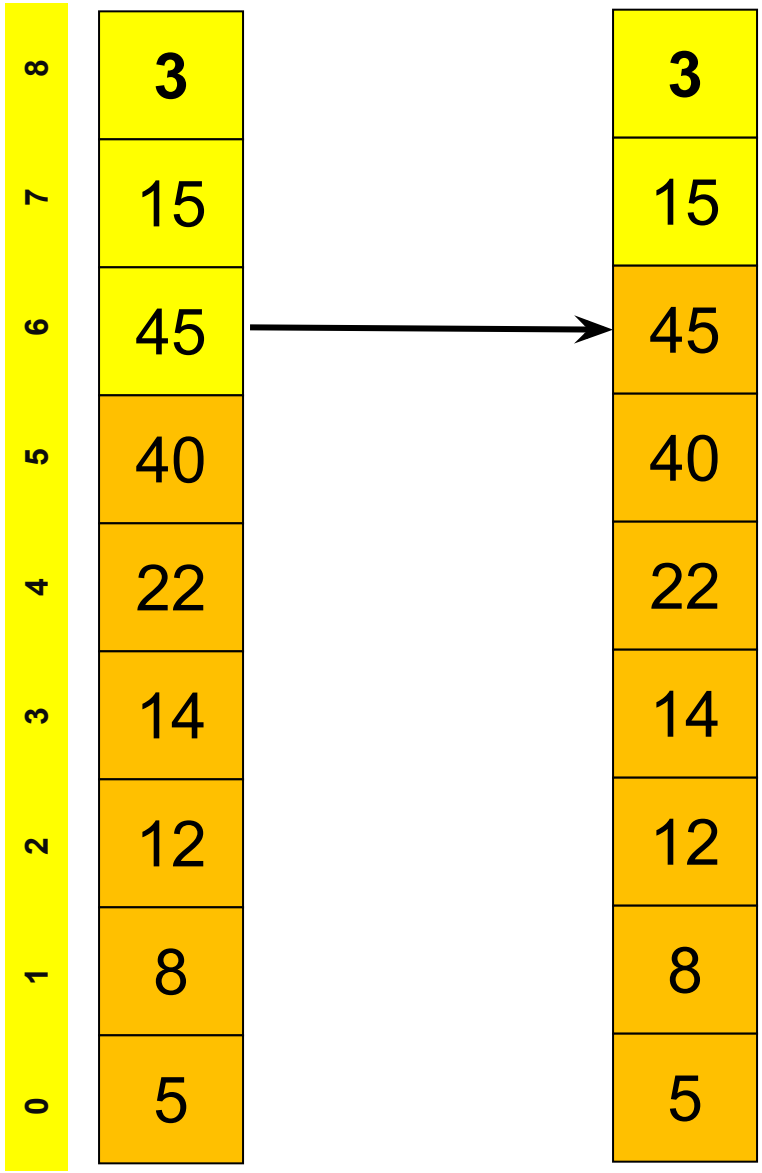
Insertion Sort example: step 4 (9)



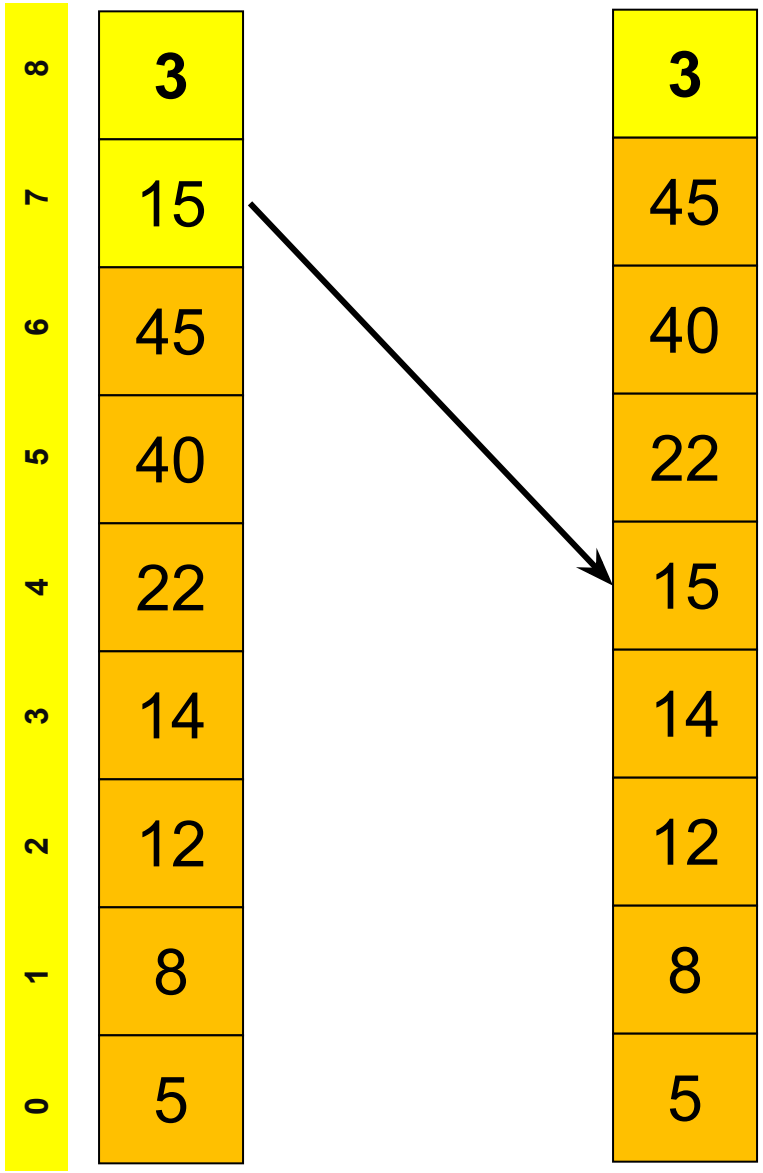
Insertion Sort example: step 5 (10)



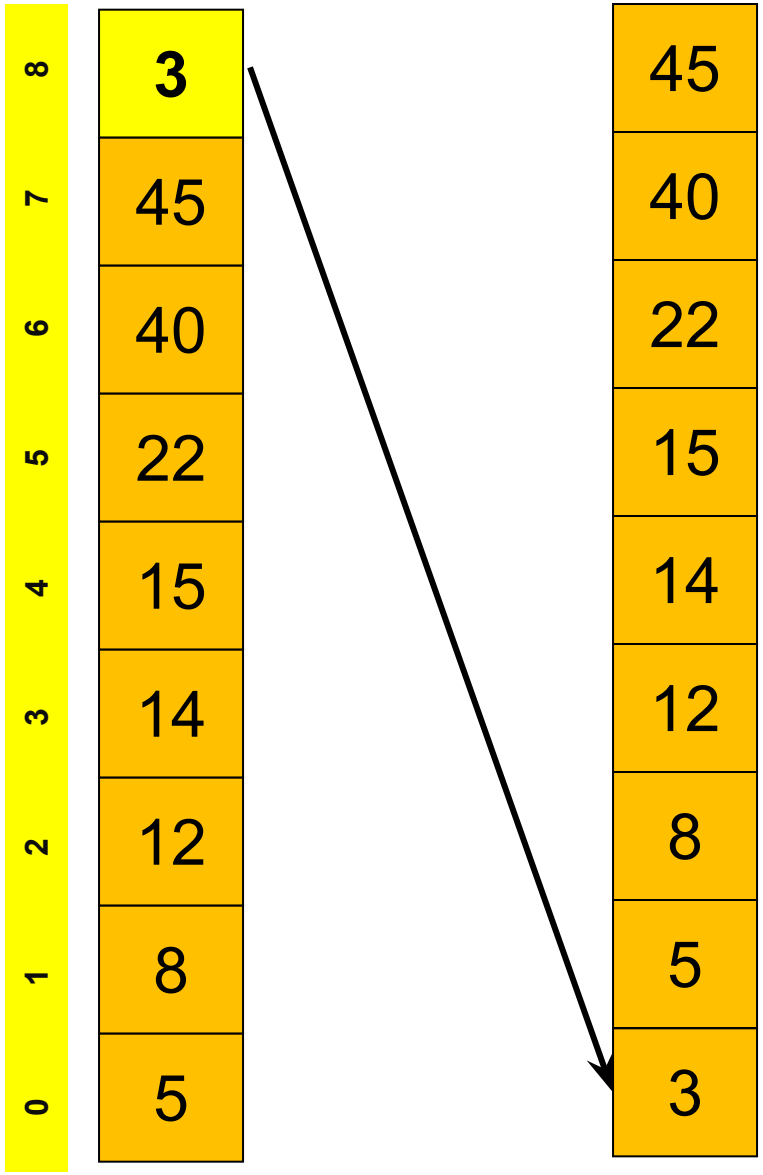
Insertion Sort example: step 6 (11)



Insertion Sort example: step 7 (12)



Insertion Sort example: step 8 (13)



Insertion Sort Algorithm (14)

Algorithm: Selection Sort
Input: List with size n
Output: List (sorted)

```
for  i = 1; i < size; i++
    j = i
    temp = List[i]
    while (j > 0  and    temp < List[j - 1])
        List[j] = List[j - 1]   // right shifting
        j - -
    end while
    List[j] = temp
End for
```

Insertion Sort Algorithm in Java (15)

```
void insertionSort(int List[]){
    int temp;
    int size = List.length;
    for (int i = 1; i < size; i++){
        int j = i;
        temp = List[i];
        while (j > 0 && temp < List[j - 1]){
            List[j] = List[j - 1]; // right shifting
            j--;
        }
        List[j] = temp;
    }
}
```

☞Time complexity of the Insertion Sort algorithm is $O(n^2)$. Think why?