Computer Science Department
College of Computer at Al-Lith

جامعة أم القرى
UMM AL-QURA UNIVERSITY

Data Structures

Chapter 3: Sorting

Instructor Maher Hadiji
hdiji.maher@gmail.com

2015-2016

Sorting Algorithms

1. Sorting Definition
2. Bubble Sort
3. Selection Sort
4. Insertion Sort

1. Sorting Definition

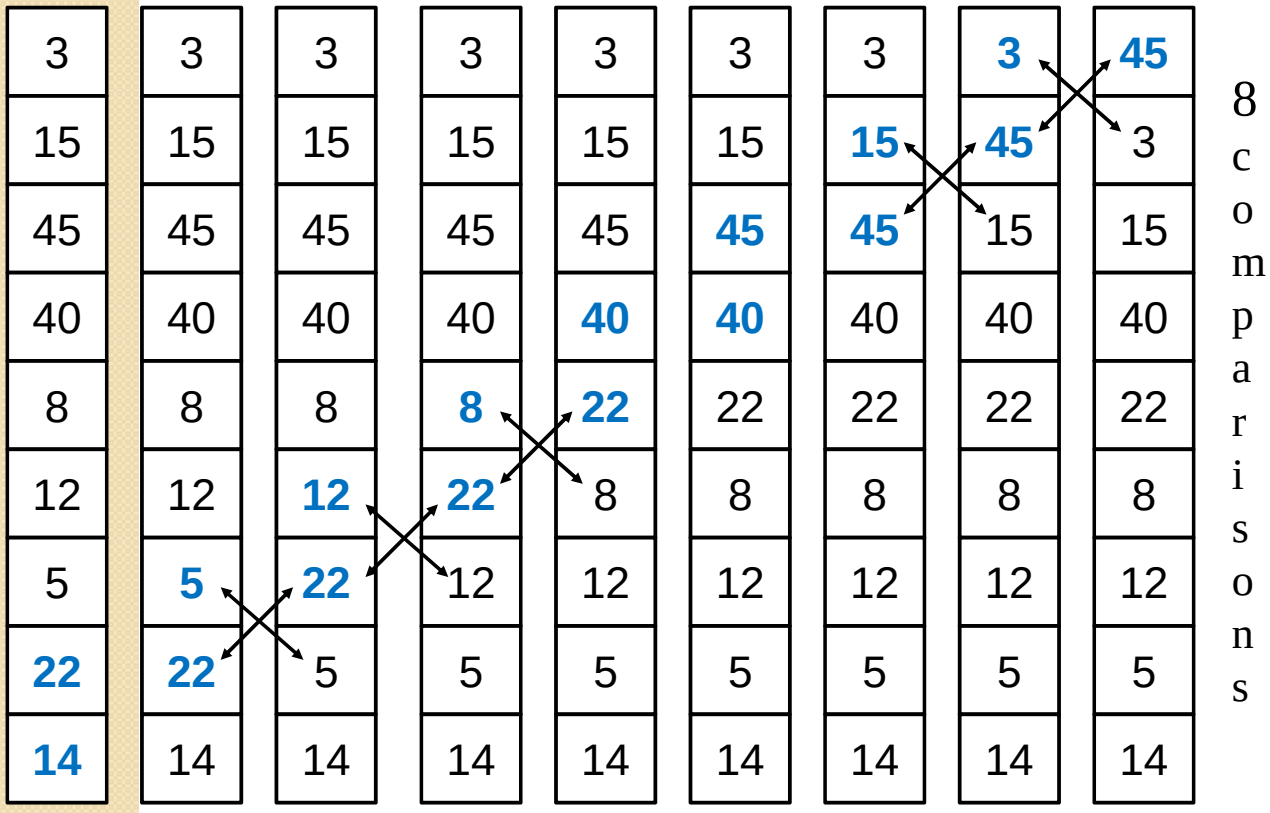
- Motivation
  - Generally, to arrange a list of elements in some order
- List of numbers
  - 10 20 50 30 40 60 25 (Unsorted list)
  - 10 20 25 30 40 50 60 (Sorted list, ascending)
  - 60 50 40 30 25 20 10 (Sorted list, descending)
- List of alphabets
  - P A K I S T A N (Unsorted list)
  - A A I K N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

2.Bubble Sort

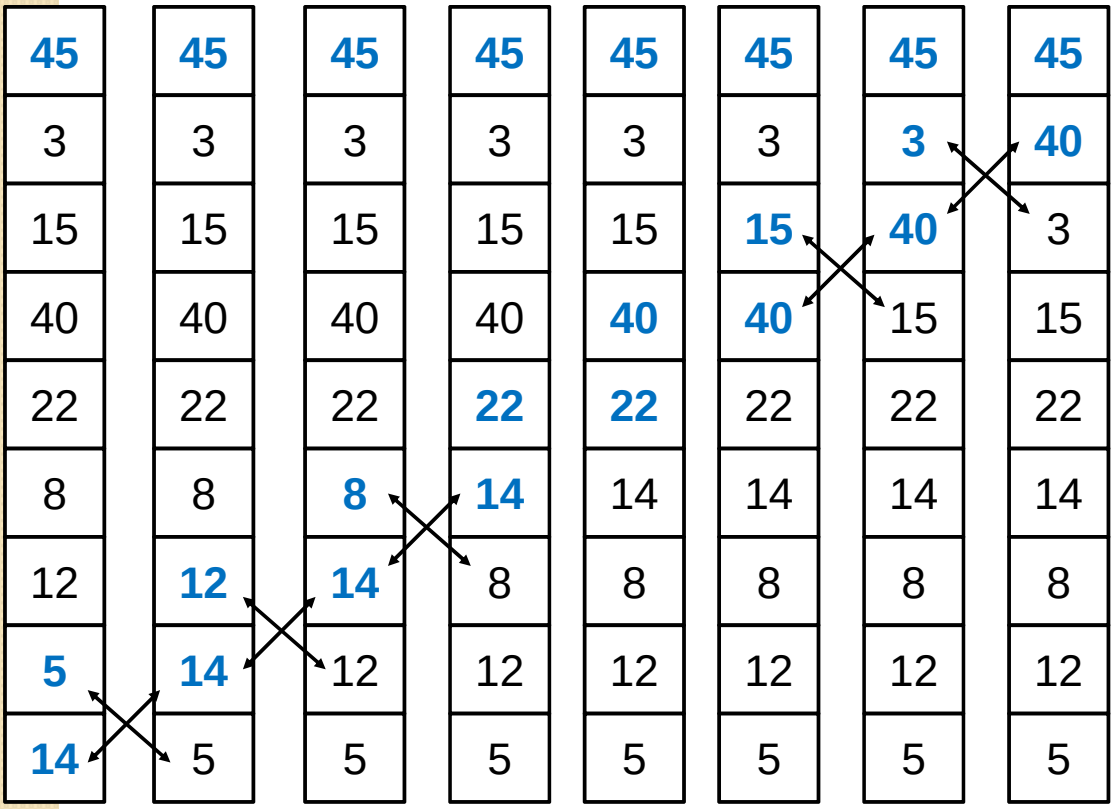
Bubble Sort Algorithm: Informal

- 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 in Action: Phase 1

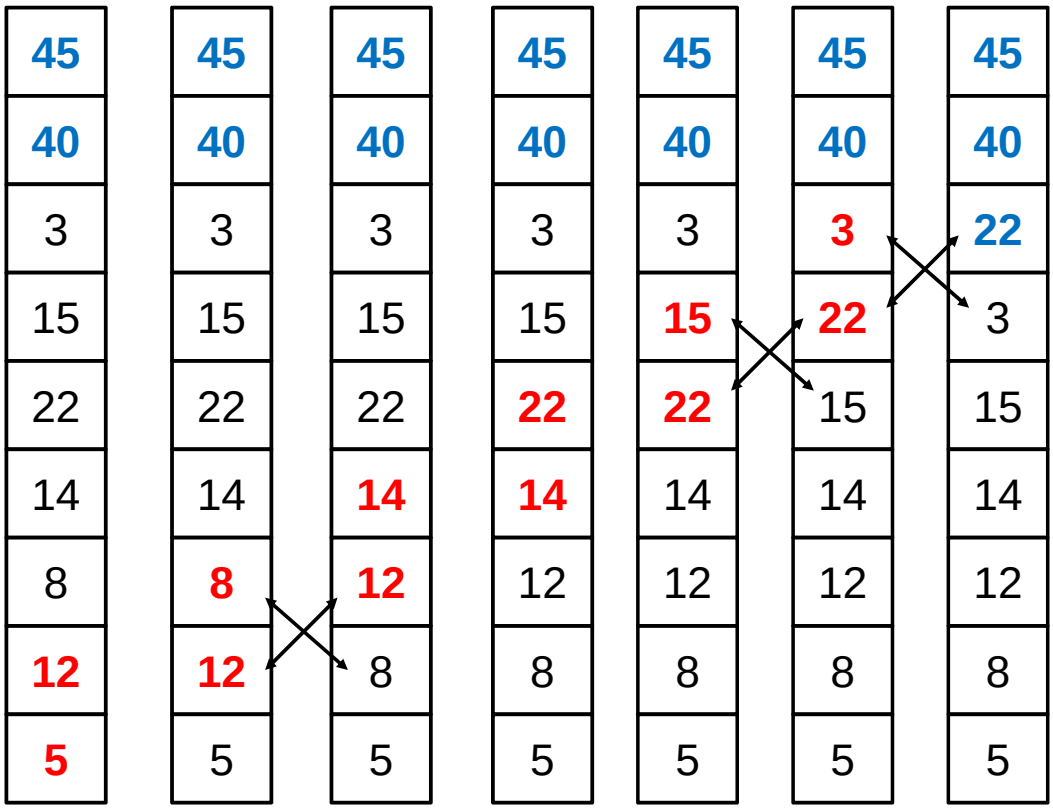


Bubble Sort in Action: Phase 2



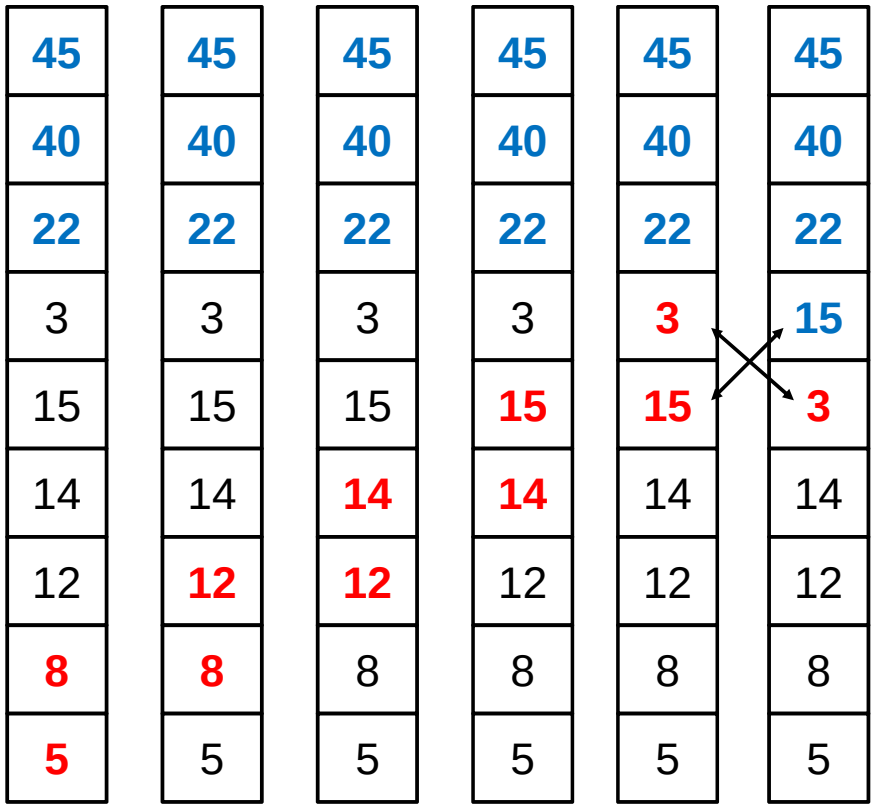
7 comparisons

Bubble Sort in Action: Phase 3



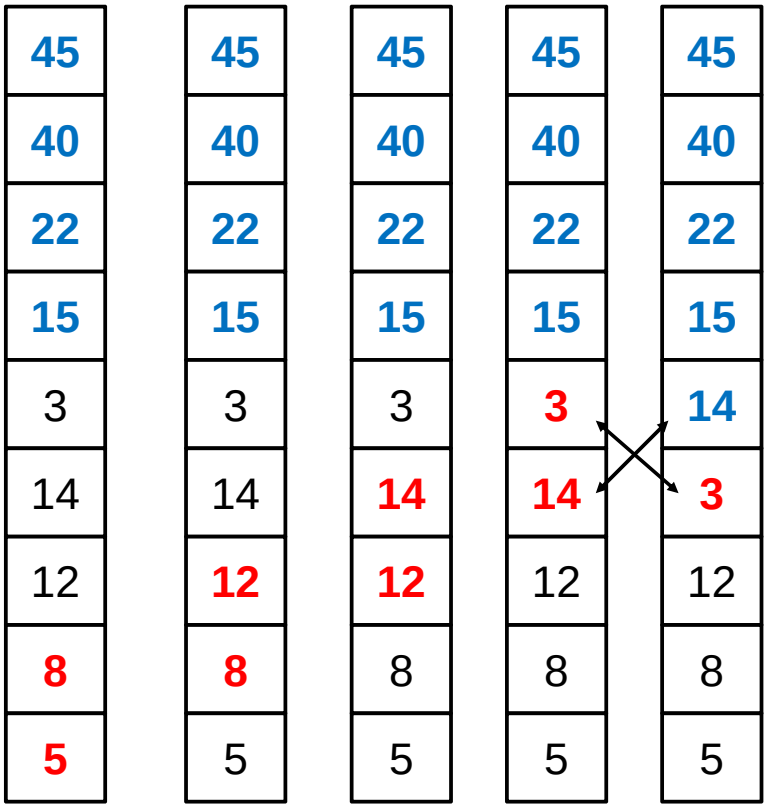
6 comparisons

Bubble Sort in Action: Phase 4



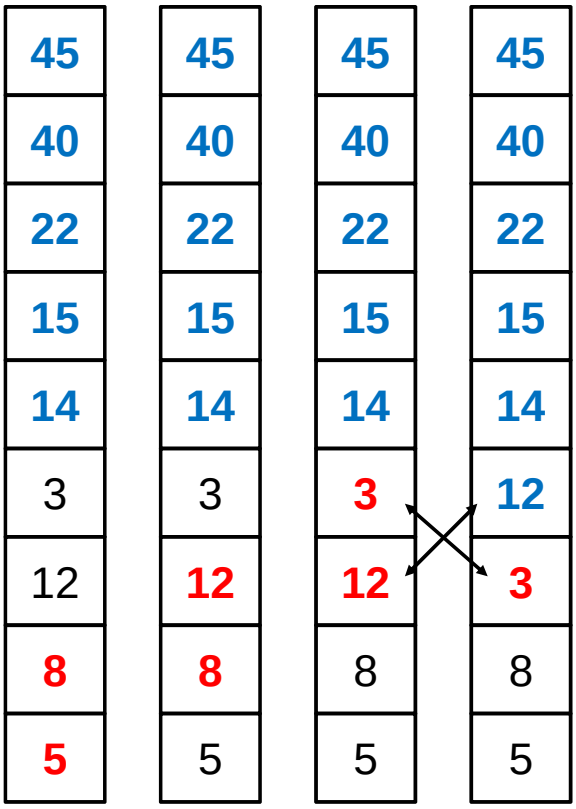
5 comparisons

Bubble Sort in Action: Phase 5



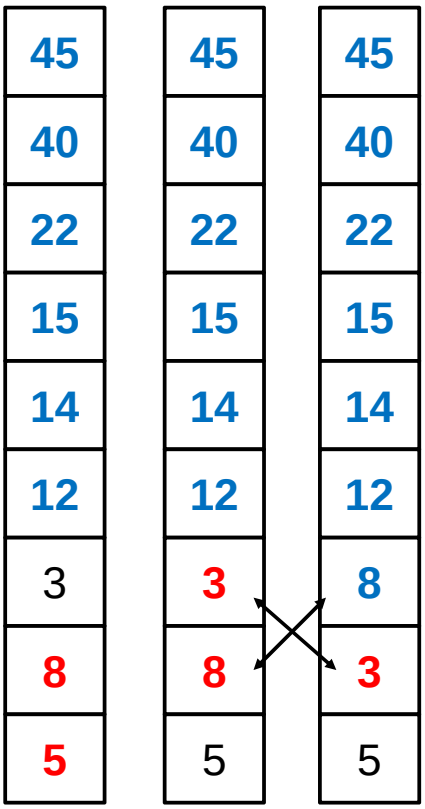
4 comparisons

Bubble Sort in Action: Phase 6



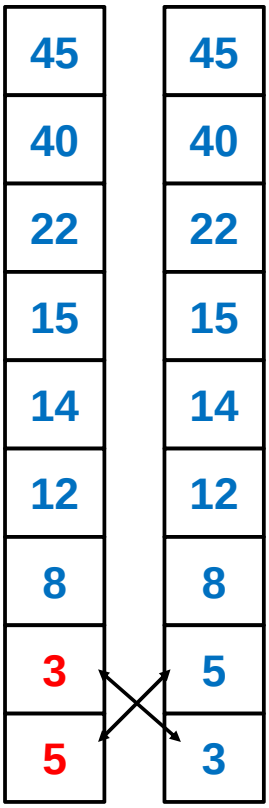
3 comparisons

Bubble Sort in Action: Phase 7



2 comparisons

Bubble Sort in Action: Phase 8



1 comparison

Bubble Sort Algorithm in Java

```
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]){
                temp = List[j];
                List[j] = List[j+1];
                List[j+1] = temp;
            }
        }
    }
}
```

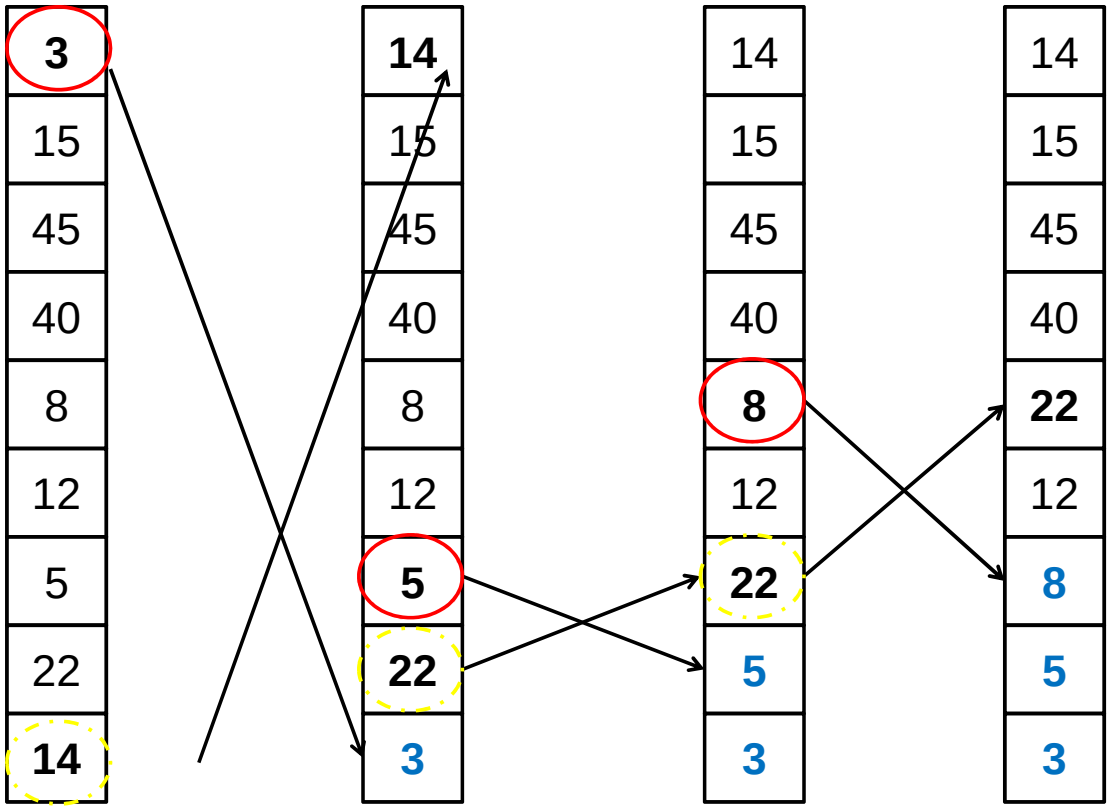
☞Time complexity of the Bubble Sort algorithm is $O(n^2)$. Think why?

3.Selection Sort

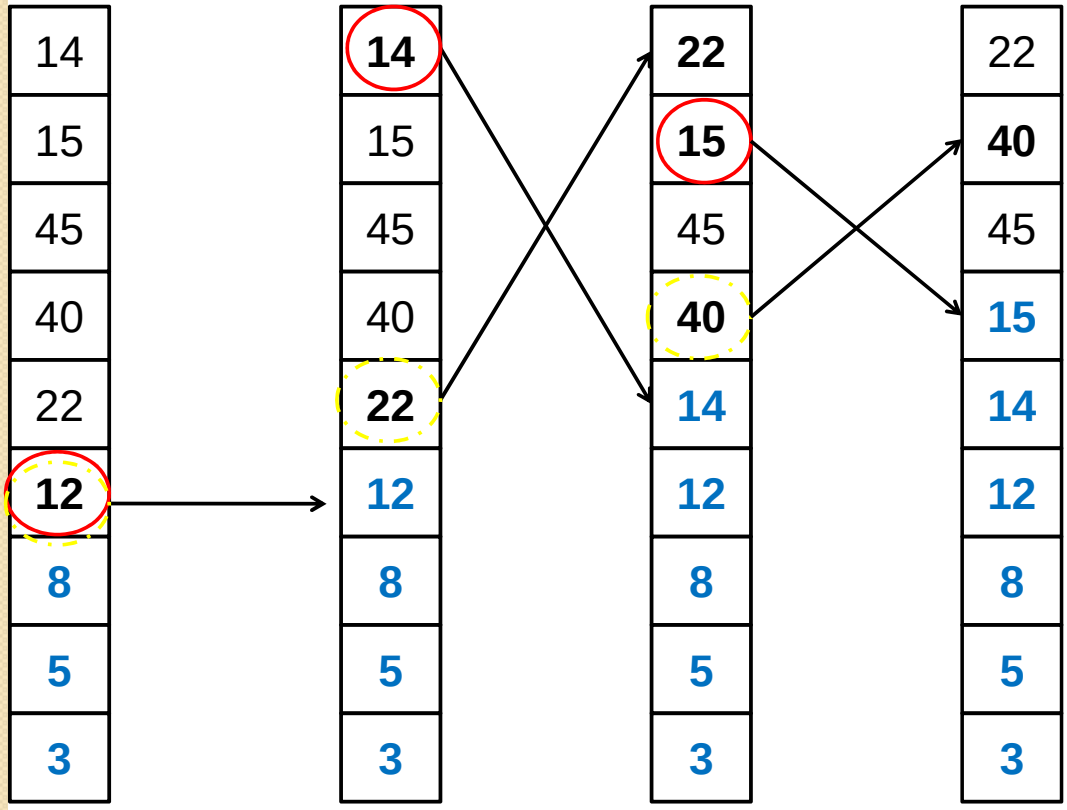
Selection Sort: Algorithm Informal

- 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

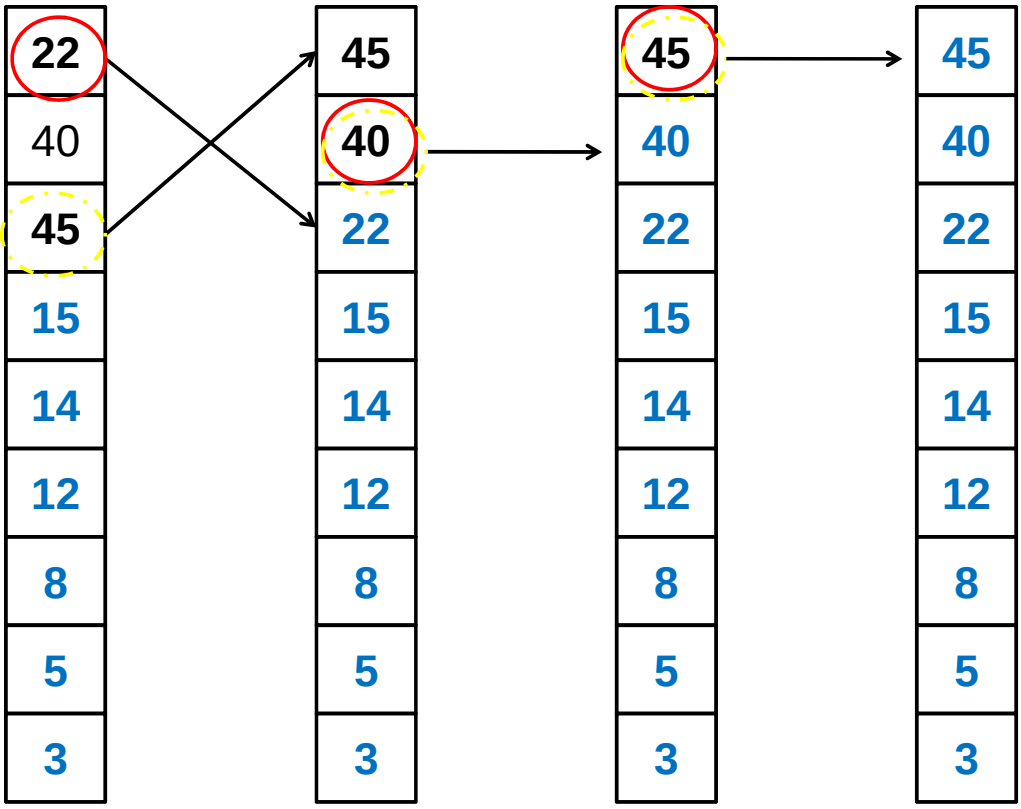
Selection Sort in Action



Selection Sort in Action



Selection Sort in Action



Selection Sort Algorithm in Java

```
void selectionSort(int List[]){
    int temp, min;
    int size = List.length;
    for (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];
        List[i] = temp;
    }
}
```

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

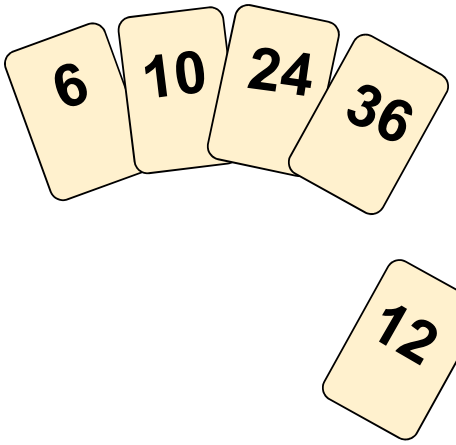
Bubble Sort vs. Selection Sort

- 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

Insertion Sort

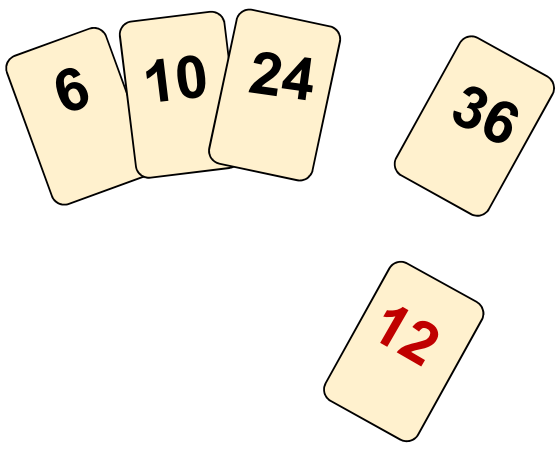
Works like someone who inserts one more card at a time into a hand of cards that are already sorted



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

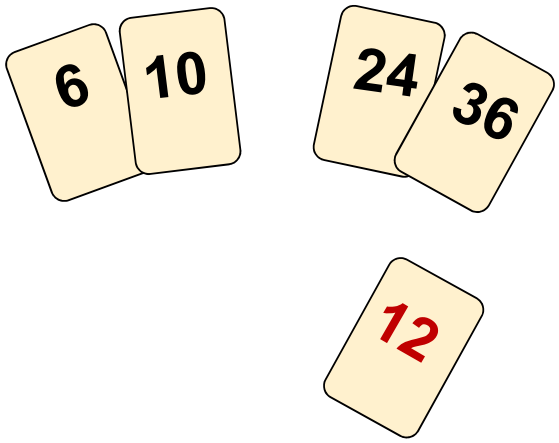
Insertion Sort

… by shifting first 36 towards right…



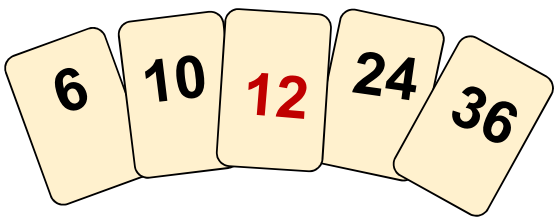
Insertion Sort

… and then shifting 24 towards right



Insertion Sort

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



Insertion Sort: Informal

- 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 Algorithm in Java

```
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?