Lab 3

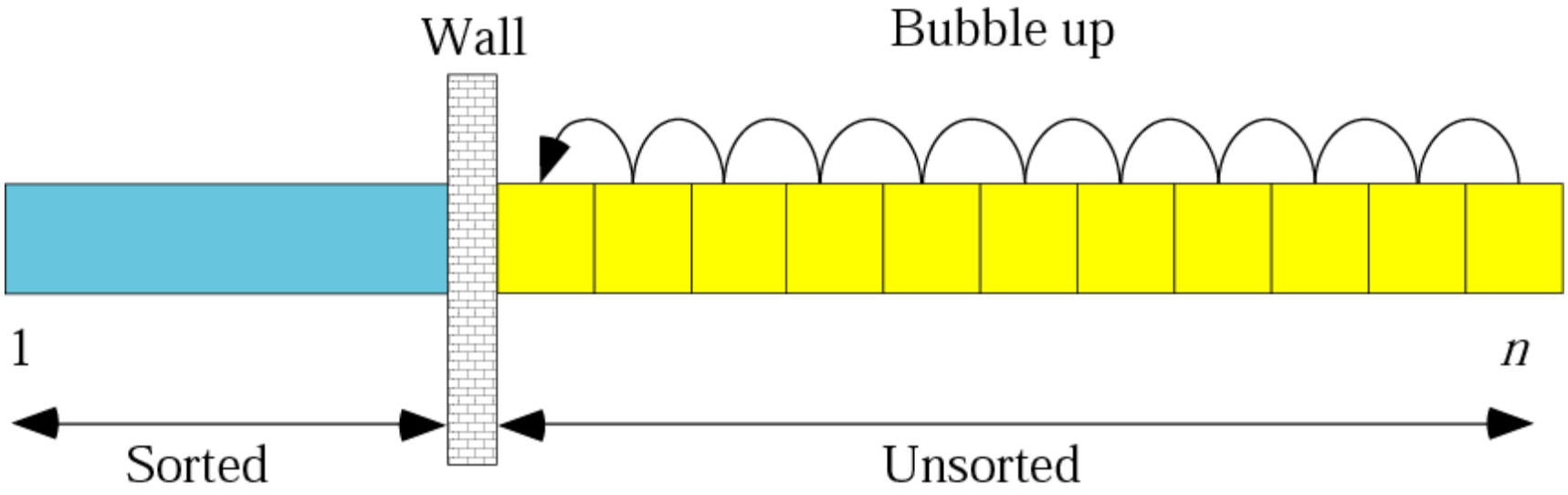
Guideline

Sort with Assembly Language

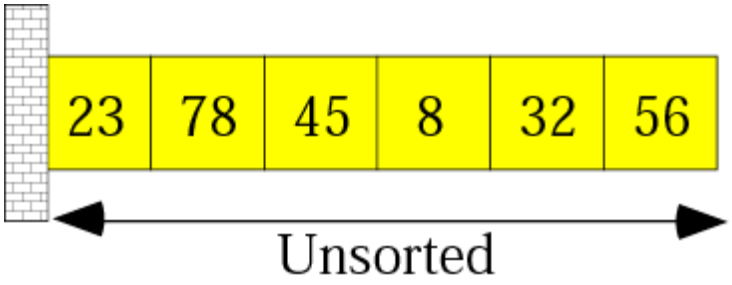
Sorting Algorithm

- Bubble Sort
- Selection Sort
- Insertion Sort
- Quick Sort

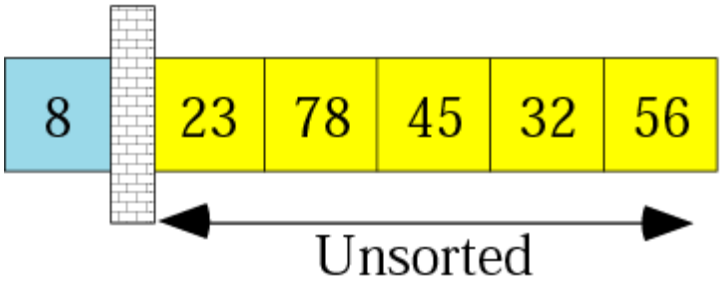
Bubble Sort



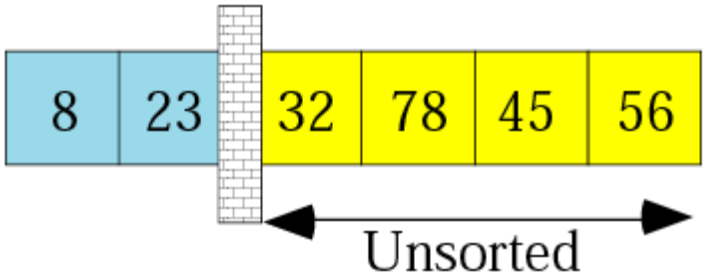
Example of Bubble Sort



Original list

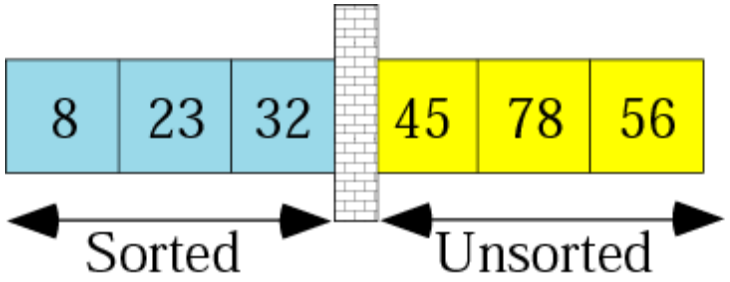


After pass 1

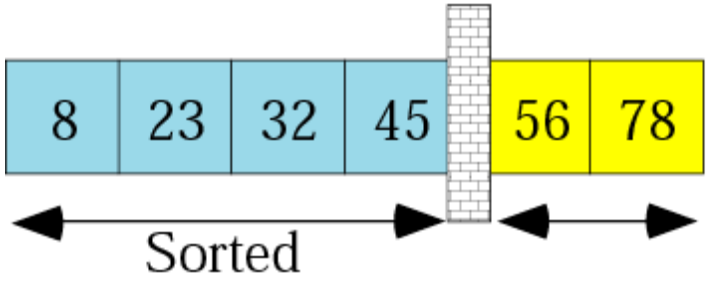


After pass 2

Example of Bubble Sort (con)

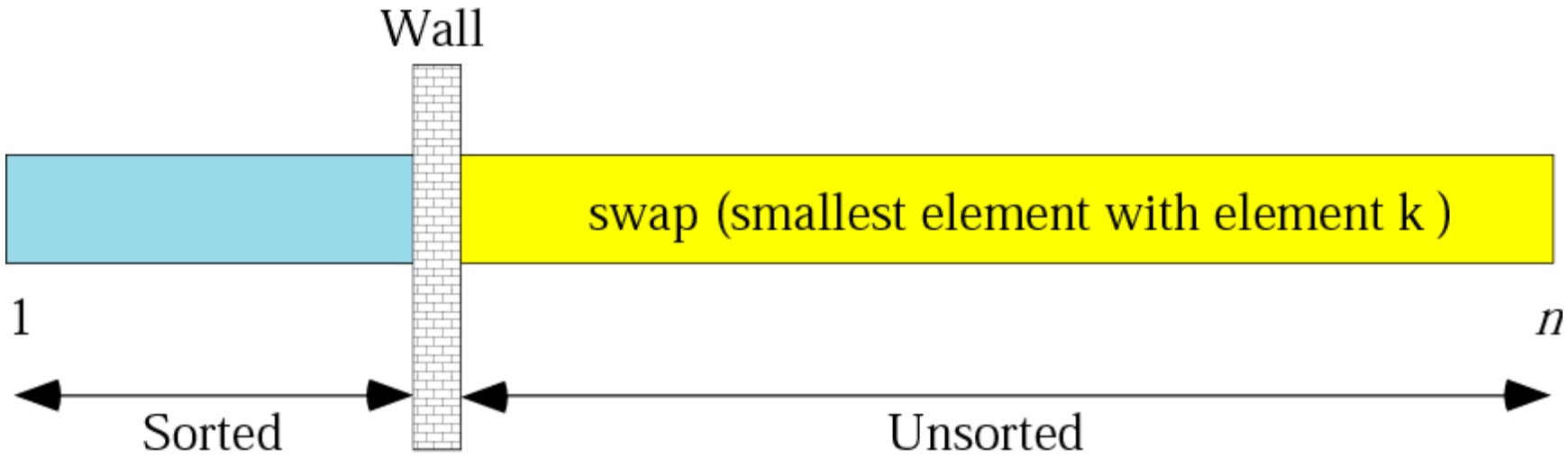


After pass 3

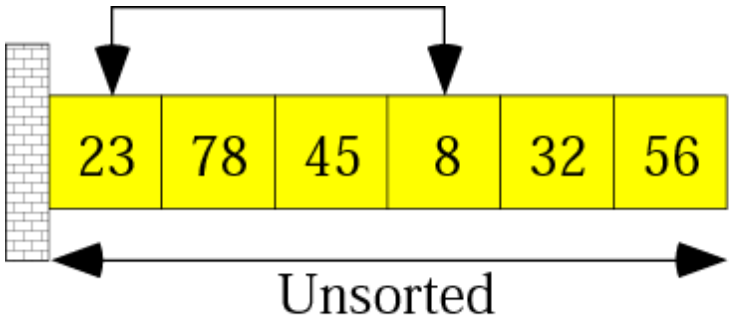


After pass 4
Sorted

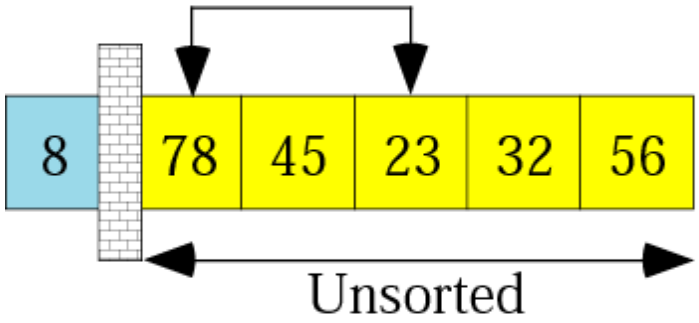
Selection Sort



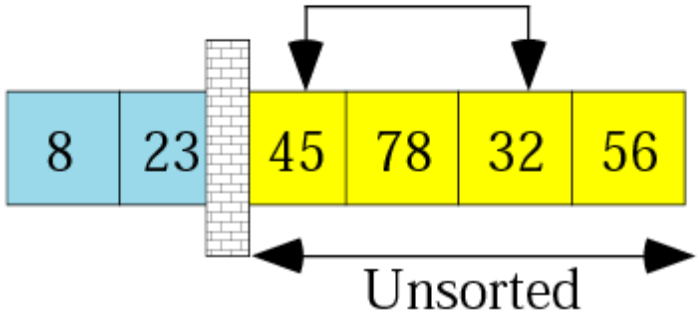
Example of Selection Sort



Original list

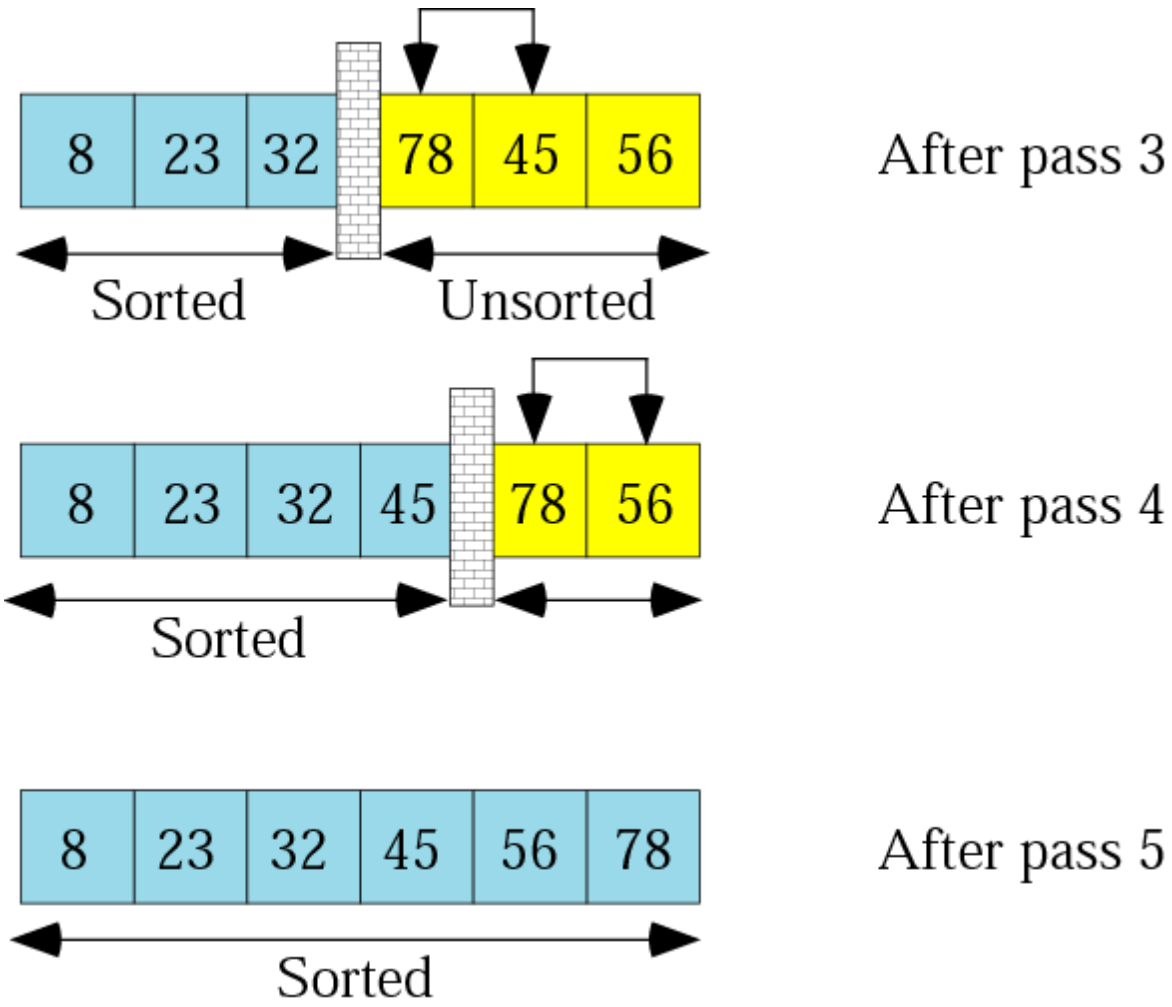


After pass 1

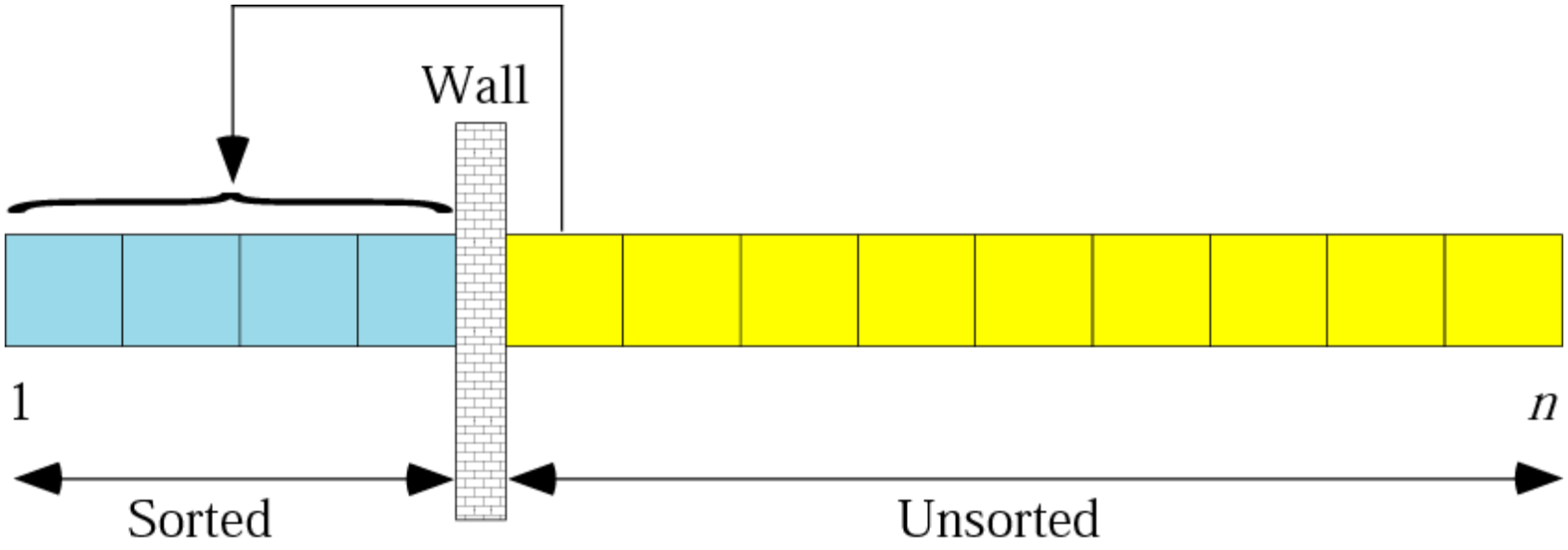


After pass 2

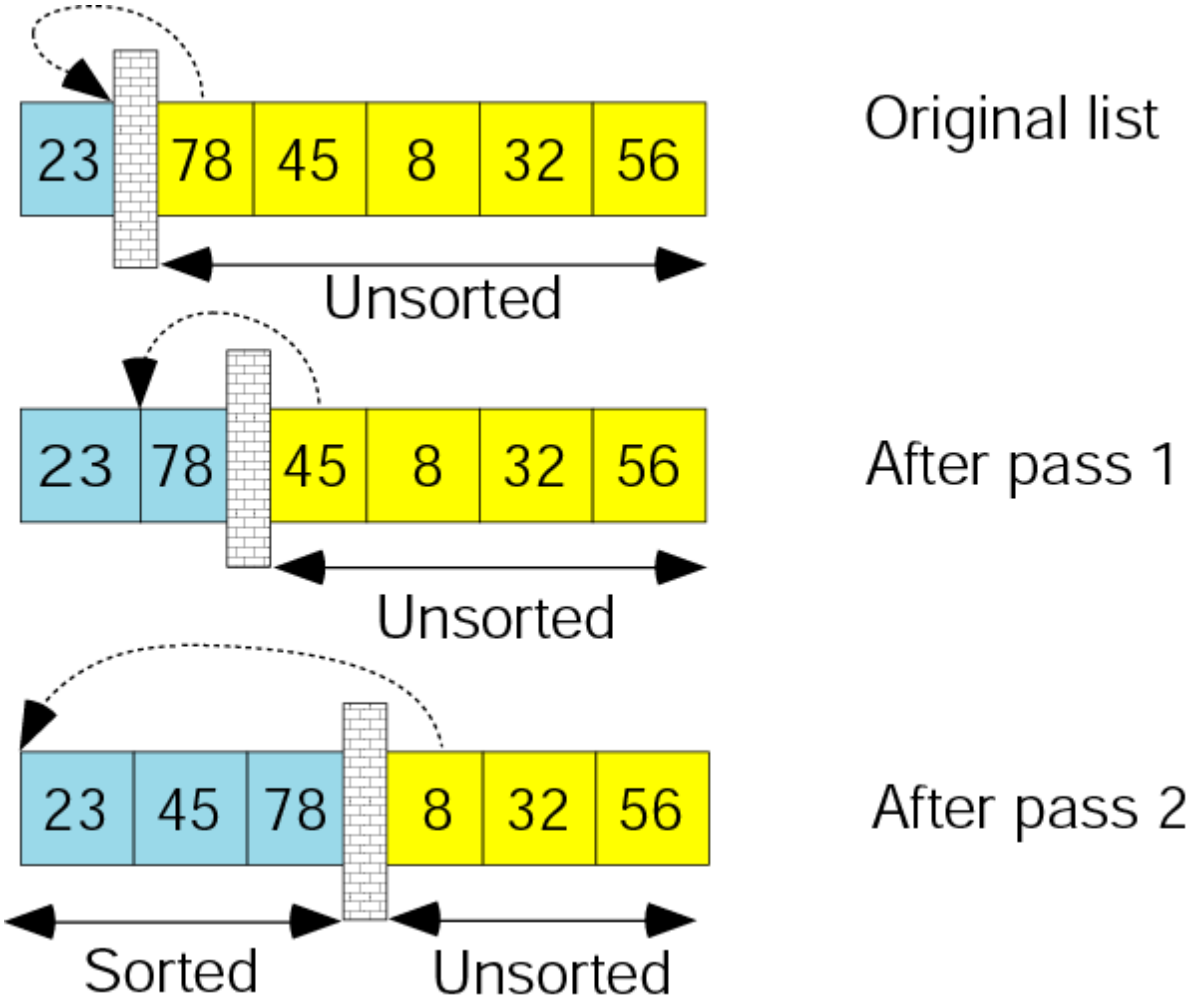
Example of Selection Sort (con)



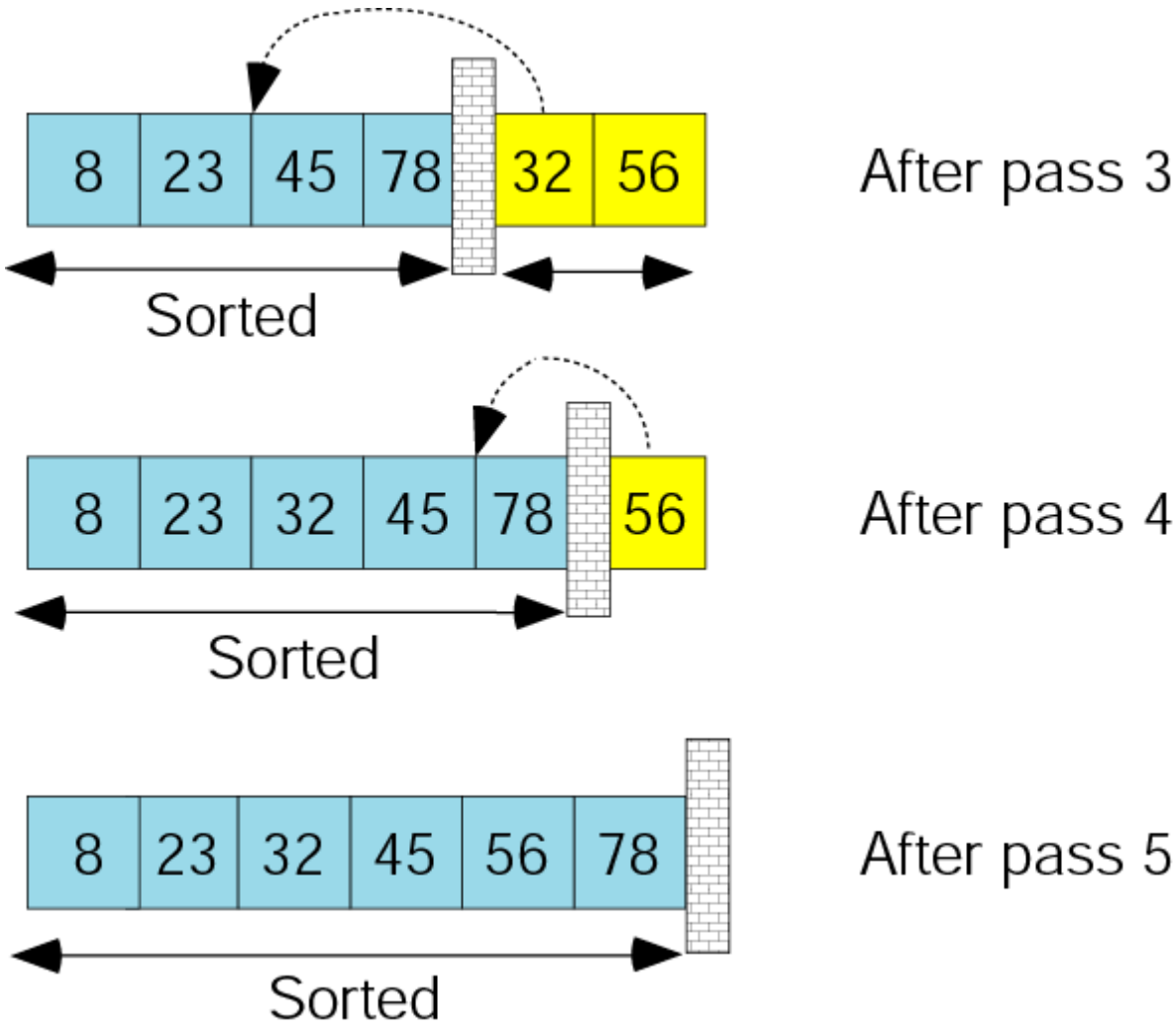
Insertion Sort



Example of Insertion Sort



Example of Insertion Sort (con)



Quick Sort

- Quick Sort is the fastest known sorting algorithm in practice.
- The algorithm is based on divide-and-conquer recursive method.
- The general idea is to
  - pick an element, called a pivot.
  - divide the array of elements into two halves excluding the pivot, *v*.
  - the first half contains all the elements with the value less than *v*,
  - the second half has all the elements with the values greater than *v*.
  - the two halves are then quicksort again recursively.

The Quick Sort Idea

Unordered list of values

The Quick Sort Idea

Choose pivot value

The Quick Sort Idea

Low list
(≤ 17)

High list
(> 17)

The Quick Sort Idea

Recursively
apply quicksort
to low list

The Quick Sort Idea

Recursively
apply quicksort
to high list

The Quick Sort Idea

Sorted list of values

The Quick Sort algorithm

- Pick a pivot, *v*, and swap it with the last element in an array.
- Start a pointer, *i*, at position 0 and start a pointer, *j*, at position *N-1*.
- Move pointer *i* to the right until *i* points at an element larger than or equal to *v*.
- Move pointer *j* to the left until *j* points at an element smaller than or equal to *v*.
- If $i > j$, swap an element that is pointed by *i* with *v*.
- If $i < j$, swap an element that is pointed by *i* with an element that is pointed by *j*.