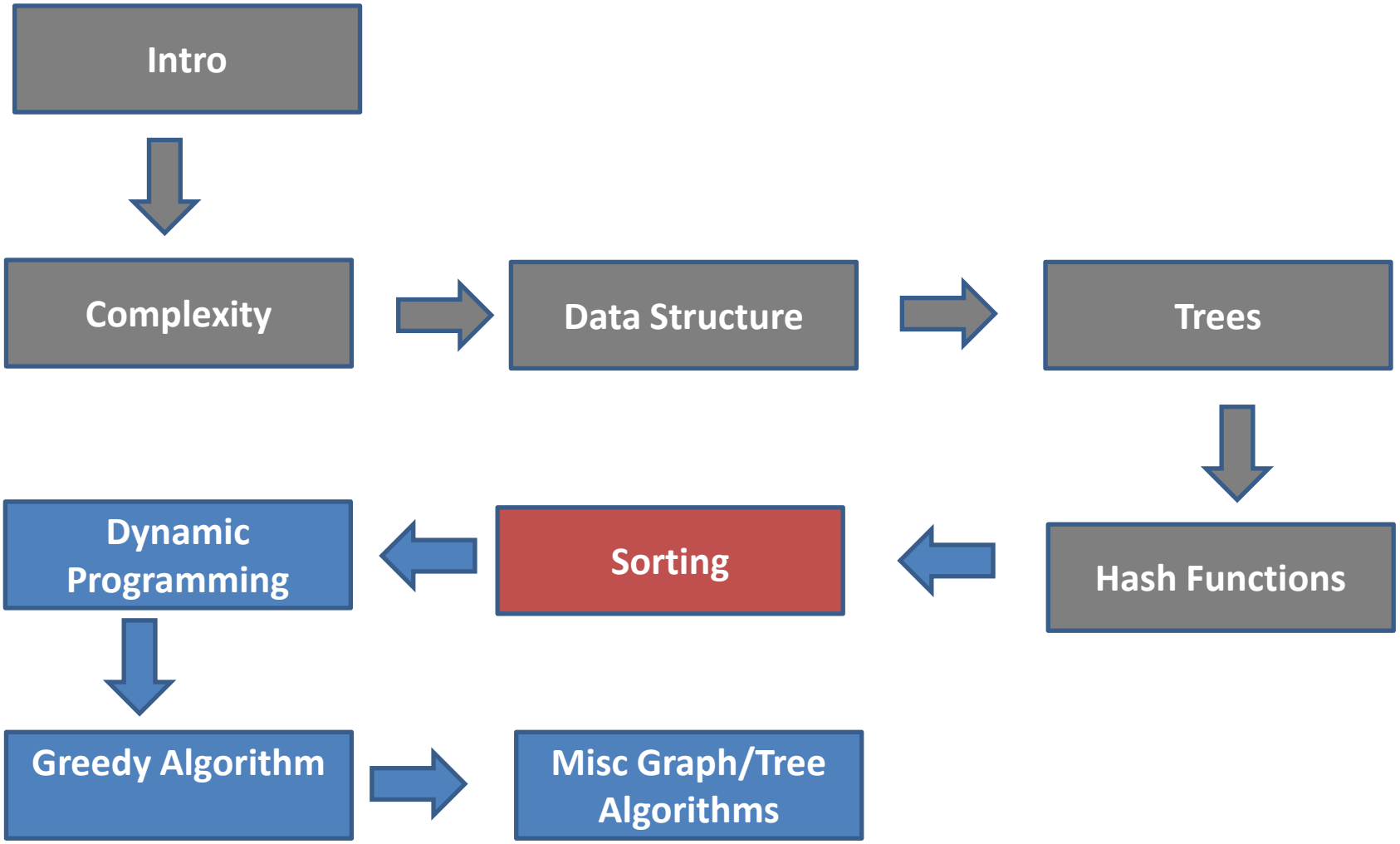
An Introduction to Algorithms

By

Hossein Rahmani

h_rahmani@iust.ac.ir

http://webpages.iust.ac.ir/h_rahmani/


Intro
Complexity
Data Structure
Trees
Hash Functions
Sorting
Dynamic Programming
Greedy Algorithm
Misc Graph/Tree Algorithms

The Sorting Problem

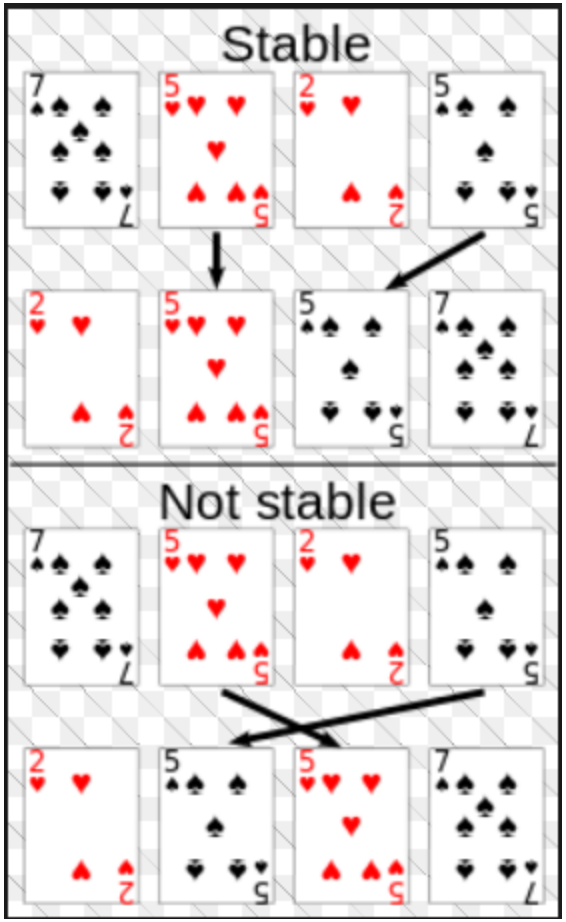
- **Input:**
  - A sequence of n numbers $a_1, a_2, \ldots, a_n$
- **Output:**
  - A permutation (reordering) $a_1', a_2', \ldots, a_n'$ of the input sequence such that $a_1' \leq a_2' \leq \cdots \leq a_n'$

Some Definitions

- Internal Sort
  - The data to be sorted is all stored in the computer's main memory.
- External Sort
  - Some of the data to be sorted might be stored in some external, slower, device.
- In Place Sort
  - The amount of extra space required to sort the data is constant with the input size.

Stability

- A STABLE sort preserves relative order of records with equal keys



Varied range of sorting algorithms

- Selection sort
- Insertion sort
- Bubble sort
- Merge sort
- Heapsort
- Quicksort
- Linear Sorting
- …

$O(N^2)$ Sorting Algorithms

Insertion Sort

Selection Sort

Bubble Sort

Insertion Sort

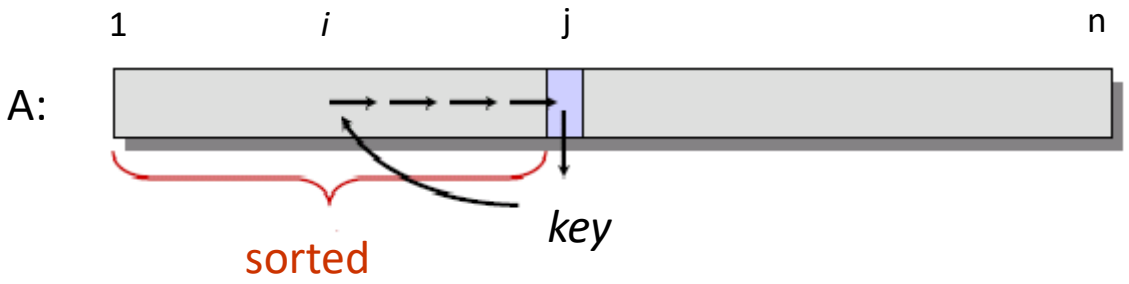
An Example: Insertion Sort

```
InsertionSort(A, n) {
  for i = 2 to n {
      key = A[i]
      j = i - 1;
      while (j > 0) and (A[j] > key) {
            A[j+1] = A[j]
            j = j - 1
      }
      A[j+1] = key
  }
}
```

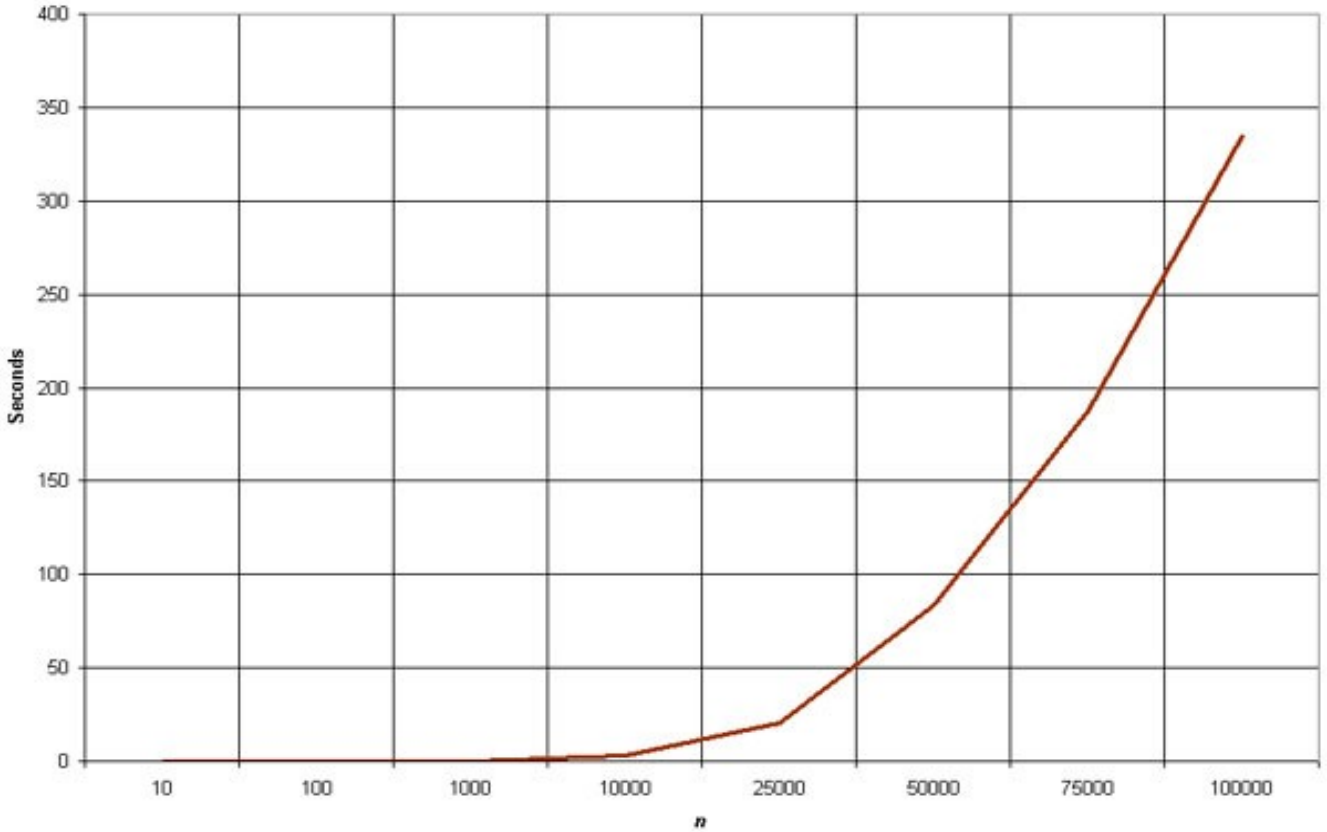
INSERTION-SORT

"pseudocode"

```
INSERTION-SORT (A, n)        ▷ A[1 . . n]
    for j ←2 to n
     do key ← A[ j]
        i ← j –1
        while i > 0 and A[i] > key
            do A[i+1] ← A[i]
               i ← i –1
        A[i+1] = key
```



Empirical Analysis of Insertion Sort



The graph demonstrates the *n*^2 complexity of the insertion sort.

Source: http://linux.wku.edu/~lamonml/algor/sort/insertion.html

Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

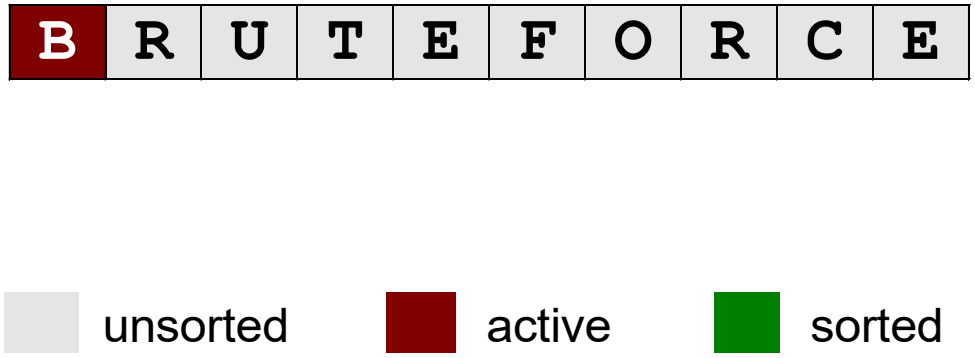
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.

B	R	U	T	E	F	O	R	C	E

Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

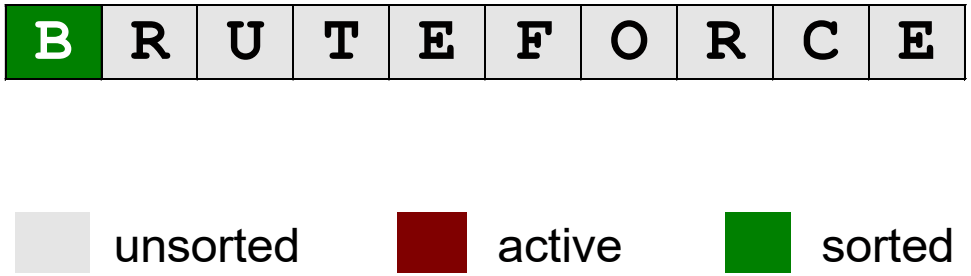
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

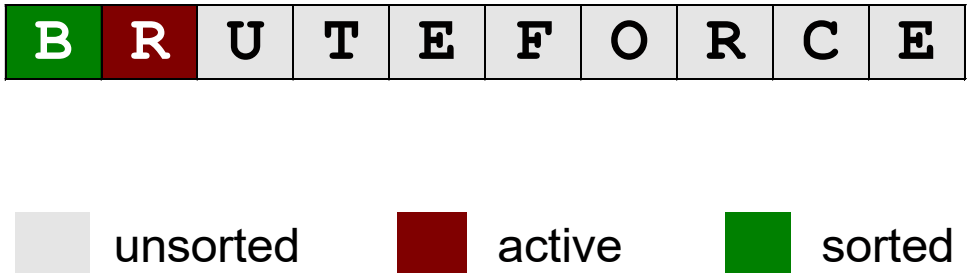
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

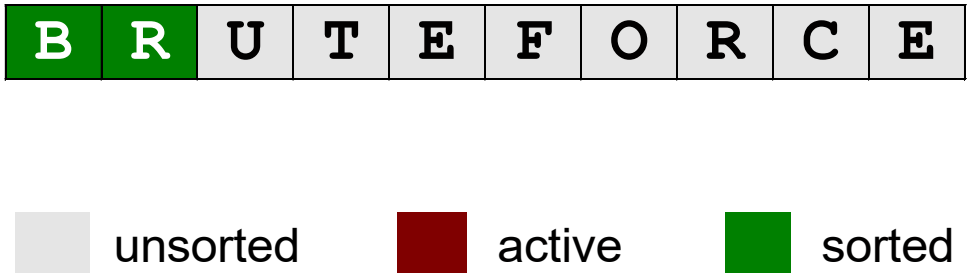
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

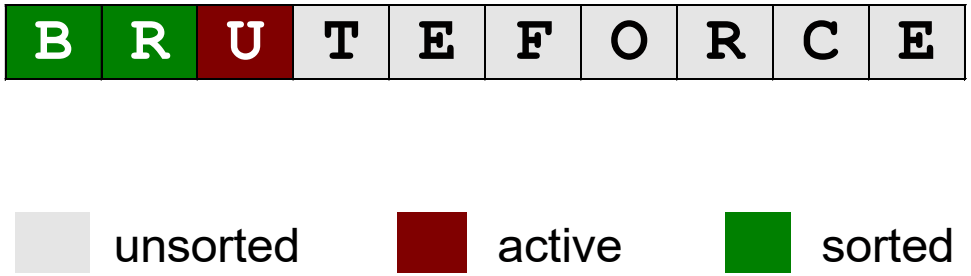
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

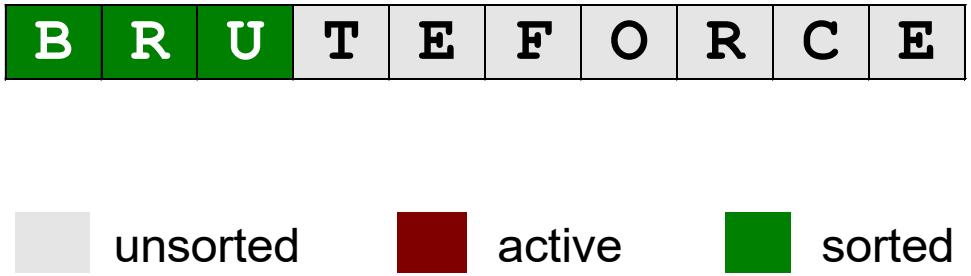
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

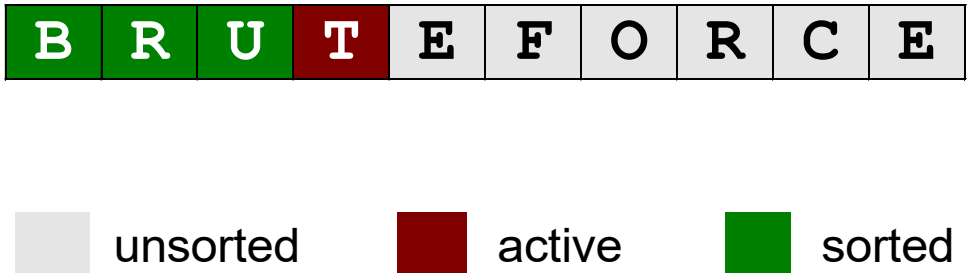
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

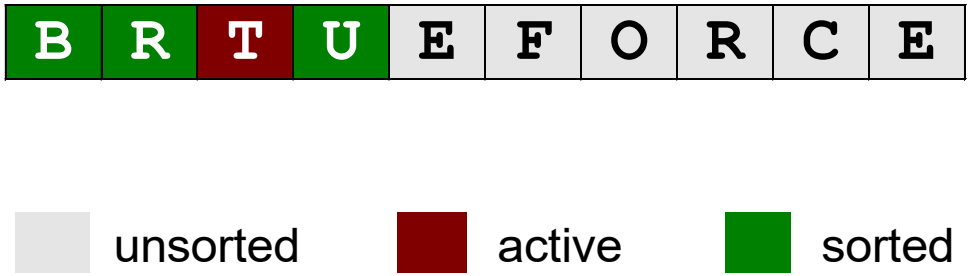
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

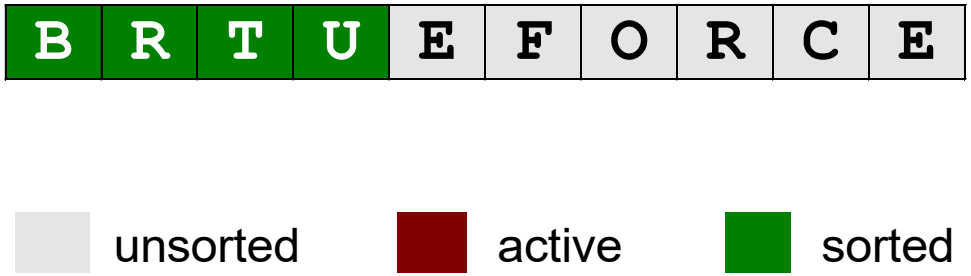
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

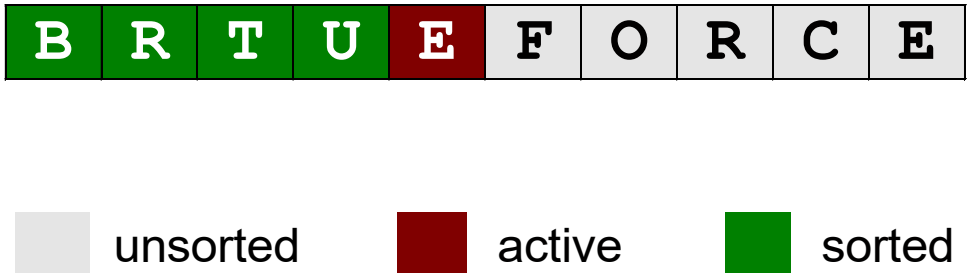
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

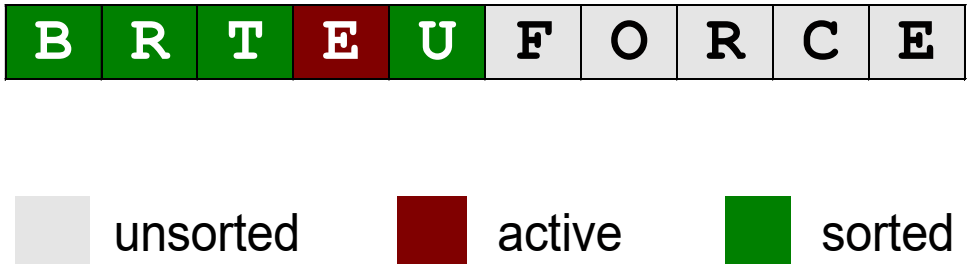
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

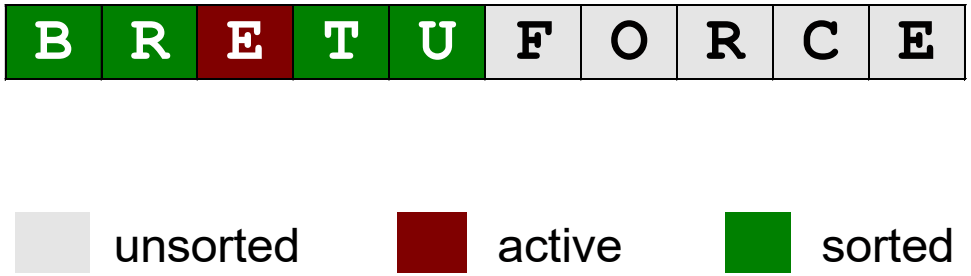
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

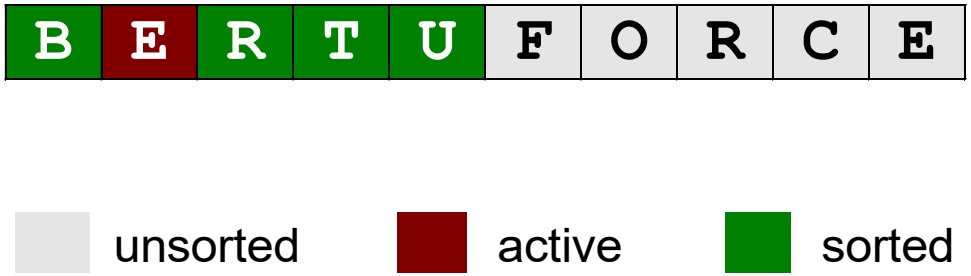
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

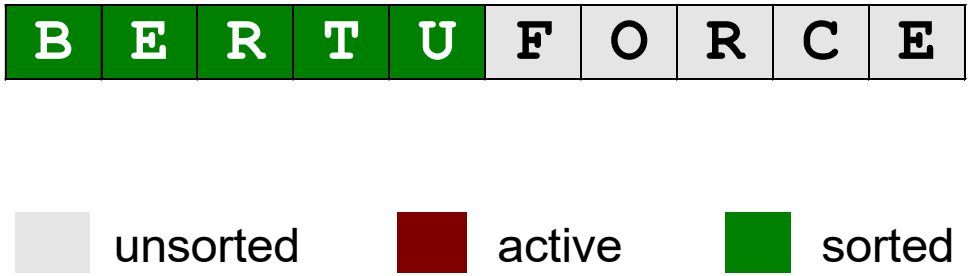
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

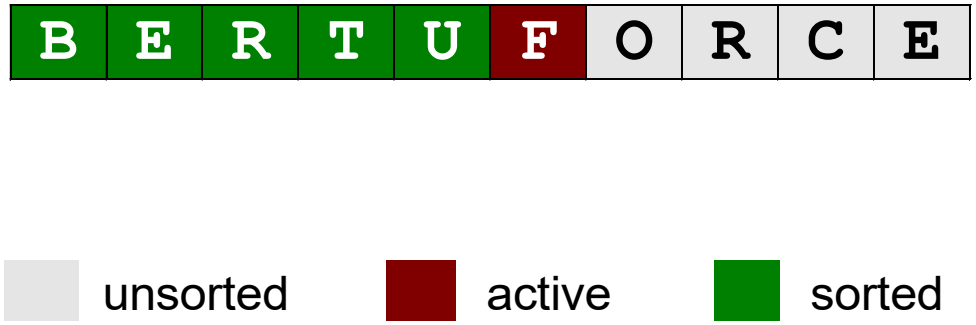
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

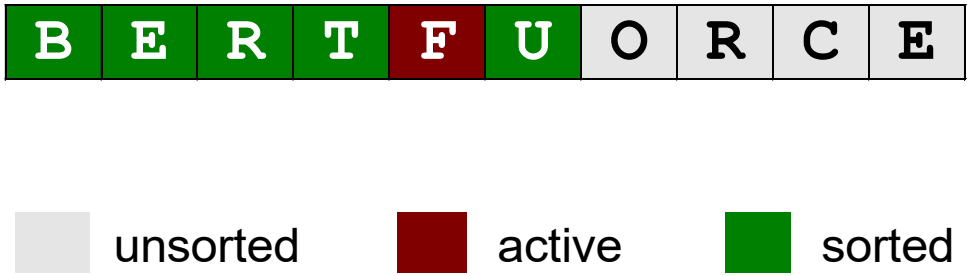
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

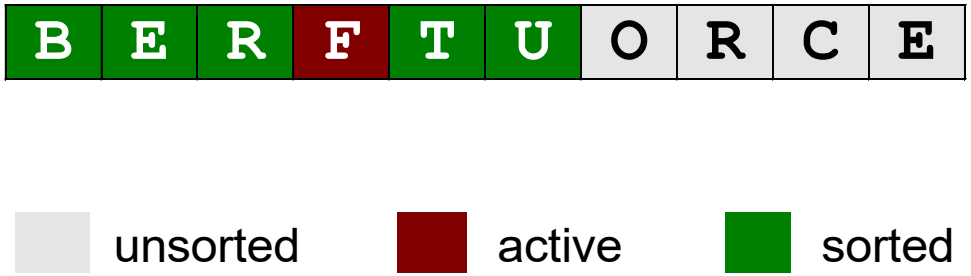
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

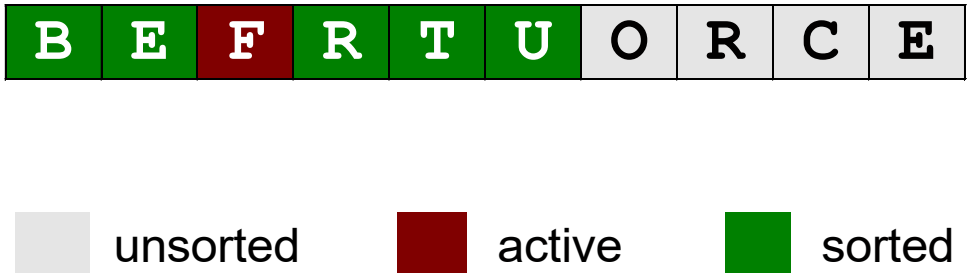
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

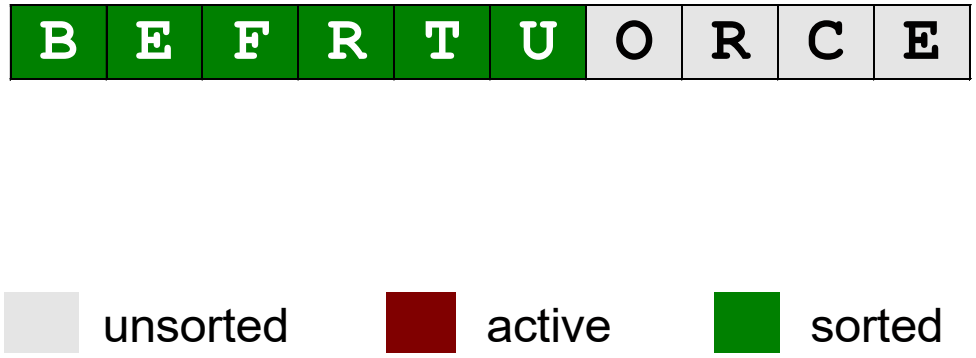
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

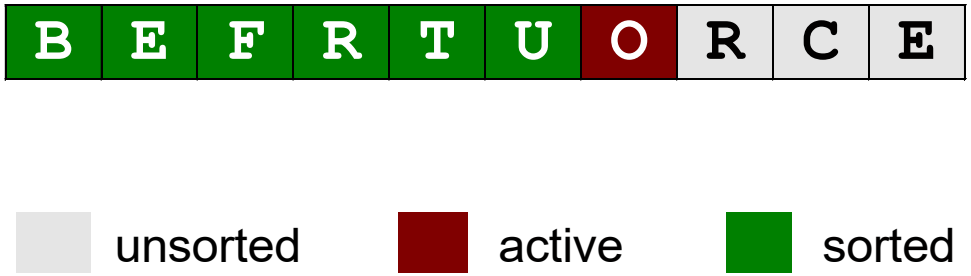
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

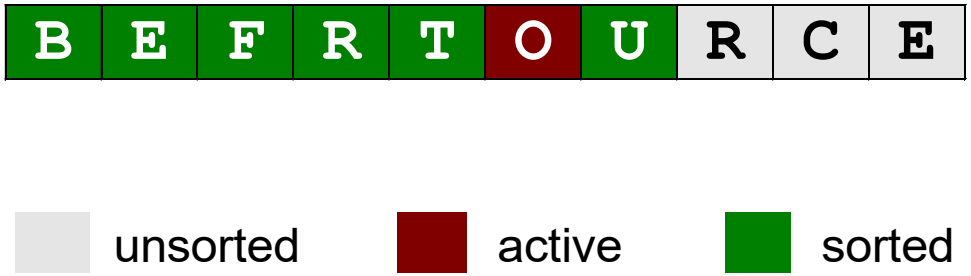
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

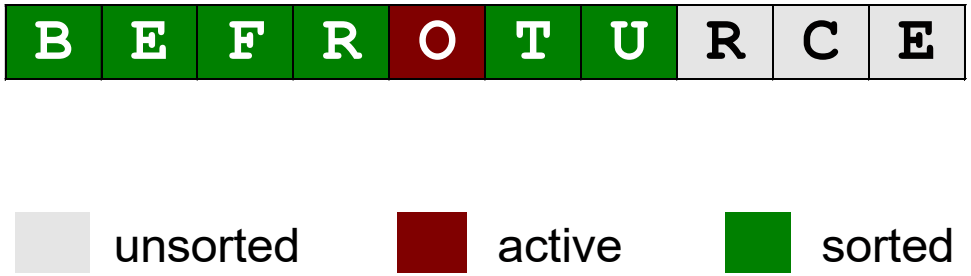
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

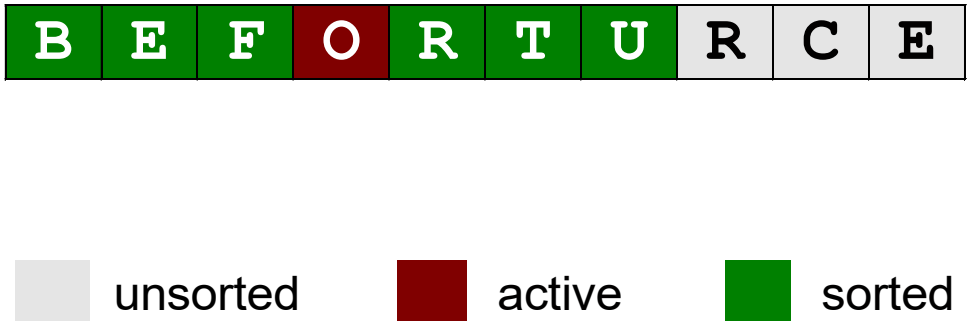
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

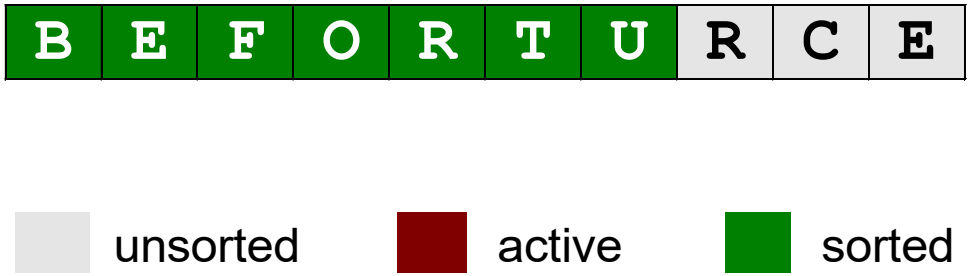
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

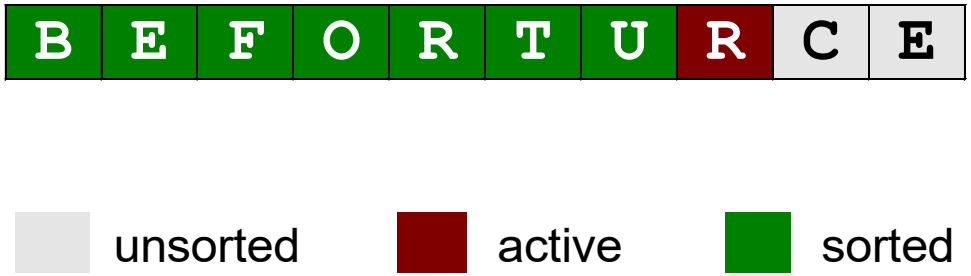
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

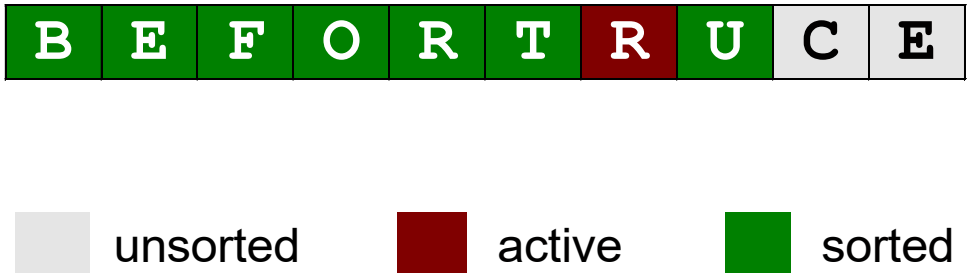
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

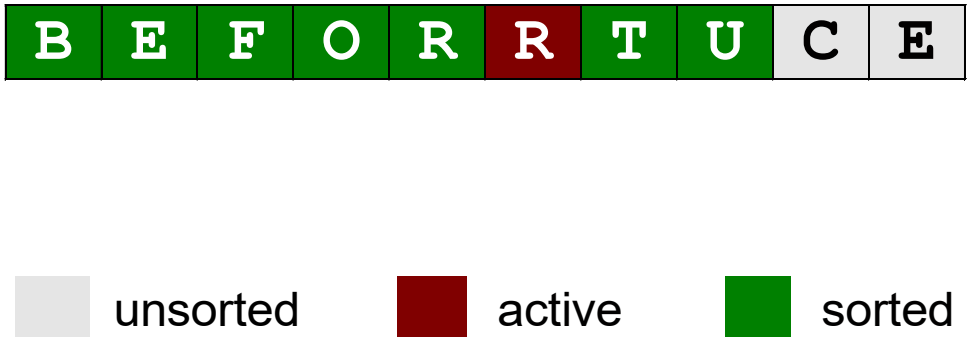
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

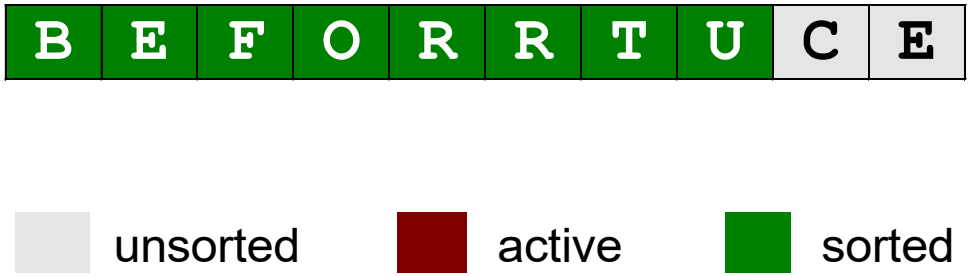
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

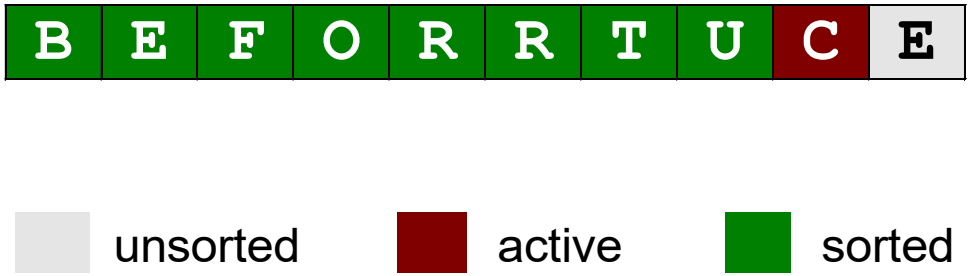
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

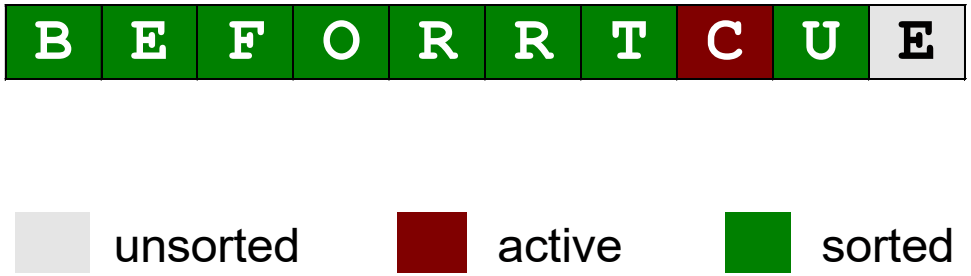
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

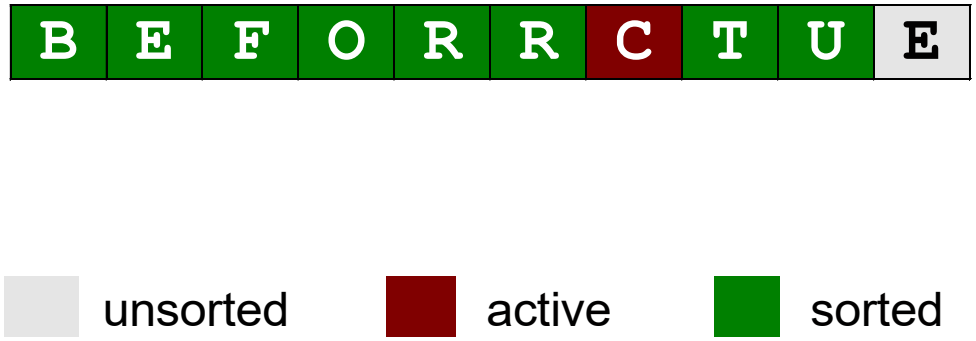
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

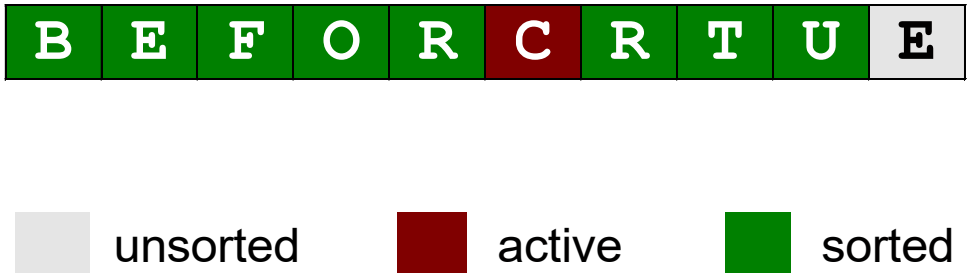
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

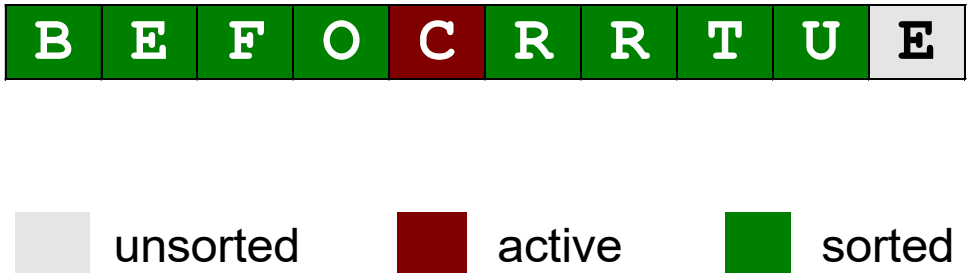
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

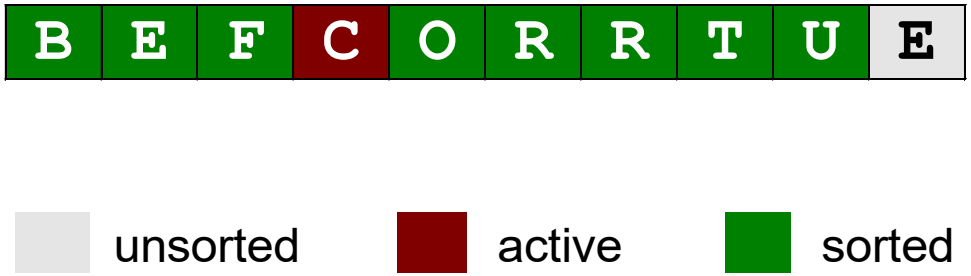
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

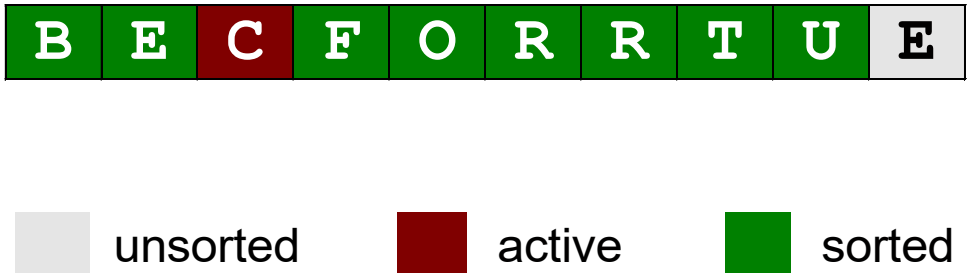
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

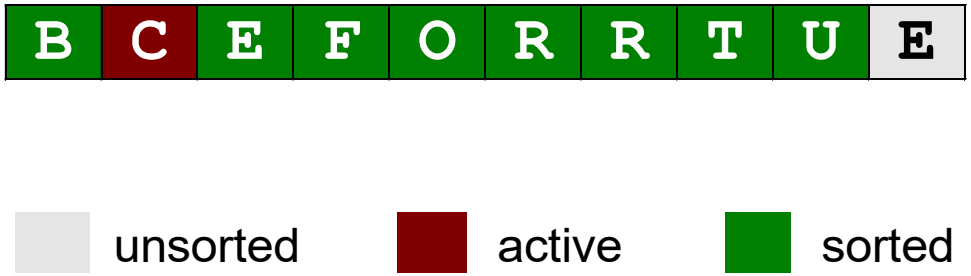
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

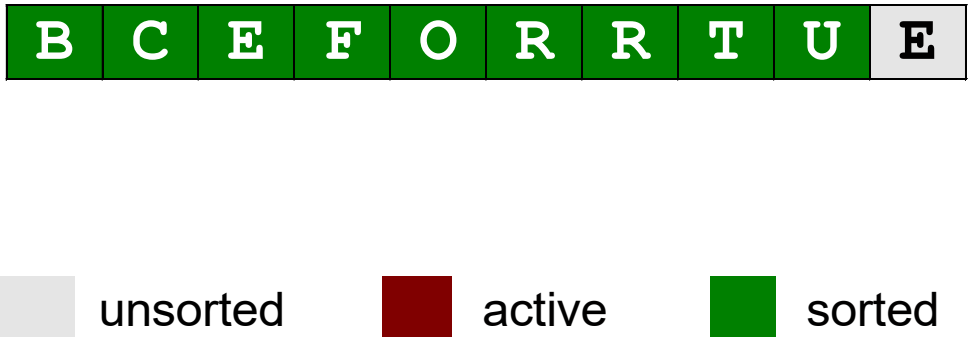
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

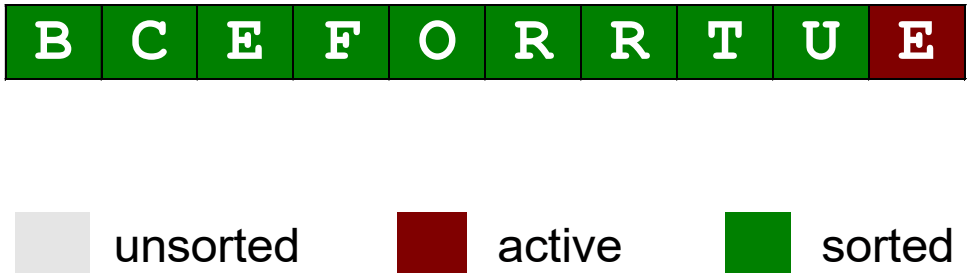
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

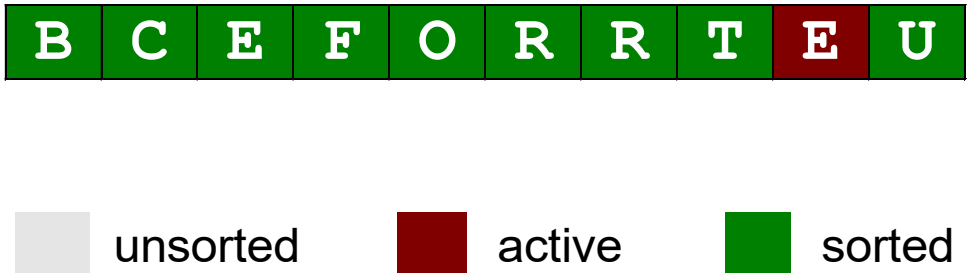
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

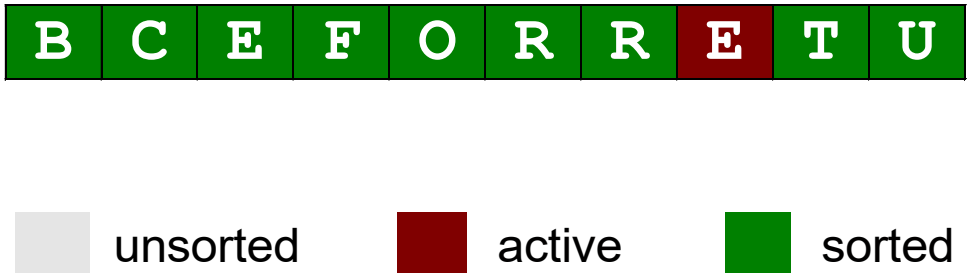
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

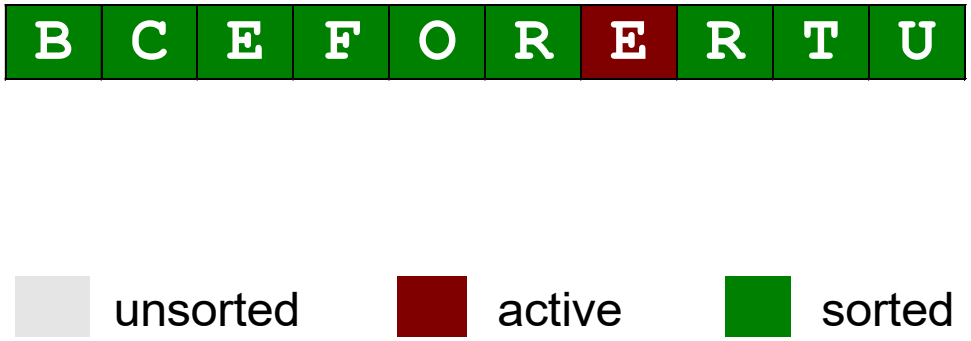
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

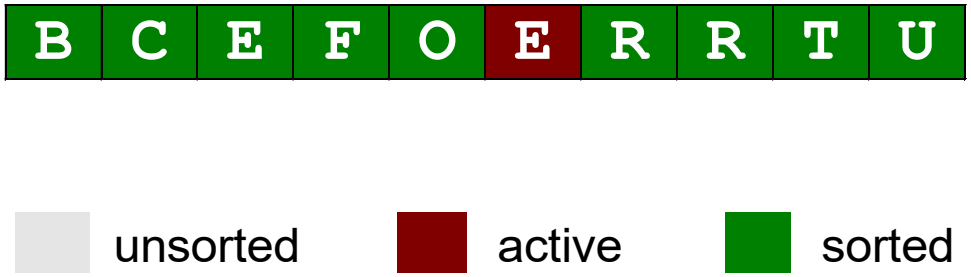
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

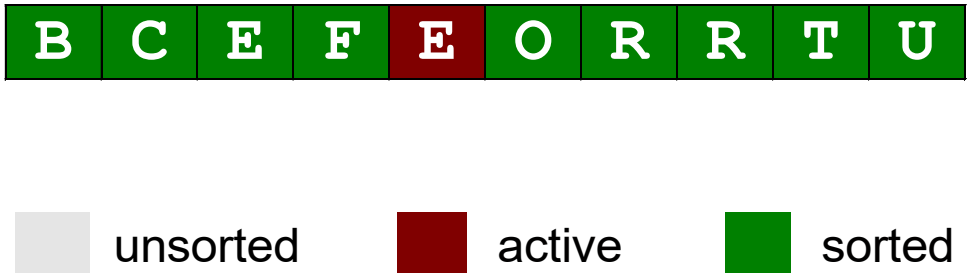
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

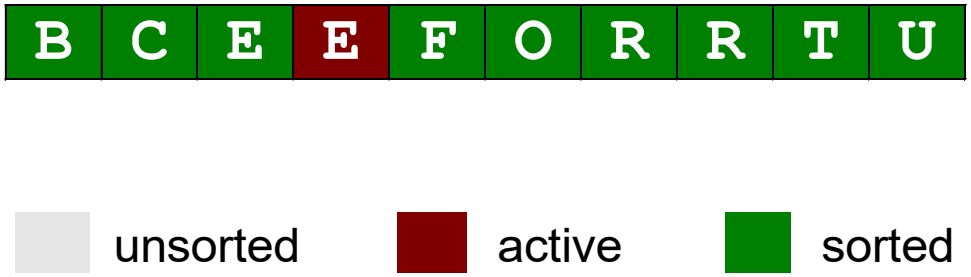
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

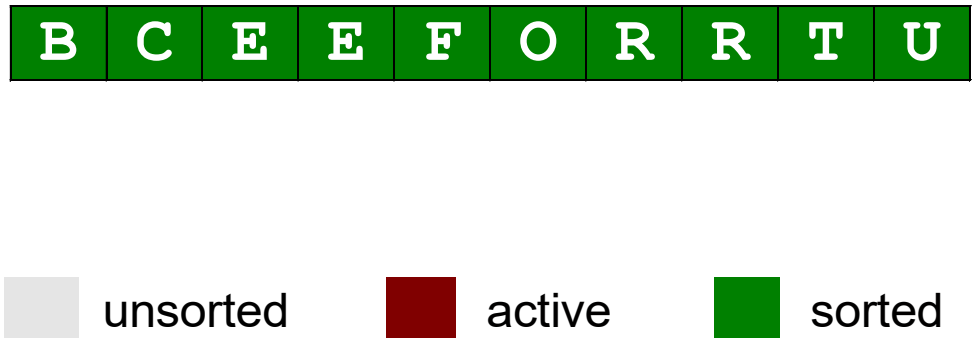
- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Insertion Sort Demo

- Sorting problem:
  - Given an array of N integers, rearrange them so that they are in increasing order.

- Insertion sort
  - Brute-force sorting solution.
  - Move left-to-right through array.
  - Exchange next element with larger elements to its left, one-by-one.



Selection Sort

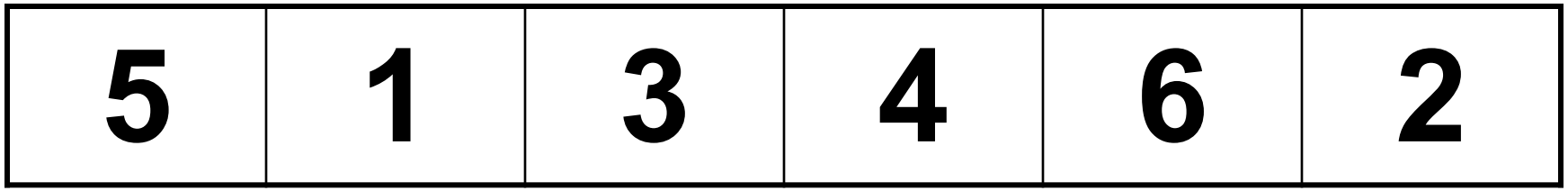
Selection Sort

- Idea:
  - Find the smallest element in the array
  - Exchange it with the element in the first position
  - Find the second smallest element and exchange it with the element in the second position
  - Continue until the array is sorted
- Disadvantage:
  - Running time depends only slightly on the amount of order in the file

Selection Sort

- SELECTION-SORT(A)
  1. for j ← 1 to n-1
  2. smallest ← j
  3. for i ← j + 1 to n
  4. if A[i] < A[smallest]
  5. smallest ← i
  6. Exchange A[j] ↔ A[smallest]

Selection Sort

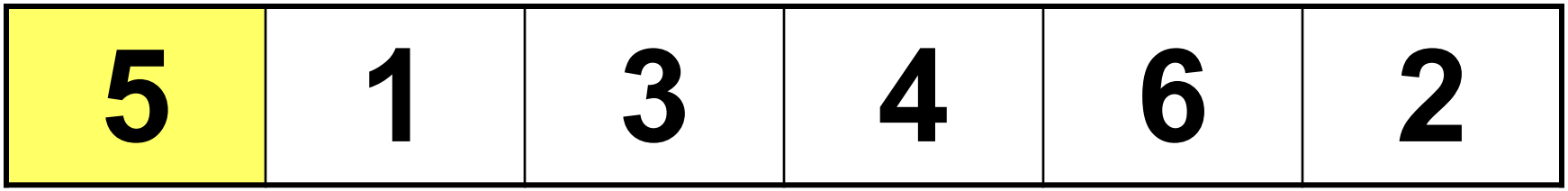


Comparison

Data Movement

Sorted

Selection Sort

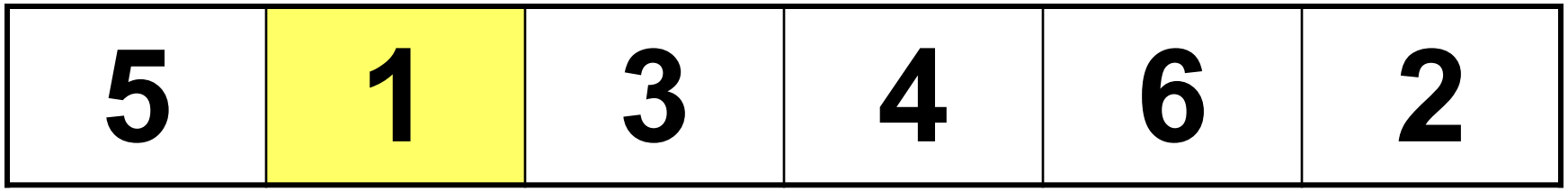


Comparison

Data Movement

Sorted

Selection Sort

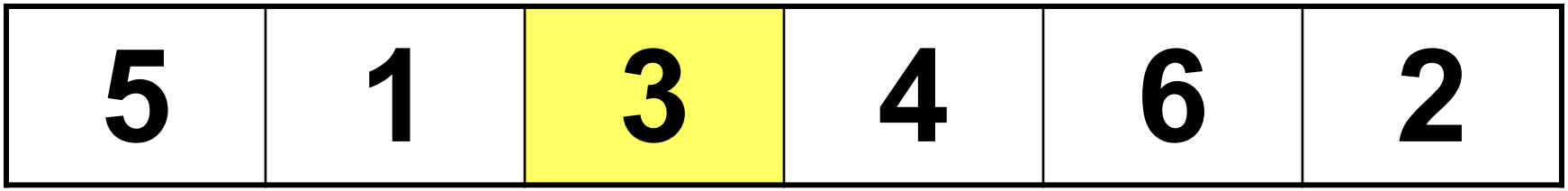


Comparison

Data Movement

Sorted

Selection Sort

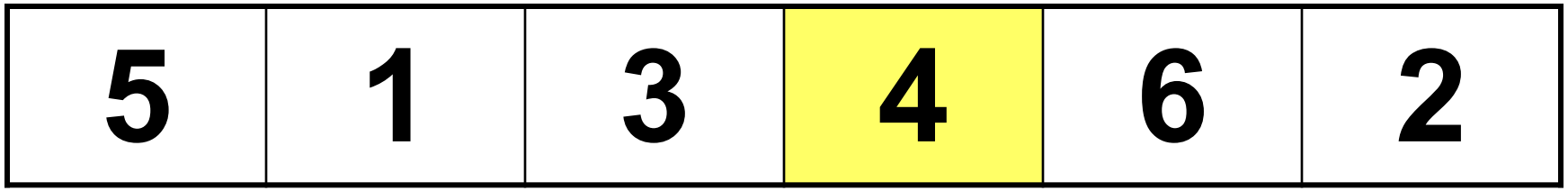


Comparison

Data Movement

Sorted

Selection Sort

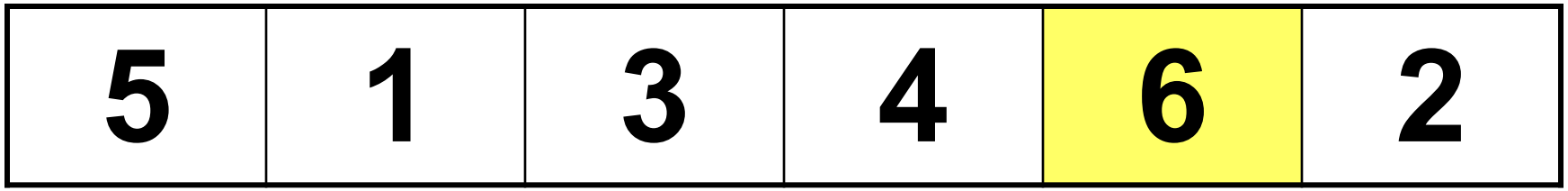


Comparison

Data Movement

Sorted

Selection Sort

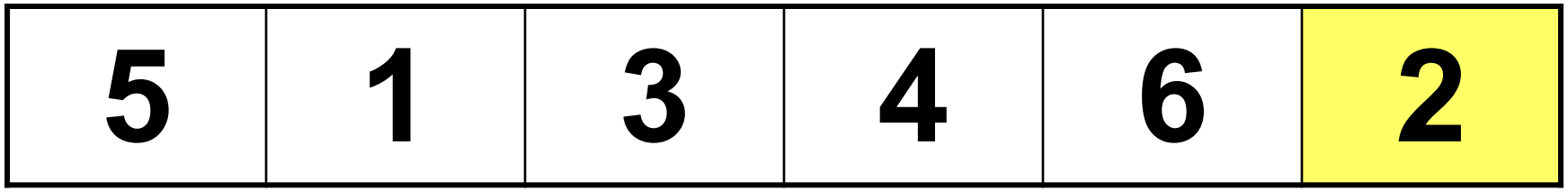


Comparison

Data Movement

Sorted

Selection Sort

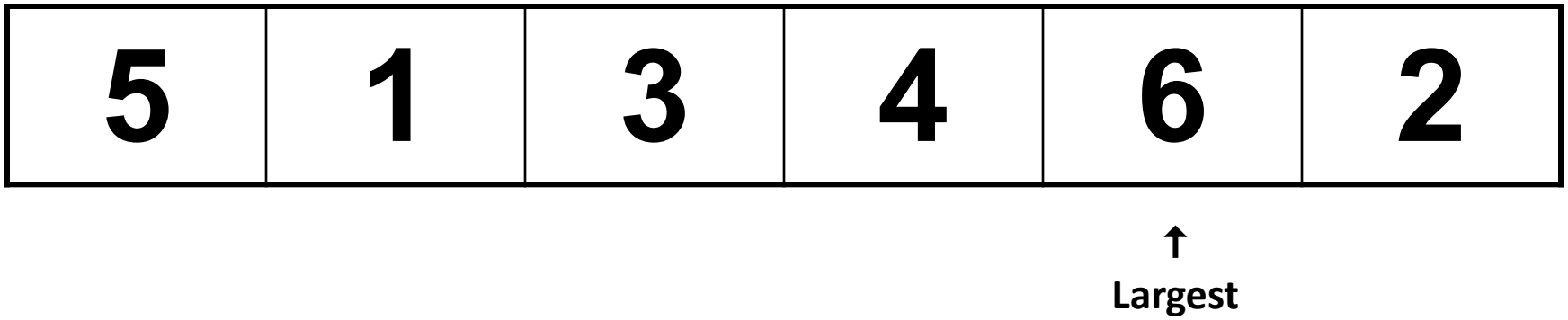


Comparison

Data Movement

Sorted

Selection Sort

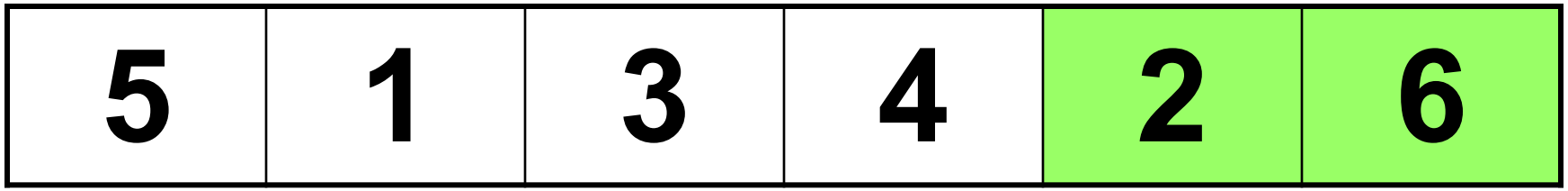


Comparison

Data Movement

Sorted

Selection Sort

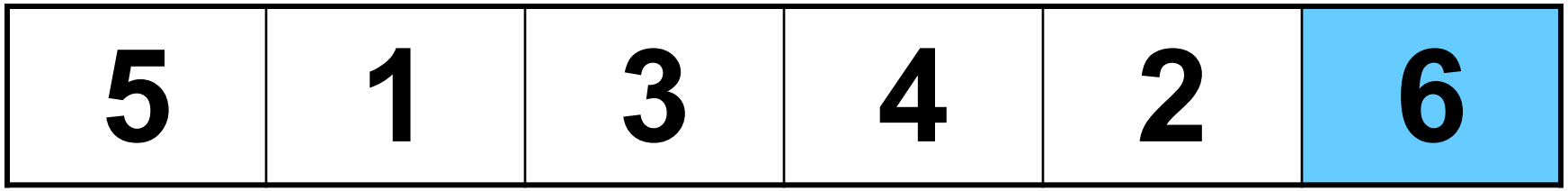


Comparison

Data Movement

Sorted

Selection Sort

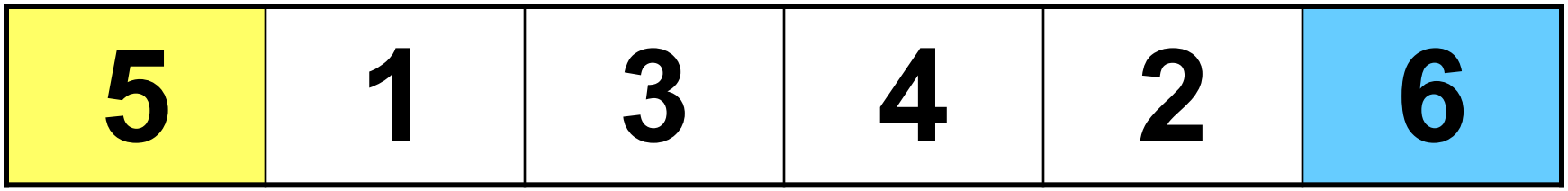


Comparison

Data Movement

Sorted

Selection Sort

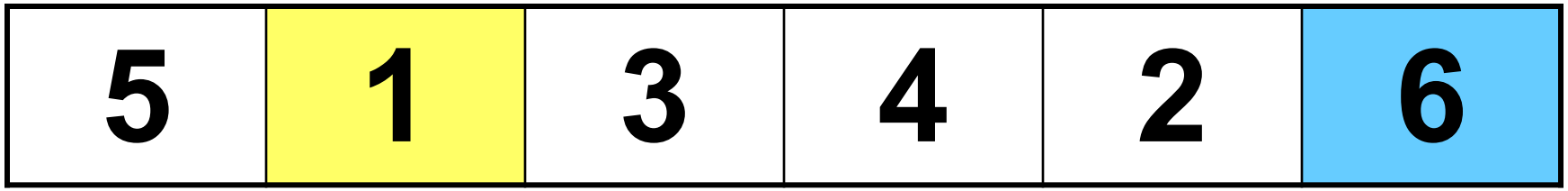


Comparison

Data Movement

Sorted

Selection Sort

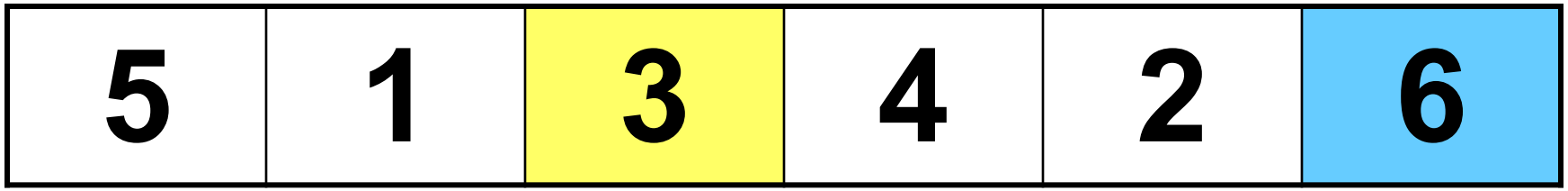


Comparison

Data Movement

Sorted

Selection Sort

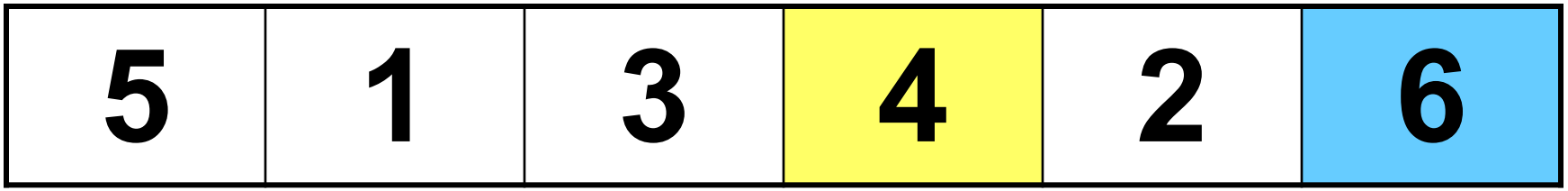


Comparison

Data Movement

Sorted

Selection Sort

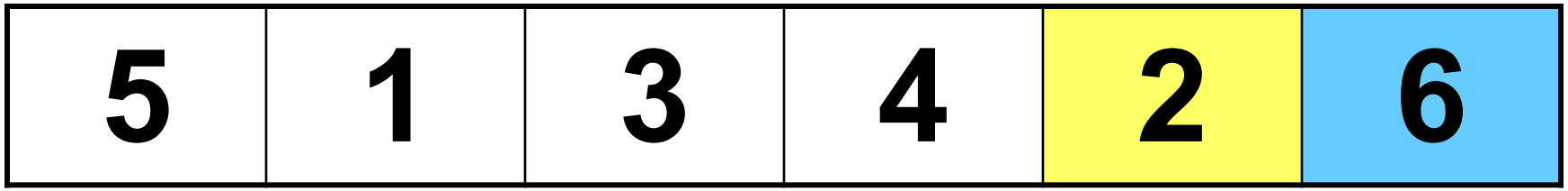


Comparison

Data Movement

Sorted

Selection Sort

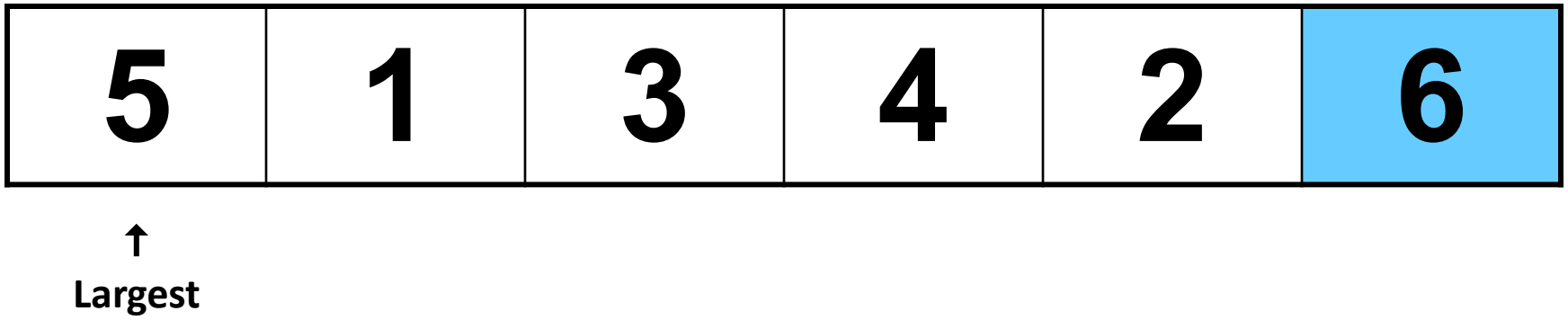


Comparison

Data Movement

Sorted

Selection Sort

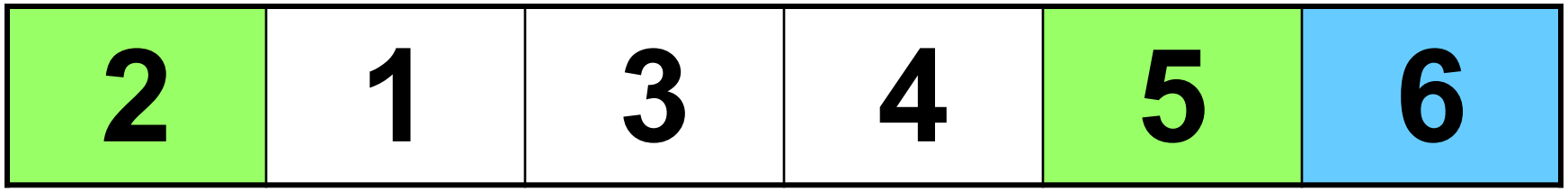


Comparison

Data Movement

Sorted

Selection Sort

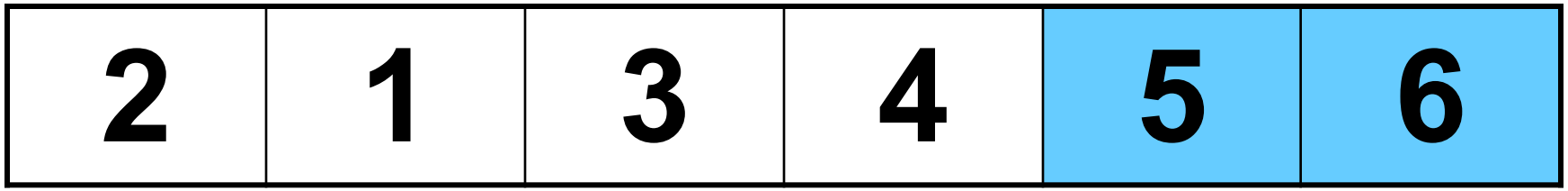


Comparison

Data Movement

Sorted

Selection Sort

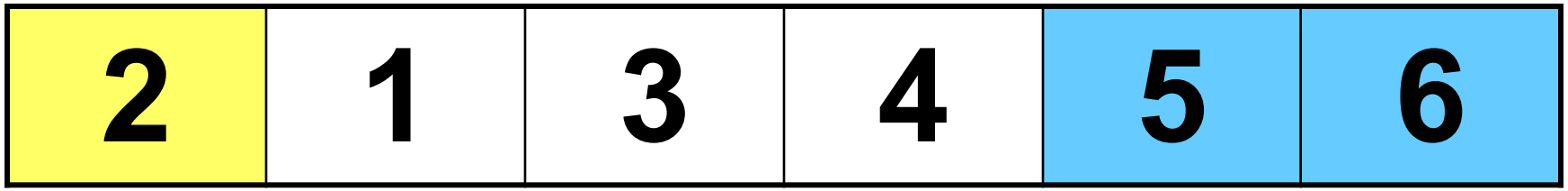


Comparison

Data Movement

Sorted

Selection Sort

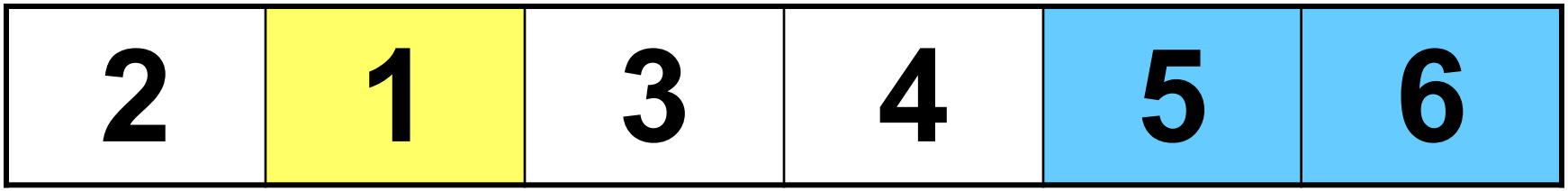


Comparison

Data Movement

Sorted

Selection Sort

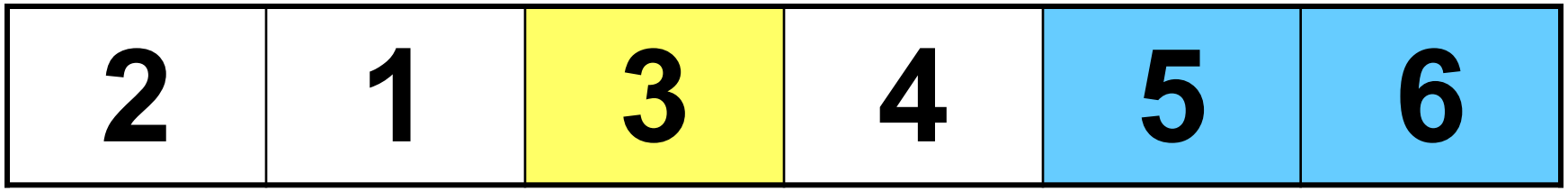


Comparison

Data Movement

Sorted

Selection Sort

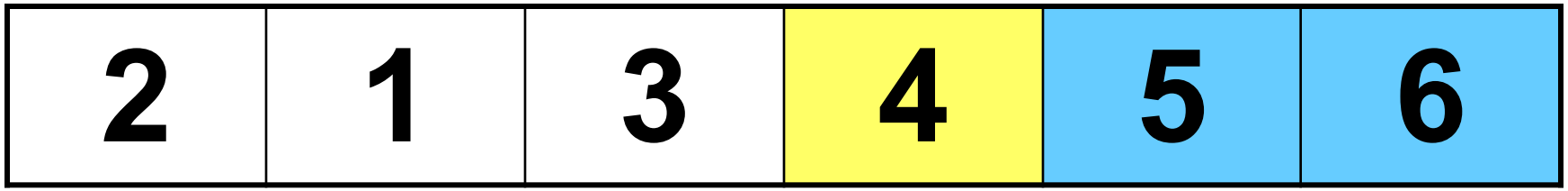


Comparison

Data Movement

Sorted

Selection Sort

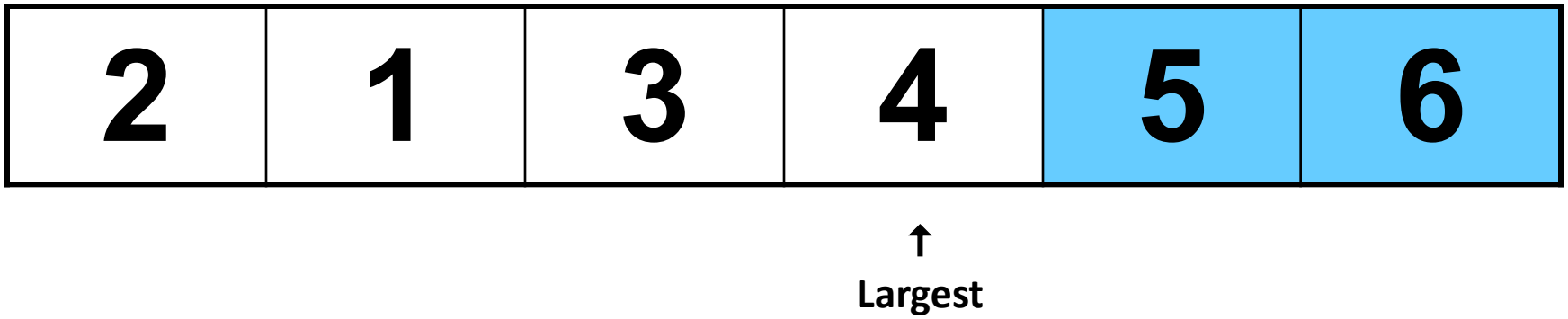


Comparison

Data Movement

Sorted

Selection Sort

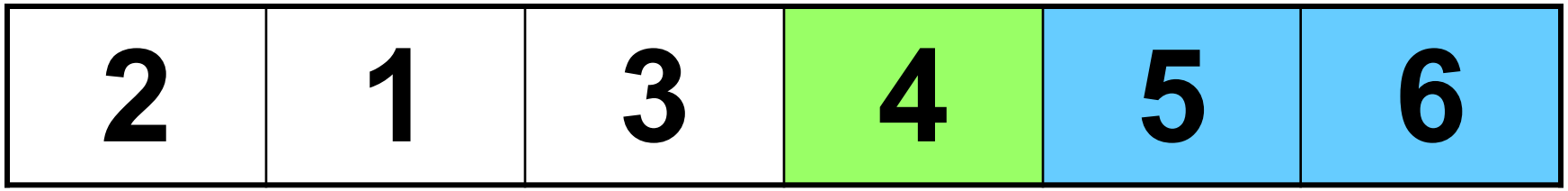


Comparison

Data Movement

Sorted

Selection Sort

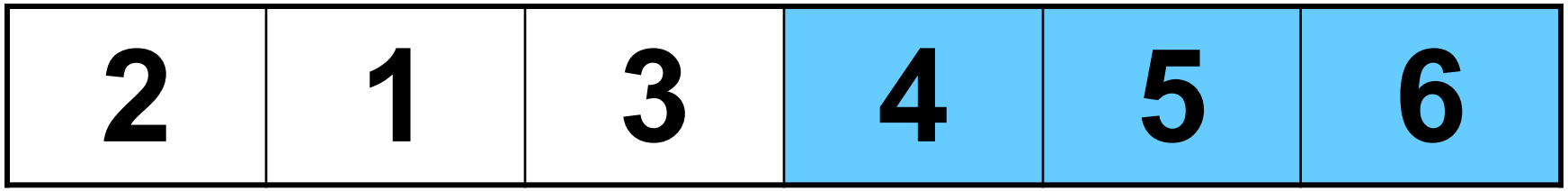


Comparison

Data Movement

Sorted

Selection Sort

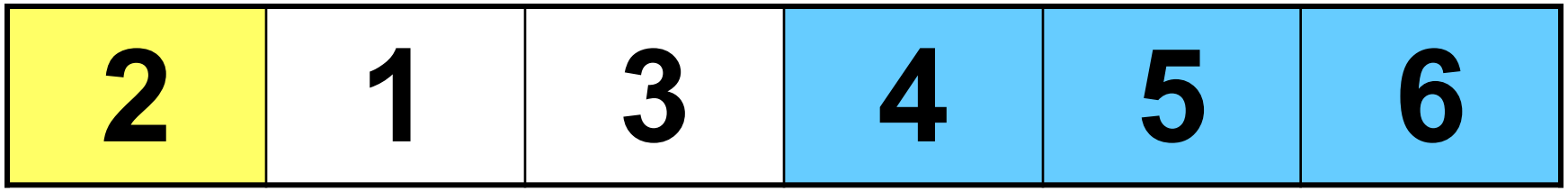


Comparison

Data Movement

Sorted

Selection Sort

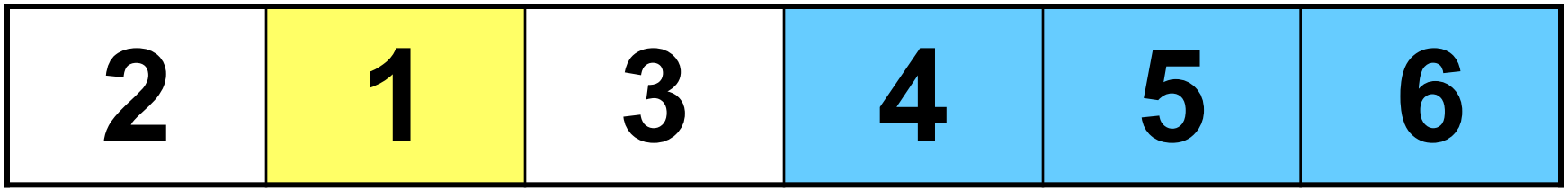


Comparison

Data Movement

Sorted

Selection Sort

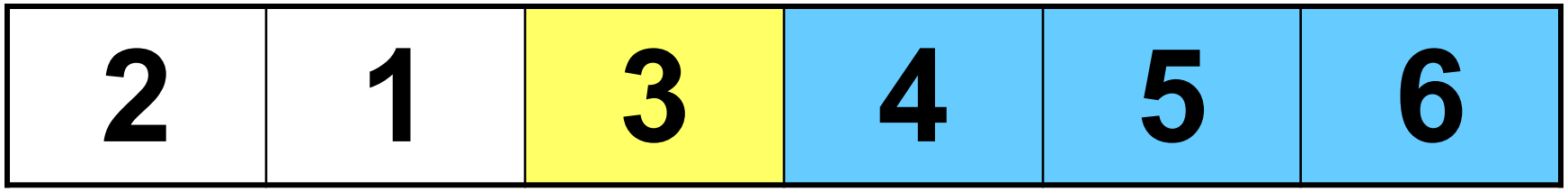


Comparison

Data Movement

Sorted

Selection Sort

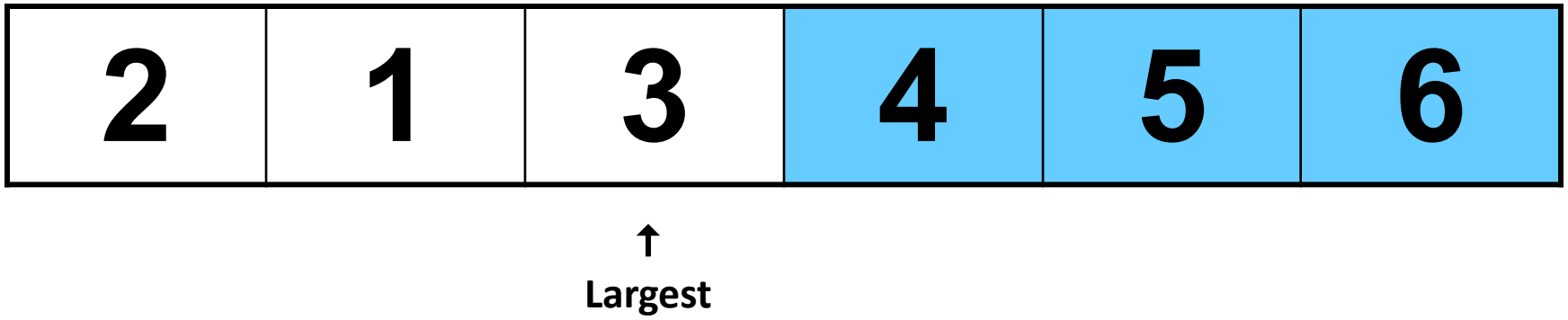


Comparison

Data Movement

Sorted

Selection Sort

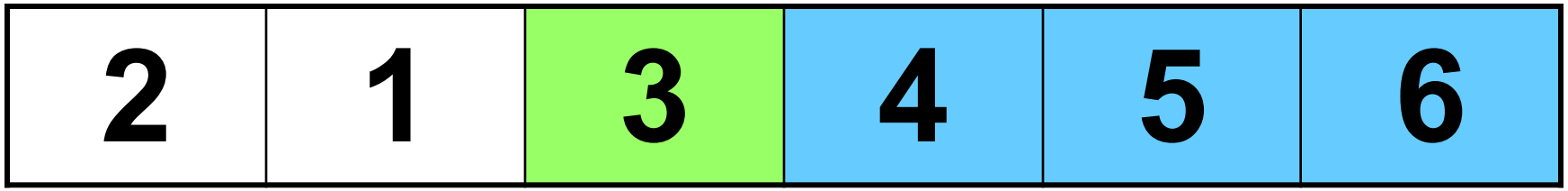


Comparison

Data Movement

Sorted

Selection Sort

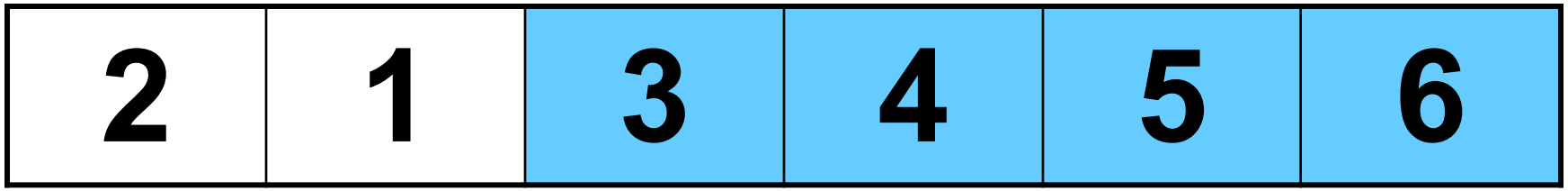


Comparison

Data Movement

Sorted

Selection Sort

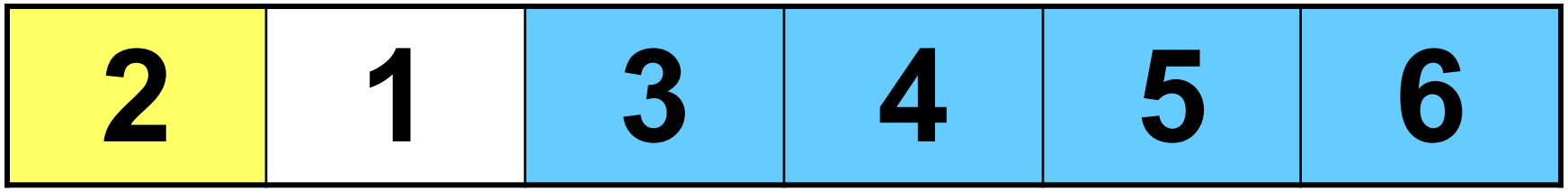


Comparison

Data Movement

Sorted

Selection Sort

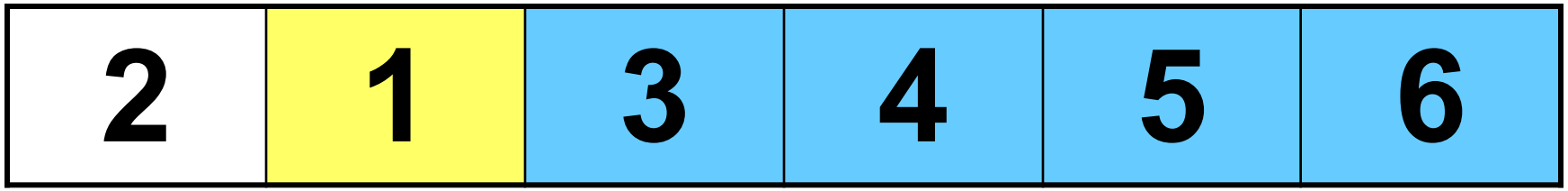


Comparison

Data Movement

Sorted

Selection Sort

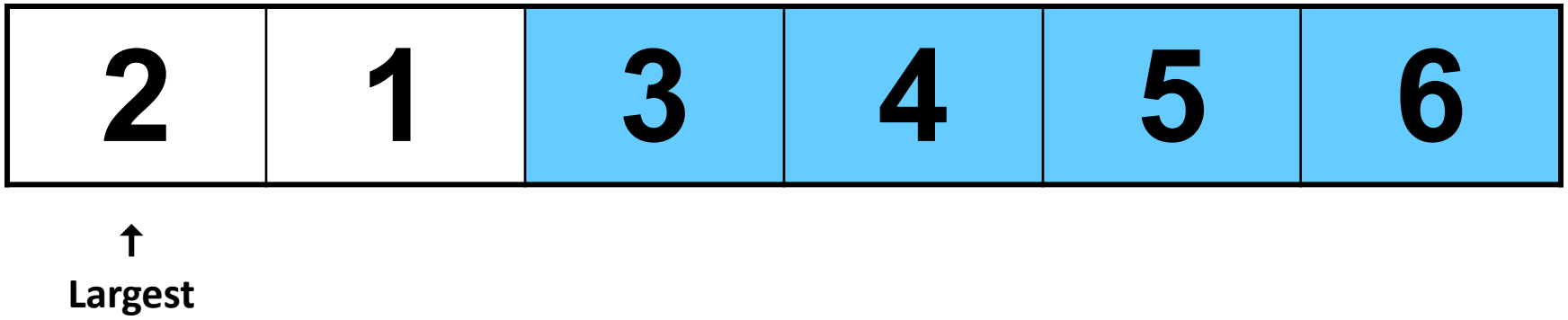


Comparison

Data Movement

Sorted

Selection Sort

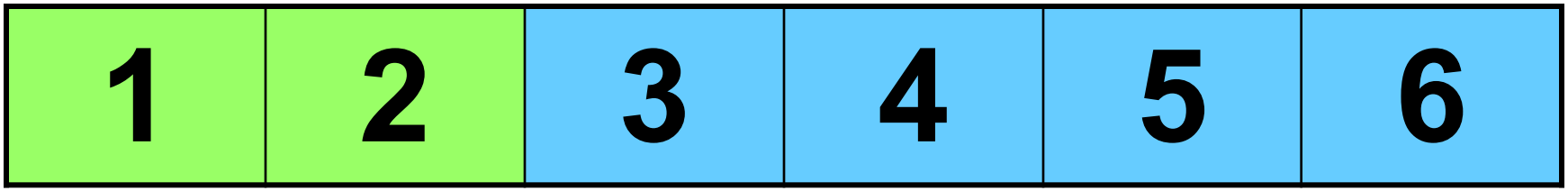


Comparison

Data Movement

Sorted

Selection Sort

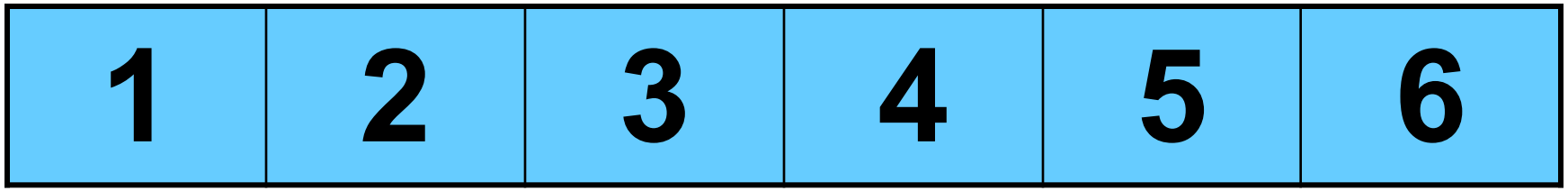


Comparison

Data Movement

Sorted

Selection Sort



DONE!

Comparison

Data Movement

Sorted

Selection Sort

- Count comparisons of *largest so far* against other values
- Find Largest, given *m* values, does *m*-1 comparisons
- Selection sort calls Find Largest *n* times,
  - Each time with a smaller list of values
  - Cost = n-1 + (n-2) + … + 2 + 1 = $n(n-1)/2$

Selection Sort

- Time efficiency
  - Comparisons: $n(n-1)/2$
  - Exchanges: n (swapping largest into place)
  - Overall: $\Theta(n^2)$, best and worst cases
- Space efficiency
  - Space for the input sequence, plus a constant number of local variables

Quiz 1

What are the correct intermediate steps of the following data set when it is being sorted with the Selection sort? 15,20,10,18

A. 10, 20,15,18 -- 10,15,20,18 -- 10,15,18,20

B. 15,20,10,18 -- 15,10,20,18 -- 10,15,20,18 -- 10,15,18,20

C. 15,18,10,20 -- 10,18,15,20 -- 10,15,18,20 -- 10,15,18,20

D. 15,10,20,18 -- 15,10,18,20 -- 10,15,18,20

Quiz 2

- The number of passes through an array of 20 elements using a selection sort is:
  - A 0
  - B 19
  - C 20
  - D 21
  - E depends on the order of the items in the array

Quiz 3

- The number of comparisons used with an array of 7 elements using a selection sort is:
  - A 6
  - B 7
  - C 8
  - D 21
  - E depends on the order of the items in the array

Quiz 4

- Suppose we are sorting an array of eight integers using some quadratic sorting algorithm. After four iterations of the algorithm's main loop, the array elements are ordered as shown here: 2 4 5 7 8 1 3 6. Which statement is correct? (Note: Our selection sort picks largest items rst.)

- A. The algorithm might be either selection sort or insertion sort.
- B. The algorithm might be selection sort, but it is not insertion sort.
- C. The algorithm is not selection sort, but it might be insertion sort.
- D. The algorithm is neither selection sort nor insertion sort.