Brute Force and Exhaustive Search

Prof. Sasmita Kumari Nayak, CSE

Introduction to Brute Force

- Brute force is a straightforward approach to solve a problem.
- It is directly based on the problem statement and definitions of the concepts involved.
- some features of the brute force algorithm are:
  - It is an intuitive, direct, and straightforward technique
  - Many problems solved in day-to-day life using the brute force strategy, for example exploring all the paths to a nearby market to find the minimum shortest path, optimal algorithms etc.
  - Arranging the books in a rack using all the possibilities to optimize the rack spaces, etc.

Advantages of Brute Force

- The brute force approach is a guaranteed way to find the correct solution by listing all the possible candidate solutions for the problem.
- It is a generic method and not limited to any specific domain of problems.
- The brute force method is ideal for solving small and simpler problems.
- It is known for its simplicity and can serve as a comparison benchmark.

Disadvantages of Brute Force

- The brute force approach is inefficient. For real-time problems, algorithm analysis often goes above the O(N!) order of growth.
- This method relies more on compromising the power of a computer system for solving a problem than on a good algorithm design.
- Brute force algorithms are slow.
- Brute force algorithms are not constructive or creative compared to algorithms that are constructed using some other design paradigms.

Bubble Sort Algorithm

Prof. Sasmita Kumari Nayak, CSE

Introduction to sorting

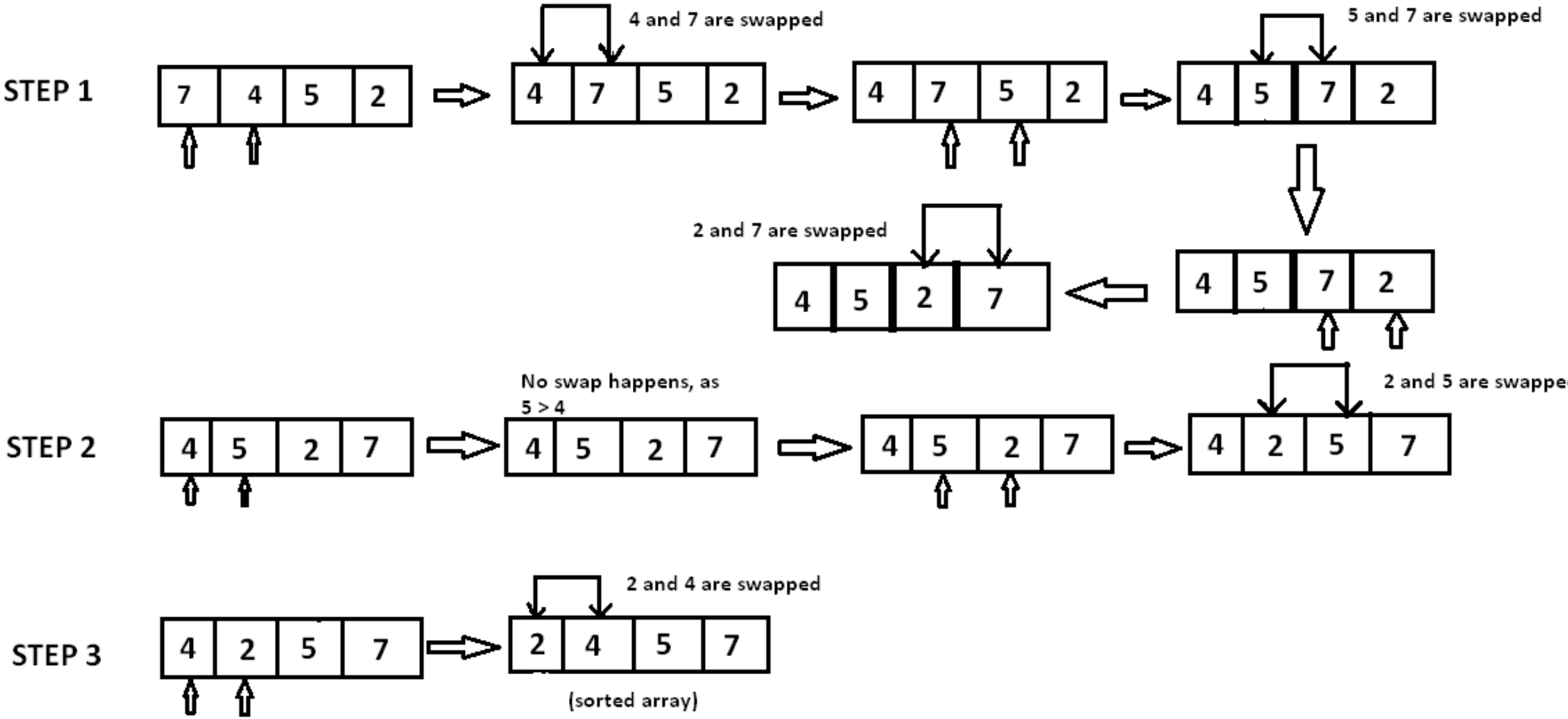
- There are two types of sorting: Sequential Sort and Recursive Sort.
- In general, Sequential Sort is simple but slower, Recursive Sort is much faster but requires more algorithmic understanding.

Sequential Sort

- 1 Selection Sort - continuously finding the smallest or the greatest element in each iterated subset and swapping with the beginning element of the subset
- 2 Insertion Sort - continuously inserting the element into an iterated subset in a given order
- 3 Bubble Sort - continuously comparing the nearing 2 elements and swapping the element in a given order

Introduction to Bubble Sort

- One of the simplest sorting algorithms proceeds by walking down the list, comparing adjacent elements, and swapping them if they are in the wrong order.
- The process is continued until the list is sorted.
- **Algorithm:**
  - Given n numbers to sort:
  - Repeat the following n-1 times:
    - For each pair of adjacent numbers:
    - If the number on the left is greater than the number on the right, swap them


STEP 1
7 4 5 2
4 and 7 are swapped
4 7 5 2
4 7 5 2
5 and 7 are swapped
4 5 7 2
2 and 7 are swapped
4 5 2 7
4 5 7 2
STEP 2
4 5 2 7
No swap happens, as
5 > 4
4 5 2 7
4 5 2 7
2 and 5 are swapped
4 2 5 7
STEP 3
4 2 5 7
2 and 4 are swapped
2 4 5 7
(sorted array)

Algorithm:

```
begin BubbleSort(list)

   for all elements of list
      if list[i] > list[i+1]
         swap(list[i], list[i+1])
      end if
   end for

   return list

end BubbleSort
```

Analysis of Bubble sort Algorithms

- Worst and Average Case Time Complexity: O(n*n). Worst case occurs when array is reverse sorted.
- Best Case Time Complexity: O(n). Best case occurs when array is already sorted.