CSCI 115 Lab

Week 4- Selection and Merge Sort

- Professor: Dr. Matin Pirouz
  Email: mpirouz@csufresno.edu

- TA: Shreeja Miyyar
  Email:shreejarao12@mail.fresnostate.edu

Table of Contents

Selection Sort

- It is a sorting algorithm where the list is divided into two parts, unsorted array on the right side and sorted array on the left side.
- The smallest element is selected from the unsorted array and moved to the sorted subarray.

Example:

Array = [4, 7, 3, 1, 20, 5]

// First iteration - find the minimum element from array[0] to array[n-1] and swap it with the first element
[**1**, 7, 3, 4, 20, 5]

// Second iteration - find the minimum element from array[1] to array[n-1] and swap it with the second element
[1, **3**, 7, 4, 20, 5]

…. and so on

Selection Sort algorithm

- **Step 1** - Set MIN to Index 0 of the list
- **Step 2** - Find the minimum element from Index MIN+1 to N-1 where N is the length of the list.
- **Step 3** - Swap the minimum element with the value at MIN index.
- **Step 4** - Increment the MIN by 1.
- **Step 5** - Repeat Step 2 - 4 until MIN reaches the end of the list.

Time Complexities:

Worst Case Complexity: O(n2)

If we want to sort in ascending order and the array is in descending order then, the worst case occurs.

Best Case Complexity: O(n2)

It occurs when the array is already sorted

Average Case Complexity: O(n2)

It occurs when the elements of the array are in jumbled order (neither ascending nor descending).

- Comparison:

Cycle	Number of Comparison
1st	(n-1)
2nd	(n-2)
3rd	(n-3)
...	...
last	1

Total number of comparison=(n - 1) + (n - 2) + (n - 3) + ... + 1 = n(n - 1) / 2 which is approximately n2/2.

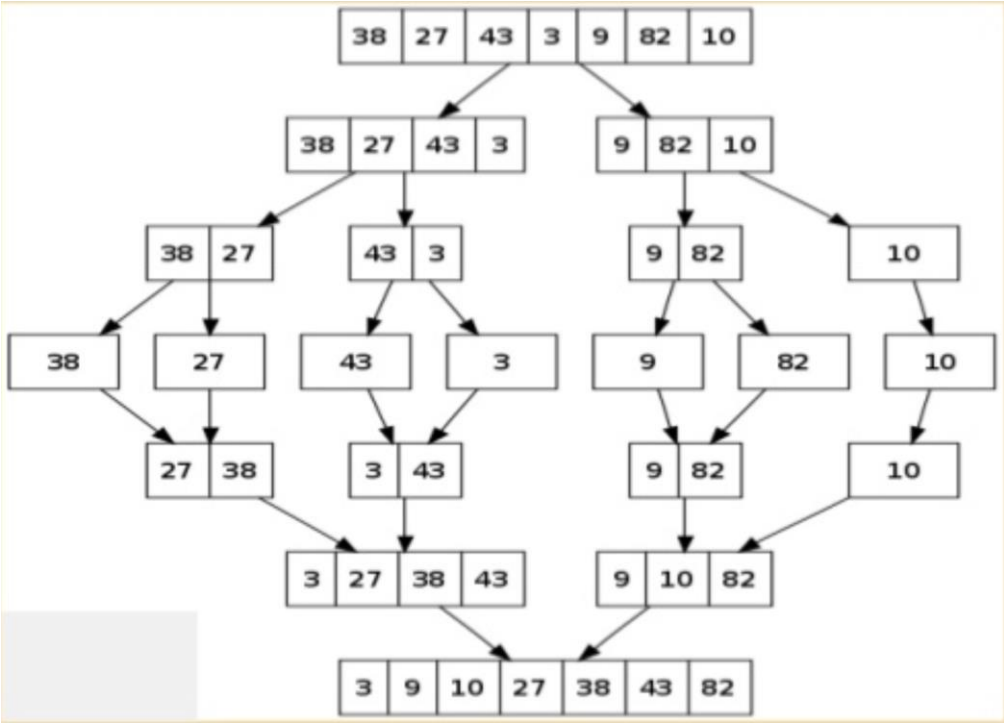
- Number of exchanges is approximately equal to n.

Merge Sort

- It uses the divide and conquer approach to sort an array.

- Using recursion, it divides the input into two halves, sorts them separately and merges them to form a sorted array.

- Time Complexity:
  - Worst case: O(nlgn)
  - Best case: O(nlgn)
  - Average case: O(nlgn)



Merge Sort algorithm

MergeSort (array, left, right)

If (left < right)

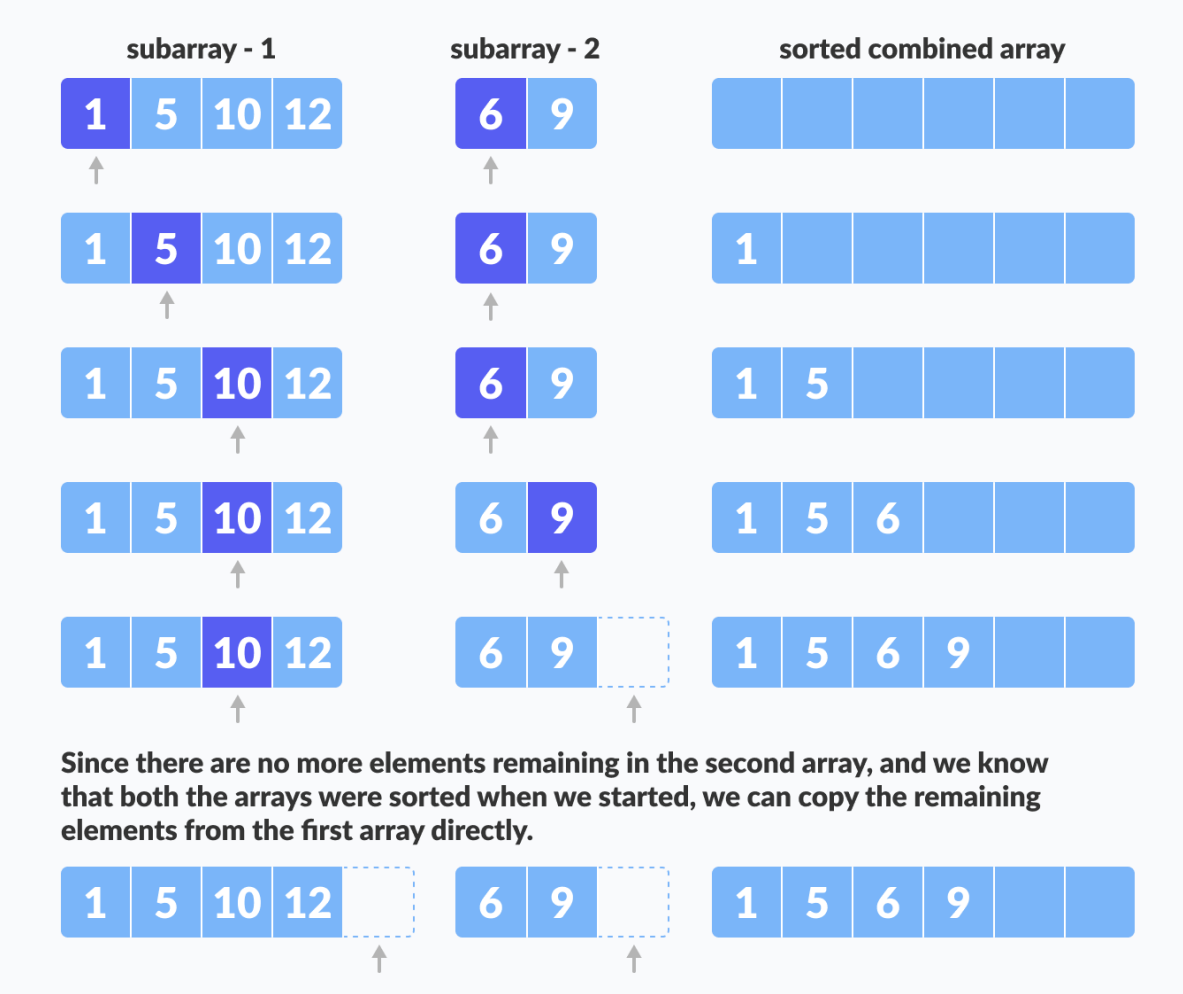
- **Step 1** - Find the middle point of the array and divide it into two halves.
  middle = (left + right) / 2
- **Step 2** - Recursively call MergeSort(array, left, middle)
- **Step 3** - Recursively call MergeSort(array, middle+1, right).
- **Step 4** - Call the function Merge(array, left, middle, right)

Merge (array, left, middle, right)

This method is used to merge two array into one sorted array (result array).

- **Step 1** - Iterate both the arrays.
- **Step 2** - Have we reached the end of any of the both array?
  - If no, then
    - Compare the elements of both arrays.
    - Copy the smaller element into the result array.
    - Move to the next element of the array containing the smaller element.
  - If yes, then
    - Copy all the remaining elements of the non-empty array to the result array.

Merge function - Visualization



Lab Assignment

1. Write a program that takes a list and sorts it using Selection Sort.

   Hint: Use the selection sort algorithm mentioned in the slides to write the program.

2. Write a program that takes a list and sorts it using Merge sort with recursion.

   Hint: Use the merge sort algorithm mentioned in the slides to write the program. You can define a separate method for merging operation.

3. Compare the execution time for the following three conditions:
   - The list is sorted
   - The list is half sorted
   - The list is reverses

   Hint: You can use clock() function to record execution time.

   E.g. for sorted list is [1, 2, 4, 7, 9], half sorted list is [1, 4, 5, 20, 7, 6] and reversed list is [6, 4, 3, 2, 1]. Use a large list so that the the execution time for the three conditions are noticeable.

4. Analyze and discuss the time complexity for cases 3.a-3.c.
5. Analyze and discuss the time complexity of the two methods.
6. Fill out the report sheet.

   Hint: Write a detailed report as per the template and provide an elaborate explanation on how the execution time is different for each algorithm.

Coding guidelines

- Selection Sort:
  - In the main function provide the input array to be sorted.
  - Create a function which takes the input array as an argument and performs the necessary operations to sort the array using selection sort.
  - Use the clock method in the main function to find the execution time for different input arrays.

- Merge Sort:
  - In the main function provide the input array to be sorted.
  - Create a function mergeSort() which takes 3 arguments - input array, left index and right index.
  - Create a function merge() which takes 4 arguments – input array, left index, middle index and right index.
  - Divide the input array using recursion in the mergeSort() function. After dividing the array, call the merge() function to merge the two sorted arrays.

Questions?