

Session: 2021 – 2025

**Submitted by:**

Muhammad Hamad Hassan

2021-CS-33

**Supervised by:**

Mr. Samyan Qayyam Wahla

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

* **Contents**

|  |  |
| --- | --- |
| Topic | PAGE NUMBER |
| Insertion Sort | 2-5 |
| Merge Sort | 6-9 |
| Selection Sort | 9 |
| Bubble Sort | 10-13 |

**Instructions:**

Green Color: Sorted

White Color: Unsorted

Gold Color: Key element

**Insertion Sort**

1. Dry run the Insertion Sort on the following inputs. Use the code given in CLRS Section 2.1. Show the detailed iteration. A= {5,43,76,2,98,23,12,32} B= {6,7,8,9,10}

**Code:**

def InsertionSort(array,start,end):

    for i in range(start,end):

        key=array[i]

        j=i-1

        while(key<array[j] and j>=start):

            array[j+1]=array[j]

            j=j-1

        array[j+1]=key

    return array

**Case1:**

A= {5,43,76,2,98,23,12,32}

InsertionSort(A,1,len(A))

**Iteration 1:**

i=1, j=0, key=43, array[j]=5, while condition: False



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 43 | 76 | 2 | 98 | 23 | 12 | 32 |

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 43 | 76 | 2 | 98 | 23 | 12 | 32 |

**Iteration 2:**

i=2, j=1, key=76, array[j]=43, while condition: False



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 43 | 76 | 2 | 98 | 23 | 12 | 32 |

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 43 | 76 | 2 | 98 | 23 | 12 | 32 |

**Iteration 3:**

i=3, key=2, j=2,array[j]=76 while condition: True :

array[j+1]=76, ,j=1



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 43 | 76 | 2 | 98 | 23 | 12 | 32 |

array[j+1]=43, ,j=0



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 43 | 2 | 76 | 98 | 23 | 12 | 32 |



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 2 | 43 | 76 | 98 | 23 | 12 | 32 |

array[j+1]=5, ,j=-1



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 76 | 98 | 23 | 12 | 32 |

array[j+1]=2

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 76 | 98 | 23 | 12 | 32 |

**Iteration 4:**

i=4, key=98, j=3,array[j]=76 while condition: False :



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 76 | 98 | 23 | 12 | 32 |

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 76 | 98 | 23 | 12 | 32 |

**Iteration 5:**

i=5, key=23, j=4,array[j]=98 while condition: True :

array[j+1]=98, ,j=3



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 76 | 98 | 23 | 12 | 32 |

array[j+1]=76, ,j=2



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 76 | 23 | 98 | 12 | 32 |

array[j+1]=43, ,j=1



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 43 | 23 | 76 | 98 | 12 | 32 |

array[j+1]=23



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 23 | 43 | 76 | 98 | 12 | 32 |

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 23 | 43 | 76 | 98 | 12 | 32 |

**Iteration 6:**

i=6, key=12, j=5,array[j]=98 while condition: True :

array[j+1]=98, ,j=4



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 23 | 43 | 76 | 98 | 12 | 32 |

array[j+1]=76, ,j=3



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 23 | 43 | 76 | 12 | 98 | 32 |

array[j+1]=43, ,j=2



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 23 | 43 | 12 | 76 | 98 | 32 |

array[j+1]=23, ,j=1



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 23 | 12 | 43 | 76 | 98 | 32 |

array[j+1]=12



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 43 | 76 | 98 | 32 |

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 43 | 76 | 98 | 32 |

**Iteration 7:**

i=7, key=32, j=6,array[j]=98 while condition: True :

array[j+1]=98, ,j=5



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 43 | 76 | 98 | 32 |

array[j+1]=76, ,j=4



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 43 | 76 | 32 | 98 |

array[j+1]=43, ,j=3



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 43 | 32 | 76 | 98 |

array[j+1]=32



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 32 | 43 | 76 | 98 |

Iterated Result

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 5 | 12 | 23 | 32 | 43 | 76 | 98 |

**Case1:**

B= {6,7,8,9,10}

InsertionSort(A,1,len(B))

**Iteration 1:**

i=1, j=0, key=7, array[j]=6, while condition: False



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

Iterated Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

**Iteration 2:**

i=2, j=1, key=8, array[j]=7, while condition: False



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

Iterated Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

**Iteration 3:**

i=3, j=2, key=9, array[j]=8, while condition: False



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

Iterated Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

**Iteration 4:**

i=4, j=3, key=10, array[j]=9, while condition: False



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

Iterated Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | 7 | 8 | 9 | 10 |

**Merge Sort**

2,3. Dry run the MergeSort on the following inputs. Use the code given in CLRS Section 2.3. Show in detail what happens in each recursive function.

A= {1,3,54,2,72,23,12,32,76,12}

**Code:**

import numpy as np

def Merge(array,start,mid,end):

    n1=mid-start+1

    n2=end-mid

    L=[0]\*(n1+1)

    R=[0]\*(n2+1)

    for i in range(n1):

        L[i]=array[start+i]

    for j in range(n2):

        R[j]=array[mid+j+1]

    L[len(L)-1]=np.inf

    R[len(R)-1]=np.inf

    i=j=0

    for s in range(start,end+1):

        if(L[i]<=R[j]):

            array[s]=L[i]

            i+=1

        else:

            array[s]=R[j]

            j=j+1

def MergeSort(array, start, end):

    if(start!=end):

        mid=start+(end-start)//2

        MergeSort(array, start, mid)

        MergeSort(array, mid+1, end)

        Merge(array, start, mid, end)

    else:

        return

A= {1,3,54,2,72,23,12,32,76,12}

MergeSort(A,0,len(A)-1)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 | 54 | 2 | 72 | 23 | 12 | 32 | 76 | 12 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 3 | 54 | 2 | 72 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 23 | 12 | 32 | 76 | 12 |

**First Call (Divide)**

**Second Call (Divide)**

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 54 |

|  |  |
| --- | --- |
| 2 | 72 |

|  |  |  |
| --- | --- | --- |
| 23 | 12 | 32 |

|  |  |
| --- | --- |
| 76 | 12 |

**Third Call (Divide)**

|  |  |
| --- | --- |
| 1 | 3 |

|  |
| --- |
| 54 |

|  |
| --- |
| 2 |

|  |
| --- |
| 72 |

|  |  |
| --- | --- |
| 23 | 12 |

|  |
| --- |
| 32 |

|  |
| --- |
| 76 |

|  |
| --- |
| 12 |

**Fourth Call (Conquer)**

|  |
| --- |
| 23 |

|  |
| --- |
| 12 |

|  |
| --- |
| 32 |

|  |
| --- |
| 76 |

|  |
| --- |
| 12 |

|  |
| --- |
| 1 |

|  |
| --- |
| 3 |

|  |
| --- |
| 54 |

|  |
| --- |
| 2 |

|  |
| --- |
| 72 |

**Fifth Call (Conquer)**

|  |  |
| --- | --- |
| 1 | 3 |

|  |
| --- |
| 54 |

|  |  |
| --- | --- |
| 2 | 72 |

|  |  |
| --- | --- |
| 23 | 12 |

|  |
| --- |
| 32 |

|  |  |
| --- | --- |
| 12 | 76 |

**Sixth Call (Conquer)**

|  |  |  |
| --- | --- | --- |
| 1 | 3 | 54 |

|  |  |
| --- | --- |
| 2 | 72 |

|  |  |
| --- | --- |
| 12 | 76 |

|  |  |  |
| --- | --- | --- |
| 23 | 12 | 32 |

**Seventh Call (Conquer)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 54 | 72 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 12 | 12 | 23 | 32 | 76 |

**Final Array**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 12 | 12 | 23 | 32 | 54 | 72 | 76 |

4. Do you think that MergeSort always takes same number of operations in sorted and unsorted array. Justify your answer.

**Answer:**

Before answering the question first understand the process of how to merge sort work. It follows the approach of Divide and Conquers means first dividing all the arrays to the smallest size which is one and then merging all elements.

Remember that only one element in an array is sorted. After dividing all arrays into a unit element, it follows the conquer approach to merge all the sorted arrays into the original array which is now sorted.

So, we say that in a sorted and unsorted array it takes n number of steps.

5. Change the pseudo code of MergeSort to arrange elements in decreasing order rather than increasing order. Write down the modified pseudo code.

**Pseudo code:**

Merge(A,p,q,r)

n1←q-p+q

n2←r-p

create arrays L[1..n1+1] and R[1..n2+1]

for i←1 to n1

do L[i] ←A[p+i-1]

for j←1 to n2

do R[j] ←A[q+j]

L[n1+1] ←∞

R[n2+1] ←∞

i←1

j←1

for k← p to r

do if L[i]≥R[j]

then A[k] ←R[j]

j←j+1

else A[k] ←L[i]

j←j+1

MergeSort(A,p,r)

If(p<r)

then q←[(p+r)/2]

MergeSort(A,p,q)

MergeSort(A,p+1,q)

Merge(A,p,q,r)

**Selection Sort**

6. Consider the following code of SelectionSort. Provide the detailed asymptotic analysis of code similar to that of InsertionSort discussed in class.

**Solution:**

2.(n-1)+(n-2)+(n-3)+(n-4)+…

3. 1

4. n

5. 1

6. 1

7. 1

9. 1

Calculation:

n2= (n-1)+(n-2)+(n-3)+(n-4)+…

T(n)=n2×1×n×1×1×1×1

T(n)=n2×n

T(n)=O(n2)

**7.** Discuss the Loop Invariant of SelectionSort.

**Answer:**

The loop invariant of Selection sort is true before(0..i), during(i..n), and after(0..n-1) the loop

**Bubble Sort**

8. Write down one paragraph description of Bubble Sort in your own words.

**Answer:**

The Bubble Sort algorithm is like a bubble. In childhood, we play with a bubble and place the big bubble on a side just like placing the big element of the array on the right side and continue till we get the smallest bubble on the left side. In this way, the array will sort

9.Provide pseudo code of BubbleSort.

**Pseudo code:**

BubbleSort(A)

For i←1 to length[A]

do for j←length[A] downto i+1

do if A[j]<A[j-i]

then exchange A[j]↔A[j-1]

10. Run your algorithm on the following input.

A= {9,8,7,6,4,3,2,1}

**Code:**

def BubbleSort(array,start,end):

    for i in range(start,end-1):

        for j in range(0, end-i-1):

            if array[j] > array[j+1]:

                array[j], array[j+1] = array[j+1], array[j]

    return array

**Dry Run**

A=[9,8,7,6,4,3,2,1]

BubbleSort(A,0,len(A)):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 9 | 8 | 7 | 6 | 4 | 3 | 2 | 1 |

**Iteration 1:**

i=0,j=0, array[j]=9 , array[j+1]=8 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 9 | 7 | 6 | 4 | 3 | 2 | 1 |

j=1, array[j]=9 , array[j+1]=7 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 9 | 6 | 4 | 3 | 2 | 1 |

j=2, array[j]=9 , array[j+1]=6 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 9 | 4 | 3 | 2 | 1 |

j=3, array[j]=9 , array[j+1]=4 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 4 | 9 | 3 | 2 | 1 |

j=4, array[j]=9 , array[j+1]=3 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 4 | 3 | 9 | 2 | 1 |

j=5, array[j]=9 , array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 4 | 3 | 2 | 9 | 1 |

j=6, array[j]=9 , array[j+1]=1 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 8 | 7 | 6 | 4 | 3 | 2 | 1 | 9 |

**Iteration 2:**

i=1,j=0, array[j]=8 , array[j+1]=7 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 8 | 6 | 4 | 3 | 2 | 1 | 9 |

j=1, array[j]=8, array[j+1]=6 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 8 | 4 | 3 | 2 | 1 | 9 |

j=2, array[j]=8, array[j+1]=4 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 4 | 8 | 3 | 2 | 1 | 9 |

j=3, array[j]=8, array[j+1]=3 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 4 | 3 | 8 | 2 | 1 | 9 |

j=4, array[j]=8, array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 4 | 3 | 2 | 8 | 1 | 9 |

j=5, array[j]=8, array[j+1]=1 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 4 | 3 | 2 | 1 | 8 | 9 |

**Iteration 3:**

i=2,j=0, array[j]=7 , array[j+1]=6 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 7 | 4 | 3 | 2 | 1 | 8 | 9 |

j=1, array[j]=7 , array[j+1]=4 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 4 | 7 | 3 | 2 | 1 | 8 | 9 |

j=2, array[j]=7 , array[j+1]=3 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 4 | 3 | 7 | 2 | 1 | 8 | 9 |

j=3, array[j]=7 , array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 4 | 3 | 2 | 7 | 1 | 8 | 9 |

j=4, array[j]=7 , array[j+1]=1 After swap

` 

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 6 | 4 | 3 | 2 | 1 | 7 | 8 | 9 |

**Iteration 4:**

i=3,j=0, array[j]=6 , array[j+1]=4 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 6 | 3 | 2 | 1 | 7 | 8 | 9 |

j=1, array[j]=6 , array[j+1]=3 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 6 | 2 | 1 | 7 | 8 | 9 |

j=2, array[j]=6 , array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 6 | 1 | 7 | 8 | 9 |

j=3, array[j]=6 , array[j+1]=1 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 1 | 6 | 7 | 8 | 9 |

**Iteration 5:**

i=4,j=0, array[j]=4 , array[j+1]=3 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 4 | 2 | 1 | 6 | 7 | 8 | 9 |

j=1, array[j]=4 , array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 4 | 1 | 6 | 7 | 8 | 9 |

j=2, array[j]=4 , array[j+1]=1 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 2 | 1 | 4 | 6 | 7 | 8 | 9 |

**Iteration 6:**

i=5,j=0, array[j]=3 , array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 1 | 4 | 6 | 7 | 8 | 9 |

j=1, array[j]=3 , array[j+1]=2 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 3 | 4 | 6 | 7 | 8 | 9 |

**Iteration 7:**

i=6,j=0, array[j]=2 , array[j+1]=1 After swap



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 |

**Final Array**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 6 | 7 | 8 | 9 |