mad-pract-3-sorting

June 28, 2024

[1]:
```
# PRACTICAL - 3.1 : INSERTION SORTING
def insertion_sort(A):
  # since A[] index will start from 0
  count_for = 0
  count_while = 0
  for i in range(1,len(A)):
    key=A[i]
    j=i-1
    count_for=count_for+1
    print('count_for',count_for)
    while (j>=0) and (A[j]>key):
      A[j+1]=A[j]
      j=j-1
      count_while=count_while+1
      print('count_while',count_while)
    A[j+1] = key
```

[2]:
```
A=[15, 9, 30,10,1]
print('Original Array',A)
insertion_sort(A)
print('Sorted Array :',A)
```

```
Original Array [15, 9, 30, 10, 1]
count_for 1
count_while 1
count_for 2
count_for 3
count_while 2
count_while 3
count_for 4
count_while 4
count_while 5
count_while 6
count_while 7
Sorted Array : [1, 9, 10, 15, 30]
```

0.1 Insertion Sorting in the worst case , average case and best case.

[3]:
```
# Best Case
print("Best Case of Insertion Sort")
B=[1 ,2, 3, 4, 5, 6, 7]
print('Original Array :',B)
insertion_sort(B)
print('Sorted Array :',B)
```

```
Best Case of Insertion Sort
Original Array : [1, 2, 3, 4, 5, 6, 7]
count_for 1
count_for 2
count_for 3
count_for 4
count_for 5
count_for 6
Sorted Array : [1, 2, 3, 4, 5, 6, 7]
```

[4]:
```
# Worst Case
print("Worst Case of Insertion Sort")
C = [100,90,80,70,60,50,40,30,20,10]
print('Original Array :',C)
insertion_sort(C)
print('Sorted Array :',C)
```

```
Worst Case of Insertion Sort
Original Array : [100, 90, 80, 70, 60, 50, 40, 30, 20, 10]
count_for 1
count_while 1
count_for 2
count_while 2
count_while 3
count_for 3
count_while 4
count_while 5
count_while 6
count_for 4
count_while 7
count_while 8
count_while 9
count_while 10
count_for 5
count_while 11
count_while 12
count_while 13
count_while 14
count_while 15
```

```
count_for 6
count_while 16
count_while 17
count_while 18
count_while 19
count_while 20
count_while 21
count_for 7
count_while 22
count_while 23
count_while 24
count_while 25
count_while 26
count_while 27
count_while 28
count_for 8
count_while 29
count_while 30
count_while 31
count_while 32
count_while 33
count_while 34
count_while 35
count_while 36
count_for 9
count_while 37
count_while 38
count_while 39
count_while 40
count_while 41
count_while 42
count_while 43
count_while 44
count_while 45
Sorted Array : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

[5]:
```
# Average Case
print("Average Case of Insertion Sort")
D = [10, 8, 7, 6, 12, 15, 20, 30]
print('Original Array :',D)
insertion_sort(D)
print('Sorted Array :',D)
```

```
Average Case of Insertion Sort
Original Array : [10, 8, 7, 6, 12, 15, 20, 30]
count_for 1
count_while 1
count_for 2
```

```
count_while 2
count_while 3
count_for 3
count_while 4
count_while 5
count_while 6
count_for 4
count_for 5
count_for 6
count_for 7
Sorted Array : [6, 7, 8, 10, 12, 15, 20, 30]
```

[6]:
```python
# PRACTICAL - 3.2 : SELECTION SORT
def selection_sort(array):
  temp=0
  count_for_i=0
  count_for_j=0
  for i in range(0,len(array)-1):
    count_for_i=count_for_i +1
    print('count_for_i :',count_for_i)
    min = i # set current element as minimum
    print('min :',min, ' and array[min] :',array[min])
    for j in range(i+1,len(array)):
      count_for_j = count_for_j + 1
      print('count_for_j :',count_for_j)
      #check the element to be minimum   ..... n - i
      if array[j] < array[min]:
        min = j
        print('min :',min, ' and array[min] :',array[min])
    if (min!=i):
      temp=array[min]
      array[min]=array[i]
      array[i]=temp
```

[7]:
```python
array=[15, 30, 25, 10, 35, 20, 45, 40]
print('Orginal Array :',array)
selection_sort(array)
print('Sorted Array :',array)
```

```
Orginal Array : [15, 30, 25, 10, 35, 20, 45, 40]
count_for_i : 1
min : 0  and array[min] : 15
count_for_j : 1
count_for_j : 2
count_for_j : 3
min : 3  and array[min] : 10
count_for_j : 4
count_for_j : 5
```

```
count_for_j : 6
count_for_j : 7
count_for_i : 2
min : 1  and array[min] : 30
count_for_j : 8
min : 2  and array[min] : 25
count_for_j : 9
min : 3  and array[min] : 15
count_for_j : 10
count_for_j : 11
count_for_j : 12
count_for_j : 13
count_for_i : 3
min : 2  and array[min] : 25
count_for_j : 14
count_for_j : 15
count_for_j : 16
min : 5  and array[min] : 20
count_for_j : 17
count_for_j : 18
count_for_i : 4
min : 3  and array[min] : 30
count_for_j : 19
count_for_j : 20
min : 5  and array[min] : 25
count_for_j : 21
count_for_j : 22
count_for_i : 5
min : 4  and array[min] : 35
count_for_j : 23
min : 5  and array[min] : 30
count_for_j : 24
count_for_j : 25
count_for_i : 6
min : 5  and array[min] : 35
count_for_j : 26
count_for_j : 27
count_for_i : 7
min : 6  and array[min] : 45
count_for_j : 28
min : 7  and array[min] : 40
Sorted Array : [10, 15, 20, 25, 30, 35, 40, 45]
```

0.2 Selection Sort- Worst case , Best case and Average Case

[8]:
```
# Best Case
print("Best Case of Selection Sort:")
best=[1,2,3,4,5,6,7,8]
print('Orginal Arrary :',best)
selection_sort(best)
print('Sorted Arrary :',best)
```

```
Best Case of Selection Sort:
Orginal Arrary : [1, 2, 3, 4, 5, 6, 7, 8]
count_for_i : 1
min : 0  and array[min] : 1
count_for_j : 1
count_for_j : 2
count_for_j : 3
count_for_j : 4
count_for_j : 5
count_for_j : 6
count_for_j : 7
count_for_i : 2
min : 1  and array[min] : 2
count_for_j : 8
count_for_j : 9
count_for_j : 10
count_for_j : 11
count_for_j : 12
count_for_j : 13
count_for_i : 3
min : 2  and array[min] : 3
count_for_j : 14
count_for_j : 15
count_for_j : 16
count_for_j : 17
count_for_j : 18
count_for_i : 4
min : 3  and array[min] : 4
count_for_j : 19
count_for_j : 20
count_for_j : 21
count_for_j : 22
count_for_i : 5
min : 4  and array[min] : 5
count_for_j : 23
count_for_j : 24
count_for_j : 25
count_for_i : 6
min : 5  and array[min] : 6
```

```
count_for_j : 26
count_for_j : 27
count_for_i : 7
min : 6  and array[min] : 7
count_for_j : 28
Sorted Arrary : [1, 2, 3, 4, 5, 6, 7, 8]
```

[9]:
```
# Average Case
print("Average Case of Selection Sort:")
average=[10,5,12,2,8,20,1]
print('Orginal Arrary :',average)
selection_sort(best)
print('Sorted Arrary :',average)
```

```
Average Case of Selection Sort:
Orginal Arrary : [10, 5, 12, 2, 8, 20, 1]
count_for_i : 1
min : 0  and array[min] : 1
count_for_j : 1
count_for_j : 2
count_for_j : 3
count_for_j : 4
count_for_j : 5
count_for_j : 6
count_for_j : 7
count_for_i : 2
min : 1  and array[min] : 2
count_for_j : 8
count_for_j : 9
count_for_j : 10
count_for_j : 11
count_for_j : 12
count_for_j : 13
count_for_i : 3
min : 2  and array[min] : 3
count_for_j : 14
count_for_j : 15
count_for_j : 16
count_for_j : 17
count_for_j : 18
count_for_i : 4
min : 3  and array[min] : 4
count_for_j : 19
count_for_j : 20
count_for_j : 21
count_for_j : 22
count_for_i : 5
min : 4  and array[min] : 5
```

```
count_for_j : 23
count_for_j : 24
count_for_j : 25
count_for_i : 6
min : 5  and array[min] : 6
count_for_j : 26
count_for_j : 27
count_for_i : 7
min : 6  and array[min] : 7
count_for_j : 28
Sorted Arrary : [10, 5, 12, 2, 8, 20, 1]
```

[10]:
```
# Worst Case
print("Worst Case of Selection Sort:")
worst=[5,4,3,2,1]
print('Orginal Arrary :',worst)
selection_sort(best)
print('Sorted Arrary :',worst)
```

```
Worst Case of Selection Sort:
Orginal Arrary : [5, 4, 3, 2, 1]
count_for_i : 1
min : 0  and array[min] : 1
count_for_j : 1
count_for_j : 2
count_for_j : 3
count_for_j : 4
count_for_j : 5
count_for_j : 6
count_for_j : 7
count_for_i : 2
min : 1  and array[min] : 2
count_for_j : 8
count_for_j : 9
count_for_j : 10
count_for_j : 11
count_for_j : 12
count_for_j : 13
count_for_i : 3
min : 2  and array[min] : 3
count_for_j : 14
count_for_j : 15
count_for_j : 16
count_for_j : 17
count_for_j : 18
count_for_i : 4
min : 3  and array[min] : 4
count_for_j : 19
```

```
count_for_j : 20
count_for_j : 21
count_for_j : 22
count_for_i : 5
min : 4  and array[min] : 5
count_for_j : 23
count_for_j : 24
count_for_j : 25
count_for_i : 6
min : 5  and array[min] : 6
count_for_j : 26
count_for_j : 27
count_for_i : 7
min : 6  and array[min] : 7
count_for_j : 28
Sorted Arrary : [5, 4, 3, 2, 1]
```

[11]:
```python
# PRACTICAL - 3.3 MERGE SORT
def merge_sort(arr):
    if len(arr) > 1:
        # Calculate the middle of the array
        mid = len(arr) // 2
        print('mid :',mid)
        left = arr[:mid]  # Divide the array into two halves
        right = arr[mid:]
        print('left arr :',left)
        print('right arr :',right)
        merge_sort(left)  # Recursively sort the left half
        merge_sort(right)  # Recursively sort the right half
        i = j = k = 0
        # Merge the two halves sorted in step 2 and 3
        while i < len(left) and j < len(right):
            if left[i] < right[j]:
                arr[k] = left[i]
                i=i+1
            else:
                arr[k] = right[j]
                j=j+1
            k=k+1
        # Check if any elements were left
        while i < len(left):
            arr[k] = left[i]
            i=i+1
            k=k+1
        while j < len(right):
            arr[k] = right[j]
            j=j+1
```

```
            k=k+1
    return arr
```

[12]:
```
# Take array with even number of elements
arr = [12, 11, 13, 5, 6, 7]
print('Original array:', arr)
print('length of array: ',len(arr))
sorted_array = merge_sort(arr)
print('Sorted array:',sorted_array)
```

```
Original array: [12, 11, 13, 5, 6, 7]
length of array:  6
mid : 3
left arr : [12, 11, 13]
right arr : [5, 6, 7]
mid : 1
left arr : [12]
right arr : [11, 13]
mid : 1
left arr : [11]
right arr : [13]
mid : 1
left arr : [5]
right arr : [6, 7]
mid : 1
left arr : [6]
right arr : [7]
Sorted array: [5, 6, 7, 11, 12, 13]
```

[13]:
```
# Take array with odd number of elements
arr = [12, 11, 13, 5, 6]
print('Original array:', arr)
print('length of array: ',len(arr))
sorted_array = merge_sort(arr)
print('Sorted array:',sorted_array)
```

```
Original array: [12, 11, 13, 5, 6]
length of array:  5
mid : 2
left arr : [12, 11]
right arr : [13, 5, 6]
mid : 1
left arr : [12]
right arr : [11]
mid : 1
left arr : [13]
right arr : [5, 6]
```

```
mid : 1
left arr : [5]
right arr : [6]
Sorted array: [5, 6, 11, 12, 13]
```

0.3 Merge Sort - Worst Case , Best Case and Average Case

[14]:
```
# Best Case
print("Best Case of Merge Sort :")
best = [1, 2, 3, 5, 6, 7]
print('Original array:', best)
print('length of array: ',len(best))
sorted_best = merge_sort(best)
print('Sorted array:',sorted_best)
```

```
Best Case of Merge Sort :
Original array: [1, 2, 3, 5, 6, 7]
length of array:  6
mid : 3
left arr : [1, 2, 3]
right arr : [5, 6, 7]
mid : 1
left arr : [1]
right arr : [2, 3]
mid : 1
left arr : [2]
right arr : [3]
mid : 1
left arr : [5]
right arr : [6, 7]
mid : 1
left arr : [6]
right arr : [7]
Sorted array: [1, 2, 3, 5, 6, 7]
```

[15]:
```
# Average Case
print("Average Case of Merge sort :")
average = [50, 40, 30, 20, 2, 6, 8, 13, 17]
print('Original array:', average)
print('length of array: ',len(average))
sorted_average = merge_sort(average)
print('Sorted array:',sorted_average)
```

```
Average Case of Merge sort :
Original array: [50, 40, 30, 20, 2, 6, 8, 13, 17]
length of array:  9
mid : 4
left arr : [50, 40, 30, 20]
```

```
right arr : [2, 6, 8, 13, 17]
mid : 2
left arr : [50, 40]
right arr : [30, 20]
mid : 1
left arr : [50]
right arr : [40]
mid : 1
left arr : [30]
right arr : [20]
mid : 2
left arr : [2, 6]
right arr : [8, 13, 17]
mid : 1
left arr : [2]
right arr : [6]
mid : 1
left arr : [8]
right arr : [13, 17]
mid : 1
left arr : [13]
right arr : [17]
Sorted array: [2, 6, 8, 13, 17, 20, 30, 40, 50]
```

[16]:
```
# Worst Case
print("Worst Case of Merge Sort :")
worst = [12, 3, 11, 5, 13, 6, 20]
print('Original array:', worst)
print('length of array: ',len(worst))
sorted_worst = merge_sort(worst)
print('Sorted array:',sorted_worst)
```

```
Worst Case of Merge Sort :
Original array: [12, 3, 11, 5, 13, 6, 20]
length of array:  7
mid : 3
left arr : [12, 3, 11]
right arr : [5, 13, 6, 20]
mid : 1
left arr : [12]
right arr : [3, 11]
mid : 1
left arr : [3]
right arr : [11]
mid : 2
left arr : [5, 13]
right arr : [6, 20]
mid : 1
```

```
left arr : [5]
right arr : [13]
mid : 1
left arr : [6]
right arr : [20]
Sorted array: [3, 5, 6, 11, 12, 13, 20]
```

[17]:
```python
# PRACTICAL - 3.4 HEAP SORT
def heapify(arr, n, i):
    largest = i  # Initialize largest as root
    L = 2 * i + 1     # left = 2*i + 1
    R = 2 * i + 2     # right = 2*i + 2
    # See if left child of root exists and is greater than root
    if L < n and arr[largest] < arr[L]:
        largest = L
    # See if right child of root exists and is greater than root
    if R < n and arr[largest] < arr[R]:
        largest = R
    # Change root, if needed
    if largest != i:
        arr[i], arr[largest] = arr[largest], arr[i]  # swap
        # Heapify the root.
        heapify(arr, n, largest)
```

[18]:
```python
def heap_sort(arr):
  n = len(arr)
  # Build a maxheap.
  for i in range(n//2 - 1, -1, -1):
    heapify(arr, n, i)
  # One by one extract elements
  for i in range(n-1, 0, -1):
    arr[i], arr[0] = arr[0], arr[i]  # swap
    heapify(arr, i, 0)
```

[19]:
```python
# Test Code for Heap Sort
arr=[10, 2 , 3, 12, 11, 23, 34, 8,]
print("Original Array :",arr)
# Function call
heap_sort(arr)
n = len(arr)
print("Sorted array is",arr)
```

```
Original Array : [10, 2, 3, 12, 11, 23, 34, 8]
Sorted array is [2, 3, 8, 10, 11, 12, 23, 34]
```