



Abbottabad University Of Sciences And Technology

Name:	Sanaullah Baig
Roll no:	12402
Section:	C
Subject:	Data Structure And Algorithm
Lab no:	4
Teacher:	Mr Jamal Abdul Ahad
Class:	BSSE 3 rd
Date:	05/11/2023

Question # 1:

Modify the merge sort algorithm to count the number of inversions in arrays.

An inversion is a pair of indices (i, j) such that $i < j$ and $arr[i] > arr[j]$.

Program And Output:

```
no of inversion.py X
no of inversion.py > ...
1  def merge(arr, left, mid, right):
2      inv_count = 0
3
4      left_arr = arr[left:mid + 1]
5      right_arr = arr[mid + 1:right + 1]
6
7      i = j = 0
8      k = left
9
10     while i < len(left_arr) and j < len(right_arr):
11         if left_arr[i] <= right_arr[j]:
12             arr[k] = left_arr[i]
13             i += 1
14         else:
15             arr[k] = right_arr[j]
16             j += 1
17             inv_count += (len(left_arr) - i) # Counting inversions
18             k += 1
19
20     while i < len(left_arr):
21         arr[k] = left_arr[i]
22         i += 1
23         k += 1
24
25     while j < len(right_arr):
26         arr[k] = right_arr[j]
27         j += 1
28         k += 1
29
30     return inv_count
31
32 def mergeSort(arr, left, right):
33     inv_count = 0
34     if left < right:
35         mid = (left + right) // 2
36
37         inv_count += mergeSort(arr, left, mid)
38         inv_count += mergeSort(arr, mid + 1, right)
```

Question # 2:

Implement the merge sort algorithm for sorting linked lists instead of arrays.

This exercise will require modifying the merge process..

Program And Output:

```
no of inversion.py  sort linked list.py X
sort linked list.py > ...
1 class ListNode:
2     def __init__(self, value=0, next=None):
3         self.val = value
4         self.next = next
5
6 def findMiddle(head):
7     slow = head
8     fast = head.next if head else None
9
10    while fast and fast.next:
11        slow = slow.next
12        fast = fast.next.next
13
14    return slow
15
16 def merge(left, right):
17     dummy = ListNode(0)
18     current = dummy
19
20    while left and right:
21        if left.val < right.val:
22            current.next = left
23            left = left.next
24        else:
25            current.next = right
26            right = right.next
27        current = current.next
28
29    current.next = left or right
30
31    return dummy.next
32
33 def mergeSort(head):
34     if not head or not head.next:
35         return head
36
37     middle = findMiddle(head)
```

```
no of inversion.py  sort linked list.py X
sort linked list.py > ...
36
37     middle = findMiddle(head)
38     next_to_middle = middle.next
39     middle.next = None
40
41     left = mergeSort(head)
42     right = mergeSort(next_to_middle)
43
44     return merge(left, right)
45
46 head = ListNode(4)
47 head.next = ListNode(2)
48 head.next.next = ListNode(1)
49 head.next.next.next = ListNode(3)
50
51 sorted_list = mergeSort(head)
52
53 while sorted_list:
54     print(sorted_list.val, end=" ")
55     sorted_list = sorted_list.next
56
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\PC\Desktop\Python> & "C:\Program Files\Python311\python.exe" "c:/Users/PC/Desktop/Python/sort linked list.py"

1 2 3 4

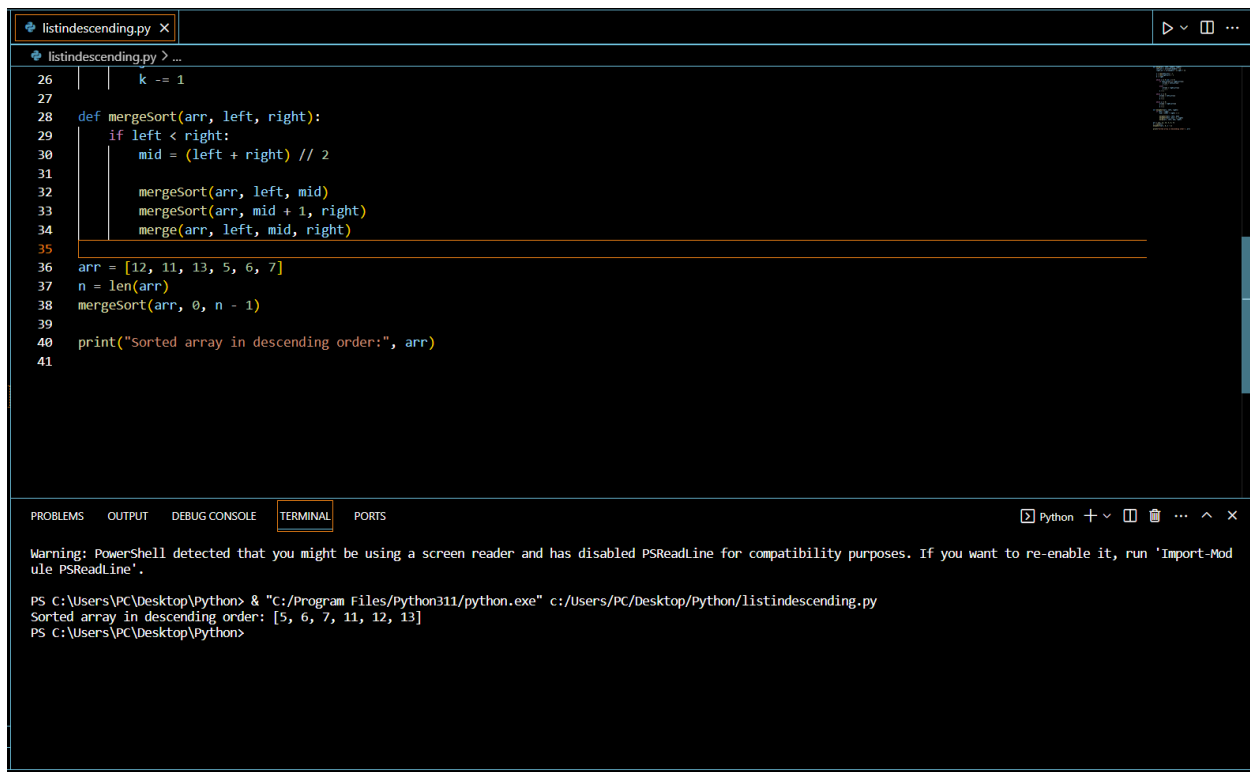
PS C:\Users\PC\Desktop\Python>

Question # 3:

Modify the merge sort algorithm for a sort list in descending order instead of ascending order. This will require changes to the merging step.

Input And Output:

```
listindescending.py
listindescending.py > ...
1 def merge(arr, left, middle, right):
2     left_arr = arr[left:middle + 1]
3     right_arr = arr[middle + 1:right + 1]
4
5     i = len(left_arr) - 1
6     j = len(right_arr) - 1
7     k = right
8
9     while i >= 0 and j >= 0:
10        if left_arr[i] >= right_arr[j]:
11            arr[k] = left_arr[i]
12            i -= 1
13        else:
14            arr[k] = right_arr[j]
15            j -= 1
16        k -= 1
17
18    while i >= 0:
19        arr[k] = left_arr[i]
20        i -= 1
21        k -= 1
22
23    while j >= 0:
24        arr[k] = right_arr[j]
25        j -= 1
26        k -= 1
27
28 def mergeSort(arr, left, right):
29     if left < right:
30         mid = (left + right) // 2
31
32         mergeSort(arr, left, mid)
33         mergeSort(arr, mid + 1, right)
34         merge(arr, left, mid, right)
35
36 arr = [12, 11, 13, 5, 6, 7]
37 n = len(arr)
```



The screenshot shows a Python IDE with a file named `listindescending.py`. The code implements a merge sort algorithm. The `mergeSort` function is recursive, splitting the array into two halves and merging them back in sorted order. The array `arr` is initialized with the values `[12, 11, 13, 5, 6, 7]`. The output of the program is printed as "Sorted array in descending order:", followed by the sorted array.

```
26 | k -= 1
27 |
28 | def mergeSort(arr, left, right):
29 |     if left < right:
30 |         mid = (left + right) // 2
31 |         mergeSort(arr, left, mid)
32 |         mergeSort(arr, mid + 1, right)
33 |         merge(arr, left, mid, right)
34 |
35 |
36 | arr = [12, 11, 13, 5, 6, 7]
37 | n = len(arr)
38 | mergeSort(arr, 0, n - 1)
39 |
40 | print("Sorted array in descending order:", arr)
41 |
```

The terminal output shows the command executed and the resulting sorted array:

```
PS C:\Users\PC\Desktop\Python> & "C:/Program Files/python311/python.exe" c:/Users/PC/Desktop/Python/listindescending.py
Sorted array in descending order: [5, 6, 7, 11, 12, 13]
PS C:\Users\PC\Desktop\Python>
```

Question # 4:

Extend the merge sort algorithm to work with three or more sub lists at each stop. not just two. This is called a three way (or multi-way) merge sort.

Input And Output:

```
no of inversion.py  sort linked list.py  extend mergesort.py X
extend mergesort.py > merge
1 def merge(arr, left, middle1, middle2, right):
2
3     temp = [0] * (right - left + 1)
4     i = left
5     j = middle1 + 1
6     k = middle2 + 1
7     t = 0
8
9     while i <= middle1 and j <= middle2 and k <= right:
10         if arr[i] <= arr[j] and arr[i] <= arr[k]:
11             temp[t] = arr[i]
12             i += 1
13         elif arr[j] <= arr[i] and arr[j] <= arr[k]:
14             temp[t] = arr[j]
15             j += 1
16         else:
17             temp[t] = arr[k]
18             k += 1
19         t += 1
20
21     while i <= middle1 and j <= middle2:
22         if arr[i] <= arr[j]:
23             temp[t] = arr[i]
24             i += 1
25         else:
26             temp[t] = arr[j]
27             j += 1
28         t += 1
29
30     while j <= middle2 and k <= right:
31         if arr[j] <= arr[k]:
32             temp[t] = arr[j]
33             j += 1
34         else:
35             temp[t] = arr[k]
36             k += 1
37         t += 1
38
```

```
no of inversion.py  sort linked list.py  extend mergesort.py X
extend mergesort.py > merge
36         temp[t] = arr[k]
37         k += 1
38         t += 1
39
40     while i <= middle1 and k <= right:
41         if arr[i] <= arr[k]:
42             temp[t] = arr[i]
43             i += 1
44         else:
45             temp[t] = arr[k]
46             k += 1
47         t += 1
48
49     while i <= middle1:
50         temp[t] = arr[i]
51         i += 1
52         t += 1
53
54     while j <= middle2:
55         temp[t] = arr[j]
56         j += 1
57         t += 1
58
59     while k <= right:
60         temp[t] = arr[k]
61         k += 1
62         t += 1
63
64     for p in range(left, right + 1):
65         arr[p] = temp[p - left]
66
67 def threeWayMergeSort(arr, left, right):
68     if right > left:
69         mid1 = left + (right - left) // 3
70         mid2 = left + 2 * (right - left) // 3
71
72         threeWayMergeSort(arr, left, mid1)
73         threeWayMergeSort(arr, mid1 + 1, mid2)
```

no of inversion.py sort linked list.py extend mergesort.py X

extend mergesort.py > merge

```
71         threeWayMergeSort(arr, left, mid1)
72         threeWayMergeSort(arr, mid1 + 1, mid2)
73         threeWayMergeSort(arr, mid2 + 1, right)
74
75         merge(arr, left, mid1, mid2, right)
76
77     arr = [12, 11, 13, 5, 6, 7]
78     n = len(arr)
79     threeWayMergeSort(arr, 0, n - 1)
80
81     print("Sorted array:", arr)
82
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\PC\Desktop\Python> & "C:/Program Files/Python311/python.exe" "c:/Users/PC/Desktop/Python/extend mergesort.py"
Sorted array: [5, 6, 7, 11, 12, 13]
PS C:\Users\PC\Desktop\Python>