

```
1 ## Function to Use Inversion Count
2
3 def mergeSort(arr, n):
4
5     # A temp_arr is created to store
6
7     # sorted array in merge function
8
9     temp_arr = [0]*n
10
11     return _mergeSort(arr, temp_arr, 0, n-1)
12
13
14 ## This Function will use MergeSort to count inversions
15
16
17
18 def _mergeSort(arr, temp_arr, left, right):
19
20
21
22     # A variable inv_count is used to store
23
24     # inversion counts in each recursive call
25
26
27
28     inv_count = 0
29
30
31
32     # We will make a recursive call if and only if
33
34     # we have more than one elements
35
36
37
38     if left < right:
39
40
41
42         # mid is calculated to divide the array into two subarrays
43
44         # Floor division is must in case of python
45
46
47
48         mid = (left + right)//2
49
50
51
52         # It will calculate inversion
53
54         # counts in the left subarray
55
56
57
58         inv_count += _mergeSort(arr, temp_arr,
59
60                                 left, mid)
61
62
63
64         # It will calculate inversion
65
66         # counts in right subarray
67
68
69
70         inv_count += _mergeSort(arr, temp_arr,
71
72                                 mid + 1, right)
73
74
75
76         # It will merge two subarrays in
77
78         # a sorted subarray
```

```
79
80
81
82     inv_count += merge(arr, temp_arr, left, mid, right)
83
84     return inv_count
85
86
87 ## This function will merge two subarrays
88 # in a single sorted subarray
89
90 def merge(arr, temp_arr, left, mid, right):
91
92     i = left      # Starting index of left subarray
93
94     j = mid + 1 # Starting index of right subarray
95
96     k = left      # Starting index of to be sorted subarray
97
98     inv_count = 0
99
100
101
102     # Conditions are checked to make sure that
103
104     # i and j don't exceed their
105
106     # subarray limits.
107
108
109
110     while i <= mid and j <= right:
111
112
113
114         # There will be no inversion if arr[i] <= arr[j]
115
116
117
118         if arr[i] <= arr[j]:
119
120             temp_arr[k] = arr[i]
121
122             k += 1
123
124             i += 1
125
126         else:
127
128             # Inversion will occur.
129
130             temp_arr[k] = arr[j]
131
132             inv_count += (mid - i + 1)
133
134             k += 1
135
136             j += 1
137
138
139
140     # Copy the remaining elements of left
141
142     # subarray into temporary array
143
144     while i <= mid:
145
146         temp_arr[k] = arr[i]
147
148         k += 1
149
150         i += 1
151
152
153
154     # Copy the remaining elements of right
155
156     # subarray into temporary array
157
158     while j <= right:
```

```
158         while j < right:
159
160             temp_arr[k] = arr[j]
161
162             k += 1
163
164             j += 1
165
166
167
168     # Copy the sorted subarray into Original array
169
170     for loop_var in range(left, right + 1):
171
172         arr[loop_var] = temp_arr[loop_var]
173
174
175
176     return inv_count
177
178
179
180
181 arr = [1, 20, 6, 4, 5]
182
183 n = len(arr)
184
185 result = mergeSort(arr, n)
186
187 print("Number of inversions are", result)
```

☞ Number of inversions are 5