Practice 02:

Implementation of Tim Sort in Python

Code:

MIN\_MERGE = 32

def calcMinRun(n):

r = 0

while n >= MIN\_MERGE:

r |= n & 1

n >>= 1

return n + r

# This function sorts array from left index to right index which is of size at most RUN

def insertionSort(arr, left, right):

for i in range(left + 1, right + 1):

j = i

while j > left and arr[j] < arr[j - 1]:

arr[j], arr[j - 1] = arr[j - 1], arr[j]

j -= 1

# Merge function merges the sorted runs

def merge(arr, l, m, r):

len1, len2 = m - l + 1, r - m

left, right = [], []

for i in range(0, len1):

left.append(arr[l + i])

for i in range(0, len2):

right.append(arr[m + 1 + i])

i, j, k = 0, 0, l

# after comparing, we merge those two array in larger sub array

while i < len1 and j < len2:

if left[i] <= right[j]:

arr[k] = left[i]

i += 1

else:

arr[k] = right[j]

j += 1

k += 1

# Copy remaining element of left

while i < len1:

arr[k] = left[i]

k += 1

i += 1

# Copy remaining element of right

while j < len2:

arr[k] = right[j]

k += 1

j += 1

# Iterative Timsort function to sort the array

def timSort(arr):

n = len(arr)

minRun = calcMinRun(n)

for start in range(0, n, minRun):

end = min(start + minRun - 1, n - 1)

insertionSort(arr, start, end)

size = minRun

while size < n:

for left in range(0, n, 2 \* size):

mid = min(n - 1, left + size - 1)

right = min((left + 2 \* size - 1), (n - 1))

if mid < right:

merge(arr, left, mid, right)

size = 2 \* size

# Main Function

if \_\_name\_\_ == "\_\_main\_\_":

arr = [-2, 7, 15, -14, 0, 15, 0,

7, -7, -4, -13, 5, 8, -14, 12]

print("Given Array is")

print(arr)

timSort(arr)

print("After Sorting Array is")

print(arr)