- **Left Subarray**: Contains n1 = mid - i + 1 elements.
- **Right Subarray**: Contains n2 = j - mid elements.

For example, if i = 0, mid = 3, and j = 7, then:

- Left Subarray size (n1) = 3 - 0 + 1 = 4
- Right Subarray size (n2) = 7 - 3 = 4

```
for m in range(n1):
    leftSubarray[m] = arr[i + m]

for n in range(n2):
    rightSubarray[n] = arr[mid + 1 + n]
```

- **First loop**: Copies elements from **arr[i] to arr[mid]** into **leftSubarray**.
- **Second loop**: Copies elements from **arr[mid+1]** to **arr[j]** into **rightSubarray**.

Example:
Given **arr = [50, 70, 65, 13, 80, 62, 98, 27]**
If **i = 0**, **mid = 3**, **j = 7**, then:

- **Left Subarray**: [50, 70, 65, 13]
- **Right Subarray**: [80, 62, 98, 27]

Merge the Two Sorted Subarrays

```
p = 0
q = 0
k = i

while p < n1 and q < n2:
    if leftSubarray[p] <= rightSubarray[q]:
        arr[k] = leftSubarray[p]
        p += 1
    else:
        arr[k] = rightSubarray[q]
        q += 1
    k += 1
```

- p → Tracks index in leftSubarray
- q → Tracks index in rightSubarray
- k → Tracks index in arr (main array)

Copy Any Remaining Elements

```
while p < n1:
    arr[k] = leftSubarray[p]
    p += 1
    k += 1
```

If **leftSubarray** still has elements left, copy them to **arr**

```
while q < n2:
    arr[k] = rightSubarray[q]
    q += 1
    k += 1
```

If **rightSubarray** still has elements left, copy them to **arr**

Example Execution

Given Input Array:

```
arr = [50, 70, 65, 13, 80, 62, 98, 27]
```

We assume **mergeProcedure()** is called after sorting two halves:

- **Left Subarray (sorted):** [13, 50, 65, 70]
- **Right Subarray (sorted):** [27, 62, 80, 98]

Now, we merge them step by step.

Step-by-Step Merging Process

Step	Left (p)	Right (q)	Comparison	Result (arr[k])
1	13	27	13 ≤ 27	13
2	50	27	50 > 27	27
3	50	62	50 ≤ 62	50
4	65	62	65 > 62	62
5	65	80	65 ≤ 80	65
6	70	80	70 ≤ 80	70
7	**(Left exhausted)**	80	Copy remaining	80
8	**(Left exhausted)**	98	Copy remaining	98

Final Merged Array:

```
[13, 27, 50, 62, 65, 70, 80, 98]
```