# 2. Merge Algorithm for two Sorted Arrays

```
Algorithm merge (arrA , arrB)

    index A ← 0
    index B ← 0
    index C ← 0
    arrC ← int [arrA.size() + arrB.size()]

    while indexA < arrA.size() and indexB < arrB.size() do
        if arrA[indexA] < arrB[indexB] then
            arrC [index C] ← arrA [index A]
            indexA = indexA + 1
        else
            arrC [index C] ← arrB[indexB]
            indexB ← indexB + 1
        indexC ← index C + 1

    if indexA < arrA.size () then
        for i ← indexA to arrA.size() - 1 do
            arrC [index C] ← arrA [i]
            indexC ← index C + 1
            {increment counter i}

    else if indexB < arrB.size() then
        for i ← indexB to arrB.size() - 1 do
            arrC [indexC] ← arrB [i]
            indexC = indexC + 1
            {increment counter i}

    return arrC
```

## 2. Merge Algorithm for two Sorted Arrays

$\overset{\swarrow \text{arrA.size()}}{m} > \overset{\swarrow \text{arrB.size()}}{n}$

```
Algorithm merge (arrA , arrB)

        indexA ← 0
        indexB ← 0
        indexC ← 0
        arrC ← int [arrA.size() + arrB.size()]

        while indexA < arrA.size() and indexB < arrB.size() do
  n  {      if arrA[indexA] < arrB[indexB] then
                    arrC[indexC] ← arrA[indexA]
                    indexA = indexA + 1

            else
                arrC[indexC] ← arrB[indexB]
                indexB ← indexB + 1

            indexC ← indexC + 1
           }
        if indexA < arrA.size() then
  m  {      for i ← indexA to arrA.size()-1 do
                    arrC [indexC] ← arrA[i]
                    indexC ← indexC + 1
                    {increment counter i}
           }
        else if indexB < arrB.size() then
            for i ← indexB to arrB.size()-1 do
                    arrC [indexC] ← arrB[i]
                    indexC = indexC +1
                    {increment counter i}


        return arrC
```

∴ Running time is $O(m+n)$