**LC#88. Merge Sorted Array**

**APPROACH1:**

**->using two pointer put the value in a third array by comparing both nums1 and nums2 as it is sorted**

class Solution {

    public void merge(int[] nums1, int m, int[] nums2, int n) {

        int arr[] = new int[m+n];

        int left = 0 , right = 0 , idx =0;

        while(left<m && right<n)

        {

            if(nums1[left]<nums2[right])

            {

                arr[idx++] = nums1[left];

                left++;

            }

            else

            {

                arr[idx++] = nums2[right];

                right++;

            }

        }

        while(left<m)

            arr[idx++] = nums1[left++];

        while(right<n)

            arr[idx++] =nums2[right++];

        for(int i = 0 ; i< m+n;i++)

        {

            nums1[i] =arr[i];

        }

    }

**APP2:OPTIMAL**

**->shell sort**

**TC:O(nlogn)+O(m+n) SC:O(1)**

class Solution {

    void swap(int a[] , int b[] , int x , int y)

    {

        a[x] = a[x]^b[y];

        b[y] = a[x]^b[y];

        a[x] = a[x]^b[y];

    }

    public void merge(int[] nums1, int m, int[] nums2, int n) {

        int len = m+n;

        int gap = (len/2) +(len%2);

        while(gap>0)

        {

            int left = 0 ;

            int right = left +gap;

            while(right<len)

            {

                if(left<m && right>=m)//**left in arr1 and right in arr2**

                {

                    if(nums1[left]>nums2[right-m])

                    swap(nums1,nums2,left,right-m);

                }

                else if(left>=m)//**left and right in arr2**

                {

                    if(nums2[left-m]>nums2[right-m])

                    swap(nums2,nums2,left-m,right-m);

                }

                else

                {

                    if(nums1[left]>nums1[right])

                    swap(nums1,nums1,left ,right);// **in arr1**

                }

                left++;

            right++;

            }

            gap = (gap == 1) ? 0 : (gap / 2 + gap % 2);//**updating gap**

        }

        for (int i = 0; i < n; i++) {

            nums1[m + i] = nums2[i];//**copying sorted nums2 to nums1**

        }

    }

}