Sort by Set Bit Count

Given an array of integers, sort the array (in descending order) according to count of set bits in binary representation of array elements.

Note: For integers having same number of set bits in their binary representation, sort according to their position in the original array i.e., a stable sort.

Example 1:

Input:
arr[] = {5, 2, 3, 9, 4, 6, 7, 15, 32};
Output:
15 7 5 3 9 6 2 4 32
Explanation:
The integers in their binary
representation are:
15 - 1111
7 - 0111
5 - 0101
3 - 0011
9 - 1001
6 - 0110
2 - 0010
4 - 0100
32 - 10000
hence the non-increasing sorted order is:
{15}, {7}, {5, 3, 9, 6}, {2, 4, 32}

Example 2:

Input: arr[] = {1, 2, 3, 4, 5, 6};
Output:
3 5 6 1 2 4
Explanation:
3 - 0110
5 - 0101
6 - 0110
1 - 0001
2 - 0010
4 - 0100

hence the non-increasing sorted order is
{3, 5, 6}, {1, 2, 4}

Your Task:
You don't need to print anything, printing is done by the driver code itself. You just need to complete the function **sortBySetBitCount()** which takes the array **arr[]** and its size **N** as inputs and sort the array **arr[]** inplace. Use of extra space is prohibited.

Expected Time Complexity: O(N.log(N))
Expected Auxiliary Space: O(1)

```
class Solution:
    def sortBySetBitCount(self, arr, n):
        # Your code goes here
        Space is O(n)
        # res = arr
        # res = [(x,bin(x).count('1')) for x in res]
        # res = sorted(res,key=lambda x:-x[1])
        # res = [x[0] for x in res]
        # for i in range(n):
        #     arr[i] = res[i]
        space is O(1)
        arr.sort(key=lambda x:-bin(x).count('1'))
```