```python
# 冒泡排序
# 遍历数组，每次都相互比较，直到把大的数沉到数组尾部
# 冒泡排序的第一次循环实际上就是找最后会比较的位置，由于不停的两两比较，所l
def bubbleSort(nums):
    n = len(nums)
    for i in range(n - 1):
        for j in range(n - i - 1):
            if nums[j] > nums[j + 1]:
                nums[j], nums[j + 1] = nums[j + 1], nums[j]
    return nums


# 选择排序
# 遍历数组，找到最小的值，记录其下标，然后和最初值交换
# 因为要找一个比当前值还要小的数值的下标，若有相同的值，根据算法会选择后面
def selectionSort(nums):
    n = len(nums)
    for i in range(n - 1):
        minIndex = i
        for j in range(i + 1, n):
            if nums[j] < nums[minIndex]:
                minIndex = j
        nums[i], nums[minIndex] = nums[minIndex], nums[i]
    return nums
```

```python
# 插入排序
# 遍历数组，相互比较元素，小的被一步步往前交换
# 因为是一个个比较的向前，所以排序稳定
def insertionSort(nums):
    n = len(nums)
    for i in range(1, n):
        while i - 1 >= 0 and nums[i - 1] > nums[i]:
            nums[i - 1], nums[i] = nums[i], nums[i - 1]
            i -= 1
    return nums


# 希尔排序
# 插入排序提升版
# 因为不是逐个比较，因此对于相同的元素，后面的可能会被移到前面去，因此不稳
def shellSort(nums):
    n = len(nums)
    gap = n // 2
    while gap:
        for i in range(gap, n):
            while i - gap >= 0 and nums[i - gap] > nums[i]:
                nums[i - gap], nums[i] = nums[i], nums[i - gap
                i -= gap
        gap //= 2
    return nums
```

```python
def heapify(array, n, k):
    left = 2 * k + 1
    right = 2 * k + 2
    max_i = k
    if left < n and array[max_i] < array[left]:
        max_i = left
    if right < n and array[max_i] < array[right]:
        max_i = right
    if max_i != k:
        array[max_i], array[k] = array[k], array[max_i]
        heapify(array, n, max_i)

def heapSort(arr):
    n = len(arr)
    for i in range(n // 2 - 1, -1, -1):
        heapify(arr, n, i)
    for i in range(n - 1, 0, -1):
        arr[0], arr[i] = arr[i], arr[0] # swap
        heapify(arr, i, 0)
```

```python
def mergeSort(nums, left, right):
    if right <= left:
        return

    mid = (left + right) >> 1
    mergeSort(nums, left, mid)
    mergeSort(nums, mid + 1, right)
    merge(nums, left, mid, right)

def merge(nums, left, mid, right):
    tmp = []
    i = left
    j = mid + 1
    while i <= mid and j <= right:
        if nums[i] <= nums[j]:
            tmp.append(nums[i])
            i += 1
        else:
            tmp.append(nums[j])
            j += 1
    while i <= mid:
        tmp.append(nums[i])
        i += 1
    while j <= right:
        tmp.append(nums[j])
        j += 1
    nums[left:right + 1] = tmp
```

```python
import random
from typing import List

nums = [9,4,3,2,6]
def sortArray(nums: List[int]) -> List[int]:
    n = len(nums)

    def quick(left, right):
        if left >= right:
            return nums

        # 如果需要变成随机的快速排序，就加入下面两行
        t = random.randint(left, right) # 生成[left,right]之间的
        nums[t], nums[left] = nums[left], nums[t]

        pivot = left
        i = left
        j = right
        while i < j:
            while i < j and nums[j] > nums[pivot]:
                j -= 1
            while i < j and nums[i] <= nums[pivot]:
                i += 1
            nums[i], nums[j] = nums[j], nums[i]
        nums[pivot], nums[j] = nums[j], nums[pivot]
        quick(left, j - 1)
        quick(j + 1, right)
        return nums
    return quick(0, n - 1)
```