```python
'''冒泡排序'''
def bubble_sort(lst):
    for i in range(len(lst)-1):
        target = True
        for j in range(len(lst)-i-1):
            if lst[j] > lst[j+1]:
                lst[j],lst[j+1] = lst[j+1],lst[j]
                target = False
        if target:
            break
    return lst



'''选择排序'''
def select_sort(lst):
    for i in range(len(lst)-1):
        min_index = i
        for j in range(i+1,len(lst)):
            if lst[j] < lst[min_index]:
                min_index = j
        if min_index != i:
            lst[i],lst[min_index] = lst[min_index],lst[i]
    return lst


'''插入排序'''
def insert_sort(lst):
    for i in range(1,len(lst)):
        for j in range(i,0,-1):
            if lst[j] < lst[j-1]:
                lst[j],lst[j-1] = lst[j-1],lst[j]
    return lst


'''快速排序'''
def quick_sort(lst,left,right):
    if left > right:
        return lst
    pivot = lst[left]
    start = left
    end = right
    while start < end:
        while start < end and pivot < lst[end]:
            end -= 1
        lst[start] = lst[end]
        while start < end and pivot > lst[start]:
            start += 1
        lst[end] = lst[start]
    lst[start] = pivot
    quick_sort(lst,left,start-1)
    quick_sort(lst,start+1,right)
    return lst



'''合并排序'''
def merge_sort(lst):
    if len(lst) <= 1:
        return lst
    else:
        mid = len(lst)//2
        lst_l = merge_sort(lst[:mid])
        lst_r = merge_sort(lst[mid:])
    return merge(lst_l,lst_r)

def merge(left,right):
    l,r = 0,0
    rst = []
    while l < len(left) and r < len(right):
        if left[l] <= right[r]:
            rst.append(left[l])
            l += 1
        else:
            rst.append(right[r])
            r += 1
    rst += left[l:]
    rst += right[r:]
    return rst



'''希尔排序'''
def shell_sort(lst):
    gap = len(lst)//2
    while gap > 0:
        for i in range(gap,len(lst)):
            while i-gap >= 0 and lst[i] < lst[i-gap]:
                lst[i],lst[i-gap] = lst[i-gap],lst[i]
                i -= gap
        gap = gap//2
    return lst


'''堆排序'''
def sift(data, low, high):
    i = low       # 父节点
    j = 2 * i + 1    # 左子节点
    tmp = data[i]    # 父节点值
    while j <= high:    # 子节点在节点中
        if j < high and data[j] > data[j + 1]:  # 有右子节点且右节点比父节点值大
            j += 1
        if tmp > data[j]:
            data[i] = data[j]   # 将父节点替换成新的子节点的值
            i = j   # 变成新的父节点
            j = 2 * i + 1   # 新的子节点
        else:
            break
    data[i] = tmp   # 将替换的父节点值赋给最终的父节点

def heap_sort(data):
    n = len(data)
    # 创建堆
    for i in range(n//2-1, -1, -1):
        sift(data, i, n-1)

    # 挨个出数
    for i in range(n-1, -1, -1):    # 从大到小
        data[0], data[i] = data[i], data[0]     # 将最后一个值与父节点交互位置
        sift(data, 0, i-1)
    return data
```