# 基础算法

## 快速排序

```
def putpivot(arr,i,j):
    pivot = arr[i]
    iturn = 0
    jturn = 1
    while(i < j):
        if jturn:
            if arr[j]<pivot:
                arr[i],arr[j] = arr[j],arr[i]
                iturn = 1
                jturn = 0
            elif arr[j]>=pivot:
                j = j -1
        if iturn:
            if arr[i]>pivot:

                arr[i],arr[j] = arr[j],arr[i]
                iturn = 0
                jturn = 1
            elif arr[i]<=pivot:
                i = i + 1
    arr[i] = pivot
    return i

def quickpx(arr,low,high):
    if low<high:
        index = putpivot(arr,low,high)
        quickpx(arr,low,index-1)
        quickpx(arr,index+1,high)

print(quickpx(arr,0,14))
```

# 冒泡排序

```
arr = ARR
for i in range(len(arr),0,-1):#一共要遍历len(arr)-1遍
    for j in range(0,i-1):#从前往后遍历一遍 找出每趟最大的
        if arr[j]>arr[j+1]:
            temp = arr[j]
            arr[j] = arr[j+1]
            arr[j+1] = temp
print(arr)
```

# 插入排序

```
for i in range(0,len(arr)):#遍历列表，每次将元素插入之前已有序的列表中合适的位置
    for j in range(i,1,-1):
        if arr[j]<arr[j-1]:
            arr[j], arr[j-1] = arr[j-1],arr[j]
print(arr)
```

# 选择排序

```
for i in range(0,len(arr)):
    temp_min = i
    for j in range(i,len(arr)):#每趟选一个最小的，与这一趟的第一个元素交换
        if arr[j]<arr[temp_min]:
            temp_min = j
    temp = arr[i]
    arr[i]=arr[temp_min]
    arr[temp_min] = temp

print(arr)
```

# 堆排序

```
arr = [0]+[3,2,1,5,6,4]
def BuildMaxHead(A,l):
    for i in range(l//2,0,-1):
        HeadAdjust(A,i,l)

def HeadAdjust(A,k,l):#调整A数组以k为根的子树为大根树
    A[0] = A[k]       #A[0]存放根的值
    i = 2*k           #i为k的左子树
    while i<=l:
        if i == l:
            pass
        elif A[i]<A[i+1]:#让i指向两个子节点的较大的那个
            i = i + 1
        if A[0]>A[i]:
            break
        else:
            A[k] = A[i]
            k = i   #k指向其大的那个子树的根，小元素不断下降
        i = i *2
    A[k] = A[0]

n = len(arr)-1
BuildMaxHead(arr,n)
print(arr)
arr[1],arr[6] = arr[6],arr[1]
for i in range(5,0,-1):
    HeadAdjust(arr,1,i)
    arr[1],arr[i] = arr[i],arr[1]
```

# 归并排序

```
def merge(A,low,mid,high):
    #A[low,mid]、A[mid,high+1]都是有序数列
    #将A[low,mid]和A[mid,high+1]归并
    for i in range(low,high+1):
        B[i] = A[i]

    i = low
    j = mid+1
    k = low

    while (i<=mid) and (j<=high):
        if B[i]<= B[j]:
            A[k] = B[i]
            k = k + 1
            i = i + 1
        else:
            A[k] = B[j]
            k = k + 1
            j = j + 1

    while i<=mid:
        A[k] = B[i]
        k = k + 1
        i = i + 1
    while j <=high:
        A[k] = B[j]
        k = k + 1
        j = j + 1

def MergeSort(arr,low,high):
    mid = (low + high)//2
    if low<high:
        MergeSort(arr,low,mid)
        MergeSort(arr,mid+1,high)
        merge(arr,low,mid,high)
B = [0]*15
MergeSort(arr,0,14)
```