**希尔排序：**

**1//**取一个整数d1 = n/2,，将元素分为d1个组，每组相邻距离为d1,在各组进行直接插入排序

**2//**取第二个整数d2=d1/2，重复上述分组排序过程，指导di = 1

**3//**使整体数据越来越接近有序

def insertSortGap(dataSet , gap):  
 for i in range(1, len(dataSet)): #i 表示摸到的牌  
 temp = dataSet[i]  
 j = i - gap #j 表示手里的牌  
 while j >= 0 and dataSet[j] < temp:  
 dataSet[j+gap] = dataSet[j]  
 j -= gap  
 dataSet[j+gap] = temp  
 print(dataSet)  
  
def shellSort(dataSet ):  
 d = len(dataSet)//2  
 while d >= 1:  
 insertSortGap(dataSet , d )  
 d //= 2

**计数排序：**

**已知范围，把等于n的数计数出来再排序**

def countSort(dataSet , max\_count = 100):  
 count = [0 for \_ in range(max\_count+1)]  
 for val in dataSet:  
 count[val] += 1  
 dataSet.clear()  
 for ind, val in enumerate(count):  
 for i in range(val):  
 dataSet.append(ind)  
  
import random  
dataSet = [random.randint(0,100) for \_ in range(1000)]  
print(dataSet)  
countSort(dataSet)  
print(dataSet)

**桶排序：**

**计数排序分多个区域**

def bucketSort(dataSet , n = 100 ,max\_num = 10000):  
 buckets = [[] for \_ in range(n)]  
 #创建桶  
 for var in dataSet:  
 i = min( var // (max\_num // n ), n - 1 )  
 buckets[i].append(var)  
 #i 表示var放到几号桶里  
 for j in range(len(buckets[i]) - 1, 1, -1):  
 if buckets[i][j] < buckets[i][j - 1]:  
 buckets[i][j], buckets[i][j - 1] = buckets[i][j - 1], buckets[i][j]  
 else:  
 break  
 # 保持桶内的顺序  
 sortedList = []  
 for buc in buckets:  
 sortedList.extend(buc)  
 return sortedList

**基数排序：**

多关键字排序：加入现在有一个员工表，要求按照薪资排序，年龄相同的员工按照年龄排序

按位数分桶在装桶，多少位就多少次分桶

def radixSort(dataSet ):  
 max\_num = max(dataSet)  
 it = 0  
 while 10 \*\* it <= max\_num:  
 buckets = [[] for \_ in range(10)]  
 for var in dataSet:  
 digit = (var // 10 \*\* it) % 10  
 buckets[digit].append(var)  
 #分桶完成  
 dataSet.clear()  
 for buc in buckets:  
 dataSet.extend(buc)  
 #重新写回dataSet  
 it += 1 #位数分桶