shellsort_test.py

```python
import random
import copy
import time
import matplotlib.pyplot as plt
import numpy as np

a = [8,27,1,2,39,36,14,19,4,30]

def bubble_sort(list_org):
    a_deep = copy.deepcopy(a)
    Change = True
    while Change == True:
        Change = False
        for elem in range(len(a_deep)-1): #リストの中の数だけループされる
            if a_deep[elem] > a_deep[elem+1]: #もし左の数より右の数が小さければ実行される
                a_deep[elem], a_deep[elem+1] = a_deep[elem+1], a_deep[elem] #左右のリストの要素を変える
                Change = True #すべての数で左から小さい順に並ばない限り、ループさせる為にTrueを変えさせる


def merge_sort(a):
    if len(a) > 1:
        mid = len(a) // 2 #分割を行う
        left = a[:mid]
        right = a[mid:]
        left = merge_sort(left) #再帰を利用して再度分割していく
        right = merge_sort(right) #再帰を利用して再度分割していく
        merged = merge(left, right)
        return merged #mergedに結合しているのでそれをreturnでmergeを再帰する
    return a

def merge(left, right):
    merged = []
    l_i, r_i =0, 0

    while l_i < len(left) and r_i < len(right): #ソート済みの配列をマージする為にそれぞれ左から見ていく
        if left[l_i] <= right[r_i]:
            merged.append(left[l_i])
            l_i = l_i + 1
        else:
            merged.append(right[r_i])
            r_i = r_i + 1

    # 上のwhile文のどちらかがFalseになった場合終了する為、あまりをextendで追加する
    if l_i < len(left):
        merged.extend(left[l_i:])
    if r_i < len(right):
        merged.extend(right[r_i:])
    return merged

def insertion_sort(a):
    a_deep = copy.deepcopy(a)
    for i in range(len(a_deep)):
        j = i
        while j >= 1 and a_deep[j-1] > a_deep[j]: #jが0以下になるか、elemがlist_copied[i]より小さかったら実行
            a_deep[j], a_deep[j-1] = a_deep[j-1], a_deep[j]
            j -= 1


def shell_sort(a):
    a_deep = copy.deepcopy(a)
    gap = len(a_deep)//2
    while gap > 0:
        for i in range(gap, len(a_deep)):
            j = i
            while j >= gap and a_deep[j-gap] > a_deep[j]:
                a_deep[j], a_deep[j-gap] = a_deep[j-gap], a_deep[j]
                j -= gap
        gap //= 2
    return a_deep

def qsort(a, l, r):
    if l >= r:
```

```
        return

    #pivot = a[l + int((r-l)/2)]
    pivot = a[r]

    i,j =l, r #リストの1つ目の値をi, 後ろから一つ目をjとする

    while i < j: #i(最初は0が入る)がjより小さいと入る
        while a[i] < pivot: #a[i]がより小さかったらi+1
            i += 1
        while a[j] > pivot: #j番目の値がpivotより小さければpivotより、小さくなるまで探す。
            j -= 1
        if i < j:
            if a[i] == a[j]:
                j -= 1
            else:
                a[i], a[j] = a[j], a[i] #iとjが違う値だった場合、iとjの値を変える。(マイナスの値がない場合は、i=0なので最初に来る。)
    qsort(a, l, i-1) #i-1番目の値がjになり、再度繰り返される。
    qsort(a, j+1, r) #j+1番目の値がiになり、再度繰り返される。

num = 2
qtimes1 = []
qtimes2 = []
qtimes3 = []
qtimes4 = []
qtimes5 = []
nums = [10**i for i in range(num)]
for i in range(num):
    a = np.random.rand(10**i) #10**1以上、10**i未満
    length = len(a)
    start1 = time.time()
    bubble_sort(a) #aに関してbubble_sortで呼び出す
    end1 = time.time()
    start2 = time.time()
    merge_sort(a) #aに関してmerge_sortで呼び出す
    end2 = time.time()
    start3 = time.time()
    insertion_sort(a) #aに関してinsertion_sortで呼び出す
    end3 = time.time()
    start4 = time.time()
    shell_sort(a) #aに関してshell_sortで呼び出す
    end4 = time.time()
    start5 = time.time()
    qsort(a, 0, length-1) #aに関してq_sortで呼び出す
    end5 = time.time()
    qtimes1.append(end1 - start1)
    qtimes2.append(end2 - start2)
    qtimes3.append(end3 - start3)
    qtimes4.append(end4 - start4)
    qtimes5.append(end5 - start5)

plt.title("Sort Time Comparison") #タイトル
plt.xlabel("# of date") #x軸ラベル
plt.ylabel("Time") #y軸ラベル
plt.plot(nums, qtimes1, marker = "o", label = "bubble")
plt.plot(nums, qtimes2, marker = ".", label = "merge")
plt.plot(nums, qtimes3, marker = ">", label = "insertion")
plt.plot(nums, qtimes4, marker = "^", label = "shell")
plt.plot(nums, qtimes5, marker = "v", label = "q")
plt.legend() #凡例表示
plt.show()
```