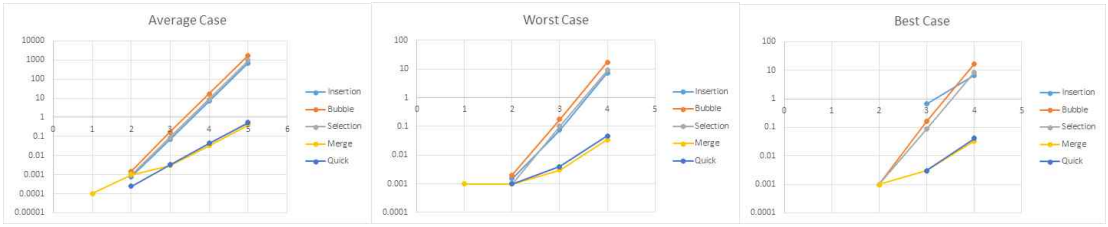
알고리즘의 기초 assignment1

전기정보공학부
2018-17824
손창대



먼저 가로축과 세로축이 모두 log 눈금단위임에 유의하자. n=10,100,1000일때는 Elapsed Time이 모두 적어 비교가 쉽지않지만, 전반적으로 Bubble Sort가 가장 오래걸렸고, 그 다음으로 Selection Sort, 그 다음으로 Insertion Sort, 그 다음으로 Quick Sort, 마지막으로 Merge Sort가 제일 시간이 적게 걸렸다. n=10000이상부터는 Merge와 Quick Sort를 제외한 3개의 Sorting은 시간이 제법 걸렸고, 이는 Average Case의 경우가 O(n^2)이기 때문으로 보인다. Insertion Sort와 Bubble Sort의 경우 Best Case의 경우 O(n)이지만, 이런 경우가 생기기가 쉽지 않기 때문에 이번 과제에서는 찾아볼 수 없었다. 마지막으로 n=100000인 경우 앞의 세 Sort는 걸리는 시간이 너무 길어 여러번 실행하여 Worst, Avg, Best를 모두 산출하지 못했다. 따라서 한번만 실행하여 Avg경우의 그래프에 첨부하였다. 앞의 세 경우는 10분이상걸리며 O(n^2)이 얼마나 오래걸리는지 확인할 수 있었다. Merge 및 Quick Sort는 여전히 1초미만의 경과시간을 보여주며 O(nlogn)임을 확인할 수 있었다. 아래는 Test에 쓰인 코드이다.

test.py - C:₩Users₩chang₩Desktop₩assignment1₩test.py (3.6.7)

File Edit Format Run Options Window Help

```
import numpy as np
import time
from insertion import *
from bubble import *
from selection import *
from merge import *
from quick import *

n = [100000]
#n = [10,100,1000]
test_time = 1
for n_idx in range(0,len(n)):
    InsertionTime = []
    BubbleTime = []
    SelectionTime = []
    MergeTime = []
    QuickTime = []
    for t in range(0,test_time):
        x = np.arange(n[n_idx])
        x=x+1
        np.random.shuffle(x)
        start = time.time()
        InsertionSort(x)
        end = time.time()
        InsertionTime.append(end-start)

        x = np.arange(n[n_idx])
        x=x+1
        np.random.shuffle(x)
        start = time.time()
        BubbleSort(x)
        end = time.time()
        BubbleTime.append(end-start)

        x = np.arange(n[n_idx])
        x=x+1
        np.random.shuffle(x)
        start = time.time()
        SelectionSort(x)
        end = time.time()
        SelectionTime.append(end-start)

        x = np.arange(n[n_idx])
        x=x+1
        np.random.shuffle(x)
        start = time.time()
        MergeSort(x,0,n[n_idx]-1)
        end = time.time()
        MergeTime.append(end-start)

        x = np.arange(n[n_idx])
        x=x+1
        np.random.shuffle(x)
        start = time.time()
        QuickSort(x,0,n[n_idx]-1)
        end = time.time()
        QuickTime.append(end-start)
        print("%d/%d cycle finished..."%(t+1,test_time))

    print("##########################################")
    print("###        n = %d case finished         ###"%(n[n_idx]))
    print("##########################################")
    print("Insertion Sort : ")
    print("Best Case : %.8f"%(min(InsertionTime)))
    print("Average Case : %.8f"%(sum(InsertionTime)/len(InsertionTime)))
    print("Worst Case : %.8f"%(max(InsertionTime)))
    print("##########################################")
    print("Bubble Sort : ")
    print("Best Case : %.8f"%(min(BubbleTime)))
    print("Average Case : %.8f"%(sum(BubbleTime)/len(BubbleTime)))
    print("Worst Case : %.8f"%(max(BubbleTime)))
    print("##########################################")
    print("Selection Sort : ")
    print("Best Case : %.8f"%(min(SelectionTime)))
    print("Average Case : %.8f"%(sum(SelectionTime)/len(SelectionTime)))
    print("Worst Case : %.8f"%(max(SelectionTime)))
    print("##########################################")
    print("Merge Sort : ")
    print("Best Case : %.8f"%(min(MergeTime)))
    print("Average Case : %.8f"%(sum(MergeTime)/len(MergeTime)))
    print("Worst Case : %.8f"%(max(MergeTime)))
    print("##########################################")
    print("Quick Sort : ")
    print("Best Case : %.8f"%(min(QuickTime)))
    print("Average Case : %.8f"%(sum(QuickTime)/len(QuickTime)))
    print("Worst Case : %.8f"%(max(QuickTime)))
    print("##########################################")
```

Ln: 87 Col: 56