**Algorithm Assignment 6**

**Quick Sort (Recursion version, Sequential version)**

**2015003654 Lee Hoyoung**

1. Code explanation

main.c – 파일 입력을 통해 데이터를 받고 sorting 방법을 선택할 수 있는 소스

quickSort.c – quick sort의 recursion version과 sequential version 그리고 partition을 구현한 소스

input\_i, input\_c: 과제 pdf에 나온 데이터들을 적은 텍스트 파일

output\_i, output\_c: 위의 입력에 대한 출력 텍스트 파일

real\_data: 전 세계 약 5개월 간의 코로나 바이러스 확진자 증가 감소 인원 수

real\_data2: 2006년 가장 빈번하게 사용된 상위 10000개의 단어 데이터

output1, output2: 그에 대한 정렬 결과

real\_data3: 2006년 가장 빈번하게 사용된 상위 10000개의 단어들의 사용빈도

output3: 그에 대한 정렬 결과

소스코드는 수업자료와 Introduction to Algorithms 3rd 를 참고하였습니다.

코드 구현은 Visual Studio Code 1.49.3 version에서 구현하였습니다.

1. 가설 설정

가설 1: real\_data3는 가장 많이 사용된 단어의 빈도수부터 내림차순으로 정렬되어 있는 값이기 때문에 merge sort의 수행시간이 제일 짧을 것이고 quick sort는 O(n^2)으로 insertion, selection, bubble sort와 수행시간이 비슷할 것이다.

가설 2: data의 개수가 많아지면 recursion이 많이 발생하게 되어 quick sort의 sequential version이 더 좋은 성능을 보일 것이다.

가설 3: data들이 무작위로 흩어져 있는 경우에는 O(nlogn)인 merge sort와 quick sort의 수행 시간은 차이가 거의 없을 것이다.

1. Data 결과 및 가설 검증

1번 가설(내림차순으로 정렬된 10000개의 숫자 데이터)

Insertion: 287ms

Bubble: 410ms

Selection: 230ms

Merge (recursion, sequential): 1ms

Quick(recursion): 298ms

Quick(sequential): 248ms

가설처럼 quick sort는 O(n^2)의 성능을 보여주었고 selection sort는 내림차순 정렬이 worst case가 아니기 때문에 O(n^2) 중에 제일 좋은 성능을 보여주었다.

Merge sort는 다른 것들보다 200배 이상의 성능을 보여주었고 O(nlogn)의 성능이 O(n^2)보다 훨씬 좋음을 나타냈다.

2번 가설(10000개 역순 정렬 숫자, 4만 개 랜덤 배열 숫자, 7만개 랜덤 배열 문자)

Data3

Quick(recursion): 298ms

Quick(sequential): 248ms

Data1

Quick(recursion): 308ms

Quick(sequential): 312ms

Data2

Quick(recursion): 432ms

Quick(sequential): 564ms

데이터가 10000~70000은 그렇게 차이가 나지 않는다. 더 큰 데이터와 비교를 해야 차이가 날 것 같다. 그리고 recursion과 sequential의 수행시간은 데이터의 양과는 크게 관련이 없는 것 같고 랜덤 배열된 곳에서는 sequential이 더 느렸고 역순으로 정렬된 곳에서는 recursion이 더 느렸다.

3번 가설

Data1

Quick(recursion): 308ms

Quick(sequential): 312ms

Merge(recursion): 22ms

Merge(sequential): 13ms

Data2

Quick(recursion): 432ms

Quick(sequential): 564ms

Merge(recursion): 16ms

Merge(sequential): 19ms

두 data 모두 quick sort가 merge sort보다 느리게 나왔다.

데이터의 분포로 예상을 해보자면 문자는 알파벳들로 전체 집합이 52개(대소문자)이기 때문에 정렬하다보면 같은 알파벳이 정렬된 경우가 나오기 때문에 sorting 중간에 어느 구간에서는 정렬된 데이터를 sorting하게 되고 그래서 merge sort보다 느려진 게 아닐까한다. 숫자 데이터는 0에서 10까지의 수가 많은 부분을 차지해서 정렬 도중에 이미 어느정도 정렬된 데이터를 다시 정렬해야해 느려지지 않을까한다.