CSE3081_MP2_20200901 REPORT

20200901 이효주

(1) Comparison ALGORITHM

먼저, 내가 사용한 algorithm1~4까지는 아래와 같다.

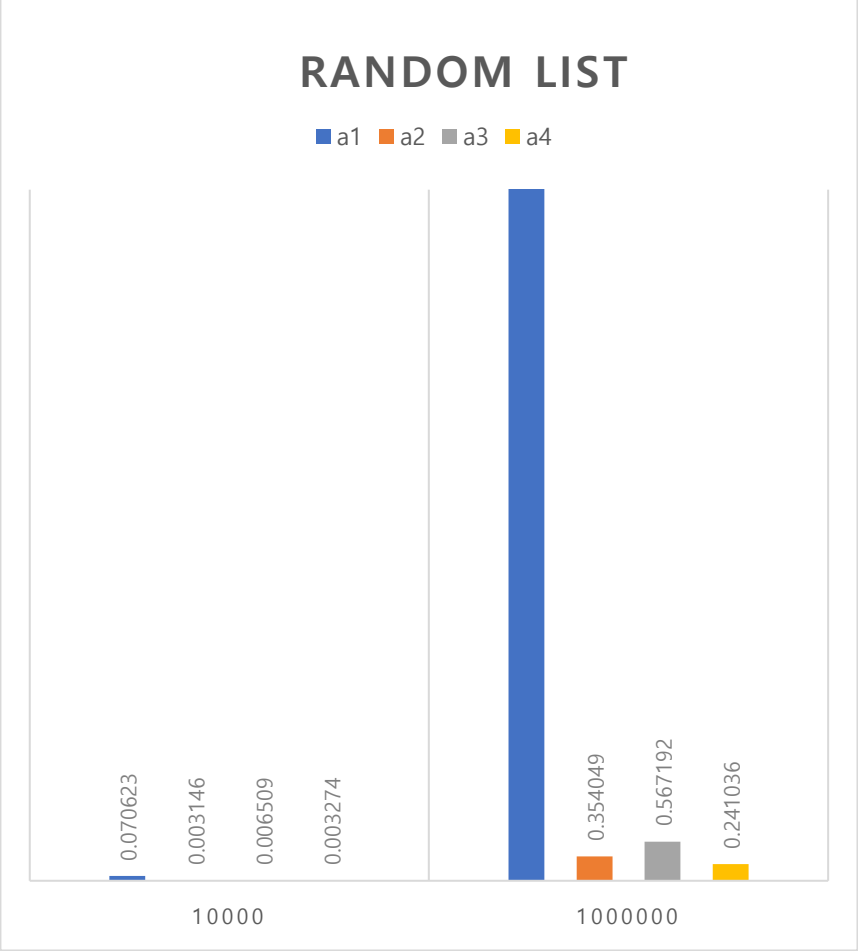
Algorithm1 - Insertion Sort

Algorithm2 – quick sort

Algorithm3 – Heap Sort

Algorithm4 – Intro Sort

1) Random list에 대한 정렬 비교



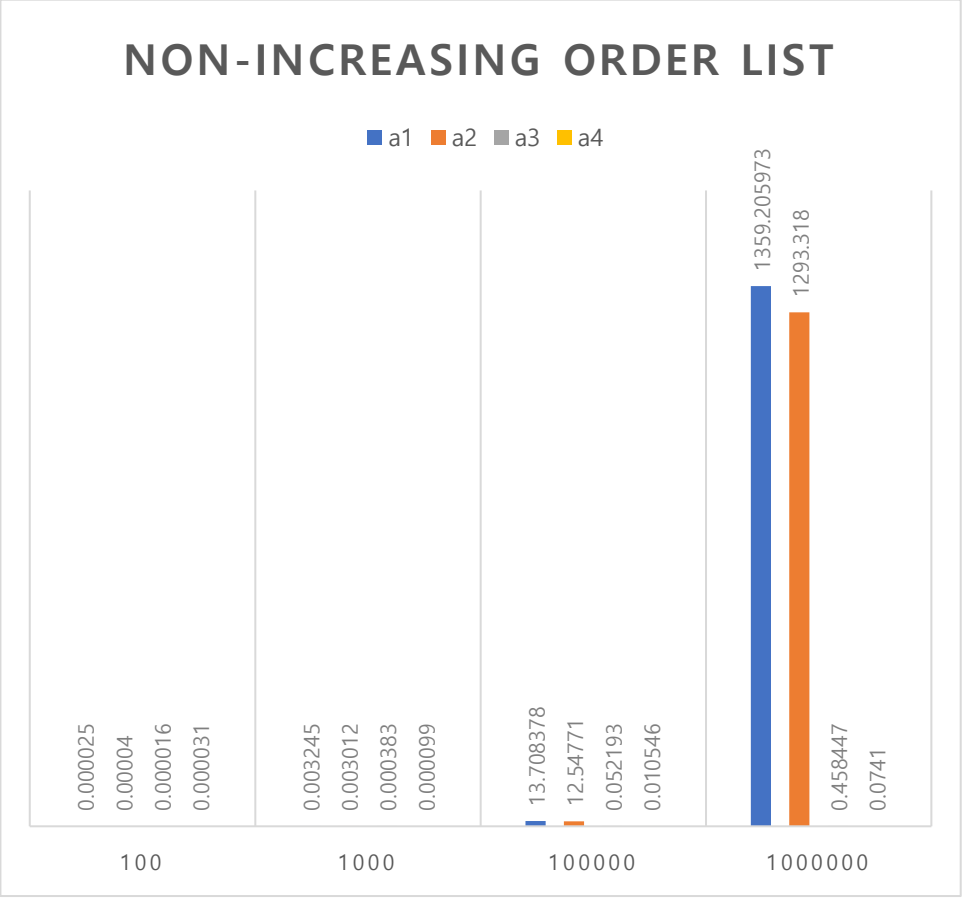
100개부터 1000000개 까지의 Random list들을 input size마다 5개씩 만들어, 평균 시간을 비교하였다.

Input size가 커질수록 insertion sort의 running time은 시간이 아주 오래 걸렸다.

이외의 algorithm2, 3, 4의 시간은 algorithm1보다 훨씬 빠르게 작동하였다. 나머지 3개를 비교한 결과, algorithm 4 < 2 < 3의 순서로, quick sort가 heap sort보다 평균적으로 빠르게 작동하였다.

따라서, random list를 정렬할 때 input size가 클수록, 시간 복잡도는 insertion sort > heap sort > quick sort > algorithm 4(intro sort) 순으로 작동하였다.

2) non-increasing order list에 대한 정렬 비교



Input size 100, 1000, 100000, 1000000의 Non-increasing order list를 만들어, running time을 비교하였다.

Algorithm 1, 2는 실행 시간이 1359s, 1293s로 아주 오래 걸렸다. 그리고 algorithm3, 4는 algorithm 1, 2에 비해 빠르게 작동하였다.

따라서, non-increasing order list를 정렬할 때 input size가 클수록, 시간 복잡도는 insertion sort > quick sort > heap sort > algorithm 4(intro sort) 순으로 작동하였다.

(2)

<Experiment environment>

(1) CPU speed

CPU MHz: 3600.000

CPU max MHZ: 3600.0000

(2) Memory size

MemTotal: 16725240kB

(3) OS

Ubuntu 20.04.2 LTS

<Experiment setup>

insertionSort, quickSort, heapSort는 모두 Algorithm4 (IntroSort)에서 사용되어서 부분적으로 정렬할 수 있도록 배열 arr와 인덱스 left와 right값을 파라미터로 주었다.

IntroSort에 대한 설명:

리스트의 크기가 32보다 작은 경우 삽입 정렬 수행하고, 그렇지 않은 경우 전체 리스트에 대해 퀵정렬을 수행한다. (이 때 피벗 값은 중앙) 수행하면서 퀵정렬의 재귀호출 깊이가 2logN이 넘어가게 되면 정렬방식을 바꾼다. 재귀호출 상 나누어진 배열의 크기가 32보다 크면 힙정렬을 수행하고 그렇지 않으면 정렬을 수행하지 않는다. 정렬이 끝나면 전체 리스트에 대해 삽입정렬을 다시 수행한다. (거의 정렬이 되어있는 상태)

데이터가 적은 경우 삽입 정렬이 퀵 정렬보다 빠르고(기준을 32로 잡음) 데이터가 정렬이 거의 된 상태에서도 퀵정렬보다 삽입 정렬이 빠르다는 것을 이용한다. 퀵 정렬이 재귀호출의 깊이가 2lonN까지만 수행되기 때문에 최악의 경우에도 O(n^2)이 나오지 않는다.

My Opinion about this experience:

Input size가 작을 때는, insertion sort를 써도 시간이 그렇게 오래 걸리지 않기 때문에 어떤 알고리즘을 쓰는 것이 좋다는 것을 비교하기가 어려웠다. Input size가 아주 작을 때는 quick sort가 정말 빨리 걸렸다. (random한 상황에서)

그리고 input size가 커지면 insertion sort는 input.txt를 읽고, sorting하는데 아주 오랜 시간이 걸려서 결과 파일을 얻기 위해 아주 오래 기다려야 했다.

특히 worst case(non-increasing order)에서는, quick sort가 heap sort보다 아주 오래 걸리며, insertion sort와 비슷한 시간이 걸림을 알 수 있었다. 그러나 random list에서, input size가 클 때는 insertion sort를 제외하고 빠르게 작동하는 편이였고, 평균적으로 heap sort보다는 quick sort가 더 빠른 시간 내에 작동함을 알 수 있었다.