**CSE3081-2: Design and Analysis of Algorithms (Fall 2019)**

**Machine Problem 2: Master of Sorting**

학과 : 컴퓨터공학 학번 : 20181668 이름 : 이예진

**1. Experiment environment**

\*The hardware specification of the machine

-CPU speed : 1.80GHz (8CPUs), ~2.0GHz

-Memory size : 8192MB RAM (8GB)

-OS type and version : Windows 10 Home 64비트 (10.0, 빌드 18362)

-개인 노트북(LG전자 올뉴그램 15Z980-GA70K, 코어i7-8550U)으로 모든 실험을 진행했다.

-Linux 서버에서 개인 계정으로 모든 실험을 진행했다.

**2. Experiment setup**

(1) 변수

-int\* list : 입력 파일에 존재하는 배열을 저장하기 위해 사용한 정수형 배열 변수이다. 동적 할당을 사용하여 주어진 크기만큼의 공간만 사용하도록하였다.

-int listsize : 입력 파일에 존재하는 배열의 크기를 저장하기 위해 사용하였다.

(2) 함수

-algorithm\_1 : insertion 정렬 알고리즘을 구현하였다.

-algorithm\_2 : quick 정렬 알고리즘을 구현하였다.

-algorithm\_3 : heap 정렬 알고리즘을 구현하였다.

-algorithm\_4 : pivot을 2개 사용한 quick 정렬 알고리즘을 구현하였다. count가 0이 된 경우, 입력 크기가 45보다 작거나 같은 경우에는 삽입 정렬을 이용하도록 하고 나머지 경우에는 힙 정렬을 이용하도록 알고리즘을 수정하였다.

(3) 단위

-알고리즘 시간 측정

알고리즘을 끝낸 시간을 저장한 변수 finish에서 알고리즘을 시작한 시간을 나타내는 변수 start를 뺀 다음 CLOCKS\_PER\_SEC로 나누어 seconds 단위로 나타냈다. double형으로 형변환을 한 다음 출력하도록 했다.

(4) The range of input size

-Random input : 음수와 0을 포함하여 배열을 생성했다. 배열의 개수는 각각 3개, 100개, 10000개, 1000000개를 사용하였다.

-Decreasing input : 0에서 n-1 범위의 내림차순 배열을 생성했다. 배열의 개수는 각각 3개, 100개, 10000개를 사용하였다.

-Increasing input : 1에서 n 범위의 오름차순 배열을 생성했다. 배열의 개수는 각각 3개, 100개, 10000개를 사용하였다.

**3. 실험**

(1) 실험 목적

동일한 문제에 대하여 동일한 결과를 생성하는 4가지의 서로 다른 정렬 알고리즘들이 어떻게 다르게 작동하는 지를 보인다. 4번째 알고리즘은 자신만의 효율적인 정렬 알고리즘을 설계한다. 각 정렬 알고리즘의 수행 시간은 입력의 크기가 커질수록 증가한다.

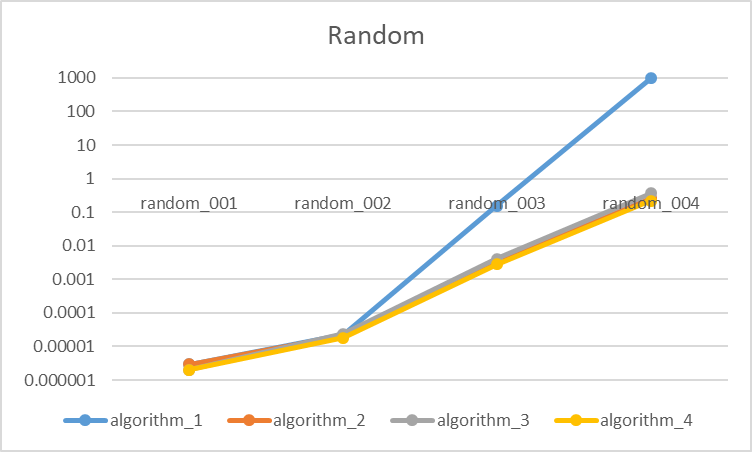
(2) 실험 결과

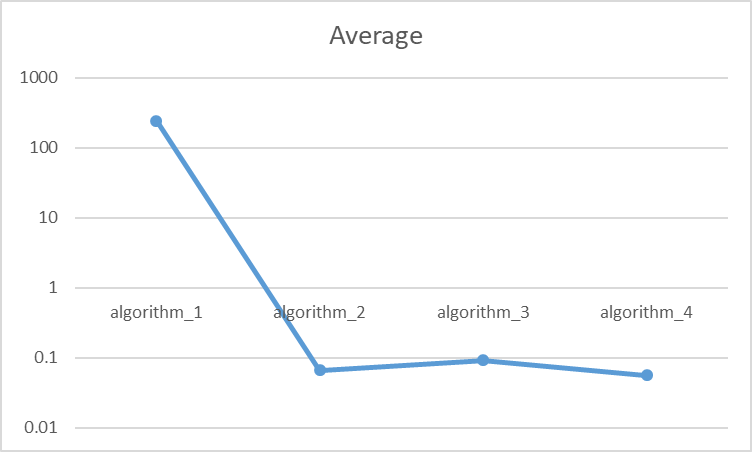
-수행 시간의 단위는 seconds이다.

-Random, Decreasing, Increasing 으로 종류를 나누어 input 파일들을 실험했다. 모든 경우에서 입력 크기가 증가할수록 실행 시간도 증가하는 경향을 보였다.

-Random : 음수와 0, 양수를 모두 포함하여 입력 파일을 작성했다. 배열의 모든 원소들은 무작위로 생성하였다.

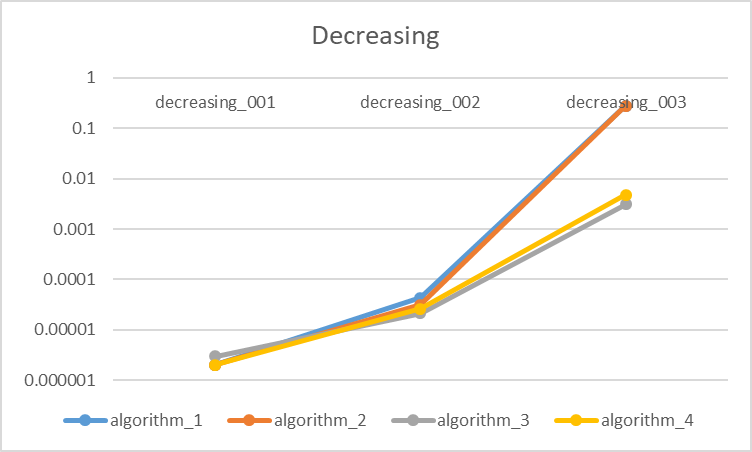






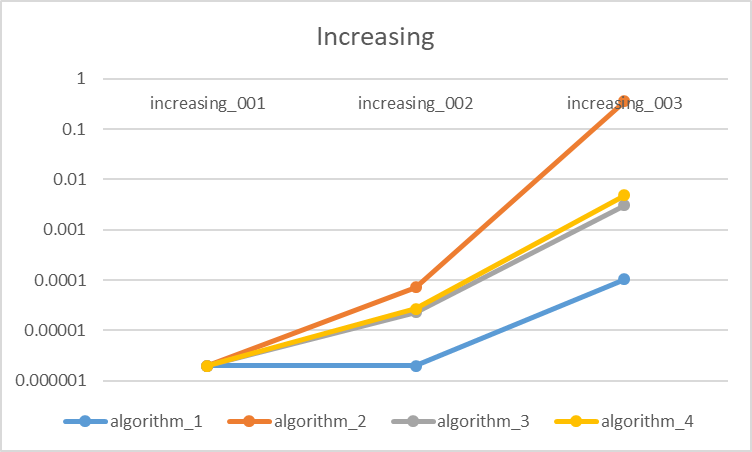
-Decreasing : non-increasing 으로 저장된 배열을 입력 파일로 이용했다.





-Increasing





(3) 4번째 정렬 알고리즘

기존의 quick 정렬은 pivot을 하나만 사용하였지만 이를 응용하여 4번째 정렬 알고리즘에서는 pivot을 left pivot과 right pivot으로 나누어서 두 개를 사용하였다. (left pivot < right pivot) 이 경우 역시 worst case의 경우 시간복잡도가 O(n2)일 수 있지만 pivot을 2개 사용했기 때문에 worst case가 될 수 있는 확률을 줄였다.

또한 count 라는 변수를 선언하여 일정 크기(depthlimit) 이상의 재귀함수가 호출되었다면 정렬 방법을 바꾸도록 했다. 입력의 크기가 45보다 작거나 같은 경우 insertion 정렬을 이용하여 시간을 단축할 수 있었다. 이러한 경우에는 배열 원소의 개수가 작기 때문에 insertion 정렬을 사용하는 것이 더 효율적일 확률이 더 높기 때문이다. 그 이외의 경우에는 주어진 시작 원소와 마지막 원소 사이의 크기를 이용하여 heap 정렬을 수행하도록 했다. heap 정렬은 worst case에 대하여 시간복잡도가 O(nlog n)을 유지한다. 따라서 quick 정렬에서 일정 크기 이상의 재귀 함수 호출이 반복되었을 때 worst case인 O(n2)이 되는 것을 방지할 수 있고 실행 시간을 단축시킬 수 있다.

실험을 통해 음수와 0, 양수를 모두 포함하는 정수형 배열을 정렬할 때는 pivot을 2개 사용한 quick 정렬을 사용하는 것이 가장 안정적이고 빠르다는 사실을 알게 되었다.

**4. Your comments on the experience**

배열의 개수가 큰 배열일수록 O(n2)의 시간복잡도를 가진 정렬 알고리즘보다 O(nlog n)의 시간복잡도를 가진 정렬 알고리즘이 훨씬 효율적이었다. 배열의 개수가 증가할수록 모든 정렬 알고리즘의 수행 시간도 같이 증가하는 경향을 보였다. 입력 크기가 매우 작은 경우에는 수행 시간이 같거나 역전되는 현상도 보였다. Decreasing 특성에 대하여 heap 정렬이, Increasing 특성에 대하여 insertion 정렬이 다른 정렬에 비해 빨랐다. 특성을 모르는 Random 배열에 대해서는 4번째 알고리즘이 다른 알고리즘에 비해 빨랐다.