[Data Structure 2022 Spring]
HW4 report

컴퓨터공학부
2017-18538
황선영

1. 정렬 알고리즘의 동작 방식

1.1. Bubble Sort

① 가장 큰 수를 찾아 맨 뒤 요소와 바꾸고 맨 뒤 요소를 관심 대상에서 제외한다.

② 이를 원소가 하나 남을 때까지 반복한다.

1.2. Insertion Sort

① 배열의 처음부터 시작한다. 길이를 1부터 시작하여 길이에 포함되는 요소들을 정렬한다.

② 길이를 늘려가며 관심 대상을 늘린다.

③ 배열의 끝까지 진행하여 정렬을 완료한다.

1.3. Heap Sort

① 주어진 배열을 힙으로 만든다. (max heap)

② 이후 원소를 하나씩 제거하며 percolateDown함수를 이용하여 수선한다.

③ 제거된 원소들은 각 단계에서 최댓값이므로 배열의 뒤부터 채워나가 정렬을 완료한다.

1.4. Merge Sort

① 입력을 반으로 나눈다.

② 전반부와 후반부를 각각 독립적으로 정렬한다.

③ 정렬된 두 부분을 합쳐서 정렬된 배열을 얻는다.

④ 이 작업을 재귀적으로 반복하여 정렬을 완료한다.

1.5. Quick Sort

① 기준 원소를 정하여 분할한다.

② 분할한 두 부분을 각각 독립적으로 정렬한다.

③ 이 작업을 재귀적으로 반복하여 정렬을 반복한다.

1.6. Radix Sort

① 모두 상수 k개 이하의 자릿수를 가진 경우 사용할 수 있는 방법으로, 우선 가장 낮은 자리수 만으로 모든 수를 정렬한다.

② 다음으로 둘째 자릿수를 대상으로 정렬한다.

③ 가장 높은 자릿수까지 정렬하여 정렬을 완료한다.

④ 이 과제에서는 양수와 음수를 따로 정렬한 후 합치는 과정을 추가하였다.

2. 동작 시간 분석

한 케이스에 대한 실험 횟수를 3회로 하고, data의 개수와 element의 range에 따른 정렬 시간을 측정하여 표로 나타내었다. N은 data의 개수를 나타낸다. (같은 배열에 대해 3회 이상 정렬하면 정렬 속도가 급격히 떨어지는 것을 감안하여, 횟수가 3회라는 것은 data 개수와 range는 같되 내용은 다른 배열 3개를 이용하여 정렬한다는 것을 의미한다.) 값의 단위는 ms이다.

N=1000	최댓값	최솟값	평균	표준편차
Bubble sort	9	5	6.67	2.08
Insertion sort	5	4	4.67	0.58
Heap sort	3	2	2.33	0.58
Merge sort	2	1	1.33	0.58
Quick sort	1	0	0.67	0.58
Radix sort	7	3	4.67	4.33

[table 1.] Data size=1000, Range -1000 ~ 1000

N=1000	최댓값	최솟값	평균	표준편차
Bubble sort	8	5	6	1.73
Insertion sort	5	3	4	1
Heap sort	2	1	1.67	0.58
Merge sort	3	1	1.67	1.33
Quick sort	1	1	1	0
Radix sort	9	8	8.67	0.33

[table 2.] Data size=1000, Range -10000000 ~ 10000000

N=10000	최댓값	최솟값	평균	표준편차
Bubble sort	162	155	159.33	14.33
Insertion sort	30	21	26.33	22.33
Heap sort	6	5	5.67	0.33
Merge sort	4	3	3.67	0.33
Quick sort	3	2	2.67	0.33
Radix sort	25	22	23.33	1.53

[table 3.] Data size=10000, Range -10000 ~ 10000

N=10000	최댓값	최솟값	평균	표준편차
Bubble sort	162	154	159	4.36
Insertion sort	30	21	26.67	4.93
Heap sort	8	7	7.33	0.33
Merge sort	6	5	5.33	0.33
Quick sort	6	3	4.33	2.33
Radix sort	28	24	26.67	5.33

[table 4.] Data size=10000, Range -10000000 ~ 10000000

N=100000	최댓값	최솟값	평균	표준편차
Bubble sort	19348	18545	18966	402.92
Insertion sort	1033	996	1017.33	19.14
Heap sort	17	16	16.67	0.33
Merge sort	18	18	18	0
Quick sort	14	13	13.33	0.33
Radix sort	79	76	77.33	2.33

[table 5.] Data size=100000, Range -100000 ~ 100000

N=100000	최댓값	최솟값	평균	표준편차
Bubble sort	19141	18874	18996	134.98
Insertion sort	1051	992	1022.33	29.54
Heap sort	17	16	16.67	0.33
Merge sort	18	18	18	0
Quick sort	14	13	13.67	0.33
Radix sort	102	99	100.67	2.33

[table 6.] Data size=100000, Range -10000000 ~ 10000000

Data size가 1000인 경우 수행 시간에 유의미한 차이를 보이지 못했다.

Data size가 10000인 경우 heap sort와 merge sort, quick sort의 수행시간이 비슷하였고 insertion sort와 radix sort의 수행시간이 비슷했고 bubble sort의 수행시간이 가장 느렸다. Range에 따른 영향은 미미했다.

Data size가 100000인 경우 insertion sort와 radix sort의 수행 시간의 차이가 두드러졌다. Quick sort가 가장 빨랐다. Bubble sort와 insertion sort는 같은 O(n^2)의 점근적 복잡도를 가지지만 상수 배의 차이가 커서 insertion sort가 bubble sort에 비해 월등히 빠른 수행 시간을 보였다. 마지막으로 range에 따라 radix sort에서 수행시간이 유의미하게 달라진 것을 확인할 수 있는데, 이는 자릿수에 따라 수행 시간이 결정되는 radix sort의 특성 때문이라고 추측할 수 있다.