20181360 물리학 전공 함형원

1. 알고리즘 비교

알고리즘을 비교하기 위해 입력의 크기 2^4, 2^7, 2^10, 2^14, 2^17, 2^20에 대해 수행 시간을 측정했다. 입력의 크기가 클 때, 지나치게 수행 시간이 긴 알고리즘은 측정하지 않았다. 3개의 random lists와 1개의 non-increasing list에 대해 시간을 측정했다.

알고리즘 1은 insertion sort로 worst-case time complexity가 O(n^2), average-case time complexity가 O(n^2)이다. random lists에 대해 측정한 시간은 다음과 같다.

Insertion Sort				
입력	1	2	3	평균
16	0.000004	0.000004	0.000004	0.000004
128	0.000039	0.000045	0.000055	0.000046
1024	0.002426	0.002444	0.002388	0.002419
16384	0.352737	0.353789	0.346691	0.351072
131072	15.124435	15.197015	15.098168	15.139873
1048576				

알고리즘 2는 quick sort로 worst-case time complexity가 O(nlogn), average-case time complexity가 O(n^2)이다. random lists에 대해 측정한 시간은 다음과 같다.

Quick Sort				
입력	1	2	3	평균
16	0.000005	0.000004	0.000004	0.000004
128	0.000026	0.000025	0.000025	0.000025
1024	0.000219	0.000239	0.000237	0.000232
16384	0.005056	0.005155	0.004758	0.004990
131072	0.037178	0.035359	0.036923	0.036487
1048576	0.250864	0.255659	0.251996	0.252840

알고리즘 3는 merge sort로 worst-case time complexity가 O(nlogn), average-case time complexity가 O(nlogn)이다. random lists에 대해 측정한 시간은 다음과 같다.

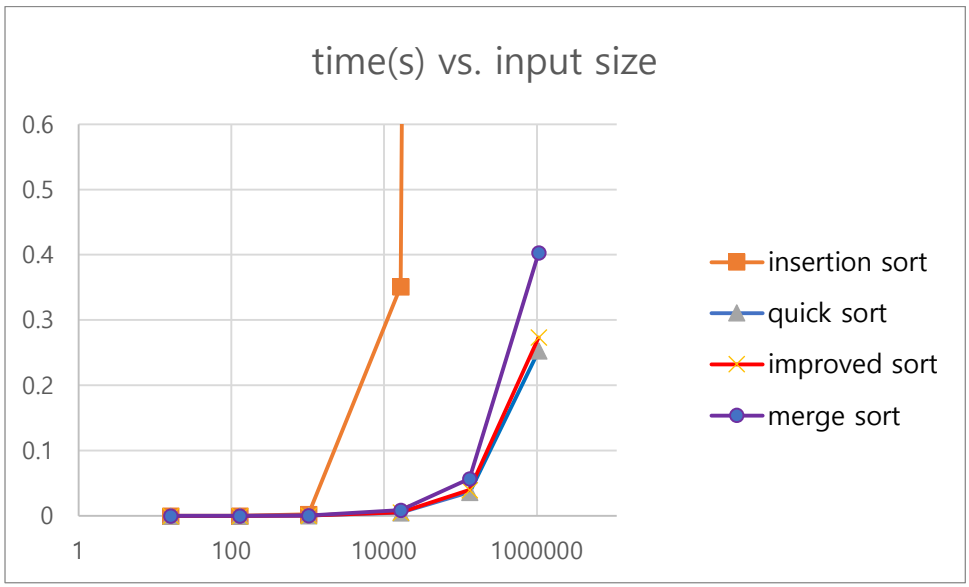
Merge Sort				
입력	1	2	3	평균
16	0.000011	0.000011	0.000010	0.000011
128	0.000056	0.000054	0.000055	0.000055

1024	0.000417	0.000417	0.000452	0.000429
16384	0.008742	0.008741	0.008723	0.008735
131072	0.055854	0.057674	0.057378	0.056969
1048576	0.402544	0.404702	0.401855	0.403034

알고리즘 4는 개선한 정렬로 worst-case time complexity가 O(nlogn), average-case time complexity가 O(n^2)이다. random lists에 대해 측정한 시간은 다음과 같다.

Improved Sort				
입력	1	2	3	평균
16	0.000005	0.000005	0.000003	0.000004
128	0.000031	0.000031	0.000030	0.000031
1024	0.000246	0.000242	0.000284	0.000257
16384	0.005477	0.005574	0.005403	0.005485
131072	0.039607	0.040821	0.038910	0.039779
1048576	0.288879	0.264297	0.267271	0.273482

각 정렬을 비교하기 위해 그래프로 나타냈다.

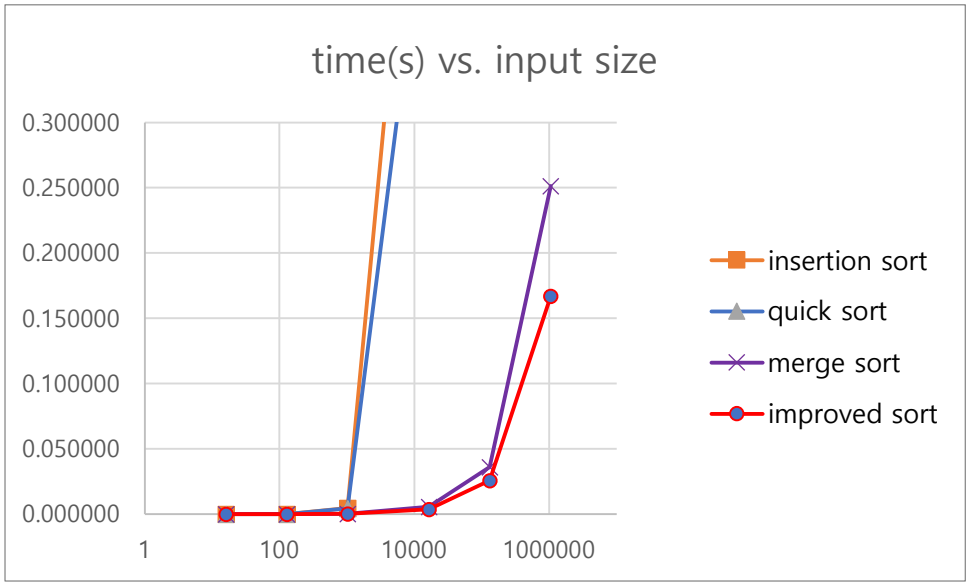


입력의 크기가 매우 작을 때(n = 16)일 때, 측정 시간이 merge sort >= quick sort >= insertion sort >= improved sort인 것을 확인할 수 있다. 입력의 크기가 클수록 insertion sort > merge sort > improved sort >= quick sort인 것을 확인할 수 있다. 데이터에 정리하지 않았지만 mp2의 크기 제한에서는 quick sort가 merge sort보다 빠르다. non-increasing list에 대해 측정한 시간은 다음과 같다.

	Algorithm			
입력	**1**	**2**	**3**	**4**

16	0.000005	0.000005	0.000009	0.000004
128	0.000078	0.000079	0.000039	0.000023
1024	0.004701	0.004318	0.000289	0.000199
16384	0.662934	0.493988	0.005632	0.003670
131072	31.193282	30.857263	0.036122	0.025743
1048576			0.251349	0.167097

각 정렬을 비교하기 위해 그래프로 나타냈다.



입력의 크기가 매우 작을 때(n = 16)일 때, 측정 시간이 merge sort > quick sort >= insertion sort >= improved sort인 것을 확인할 수 있다. 입력의 크기가 클수록 quick sort >= insertion sort > merge sort > improved sort인 것을 확인할 수 있다. non-increasing list는 insertion sort, quick sort의 worst-case이므로 수행 시간이 길 것으로 예측할 수 있고, 실험값도 이와 일치한다.

2. 알고리즘 4 설명

입력의 크기가 작을 때(n <= 16) insertion sort, 이외에는 quick sort를 사용한다. 입력의 크기가 클 때, merge sort를 사용할 계획이었지만 데이터의 경향을 보아 quick sort를 사용하는 것이 성능이 우수할 것으로 판단했다.

insertion sort는 입력의 크기가 매우 작을 때, 성능이 우수하므로 빠르게 정렬하기 위해 사용했다.

quick sort는 worst-case를 고려하여 pivot을 임의로 설정했다. 알고리즘 2에서 구현한 quick sort는 가장 오른쪽의 element를 pivot으로 사용하므로 non-increasing list가 worst-case가 된다. 이를 개선하기 위해 randomized_partition, randomized_quick_sort 함수를 구현하여 임의의 pivot

에 대해 quick sort를 한다. 또한, pivot을 기준으로 왼쪽, 오른쪽 중 길이가 작은 것에서 재귀적으로, 큰 것에서 반복적으로 수행한다. average-case에서 알고리즘 2의 quick sort보다 느리지만, worst-case에서 빠르다.