Machine Problem 2: Master of Sorting 결과 보고서

2022월 11월 13일
알고리즘 설계 및 분석 /소정민 교수님
20190018 김유이

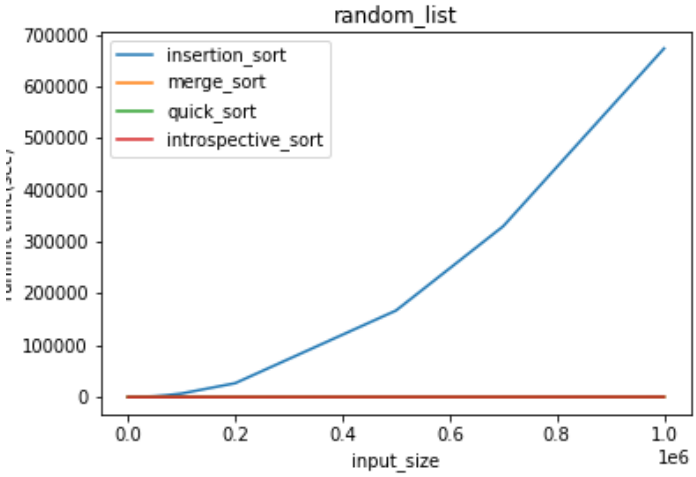
1. algorithm 4개 성능 측정 및 비교 결과

   1) random list

-experiment environment: linux (Ubuntu 20.04.2, cspro1.sogang.ac.kr)
4개의 알고리즘 모두 동일한 환경에서 구현 및 성능 측정을 진행했다.
-experiment setup: 성능 측정 기준은 argument 유효성 검사 이후, 파일 입력 시의 clock()과 정렬 결과를 반환 받을 때의 clock()의 차이를 측정하여 second 단위로 변환한 것을 기준으로 했다. range of input size는 10부터 1000000 까지의 값을 10, 20, 50, 70, 100, 200, 500, 700, 1000 등의 간격으로 측정하였다.
-comments on the experience: input의 크기가 10~100개일 때는 대체적으로 insertion sort의 running time이 introspective sort보다 빠르다. 100개 이상일 때는 introspective sort의 시간복잡도가 훨씬 빨라진다. Input의 크기가 1000개 미만일 때는 quick sort와 introspective sort를 비교해보면, running time이 introspective sort보다 조금씩 더 빠르다가 1000개 이상일 때는 in개의 알고리즘 중 가장 빠른 시간 복잡도를 가진다. Merge sort와 introspective sort를 비교해보면 input이 100개 이하일 때는 merge sort의 running time이 더 빠르고, 1000개 이상일 때는 introspective가 조금 더 빠르다가 100000개 이상일 때는 introspective가 훨씬 빨라진다. Input의 size가 작을 때, introspective sort와 나머지 3개 알고리즘의 running time은 대체적으로 큰 차이를 보이지 않고, input의 size가 커지면 introspective sort의 running time이 다른 3개의 알고리즘보다 훨씬 빠르다.

	runing time(sec)	insertion	merge	quick	introspective
input_size					
10		0.0745	0.1165	0.09	0.062
20		0.0675	0.0825	0.045	0.0935
50		0.086	0.06	0.0665	0.0735
70		0.073	0.068	0.063	0.1155

100		0.13	0.0865	0.0725	0.0965
200		0.184	0.1385	0.1185	0.1665
500		0.6535	0.262	0.2565	0.2505
700		1.0955	0.431	0.349	0.387
1000		2.021	0.632	0.513	0.487
2000		7.393	1.0985	1.0955	1.058
5000		28.6385	2.6835	2.8495	2.6255
7000		50.9915	4.683	3.8165	3.7275
10000		100.974	6.755	5.838	5.4845
20000		306.7735	10.977	12.0835	11.7025
50000		1710.11	23.1115	22.4935	21.805
70000		3383.955	30.6675	27.151	29.023
100000		6849.05	55.5685	42.041	39.6135
200000		26757.75	114.024	86.7115	82.914
500000		167161	262.1125	215.944	206.36
700000		329989.5	300.581	273.177	249.549
1000000		673542	516.819	403.142	371.962



2) non-increasing order list

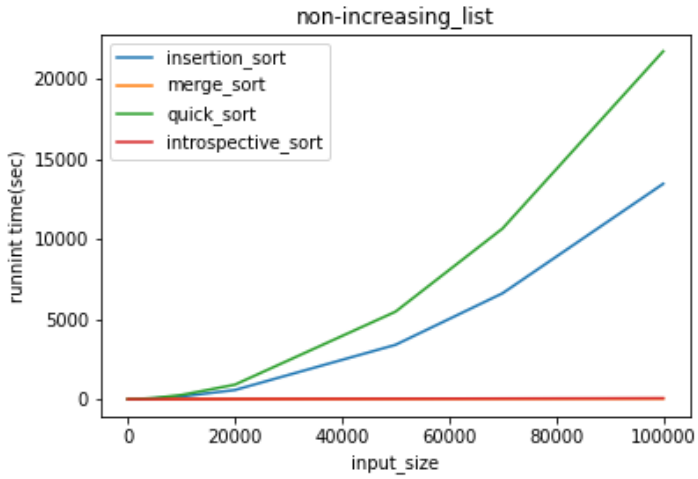
-experiment environment: linux (Ubuntu 20.04.2, cspro1.sogang.ac.kr)

4개의 알고리즘 모두 동일한 환경에서 구현 및 성능 측정을 진행했다.

-experiment setup: 성능 측정 기준은 argument 유효성 검사 이후, 파일 입력 시의 clock()과 정렬 결과를 반환 받을 때의 clock()의 차이를 측정하여 second 단위로 변환한 것을 기준으로 했다. range of input size는 10부터 100000 까지의 값을 10, 20, 50, 70, 100, 200, 500, 700, 1000 등의 간격으로 측정하였다.

-comments on the experience: insertion sort와 introspective sort의 성능을 비교했을 때, input size가 크든 작든 introspective sort의 running time이 훨씬 빠르다. Merge sort와 introspective sort의 성능을 비교했을 때, merge sort의 running time이 더 빠르다. Quick sort와 introspective sort의 성능을 비교했을 때, input이 100개 미만일 때는 quick sort의 running time이 약간 더 빠르고, 100개 이상일 때는 introspective sort의 running time이 훨씬 빠르다.

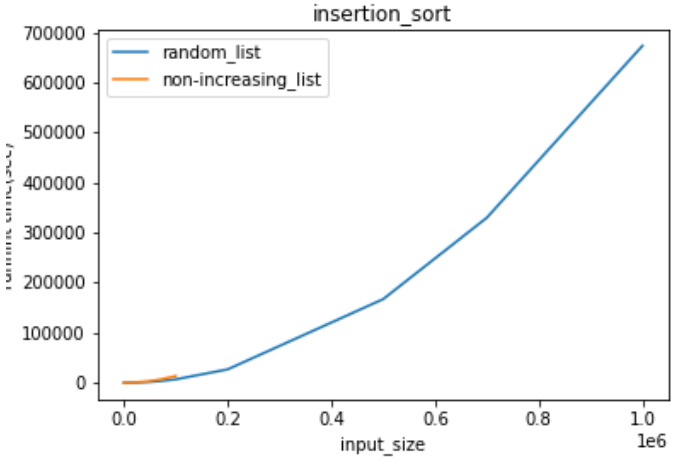
	Running time(sec)	insertion	merge	quick	introspective
input_size					
10		0.114	0.044	0.041	0.065
20		0.073	0.049	0.045	0.06
50		0.055	0.059	0.063	0.13
70		0.067	0.072	0.083	0.161
100		0.16	0.124	0.119	0.092
200		0.216	0.162	0.31	0.208
500		1.023	0.291	1.591	0.43
700		2.01	0.418	2.963	0.478
1000		3.756	0.586	5.943	0.624
2000		11.85	1.199	18.704	1.36
5000		53.169	3.035	81.347	4.057
7000		97.697	4.309	153.24	5.562
10000		168.522	6.404	263.83	8.149
20000		574.282	11.514	907.6	15.749
50000		3398.94	23.117	5468.9	31.367
70000		6621.28	33.185	10669.2	44.799
100000		13462.4	44.228	21744.6	62.009



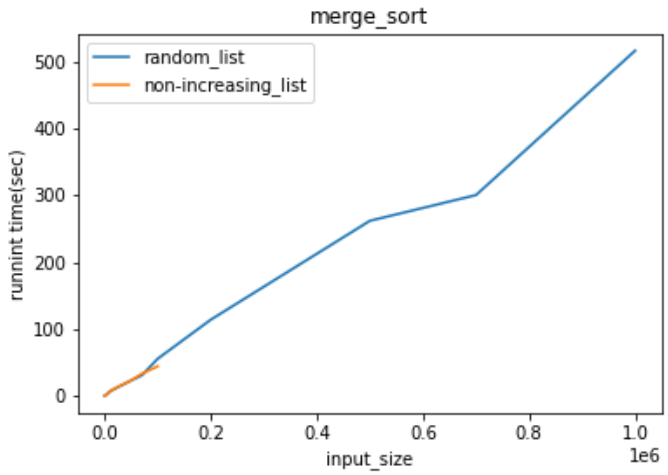
▷결론: random list를 각 input size마다 두 개의 test case를 만들어서 실험한 결과의 평균과 non-increasing list의 test case를 input으로 입력하여 실험한 결과를 모두 고려해봤을 때, insertion sort는 input의 크기가 클 때와 작을 때 running time의 차이가 너무 크고, quick sort는 input이 random list일 때와 non-increasing list일 때 running time의 차이가 너무 크다. Merge sort의 경우, non-increasing list를 input으로 입력했을 때, introspective sort보다 대체적으로 조금씩 빠르나, 100000 개 이상의 random list를 input으로 입력했을 때 introspective보다 1.5~2배 정도 느리다. 따라서, input의 size와 input element가 random으로 결정되는지, non-increasing 형태로 들어오게 되는지를 종합적으로 고려했을 때, introspective sort가 가장 running time이 빠르고 성능이 좋다.

3) 각 algorithm의 running time 그래프

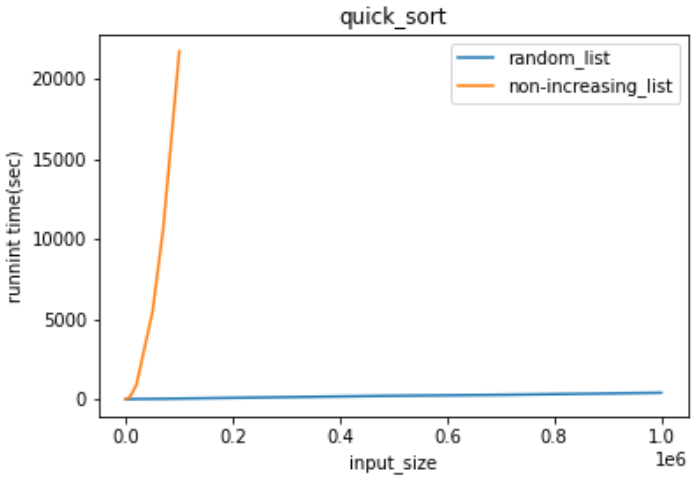
① insertion sort



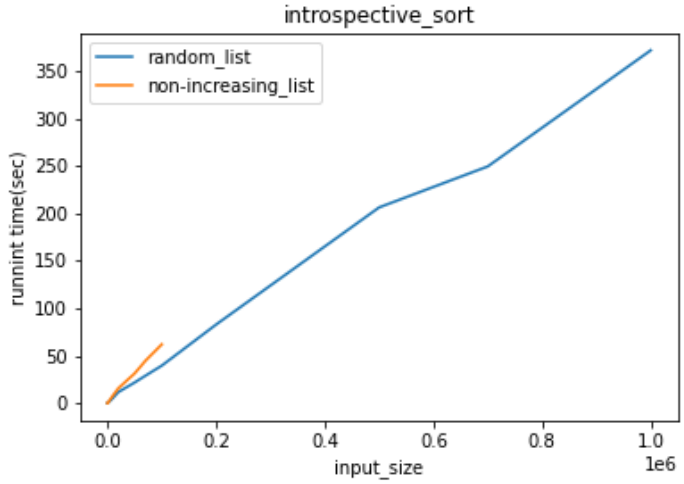
②merge sort



③quick sort



④introspective_sort



2. algorithm 4(introspective sort) 구현

- input list의 크기가 32보다 작을 때는 insertion sort를 한다. Insertion sort를 하기 위한 기준이되는 input list의 크기는 처음에는 여러 레퍼런스를 참고하여 일반적으로 16으로 설정하는 것을 보고 16으로 설정하였는데, 100개 이하의 input size에서 insertion sort의 running time이 introspective sort의 running time보다 빠른 것을 보고 32개로 늘렸다.

-recursion의 최대 횟수를 log10(input list의 size) 의 내림에 2를 곱한 수로 잡고, 해당 횟수를 넘어가면 heap sort를 한다. 해당 조건을 모두 통과하면 quick_sort를 한다. 이때, quick_sort의 각 partition마다 pivot은 rand()를 이용해서 랜덤으로 결정한다. 가장 오른쪽에 있는 원소를 pivot으로 설정했을 때, 제일 큰 원소가 가장 오른쪽에 위치해서 recursion의 횟수를 증가시키는 것을 방지하기 위해서 각 partition마다 pivot은 랜덤으로 결정하고, recursion의 depth를 제한하도록 한다. 실제 running time을 측정했을 때, left, (left+right)/2, right에 위치한 원소 중 median값을 pivot으로 정하는 것과 랜덤으로 결정하는 것 중에서 후자가 더 running time이 빨랐기 때문에 pivot은 Median Of Three가 아닌, random하게 결정하는 방식을 택하였다.