알고리즘설계와분석

MP2 보고서

20200562 신서영

Experiment environment

CPU speed : 2.10 GHz

Memory size : 8 GB

OS : Windows 11

Experiment setup

Metric : 알고리즘의 수행시간 (clock함수를 이용하여 확인하였다)

Range of input size : size가 10인 난수 배열부터 100,000인 난수 배열까지 size의 크기를 10배씩 늘려가며 테스트하였다.

각 원소는 -100,000 부터 100,000 사이의 정수로 설정하였다.

Comparing performance of the four algorithms

Random list 측정결과

1. Input size : 10

algo \ list	list 1	list 2	list 3	list 4	list 5	average
selection	0.000006	0.000005	0.000005	0.000006	0.000003	0.000005
quick	0.000005	0.000005	0.000003	0.000006	0.000005	0.0000048
merge	0.000008	0.000008	0.000008	0.000008	0.000008	0.000008
mysort	0.000038	0.000034	0.000031	0.000033	0.000038	0.0000348

2. Input size : 100

algo \ list	list 1	list 2	list 3	list 4	list 5	average
selection	0.000052	0.000075	0.000071	0.000061	0.000077	0.0000672
quick	0.000022	0.000020	0.000020	0.000012	0.000023	0.0000194
merge	0.000042	0.000030	0.000044	0.000041	0.000045	0.0000404
mysort	0.000053	0.000054	0.000043	0.000055	0.000045	0.00005

3. Input size : 1000

algo \ list	list 1	list 2	list 3	list 4	list 5	average
selection	0.005782	0.005419	0.005760	0.003389	0.004234	0.0049168
quick	0.000223	0.000250	0.000274	0.000193	0.000259	0.0002398
merge	0.000361	0.000456	0.000310	0.000486	0.000472	0.0012378
mysort	0.000255	0.000271	0.000166	0.000293	0.000351	0.0002672

4. Input size : 10,000

algo \ list	list 1	list 2	list 3	list 4	list 5	average
selection	0.465392	0.448848	0.464407	0.462961	0.461405	0.4606026
quick	0.003114	0.002418	0.002291	0.002910	0.003442	0.002835
merge	0.005340	0.003693	0.005585	0.004899	0.005682	0.0050398
mysort	0.002766	0.003431	0.002020	0.003625	0.003039	0.0029762

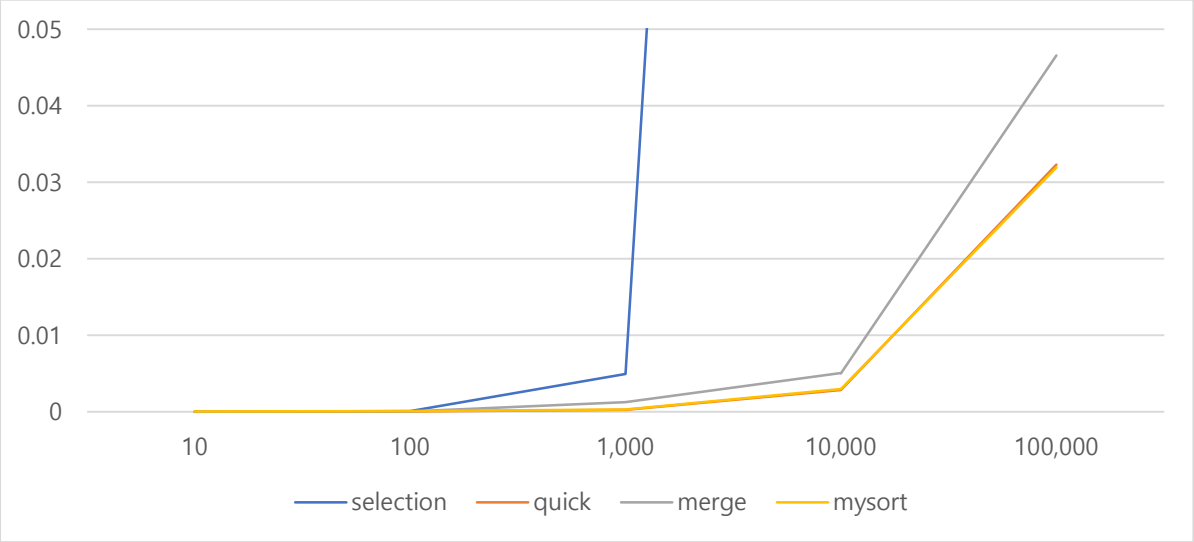
5. Input size : 100,000

algo \ list	list 1	list 2	list 3	list 4	list 5	average
selection	39.277229	42.399815	42.653706	42.220322	42.692760	41.8487664

quick	0.034128	0.028030	0.031808	0.033410	0.034054	0.032286
merge	0.051109	0.049380	0.042858	0.040431	0.049103	0.0465762
mysort	0.029759	0.030263	0.032046	0.034366	0.033301	0.031947

average 기준 random list 측정결과 표 및 추세선

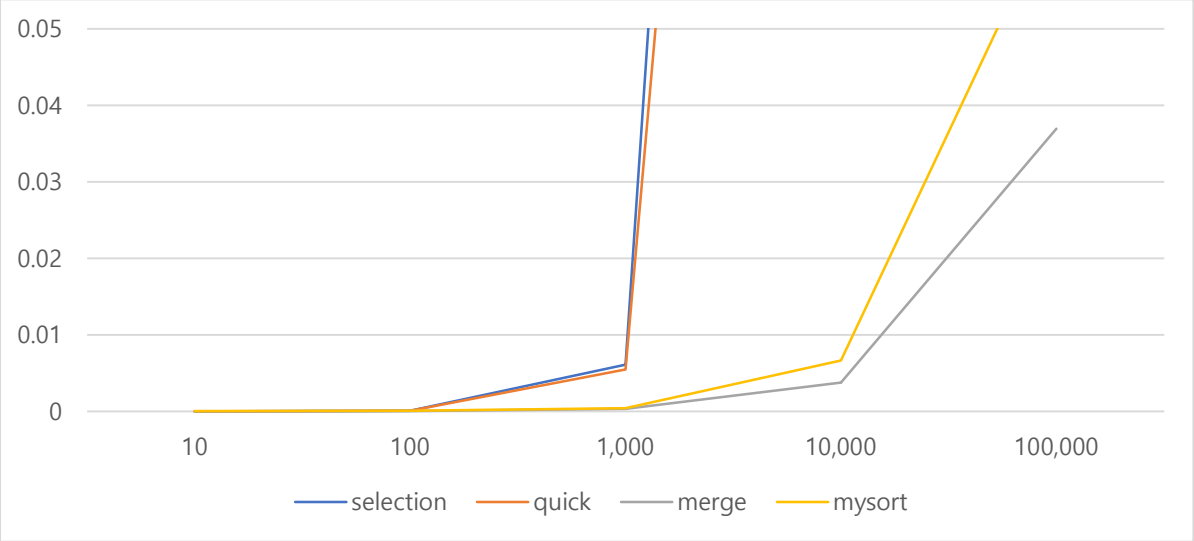
Algo \ size	10	100	1,000	10,000	100,000
selection	0.000005	0.0000672	0.0049168	0.4606026	41.8487664
quick	0.0000048	0.0000194	0.0002398	0.002835	0.032286
merge	0.000008	0.0000404	0.0012378	0.0050398	0.0465762
mysort	0.0000348	0.00005	0.0002672	0.0029762	0.031947



각 input size별 5개의 random list의 시간 측정결과를 종합하여 표를 도출하고 추세선그래프를 그렸다. selection sort는 input size가 아주 작을 때(10개 이내)를 제외하고는 매우 느렸고, merge sort는 selection sort보다는 매우 빠르고 quick sort보다는 느린 안정적인 속도를 보였다. mysort, 내가 만든 알고리즘은 input size가 작을 때는 비교적 느린 속도를 보였다. 이는 heap sort를 할지 quick sort를 할지 결정하는 과정에서 생기는 오버헤드 때문으로 추정된다. 그러나 mysort의 속도는 input size가 커질수록 매우 빨라졌다. Input size가 100,000개 이상일 때는 quick sort를 추월하는 속도를 보였다.

non-increasing list 측정결과 표 및 추세선

Algo \ size	10	100	1,000	10,000	100,000
selection	0.000005	0.000065	0.006082	0.421449	33.699020
quick	0.000006	0.000063	0.005450	0.325525	25.515999
merge	0.000005	0.000034	0.000305	0.003759	0.036951
mysort	0.000033	0.000068	0.000390	0.006643	0.066403



각 input size별 non-increasing list의 시간 측정결과를 종합하여 표를 도출하고 추세선그래프를 그렸다. 예상과 같이 selection sort는 이번에도 매우 느린 속도를 보였다. 이전에 상당히 빠른 성능을 자랑했던 quick sort는 이전과 확연히 다른 결과를 보였다. 이는 worst case를 계속해서 마주한 결과로 추정된다. merge sort는 non-decreasing list에서도 상당히 빠르고 안정적인 속도를 보였다. 내가 만든 mysort는 quicksort보다는 빠르고 merge sort보다 느린 결과를 보였다.

Algorithm 4 디자인 설명

Quicksort를 기반으로 heap sort를 추가한 instrosort를 구현하였다. depth의 초기값을 $2 * \log_2(input\ size)$로 설정한 다음, quick sort가 재귀호출될 때마다 depth를 1씩 줄이고, depth가 0이 되면 heap sort로 전환하는 알고리즘이다. 간단히 말해 quick sort를 시행하되 재귀의 깊이가 너무 커질 때에는 heap sort로 전환하여 최악의 경우에도 시간복잡도가 $O(n \log n)$을 유지할 수 있도록 하는 것이다. 최악의 경우의 시간복잡도를 줄이는 방식으로 sorting의 러닝타임을 줄일 수 있을 것이라고 예상하였다.

```
void mysort(int *arr, int p, int r, int depth) {
    if (depth == 0)
        heap(arr, r - p + 1);
    else if (p < r) {
        int q;
        q = partition(arr, p, r);
        mysort(arr, p, q - 1, depth - 1);
        mysort(arr, q + 1, r, depth - 1);
    }
}
```

코드는 매우 간단하다. arr은 정렬할 배열을 의미하고 p는 배열의 시작 인덱스를, r은 배열의 마지막 인덱스를 의미한다. 기본적으로 quick sort를 진행하다가 Depth가 0에 다다른 경우 힙소트를 호출한다.