n	Quicksort 중간 피벗		Quicksort 랜덤 피벗	
	Comparison Exchange	Time (ms)	Comparison Exchange	Time (ms)
100	184	0.018	173	0.187
200	412	0.032	425	1.289
500	1173	0.079	1188	21.005
1,000	2614	0.149	2620	161.1523
2,000	5641	0.332	5761	1368.3
3,000	9002	0.468	8917	4448.39
4,000	12273	0.619	12326	10577.6
5,000	15538	0.929	15815	20980.3

n	Quicksort 처음 피벗		Quicksort 처음,중간,마지막 중 가장 작은 피벗	
	Comparison Exchange	Time (ms)	Comparison Exchange	Time (ms)
100	190	0.021	196	0.019
200	420	0.035	415	0.028
500	1202	0.086	1202	0.071
1,000	2572	0.165	2682	0.144
2,000	5726	0.46	5801	0.287
3,000	9079	0.533	9033	0.444
4,000	12656	0.826	12681	0.642
5,000	15858	0.883	16049	0.805

n	Heapsort	
	Comparison Exchange	Time (ms)
100	413	0.027
200	1010	0.057
500	3140	0.142
1,000	7368	0.326
2,000	16689	0.708
3,000	26919	0.978
4,000	37358	1.137
5,000	48358	1.703

표와 같이 퀵소트를

1) 중간 요소 ((i +j) / 2)를 피벗으로 정했을 때
2) 처음과 마지막 사이에 랜덤 요소를 피벗으로 정했을 때
3) 처음 요소를 피벗으로 정했을 때
4) 처음 중간 마지막 요소 중 가장 작은 요소를 피벗으로 정했을 때

이 네 가지로 분류해서 실험을 진행한 결과

랜덤 요소를 피벗으로 정하는 경우 c++ rand 함수를 이용해 피벗을 정해서 교환하는 횟수는 다른 경우와 비슷했지만, 걸린 시간이 더 긴 것을 확인할 수 있었다.

그리고 4 번째 경우와 3 번째 경우의 n = 4000 행을 살펴보면 4 번째 경우가 더 교환횟수가 적지만 걸린 시간이 더 긴 것을 확인 할 수 있었다.

마지막으로 1 번째 경우와 3 번째 경우의 n = 5000 행을 살펴보면 3 번째 경우가 교환횟수는 많지만 걸린 시간이 더 적다는 것을 확인할 수 있다.

그걸로 미뤄보아 피벗을 정할 때 (i+j)/2와 quick[(i+j)/2], quick[i], quick[j]의 중간값을 구하는 연산들, rand 함수 등 더 시간을 복잡하게 하는 것으로 생각할 수 있었다.

그래서 이 실험 데이터로만 보아서는 처음 요소를 피벗으로 정했을 때가 제일 선호하는 알고리즘이다.

그리고 힙소트와 3) 퀵소트를 비교했을 때는

힙소트는 최악시간복잡도가 O(nlgn)이고 퀵소트의 최악시간복잡도는 O(n^2)인데 실험 결과로는 퀵소트가 비교횟수와 걸린 시간 모두 낮게 나왔다.