Mohib Ahmed Report Project 2(SORTING) CSCI 335

- -Heap sort and Quick Select benefit from not completing full sorts due to once they reach the median they can stop. Which results in only ½ the work being done.
- -STD:SORT and Merge do not benefit from only having to find the median.
- -WORST CASE for quickselect is super slow which caught me off guard.
- -Also for quick select the time it takes can vary a lot.
- -Once you reach a million integers the timing drastically increases which is obvious but you don't truly understand it once you do it yourself.
- -STD:SORT is the best clearly which makes sense. People much smarter then me most likely attempted to make it as efficient as it possibly could be. I did expect half heap sort and quick select to be more efficient then std::sort but they were not on my end.

Half Selection Sort	Standard Sort	Merge Sort	In Place Merge Sort	Half Heap Sort	Quick Select	
2ms	0ms	1ms	0ms	0ms	0ms	
3ms	0ms	1ms	0ms	0ms	0ms	
2ms	0ms	1ms	0ms	0ms	0ms	
2816ms	2ms	60ms	33ms	8ms	3ms	
2708ms	2ms	61ms	44ms	9ms	3ms	
2580ms	2ms	57ms	30ms	8ms	2ms	
N/A	120ms	2258ms	1111ms	679ms	157ms	
N/A	78ms	1891ms	1498ms	356ms	80ms	
N/A	82ms	1888ms	1009ms	263ms	70ms	
2.5ms	0ms	1ms	0ms	0ms	0ms	
2600ms	2ms	60ms	35ms	8.3ms	2.8ms	
N/A	90ms	1998ms	1258ms	451ms	101ms	
					10,201ms	
O(nA2)	O(plogp)	O(nlogn)	O(nlogn)	O(plogp)	Avg O(n) Worst	· ()(n/2)
· ·					Avg-2n WorstO(n^2)	
	2ms 3ms 2ms 2816ms 2708ms 2580ms N/A N/A N/A 2.5ms 2600ms	2ms 0ms 3ms 0ms 2ms 0ms 2816ms 2ms 2708ms 2ms 2580ms 2ms N/A 120ms N/A 78ms N/A 82ms 2.5ms 0ms 2600ms 2ms N/A 90ms O(n^2) O(nlogn)	2ms 0ms 1ms 3ms 0ms 1ms 2ms 0ms 1ms 2816ms 2ms 60ms 2708ms 2ms 61ms 2580ms 2ms 57ms N/A 120ms 2258ms N/A 78ms 1891ms N/A 82ms 1888ms 2.5ms 0ms 1ms 2600ms 2ms 60ms N/A 90ms 1998ms O(n^2) O(nlogn) O(nlogn)	2ms 0ms 1ms 0ms 3ms 0ms 1ms 0ms 2ms 0ms 1ms 0ms 2816ms 2ms 60ms 33ms 2708ms 2ms 61ms 44ms 2580ms 2ms 57ms 30ms N/A 120ms 2258ms 1111ms N/A 78ms 1891ms 1498ms N/A 82ms 1888ms 1009ms 2.5ms 0ms 1ms 0ms 2600ms 2ms 60ms 35ms N/A 90ms 1998ms 1258ms O(n^2) O(nlogn) O(nlogn) O(nlogn)	2ms 0ms 1ms 0ms 0ms 3ms 0ms 1ms 0ms 0ms 2ms 0ms 1ms 0ms 0ms 2816ms 2ms 60ms 33ms 8ms 2708ms 2ms 61ms 44ms 9ms 2580ms 2ms 57ms 30ms 8ms N/A 120ms 2258ms 1111ms 679ms N/A 78ms 1891ms 1498ms 356ms N/A 82ms 1888ms 1009ms 263ms 2.5ms 0ms 1ms 0ms 0ms 2600ms 2ms 60ms 35ms 8.3ms N/A 90ms 1998ms 1258ms 451ms O(n^2) O(nlogn) O(nlogn) O(nlogn) O(nlogn)	2ms 0ms 1ms 0ms 0ms 0ms 3ms 0ms 1ms 0ms 0ms 0ms 2ms 0ms 1ms 0ms 0ms 0ms 2ms 60ms 33ms 8ms 3ms 2708ms 2ms 61ms 44ms 9ms 3ms 2708ms 2ms 57ms 30ms 8ms 2ms N/A 120ms 2258ms 1111ms 679ms 157ms N/A 78ms 1891ms 1498ms 356ms 80ms N/A 82ms 1888ms 1009ms 263ms 70ms 2.5ms 0ms 1ms 0ms 0ms 0ms 2600ms 2ms 60ms 35ms 8.3ms 2.8ms N/A 90ms 1998ms 1258ms 451ms 101ms 0ms 0ms 0ms 0ms 0ms 0ms 0ms 0ms 0ms 0ms 0ms