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.

Input.txt and size	Half Selection Sort	Standard Sort	Merge Sort	In Place Merge Sort	Half Heap Sort	Quick Select	
1	2ms	0ms	1ms	0ms	0ms	0ms	
2	3ms	0ms	1ms	0ms	0ms	0ms	
3	2ms	0ms	1ms	0ms	0ms	0ms	
4	2816ms	2ms	60ms	33ms	8ms	3ms	
5	2708ms	2ms	61ms	44ms	9ms	3ms	
6	2580ms	2ms	57ms	30ms	8ms	2ms	
7	N/A	120ms	2258ms	1111ms	679ms	157ms	
8	N/A	78ms	1891ms	1498ms	356ms	80ms	
9	N/A	82ms	1888ms	1009ms	263ms	70ms	
Avg for1000	2.5ms	0ms	1ms	0ms	0ms	0ms	
Avg for 31000	2600ms	2ms	60ms	35ms	8.3ms	2.8ms	
Avg for 1000000	N/A	90ms	1998ms	1258ms	451ms	101ms	
Worst Case QS 20000						10,201ms	
Big-O	O(n^2)	O(nlogn)	O(nlogn)	O(nlogn)	O(nlogn)	Avg-O(n) Worst-O(n^2)	
Exact Big-O	n*(n-1)/2	O(n)	O(nlogn)	O(nlogn)	O(nlogn/2)	Avg-2n WorstO(n^2)	