Matt McDade 2849637

Algorithms Project 2 Analysis

(a) Complete the following 4 tables:

Table 1-1 Running time in millisecond for case 1 (points are within a circle)

n*	Running time		
	Graham Scan	Jarvis March	Quickhull
10	1	2	1
1000	3	2	4
10,000	5	6	29
100,000	56	86	422
1,000,000	578	1163	6398

Table 1-2 Running time in millisecond for case 2 (points are on a circle)

n*	Running time		
	Graham Scan	Jarvis March	Quickhull
10	3	1	1
1000	2	6	13
10,000	6	31	112
100,000	110	292	1051
1,000,000	7992	3152	12216

Table 1-3 Running time in millisecond for case 3 (points are within a rectangle)

n*	Running time		
	Graham Scan	Jarvis March	Quickhull
10	1	2	1
1000	2	2	3
10,000	7	7	14
100,000	61	49	80
1,000,000	1192	435	501

Table 1-4 Running time in millisecond for case 3 (points are within a triangle)

n*	Running time		
	Graham Scan	Jarvis March	Quickhull
10	2	1	2
1000	3	2	3
10,000	7	2	14
100,000	54	23	61
1,000,000	765	229	186

(b) What's the asymptotic time complexity of the three algorithms? Complete the following table:

	Running time complexity		
	Graham Scan	Jarvis March	Quickhull
Best case	n	nh	$n \log n$
Average case	$n \log n$	nh	$n \log n$
Worst case	$n \log n$	n^2	n^2

(c) Does your empirical analysis match with your theoretical analysis? Justify your answer.

In my testing, Jarvis's march won the most amount of times for larger datasets, which makes sense because it has the best average time complexity of nh. The only time it loses on the largest dataset is when data convex hull forms a circle around the points. Since this isn't the dataset that has the most points on the convex hull, it doesn't make complete sense. This is only one time recorded, though. If these tests were run an infinite amount of times, this empirical analysis would match closer to the asymptotic time complexities.

Quickhull had the slowest times in my testing almost across the board. I predict this is because I performed more complex operations near the end after the hull was found to order the points in a way that the provided java program drew them nicely. This required a datatype conversion (I used different datatypes to store coordinate location for quickhull), and sorting of the points on the hull. Because of these extra steps, datasets with more points on the hull took significantly longer for my quickhull implementation (see times for points on a circle).