COMP0005 Algorithms Coursework 1 (Group 5)

Methodology.

We collected the average time taken for each algorithm to execute five times. To test linearithmic complexity, which is not supported in Microsoft Excel, we plotted time against $n \log n$ and plotted a linear trendline. A graph and the best fit trendline, determined by the trendline with the highest Correlation of Determination, R^2 is chosen to be the most reliable time complexity.

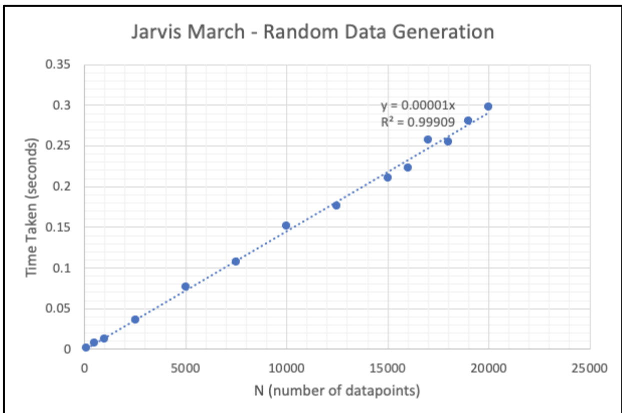
Jarvis March Algorithm – Analysis.

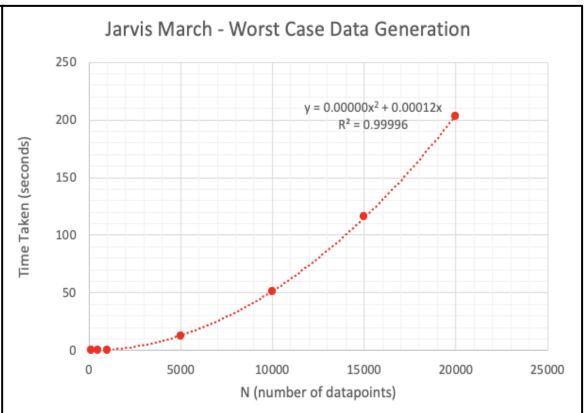
Random Input – *Theoretical Computational Complexity* – For random input the complexity would be linear $O(n)$. This is because for every point on the hull we iterate through all the other points to decide which is next. Time complexity is therefore $(x * y)$ where y is number of points we input, and x is the number of points on the hull, since in the average case x is significantly smaller than y when y is large we can drop the x hence we are left with a linear complexity.

Worst-case Scenario – *Theoretical Computational Complexity* – For the worst case, the complexity would be quadratic $O(n^2)$ because the worst case occurs when all the points lie on the convex hull. With the same reason as the random data generation $(x = y)$.

Random Input – *Experimental Computational Complexity* – The analysis shows the complexity is linear $O(n)$, as predicted from the theoretical analysis.

Worst-case Scenario – *Experimental Computational Complexity* – The analysis shows the complexity is quadratic $O(n^2)$ as predicted from the theoretical analysis.





Jarvis March - Random Data Generation (Time Taken in Seconds)						
N (Number of datapoints)	Time Taken (1)	Time Taken (2)	Time Taken (3)	Time Taken (4)	Time Taken (5)	Time Taken Avg
100	0.001505453	0.001061318	0.001559718	0.001226073	0.001696146	0.001409742
500	0.008836339	0.005434963	0.008622947	0.007491428	0.007647754	0.007606686
1000	0.010269157	0.013936317	0.014621241	0.014767947	0.012053927	0.013129718
5000	0.098301286	0.06935852	0.065438373	0.083205073	0.065320014	0.076324653
10000	0.154389193	0.150234282	0.167435136	0.139988469	0.14449148	0.151307712
15000	0.245504987	0.196717238	0.188280016	0.215625709	0.208296415	0.210884873
20000	0.3255257	0.341181615	0.323690476	0.229931769	0.267274656	0.297520843

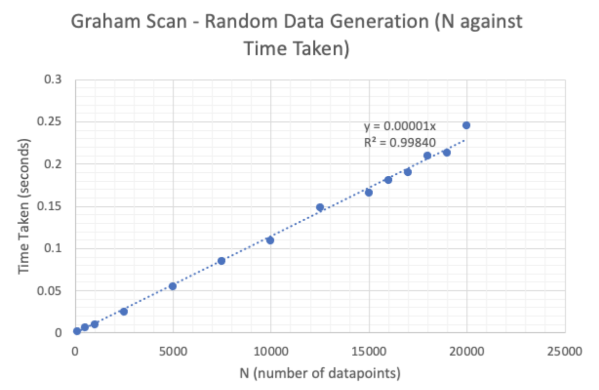
Jarvis March - Worst Case Data Generation (Time Taken in Seconds)						
N (Number of datapoints)	Time Taken (1)	Time Taken (2)	Time Taken (3)	Time Taken (4)	Time Taken (5)	Time Taken Avg
100	0.004479149	0.004480883	0.004936669	0.004262877	0.004784987	0.004588913
500	0.112923593	0.11960144	0.116664313	0.114979892	0.122623014	0.11735845
1000	0.485538348	0.46358817	0.466390849	0.463518809	0.456617537	0.467130743
5000	12.27992972	12.83753778	13.04436529	11.36576604	12.80580505	12.46668077
10000	52.41045677	51.46300434	51.14701866	49.60065452	52.81969175	51.48816521
15000	116.5793083	117.3485628	114.4379192	117.5909318	114.5712081	116.105586
20000	197.5742926	190.8834199	204.941504	202.4621609	221.8968405	203.5516436

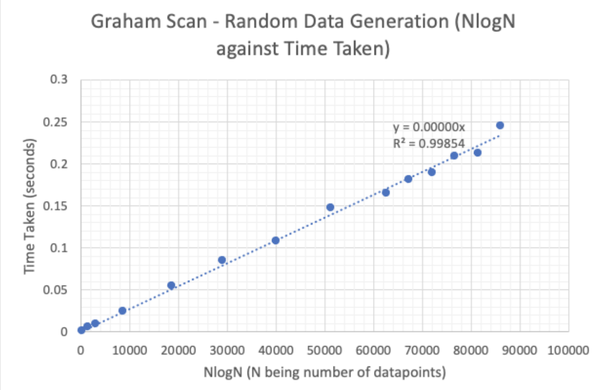
Graham Scan Algorithm – Analysis.

Random Input – *Theoretical Computational Complexity* – For random input the complexity would be $O(n \log n)$. We split the analysis of the algorithm into different parts. The order of the functions that is used in the graham scan algorithm is explained in the program's source code.

1. There are 5 parts of the program with $O(n)$ time complexity reason that it iterates through the list once, i.e. calculating point P, converting cartesian to polar, removing the polar form points with no magnitude, removing the polar coordinates with the same angle but smaller magnitude and converting points back into cartesian.
2. The sorting algorithm that sorts the angle has $O(n \log n)$ complexity on average and worst case since we used merge sort.
3. The calculation of the convex hull at most iterates through the list twice (Graham, 1972), hence time complexity $O(2n)$
4. Taking the sum, $O(n) + O(n \log n) + O(2n) = O(n \log n)$

Random Input – *Experimental Computational Complexity* – The analysis shows the complexity is linearithmic $O(n \log n)$ as predicted from the theoretical analysis.

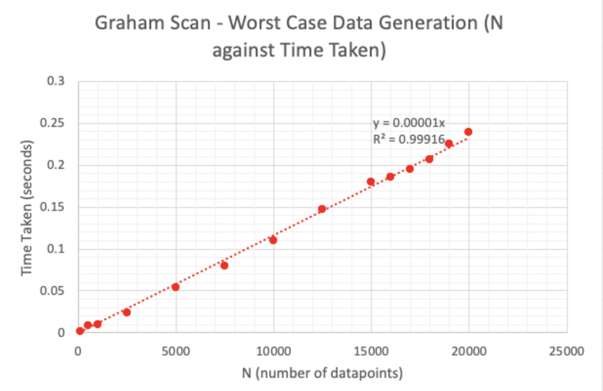


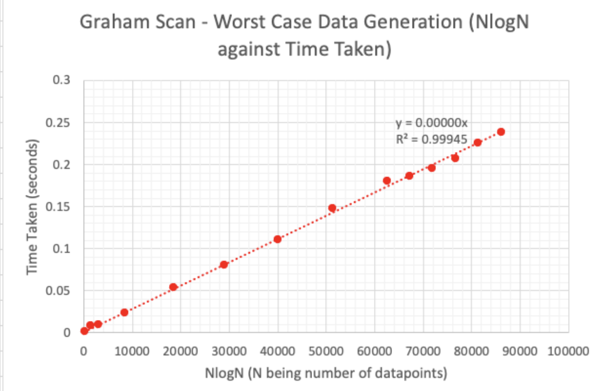


Graham Scan - Random Data Generation (Time Taken in Seconds)							
N (Number of datapoints)	NlogN (N being number of datapoints)	Time Taken (1)	Time Taken (2)	Time Taken (3)	Time Taken (4)	Time Taken (5)	Time Taken Avg
100	200	0.001434431	0.001419996	0.001336104	0.001521876	0.001276202	0.001397722
500	1349.485002	0.006377741	0.00631221	0.006321545	0.006131734	0.006129483	0.006254543
1000	3000	0.010335575	0.010246953	0.010375583	0.010304179	0.00991674	0.010235806
5000	18494.85002	0.055172615	0.056071611	0.053733189	0.053250928	0.058142637	0.055274196
10000	40000	0.119931676	0.104480877	0.116654176	0.104475282	0.099394991	0.1089874
15000	62641.36889	0.176136966	0.16695726	0.148312603	0.168528672	0.166193604	0.165225821
20000	86020.59991	0.251407147	0.231674034	0.246945552	0.270173164	0.22920046	0.245880071

Worst-case Scenario – *Theoretical Computational Complexity* – For the worst case, the complexity would be $O(n \log n)$ with the similar reasoning as random case, as merge sort contributes most to the time complexity. We have attempted to make datasets that would cause the sorting algorithm to do maximum amount of work, which happens when merge sort does the greatest number of comparisons. This is done by alternating elements within a sorted array such that at every merge stage, all elements from both auxiliary arrays are compared, rather than having one auxiliary array being concatenated with the returning merged array.

Worst-case Scenario – *Experimental Computational Complexity* – The analysis shows the complexity is linearithmic $O(n \log n)$ as predicted from the theoretical analysis.



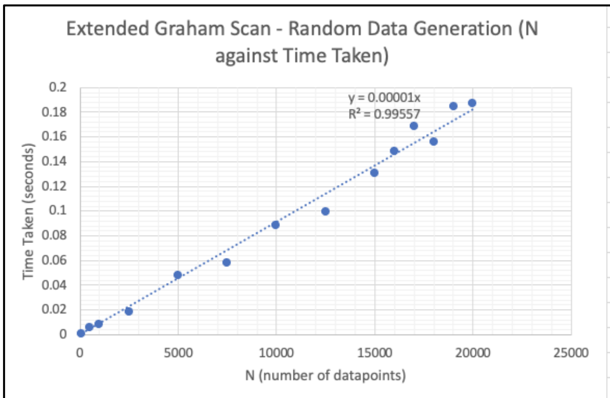


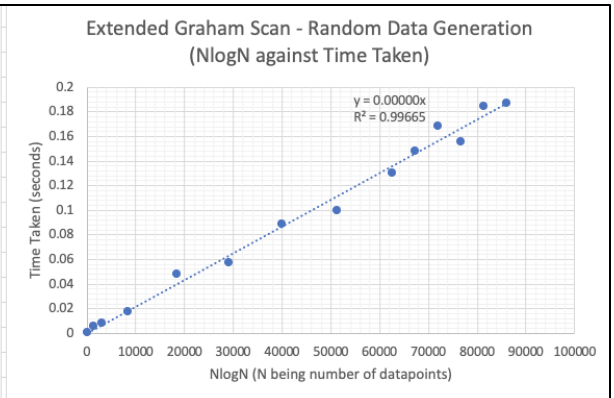
Graham Scan - Worst Case Data Generation (Time Taken in Seconds)							
N (Number of datapoints)	NlogN (N being number of datapoints)	Time Taken (1)	Time Taken (2)	Time Taken (3)	Time Taken (4)	Time Taken (5)	Time Taken Avg
100	200	0.001245045	0.00132426	0.001241884	0.002143066	0.001227256	0.001436302
500	1349.485002	0.021064736	0.005787436	0.006163269	0.006280544	0.005704009	0.008999999
1000	3000	0.009732881	0.009995924	0.009694289	0.010654749	0.009898119	0.009995192
5000	18494.85002	0.052661792	0.057290458	0.049212913	0.060825513	0.05178245	0.054354625
10000	40000	0.115094348	0.103248553	0.131598848	0.099941753	0.101988755	0.110374451
15000	62641.36889	0.174439177	0.173913745	0.202173806	0.198926295	0.152423353	0.180375275
20000	86020.59991	0.251479492	0.240125779	0.242876869	0.23360882	0.226464074	0.238911007

Extended Graham Scan Algorithm – Analysis.

Random Input – *Theoretical Computational Complexity* – Same complexity as Graham scan with the same reasoning as the heuristics has linear complexity.

Random Input – *Experimental Computational Complexity* – The analysis shows the complexity is linearithmic $O(n \log n)$ as predicted from the theoretical analysis.

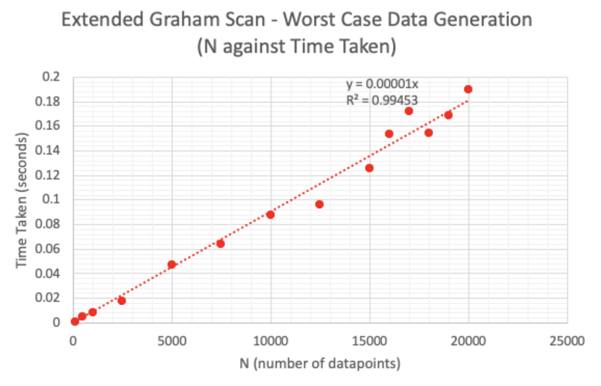


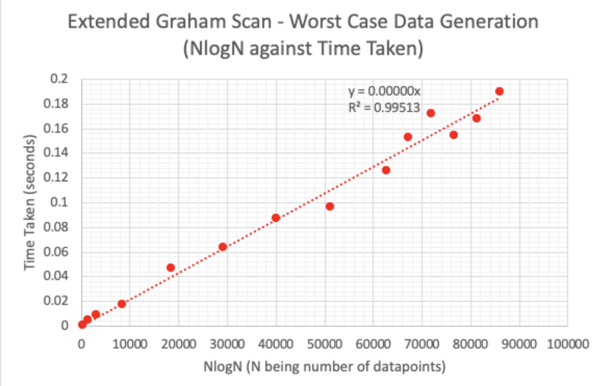


Extended Graham Scan - Random Data Generation (Time Taken in Seconds)							
N (Number of datapoints)	NlogN (N being number of datapoints)	Time Taken (1)	Time Taken (2)	Time Taken (3)	Time Taken (4)	Time Taken (5)	Time Taken Avg
100	200	0.001432738	0.001190476	0.001119257	0.000785376	0.001292776	0.001164125
500	1349.485002	0.005841164	0.007189239	0.005418537	0.004719678	0.005122216	0.005658167
1000	3000	0.006572721	0.008536474	0.008272758	0.009979134	0.008647946	0.008401807
5000	18494.85002	0.04451681	0.04584939	0.050856276	0.047144048	0.051287333	0.047930771
10000	40000	0.082596895	0.086014089	0.062520778	0.134819929	0.077954042	0.088781147
15000	62641.36889	0.13819533	0.141269757	0.10613352	0.127714697	0.138131617	0.130288984
20000	86020.59991	0.191971428	0.194368501	0.217976739	0.181993073	0.148522477	0.186966444

Worst-case Scenario – *Theoretical Computational Complexity* – Same complexity as Graham scan with the same reasoning as the heuristics has linear complexity.

Worst-case Scenario – *Experimental Computational Complexity* – The analysis shows the complexity is linearithmic $O(n \log n)$ as predicted from the theoretical analysis.





Extended Graham Scan - Worst Case Data Generation (Time Taken in Seconds)							
N (Number of datapoints)	NlogN (N being number of datapoints)	Time Taken (1)	Time Taken (2)	Time Taken (3)	Time Taken (4)	Time Taken (5)	Time Taken Avg
100	200	0.001364076	0.001075436	0.00109583	0.00077902	0.001121606	0.001087194
500	1349.485002	0.005122553	0.005795185	0.004539143	0.004425763	0.004905859	0.004957701
1000	3000	0.006366801	0.008754	0.009266057	0.010758053	0.008685794	0.008766141
5000	18494.85002	0.043050537	0.044543946	0.050371531	0.049324812	0.049063304	0.047270826
10000	40000	0.081095114	0.088625127	0.066474108	0.117646999	0.084320893	0.087632448
15000	62641.36889	0.146125747	0.136660313	0.102330073	0.124906043	0.120515157	0.126107467
20000	86020.59991	0.2208089	0.169326681	0.212395525	0.174758093	0.173914009	0.190240642

Extended Graham Scan Heuristics Explained.

The extended graham scan algorithm improves upon the original graham scan by adding a point removal algorithm. This is based on the observation that points that are enclosed within the shape formed by the points with (x_{min}, y_1), (x_1, y_{min}), (x_{max}, y_2) and (x_2, y_{max}) would not be in the set of the convex hull. By discarding these points, the graham scan goes through much less data points during the tracing phase especially significant for large datasets, hence speeding up execution time.

The algorithm is as follows:

- Compute x_{min}, y_{min}, x_{max}, y_{max} as the boundary points and the area of the shape q that the points formed.
- For each point p within the input set, compute the sum of the area of triangles that the p form with the boundary points.
- If the sum does not equal to the area of quadrilateral q, then p is discarded.

As a result of this extension, the execution time of the algorithm is 1.257 times faster for random datasets as shown by Figure 1.

Concluding Comparison.

For random case scenario Jarvis March has a linear complexity. Although Graham Scan has linearithmic complexity, it is faster for our experimental random datasets as shown by the Figure 2. For worst case scenario Jarvis march has a quadratic complexity while graham has linearithmic complexity. As a result of the comparisons above, Graham Scan has proven to be the more efficient algorithm when looking at random and worst-case datasets. Overall, Graham scan is more reliable and efficient.

References.

- Alshamrani, R., Alshehri, F. and Kurdi, H., 2020. A Preprocessing Technique for Fast Convex Hull Computation. Procedia Computer Science, 170, pp.317-324.
- Graham, R., 1972. An efficient algorith for determining the convex hull of a finite planar set. Information Processing Letters, 1(4), pp.132-133.
- https://stackoverflow.com/questions/24594112/when-will-the-worst-case-of-merge-sort-occur

Additional Figures.

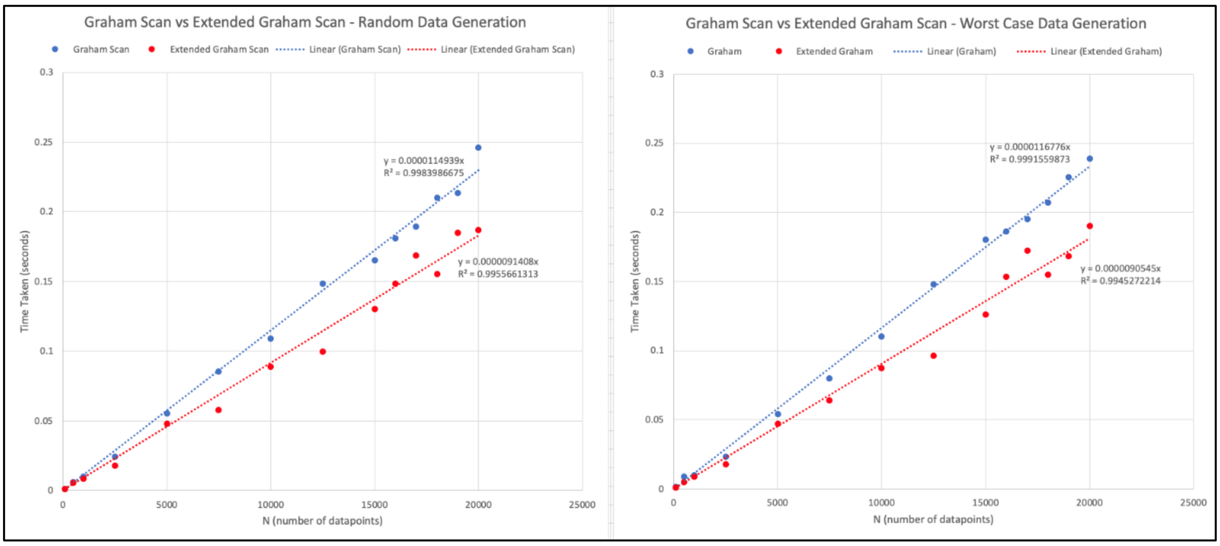


Figure 1: Graham Scan vs Extended Graham Scan

Figure 1: Jarvis March vs Graham Scan