Jarvis March Algorithm

How the algorithm works:

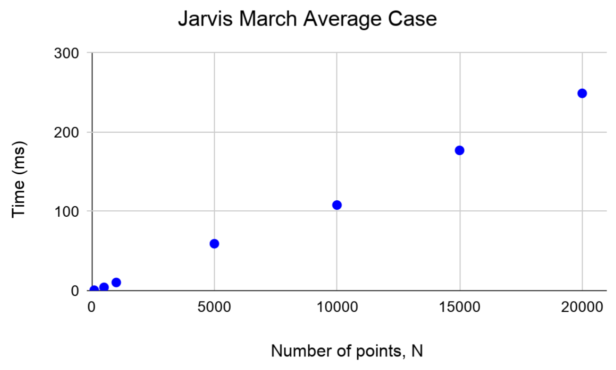
1) Finding the leftmost point which is used as the starting point, P_0
2) the algorithm will loop to pick the next point, P_i, so that all other points are to the right of the line P_{i-1} to P_i.
3) Repeat step 2 until the starting point, P_0, *h* times, where *h* is the number of points on the convex hull

Computational complexity achieved theoretically

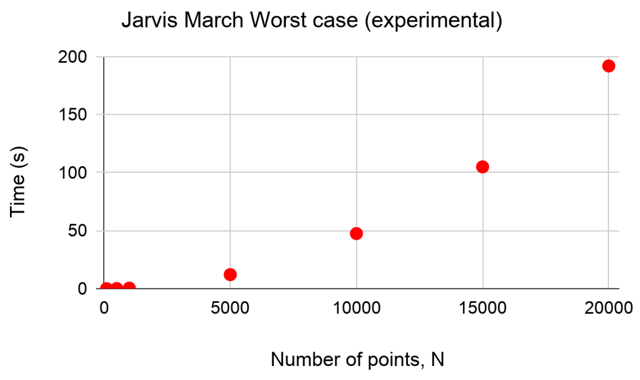
For step 1 and step 2, both for the average case and the worst case the complexity achieved theoretically would be *n,* so $\theta(n)$ and $O(n)$ for both cases respectively. Lastly, in step 3, both for the average case and the worst case the complexity achieved theoretically would be *h,* so $\theta(h)$ and $O(h)$ for both cases respectively. So overall, since both average and worst cases have the same complexity for every step, the overall computational complexity for both these cases would be $\theta(hn)$ and $O(n^2)$.
The worst case scenario in the Jarvis March Algorithm would be when all of the points in the given set of points lie on the convex hull. This would make h and n equal hence giving us $O(n^2)$ for the computational complexity achieved theoretically.

Computational Complexity achieved experimentally

Average case: Our implementation follows the theoretical complexity of $\theta(hn)$. For example, in a run of our implementation, for N = 10000 the time taken was 108ms and N = 20000 was 249ms, to 3 sig fig, which is an increase of 2.31 times. So when the input set increases by 2 times, the time taken increases by 2.31 times.



Worst case: In terms of the worst case, our implementation also follows the complexity of $O(hn)$. Since *h* = *n* in the worst case as stated before, we have $O(n^2)$ which showcased in our graph below, the worst case can be seen to be n^2. For example, when the input set is increased by 2 times, 10000 to 20000, the time taken increases 4.04 times.



Graham Scan Algorithm

Below are the 4 main computational steps for the graham scan algorithm:

Step	Big-Theta (θ)	Big-0 (O)
1. Finding bottom-most point for reference	$\theta(n)$	$O(n)$
2. Calculate polar angle for each point	$\theta(n)$	$O(n)$
3. Sort based on polar angle (**mergesort**)	$\theta(nlogn)$	$O(nlogn)$
4. Iteration through all the points	$\theta(n)$	$O(n)$

We experimented using two sorting algorithms, mergesort and quicksort for our graham scan algorithm. We ended up using **mergesort** in our implementation. Our decision in choosing mergesort will be explained below.

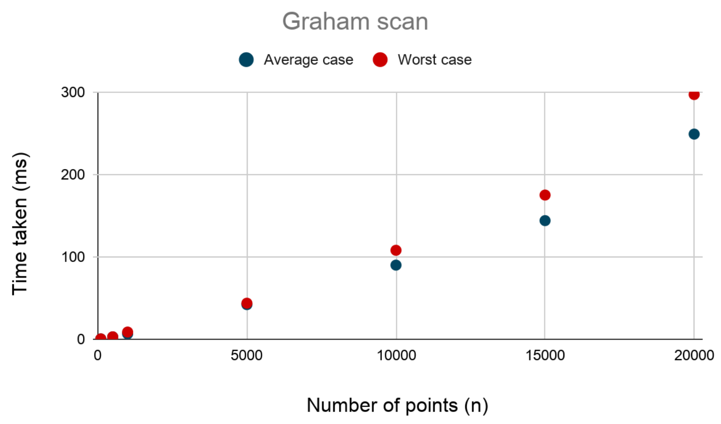
Computational complexity achieved theoretically

Average case analysis: Steps 1 and 2 require a linear scan through the array hence $\theta(n)$. The sorting step utilises **mergesort** which repeatedly halves the size of input set, hence the sorting step is of $\theta(nlogn)$. Step 4 requires a linear scan through the array of points. Although backtracking occurs, each point is only revisited at most twice, hence the average-complexity of $\theta(n)$.The overall average computational complexity is $\theta(nlogn)$ in the sorting step.

Worst case analysis: The worst case for Graham Scan is when the sequence of input points requires the maximum number of comparisons in the merge operation of mergesort. The worst case is still $O(nlogn)$ theoretically. Experimentally, the worst case takes a longer time due to more frequent comparison operations

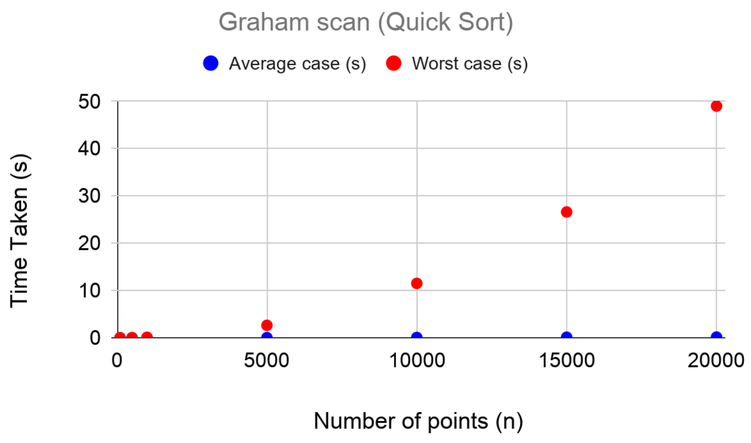
Experimentally:

Average case analysis: Overall, our implementation follows the theoretical average complexity of $\theta(nlogn)$ linearithmic time. For example, this is seen when the input set is increased by 5 times from 1000 to 5000, and the time taken increases by 5.4 times.



Worst case analysis: Overall, our implementation follows the theoretical worst case complexity of $O(nlogn)$ linearithmic time. For example, this is seen when the input set is increased by 5 times from 1000 to 5000, and the time taken increases by 8.7 times. The experimental time taken for the worst case is longer than that of the average case.

Mergesort graham scan and quicksort graham scan comparison



Although quicksort graham scan gives a slightly better experimental average complexity compared to the mergesort graham scan, the worst case for the quicksort version is $O(n^2)$ for when the input data is already sorted. Since the merge sort graham scan performs better in the worst case, we decided it would be a safer choice in our implementation even though there will be a compromise in space complexity.

<u>Extended Graham Scan:</u>
The extended version of graham scan builds off graham scan with an added heuristic, which is to **filter as many points that are within the convex hull, to reduce the size**.

In the extended version of the graham scan, we find four extreme points (top-most,bottom-most,right-most and left-most) from the input set to form a quadrilateral. The quadrilateral will adjust itself to choose four more extreme points, topleft, topright, bottomleft, and bottomright points, transforming it into an octagon.This is to filter out points that are definitely within the convex hull, and if a more extreme point is found, the respective extreme point is updated with it.

Step	Big-Theta (θ)	Worst Big-O (O)
Finding the most extreme points	$\theta(n)$	$O(n)$
Check if each point is in the shape	$\theta(n)$	$O(n)$
Perform regular graham scan	$\theta(hlogh)$	$O(nlogn)$

<u>Computational complexity achieved theoretically</u>
Average case analysis (random input): Steps 1 and 2 require a linear scan through the array hence $\theta(n)$. Since the number of random input points are significantly reduced

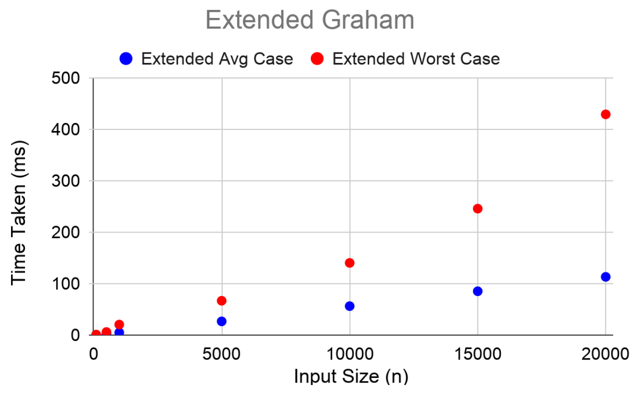
by roughly 65-84% on average (increasing with input size, reaching 82%+ after 5000 points), the merge-sort step in graham scan takes $\theta(hlogh)$ where $h < n$.

Worst case analysis: The worst case for extended graham scan occurs when the order of the order of points is increasingly more extreme, and thus nullifying the ability to filter out any points. Meaning that there is additional computation on top of the traditional graham scan, which implies $h = n$, therefore the worst case is $O(nlogn)$.

Computational complexity achieved experimentally

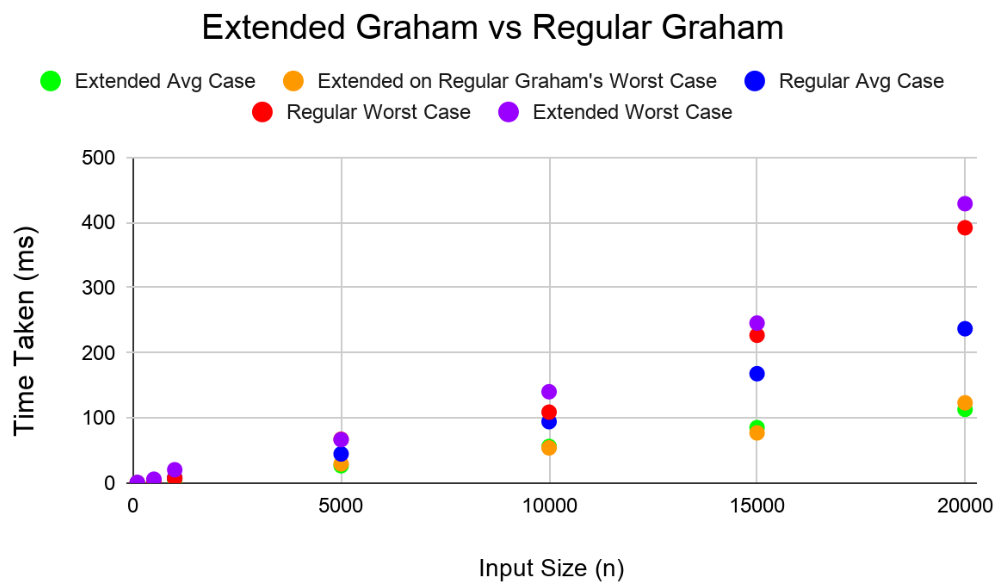
Average case analysis: Overall, our implementation follows the theoretical average complexity of $\theta(hlogh)$ linearithmic time. For example, this is seen when the input set is increased by 5 times from 1000 to 5000, and the time taken increases by 5.35 times.

Worst case analysis: Overall, our implementation follows the theoretical worst case complexity of $O(nlogn)$ linearithmic time. For example, this is seen when the input set is increased by 2 times from 5000 to 10000, and the time taken increases by 2.1 times. The experimental time taken is longer for the worst case than for the average case.



Analysis of improvements in extended Graham scan:

For the average cases, the time taken is substantially lower than their regular counterparts, and this is attributed to the magnitude of points being filtered as a result of the heuristic. The worst case for the heuristic, although still $O(nlogn)$, takes significantly longer to compute than the regular Graham scan as it takes an additional computation, filtering, which doesn't remove any points.



Reference list

Sharma, P. (2018). *Graham Scan Algorithm to find Convex Hull*. [online] OpenGenus IQ: Learn Computer Science. Available at: https://iq.opengenus.org/graham-scan-convex-hull/ [Accessed 8 Mar. 2021].

Wikipedia. (2020a). *Convex hull algorithms*. [online] Available at: https://en.wikipedia.org/wiki/Convex_hull_algorithms [Accessed 8 Mar. 2021].

Wikipedia. (2020b). *Graham scan*. [online] Available at: https://en.wikipedia.org/wiki/Graham_scan [Accessed 8 Mar. 2021].

Wikipedia. (2021). *Gift wrapping algorithm*. [online] Available at: https://en.wikipedia.org/wiki/Gift_wrapping_algorithm [Accessed 8 Mar. 2021].