COMP0005 Team15 Algorithms Coursework: Determining the Most Efficient Algorithm to Construct a Convex Hull by Computational Complexity

Abinav Baskar (20003791), Kale Champagnie (20007110),
Nathan D’Souza (20011287), Mathushan Mathiyalagan (19054897),

Jarvis March

Theoretical Time Complexity for **average case**

For an average case (random input), Time Complexity ~ $O(nh)$

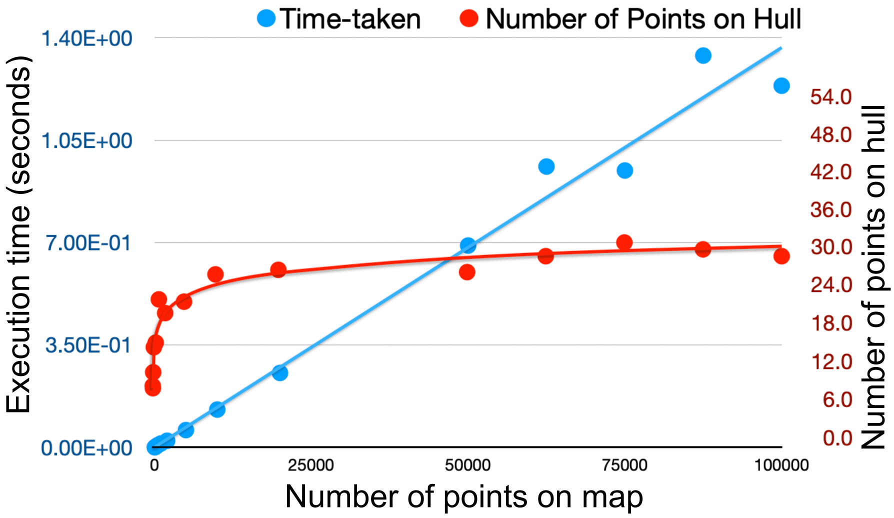
where n is the total number of points on the map, h is the number of points on the convex hull The reason for this time complexity is that the Jarvis March checks the anti-clockwise rotation angle for *all points* on the map for *each point on the hull*.

Experimental times for **average case**

The average of 3 execution times for each number of points (ranging from N=25 to N=100,000) are plotted in **blue** while the average number of points on the hull is superimposed for each value in **red** (using right axis)

The graph indicates that the points on the hull levels off logarithmically to become, essentially, constant at larger N. This is because we have a fixed map dimensions for all *N*.

Relationship between number of points, average execution time and points on hull for Jarvis March



Given this ‘constant’ h, the linear correlation between number of points and average execution time supports our theoretical claim that Time Complexity ~ $O(nh)$. It also shows Jarvis March is an output-sensitive algorithm (as h varies with n).

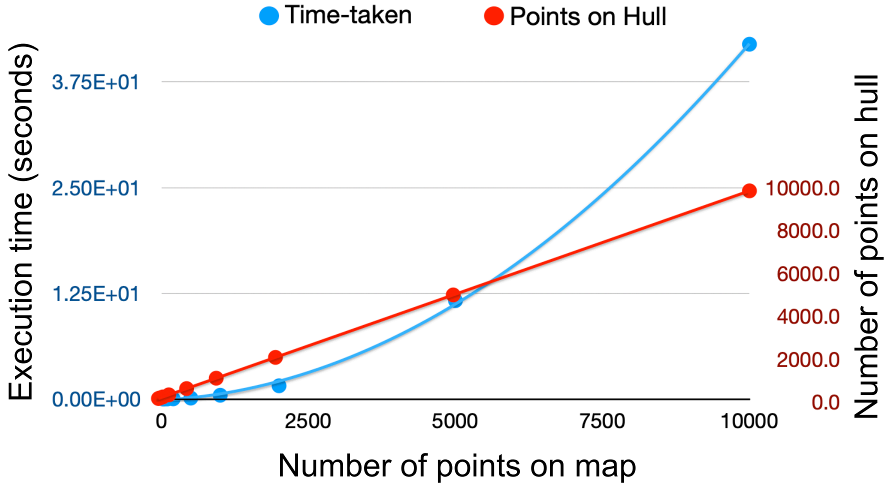
Theoretical Time Complexity for **worst-case**

Time Complexity for the worst-case scenario, given input of size n, is ~ $O(n^2)$
The reason for this time complexity is because, in the worst case scenario, all the points the Jarvis march algorithm checks are points on the hull.

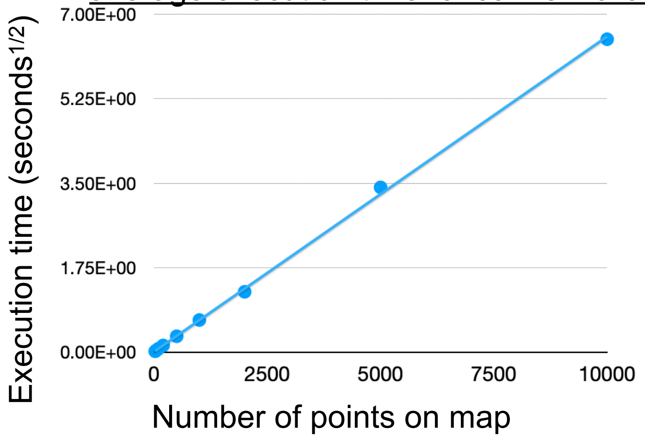
Experimental times for **worst-case**

For a data set where all points given end up on the hull, the graph supports our theoretical claim that the time complexity is $O(n^2)$ as there is a polynomial correlation between number of points on the map and the average execution time in seconds

Relationship between number of points, average execution time and points on hull for worst-case data for Jarvis March



Relationship between number of points squared and average execution time for Jarvis March



To ensure the polynomial relationship was quadratic, a scatter plot of the number of points squared on the map and the average execution time in seconds were plotted.

This produced a linear correlation which confirms that the time complexity is $O(n^2)$

Graham Scan

Theoretical Time Complexity for **average case**

For a random input of size n, Time Complexity ~ $O(nlog(n))$
This is because the graham-scan algorithm first finds the lowest leftmost point on the map and this takes $O(n)$ time. Next, all the points on the map are sorted by polar angle using quick-sort which takes $O(nlog(n))$ time for a randomly-ordered set. In the case that some points are repeated, those points are then sorted by distance which has a time complexity of $O(n)$. Finally, all points are pushed and popped on the stack at most once which is again $O(n)$.
Note: we used a stack to hold points because we can retrieve (pop) points that cause a clockwise rotation in *reverse order to how they are put in-* in a less error-prone way than indexing.

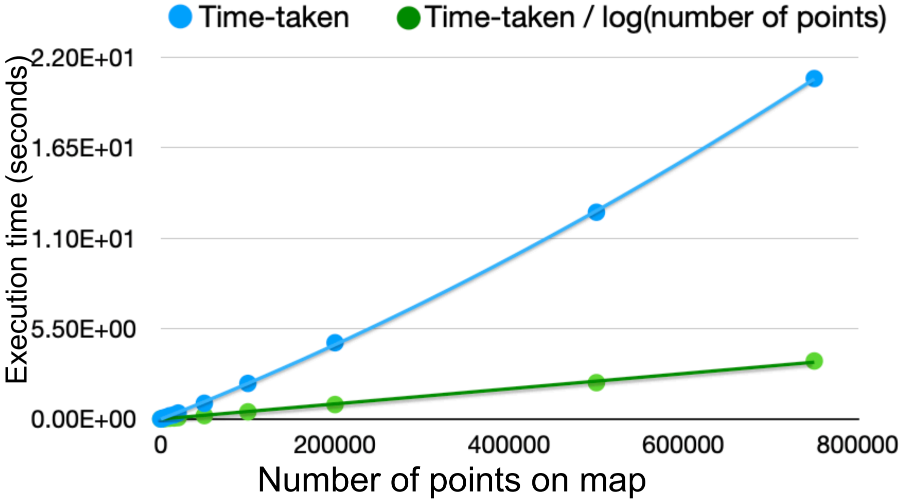
This gives an overall time complexity of $O(n) + O(nlog(n)) + O(n) + O(n) \approx O(nlog(n))$

Experimental times for **average case**

The average time of 3 applications of the graham-scan algorithm were plotted against the number of points (N=25 to N=750000) in **blue**, which appears to have a linearithmic correlation.

A **green** line (time-taken divided by the log of number of points) shows that, after negating the log(n) component of complexity, the time-taken has a linear correlation. This confirms our initial theoretical complexity of $O(nlog(n))$

Relationship between number of points, execution time and execution time divided by log(number of points)



Theoretical Time Complexity for **worst case**

As our algorithm uses quick-sort, the worst-case complexity is then O ~ (n^2), which occurs when all points in the input set are pre-sorted by polar angle. This is because, unlike the average case, the quick-sort stage will iterate through the entire (remaining) set of numbers for each increment when trying to place the pivot.
The components that contribute to time-complexity are the same as the average case, except the sorting stage, making time complexity:

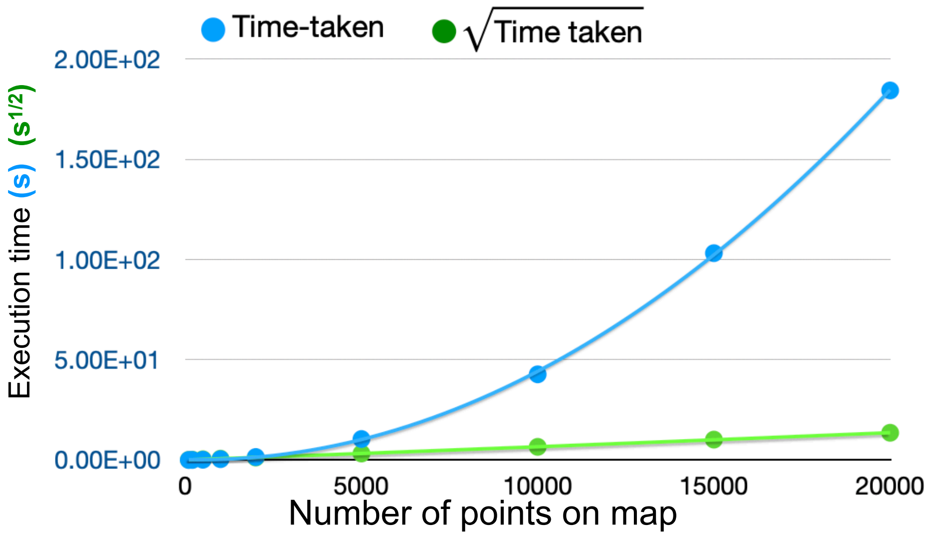
$$O(n) + O(n^2) + O(n) + O(n) \approx O(n^2)$$

If we had used merge sort, time-complexity would have been unaffected by ordered data.

Experimental times for **worst-case**

The **blue** line (time taken) is clearly non-linear when given an input set that is pre-sorted by polar angle. (An iterative quick-sort was used to circumvent python's recursion depth limit for greater N)
The **green** line, which is the square-root of time-taken, is linear, proving that the initial time complexity is indeed $O(n^2)$

Relationship between number of points, execution time and square-root of execution time



Extended Graham Scan

Heuristics Implemented to Improve Execution Time

To extend our graham-scan, we decided the sorting algorithm had to perform better in the worst case. Therefore, we decided to use the timsort algorithm, which splits a list to be sorted into blocks called runs. It then performs an insertion sort on those blocks and merges these blocks to fully sort the list. This was an improvement to the quick sort algorithm we previously used as timsort has both an average and worst case time complexity of $O(nlog(n))$

Furthermore, we came to the realisation that certain points in the input set will definitively *not* be in the convex hull and we could implement quicker methods to eliminate them.

The heuristic we used for this pre-processing is called the Akl Toussaint.
We first found the 4 points with minimum and maximum x and y coordinates.
Next, we removed all the points inside the quadrilateral formed by those four vertices.
This heuristic reduced the number of points N to approximately N/2 (as the Akl Toussaint quadrilateral encompasses about half of the map), reducing the constant terms in time-complexity to $O(\frac{N}{2}log(\frac{N}{2}))$

Theoretical Order-of-Growth for **average case**

For a **random input** of size n, Time Complexity ~ $O(\frac{n}{2}log(\frac{n}{2}))$

The reason for this time complexity is because, having eliminated about half the points through Akl Toussaint, the extended graham scan algorithm:

- Finds the lowest leftmost point on the map - $O(n/2)$ time.
- Sort all the points on the map by polar angle with timsort - $O(n/2log(n/2))$ time
- In the case that some points are repeated, those points are then sorted by distance - $O(n/2)$ time
- Finally, all points are pushed and popped on the stack at most once - $O(n/2)$ time.

This gives an overall time complexity of

$$O(\frac{n}{2}) + O((\frac{n}{2})log(\frac{n}{2})) + O(\frac{n}{2}) + O(\frac{n}{2}) \approx O((\frac{n}{2})log(\frac{n}{2}))$$

Theoretical Order-of-Growth for **worst-case**

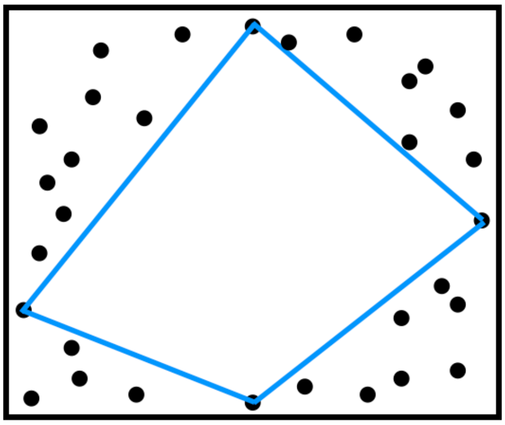


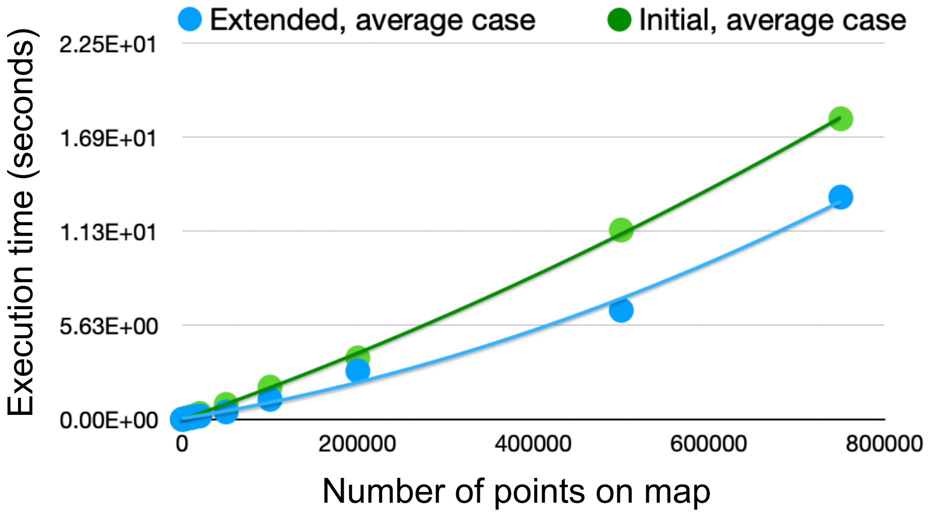
Figure1: worst-case Akl Toussaint quadrilateral. All the points are distributed in the four triangular areas outside the computed quadrilateral

While the category of order-of-growth O~$nlog(n)$ is not affected by the input-set, the number of points eliminated by the Akl Toussaint heuristic certainly is. As the diagram shows, it's possible for a set of points on the map to be arranged such that the quadrilateral eliminates none of them.

Thus the theoretical worst-case Time Complexity is that of running the extended scan without the heuristic altogether, so that no points are removed from the input set.
This is O~$(nlog(n))$.

Experimental Order-of-Growth for **average case** (vs initial graham-scan)

Relationship between number of points and average-case execution time for two graham-scan programs



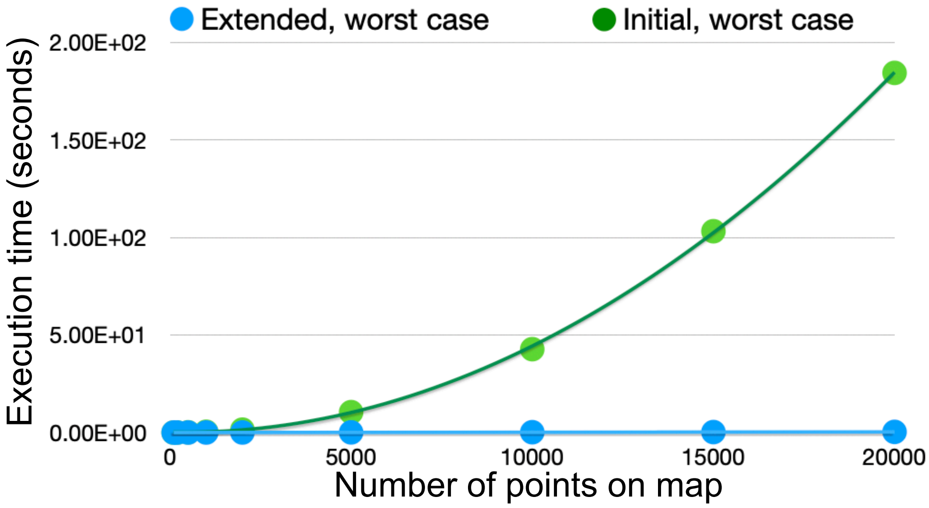
The **blue** line is the average-case for the extended graham scan, while the **green** line is the average-case for initial

The graph supports the theoretical claim that the time-complexity is $O(n/2log(n/2))$ as there is a linearithmic correlation between time-taken and average execution time.

It also has a linearithmic correlation, but is consistently slower at every data point. This is to be expected as the difference between average and worst-case is, effectively, a different constant on n.

Experimental Order-of-Growth for **worst case**

Relationship between number of points and worst-case execution time for two graham-scan programs



The worst case scenario for extended graham-scan (**blue** line) was created by manually skipping the Akl Toussaint method-call (so the set of possible points on the hull doesn't reduce as a result).

It is clearly faster than the worst-case for the initial graham-scan, which had O~(n^2) Time Complexity.

Overall, through the use of the heuristic and tim-sort, the extended graham scan is faster for both the average and worst-case. We could perhaps explore using different shapes (e.g., removing points from an octagon) for our heuristic, or using other sort methods like quad-sort, in the future.