Group 36

Darmon, Joshua
Nicholas, Rory
Rahman, Linta
See, Elyn

Investigating the Performance of Algorithms for Computing a Convex Hull

Jarvis March

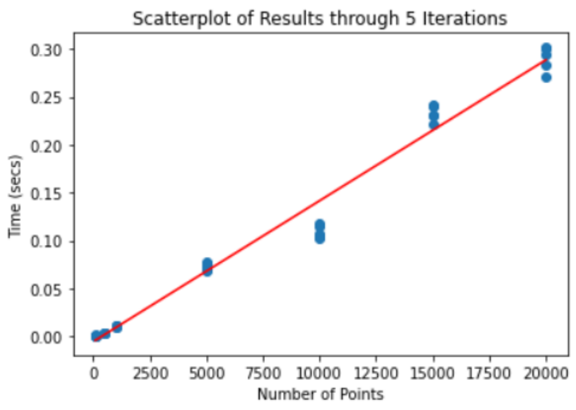
The Jarvis March algorithm, also known as the 'Gift-Wrapping' algorithm, finds the convex hull by 'wrapping' around the set of points to find the outermost path around them. This can be compared to having nails sticking into a wooden board, and using a string to wrap around it, creating an envelope with all the points inside (Cormen, 2001).

Computational Complexity

Random Case Input:

The algorithm works by starting with the leftmost point, one which will certainly be in the convex hull. Then, it compares this point with every other point in the set, using the dot product to find the most anti-clockwise point, which will be the next point in the hull. The process is repeated for every new point in the hull, until we reach the initial first point. This means for every point that is on the hull, we check all n other points. So, the complexity of the algorithm is O(nh), where n is the number of points in the set and h is the number of points on the convex hull. This means the complexity of the Jarvis-March algorithm is output-sensitive.

Random cases can be generated by producing a specific number of random data points over 5 iterations to obtain more test results. The execution time over 5 iterations is then plotted on a scatter graph, where a line of best-fit is drawn to show the trend of the results as shown below:

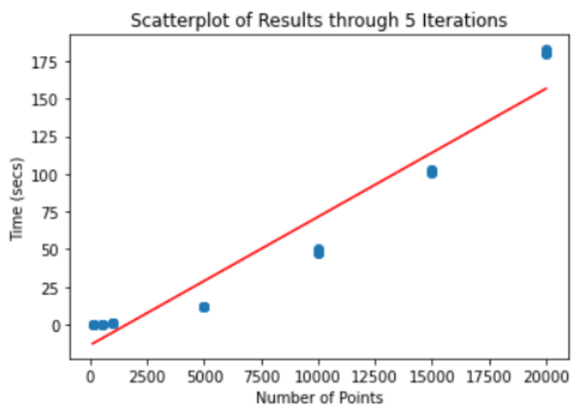


Worst Case Input:

The average case complexity for Jarvis-March is O(nh), where n is the number of points in the set and h is the number of points that lie on the hull. If the points on the hull make up a small proportion of the points, the actual complexity will be close to n.

However, in the worst case all the points in the graph lie on the convex hull. This would mean h is equal to n, giving a complexity of $O(n^2)$.

We produced the worst-case scenario of Jarvis-March by plotting a specific number of points in a circle over 5 iterations to obtain more test results. The worst-case scenario performed significantly worse than the random case scenario, due to the complexity of $O(n^2)$. Similar to the random-case scenario, we plotted the results on a scatter graph as shown below:



In the average case, the Jarvis-March algorithm grew in a somewhat consistent linear way, as the number of points in the hull is usually not very high compared to n and so doesn't impact the growth very significantly. However, in the worst-case scenario, it is evident that the rate of growth increases as n increases, showing the n^2 pattern suggested in theory.

Graham Scan

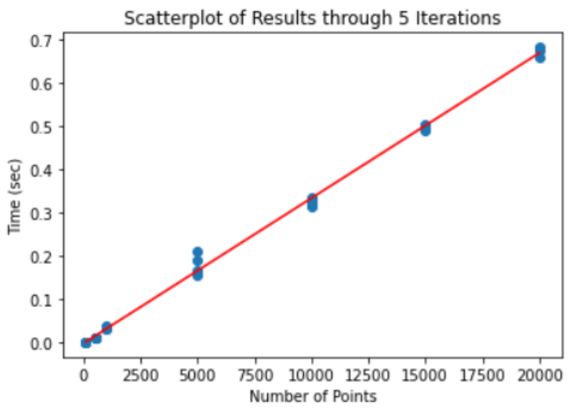
The Graham Scan algorithm computes the convex hull by going through a set of points (x, y) in a counter-clockwise way. It is achieved by finding the point with the lowest y. If there are multiple, the one with the lowest x will be selected. This point will act as a reference to sort the rest of the point in ascending order via polar angles. The set will then be iterated through, for each point the algorithm will check whether a clockwise turn or a counter-clockwise turn has been made. If a clockwise turn has been found the latest point that was added to the hull will be removed, and the current one will be rechecked. This continues until the list has been fully iterated through.

The sorting algorithm we've chosen is Heapsort because it allows consistent average cases and worst cases with O(n log n) with n being the total number of points in the set.

<u>Computational complexity:</u>

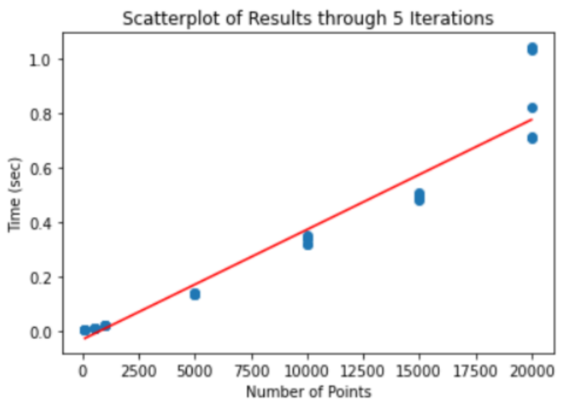
For the Graham Scan's time complexity, finding the reference point takes O(n) time, sorting the set of points according to their polar angle with the reference point with Heap sort takes O(n log n) time, removing the colinear points take O(n) time and finally the scanning itself takes O(n) time as each point will either be part of the hull or get discarded and thus each point will only be iterated through once. The highest order of growth between those steps is the sorting algorithm, which means that it has O(n log n).

Random Case Input:



Worst Case Input:

As the time complexity of Graham's scan is the same as its sorting algorithm, we can deduce that the theoretical worst-case scenario is the same as Heap sort, which is O(n log n).



Although they have the same O, the algorithm can be made slower by increasing the number of operations the algorithm will go through. As shown by the graphs, in the average case Graham Scan achieves an order of growth of (n log n) in practice. The order of growth for the worst-case input seems to stay the same, however the additional operations cause it to compute slower than the average case. The impact of this varies more as n increases, with the higher numbers of points being less consistent in terms of computation time.

Darmon, Joshua
Nicholas, Rory
Rahman, Linta
See, Elyn

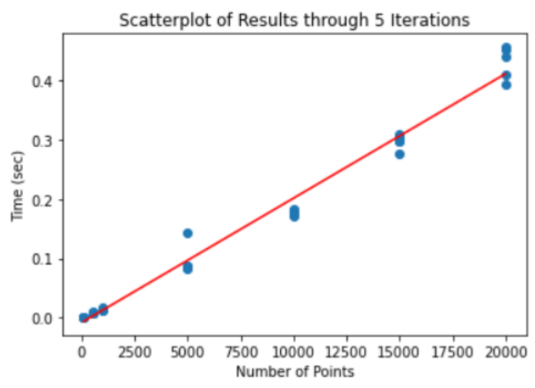
Extended Graham Scan

Akl-Toussaint Heuristic: (Souviron, 2013)

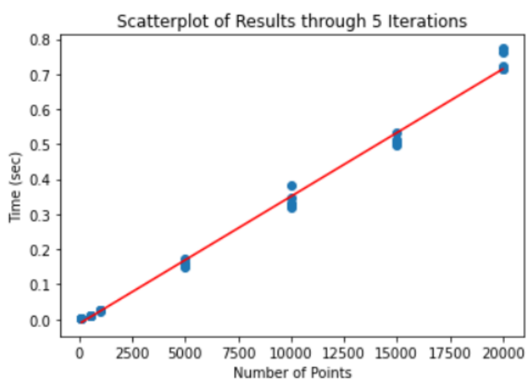
The Akl-Toussaint heuristic allows one to eliminate points which can said to certainly not be in the convex hull. By finding the highest, lowest, leftmost, and rightmost points in the set, a quadrilateral can be created such that any point falling inside this quadrilateral can be said to certainly not be in the convex hull. By pre-processing the points with this heuristic, the workload on the hull-calculating algorithm (in this case, Graham Scan) is greatly reduced.

To implement this, the algorithm first iterates through all n points, finding the corners. Then, each point is looped through again to check whether it's contained within the quadrilateral. Both steps of this process take n operations, meaning the Akl-Toussaint heuristic takes 2n operations, so the order of growth of the optimisation O(n). Overall, the order of growth of the entire algorithm will remain the same as Graham's scan, that is O(n log n).

Random Case Input:



Worst Case Input:



It is evident from the graphs that the order of growth of the extended algorithm remains the same as the original Graham Scan, however the reduced number of points causes the algorithm to run faster.

In practice, we saw that given an average input set, the optimisation reduced the number of points to around 30-50% of the original number. This is reflected in the graphs; in the average case the Graham Scan is around 40% faster after the heuristic is implemented. The difference between the time complexity for random case graphs for the Graham Scan with and without the heuristic demonstrates how the computational time has decreased with the implementation of the heuristic.

However, in the worst case the input set will form close to a convex shape already. This will reduce the number of points the heuristic eliminates, to a minimum of zero points excluded. This means the algorithm will perform at the same speed as without optimisation, which can be seen in the worst-case results for Graham Scan both with and without the optimisation.

Darmon, Joshua
Nicholas, Rory
Rahman, Linta
See, Elyn

Bibliography:

Souviron, J. (2013). Convex hull: Incremental variations on the Akl-Toussaint heuristics Simple, optimal and space-saving convex hull algorithms. *arXiv:1304.2676 [cs]*. [online] Available at: [1304.2676] Convex hull: Incremental variations on the Akl-Toussaint heuristics Simple, optimal and space-saving convex hull algorithms (arxiv.org) [Accessed 18 Mar. 2021].

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. "33.3: Finding the convex hull". Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill. pp. 955–956. ISBN 0-262-03293-7. [Accessed 1 Mar. 2021].

Graham, R.L. (1972). "An Efficient Algorithm for Determining the Convex Hull of a Finite Planar Set" (PDF). *Information Processing Letters*. **1** (4): 132–133. doi:10.1016/0020-0190(72)90045-2. [Accessed 8 Mar. 2021].