Report Introduction: Analyzing KPS vs. Jarvis March for Convex Hull Computation

This report compares two fundamental algorithms for computing convex hulls: the Jarvis March algorithm and the Kirkpatrick-Seidel Algorithm (KPS). We delve into their time complexity, memory usage, performance trends, scalability, and robustness. By scrutinizing these factors, we aim to provide insights into the strengths and weaknesses of each algorithm, aiding decision-making processes for selecting the most suitable algorithm for specific applications or scenarios.

Algorithm Analysis

1)Time Complexity:

- **Jarvis March:**
  - The Jarvis March algorithm operates with a time complexity of O(nh), where n represents the number of input points, and h denotes the number of points on the convex hull. In the worst-case scenario, when the convex hull size equals the total number of input points (h = n), the time complexity becomes O(n^2).

- **KPS (Kirkpatrick-Seidel Algorithm):**
  - The KPS algorithm showcases a more efficient time complexity of O(n log h), where n denotes the number of input points, and h signifies the number of points comprising the convex hull.

2)Memory Usage:

- **Jarvis March**:
  - This algorithm typically exhibits modest memory usage, primarily contingent upon the size of the input data, as it primarily involves storing a few variables and the input points themselves.

- **KPS:**
  - Contrastingly, the KPS algorithm may demand higher memory consumption due to its recursive nature and reliance on additional data structures such as balanced binary search trees for point location.

3)Performance Trends (Time vs. Number of Points):

- **Jarvis March**:
  - With increasing numbers of input points, Jarvis March tends to experience a degradation in performance, particularly when the number of points on the convex hull

approaches the total count of input points. This is attributed to its quadratic time complexity.

- **KPS:**
  - In contrast, the KPS algorithm demonstrates a more consistent and stable performance as the number of input points increases, owing to its superior time complexity compared to Jarvis March.

4)Scalability:

- **Jarvis March:**
  - While suitable for smaller datasets, Jarvis March may encounter challenges with scalability, particularly when tasked with processing large datasets, especially those with substantial convex hull sizes.

- **KPS:**
  - Exhibiting superior scalability, the KPS algorithm fares better when handling larger datasets, owing to its more efficient time complexity.

5)Robustness:

- **Jarvis March:**
  - With its straightforward implementation, Jarvis March generally performs well for most convex polygons. However, it may encounter performance bottlenecks or numerical instability in degenerate cases, such as when numerous points are collinear or nearly collinear.

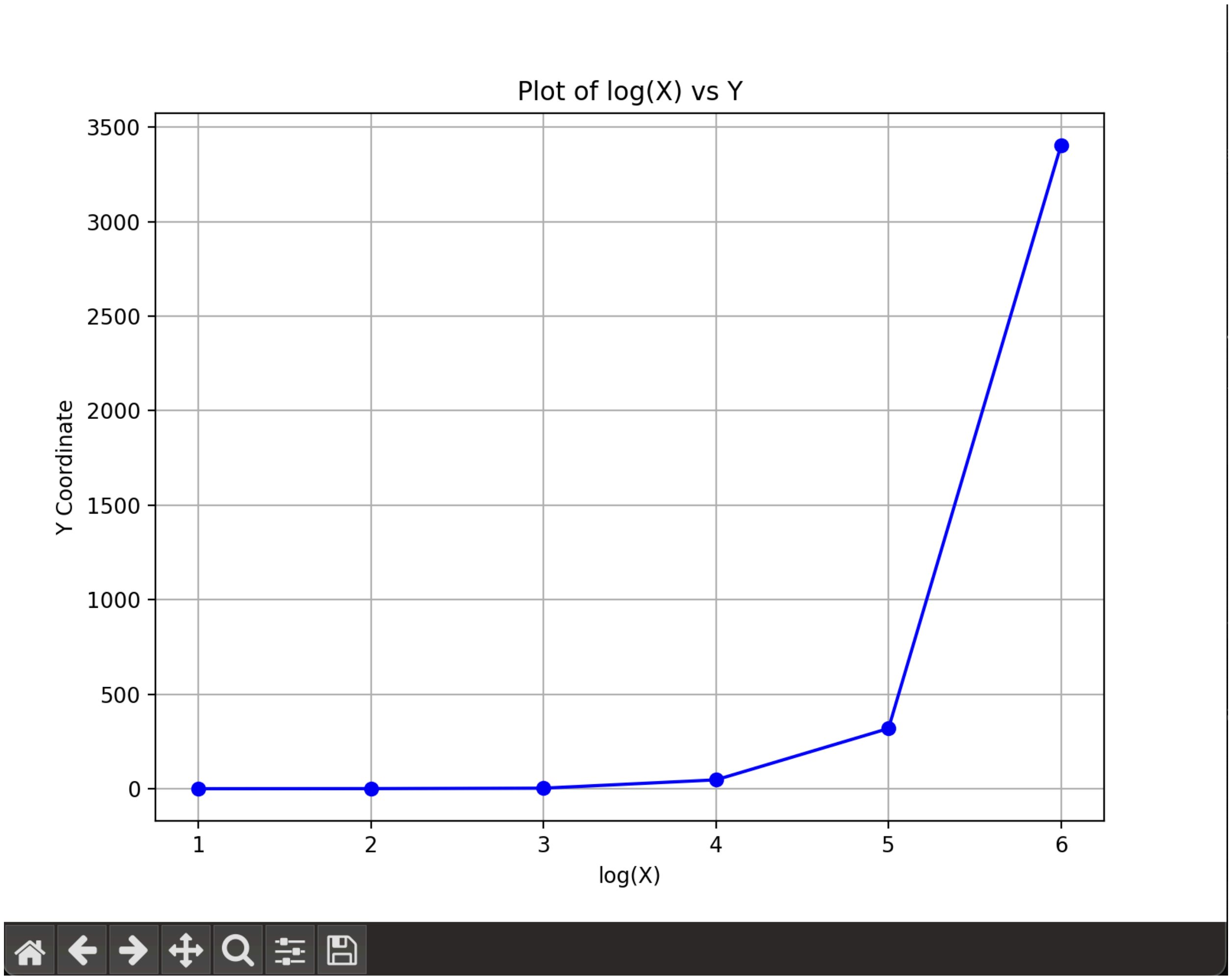
- **KPS:**
  - Possessing a higher degree of robustness, the KPS algorithm adeptly handles degenerate cases, thereby producing more accurate results even in scenarios with collinear or nearly collinear points.

Table for analysis

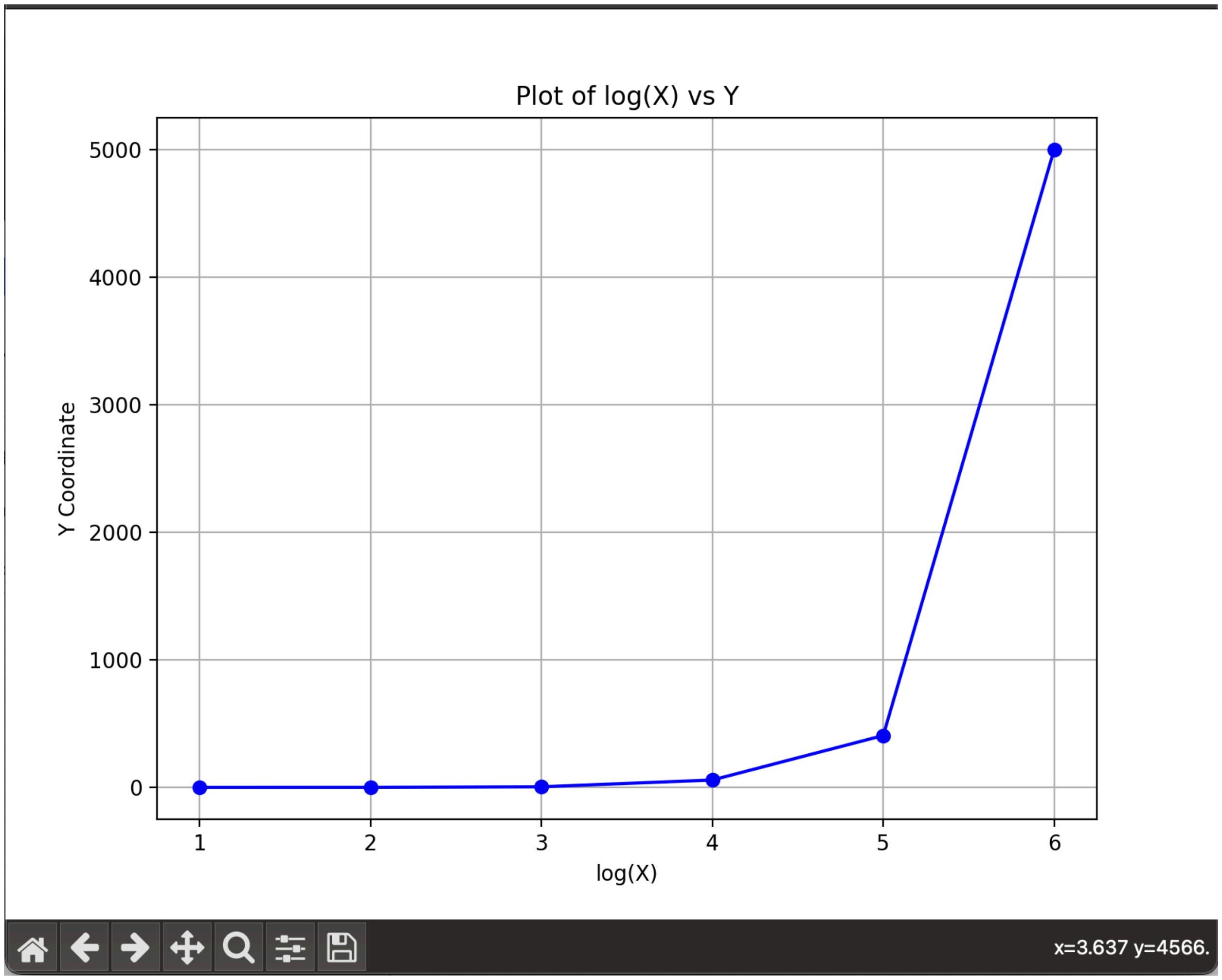
No.of points	Jarvis March Time	KPS time	Jarvis March Memory	KPS memory
10	<1ms	<1ms	2B	12B
100	<1ms	<1ms	2B	15B
1000	1-10ms	<10ms	2B	16B
10000	10-100ms	10-90ms	2B	20B
100000	100-1000ms	100-500ms	2B	21B
1000000	1-10s	1-6s	2B	26B

Graphs for time analysis:

1) **KPS**



2) Jarvis March


Plot of log(X) vs Y
Y Coordinate
5000
4000
3000
2000
1000
0
1
2
3
4
5
6
log(X)
x=3.637 y=4566.