

NATIONAL UNIVERSITY OF COMPUTER AND EMERGING SCIENCES

(KARACHI CAMPUS)

**(PARALLEL AND DISTRIBUTED COMPUTING PROJECT)**

PROJECT TITLE

**"Efficient Parallelization of Jarvis March and Binary Search Tree Algorithms with OpenMP and MPI”**

COURSE INSTRUCTOR

**Ms. Nausheen Shoaib**

**TEAM MEMBERS:**

Muhammad Tahir(21K-4503)

Insha Javed(21K-3279)

Sabika Shameel(21K-4606)

**INTRODUCTION:**

* **Optimizing Jarvis March: A Parallel vs. Serial Approach:**

Jarvis March, also known as the Gift-Wrapping algorithm, is a convex hull algorithm used in computational geometry. the algorithm efficiently computes the convex hull of a set of points in a plane, which is the smallest convex polygon that encloses all given points.

It is a key element in convex hull computation. As part of our PDC project we aim to compare the efficiencies of these implementations. Through an analysis of execution times, scalability, and resource utilization, our investigation aims to highlight the advantages and challenges associated with parallelizing the Jarvis March algorithm.

* **Binary Search Tree: A Parallel vs. Serial Approach:**

A binary search tree (BST) is a hierarchical data structure where each node has a key, and values in the left subtree are smaller while those in the right subtree are larger. This organization enables efficient search, insertion, and deletion operations with logarithmic time complexity. Widely used in computer science, BSTs are crucial for tasks like sorting, symbol tables, and database systems due to their simplicity and versatility. We analyze the performance distinctions between traditional serial BST structures and their parallelized counterparts. Examining factors like search times, scalability, and resource utilization, our study unveils the advantages and challenges in parallelizing Binary Search Tree implementation, offering insights for more efficient data structure management in parallel computing.

**Methodology:**

* Binary Search Tree Pseudocode:
* [Searching in Binary Search Tree (BST) - GeeksforGeeks](https://www.geeksforgeeks.org/binary-search-tree-set-1-search-and-insertion/)A screenshot of a computer code

  Description automatically generated

A screenshot of a computer code

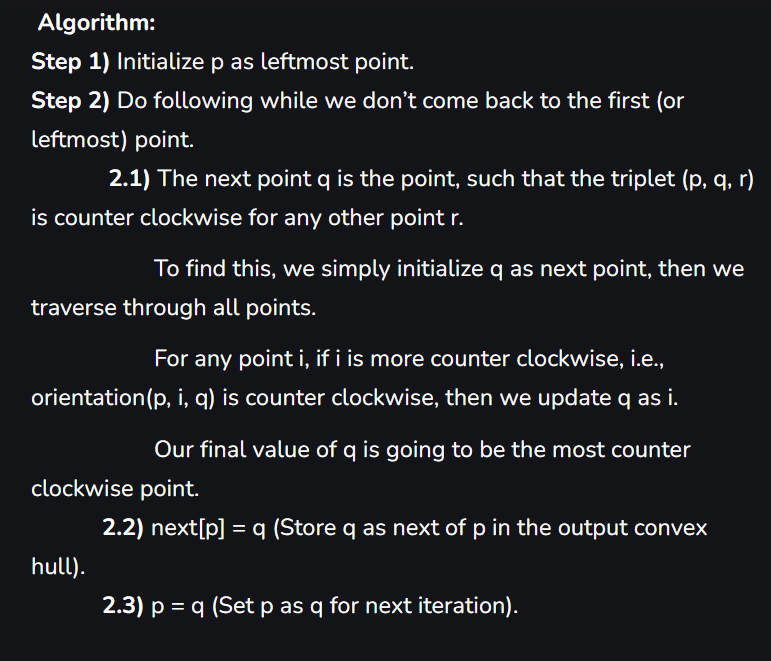
Description automatically generated

* <https://github.com/dvasavda/openmp-binary-search/blob/master/Report.pdf>

A screenshot of a computer program

Description automatically generated

* [Parallel Programming With MPI :: High Performance Computing (nmsu.edu)](https://hpc.nmsu.edu/discovery/mpi/programming-with-mpi/)
* **Jarvis March Serial Algorithm- A convex Hull solution:**



* [Convex Hull using Jarvis' Algorithm or Wrapping - GeeksforGeeks](https://www.geeksforgeeks.org/convex-hull-using-jarvis-algorithm-or-wrapping/)

**JARVIS MARCH USING MPI:**

If you want to parallelize the sorting part of the Jarvis March algorithm, you can consider using parallel sorting algorithms or parallelization frameworks. However, keep in mind that parallelizing sorting can be challenging due to dependencies between elements.

**1.**Divide the Input: Split the set of points into smaller subsets that can be processed independently.

**2**.Sort Subsets in Parallel: Use a parallel sorting algorithm (like parallel quicksort) to sort each subset independently. There are parallel sorting libraries available in various programming languages that you can leverage.

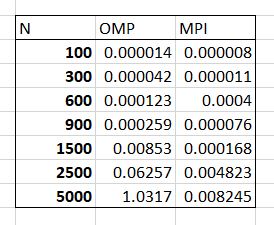
**3.**Combine Results: Once each subset is sorted, combine the results. This may involve merging the sorted subsets, or you may not need to merge if the subsequent steps of the Jarvis March algorithm can work with independently sorted subsets.

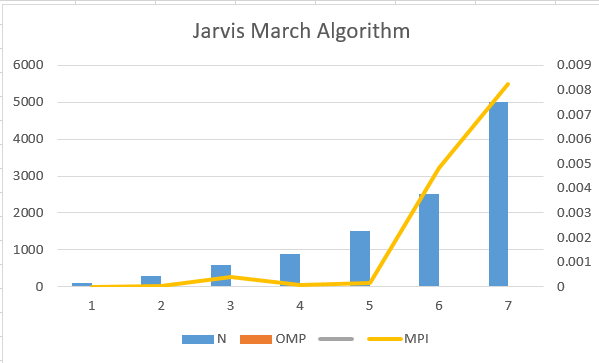
**4.**Apply Jarvis March Algorithm: After sorting, you can then apply the sequential steps of the Jarvis March algorithm to find the convex hull using the sorted points.

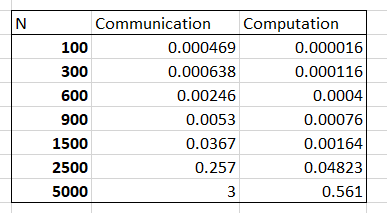
* <https://repository.lsu.edu/cgi/viewcontent.cgi?article=7028&context=gradschool_disstheses>
* [Parallel Quick Sort (opengenus.org)](https://iq.opengenus.org/parallel-quicksort/)

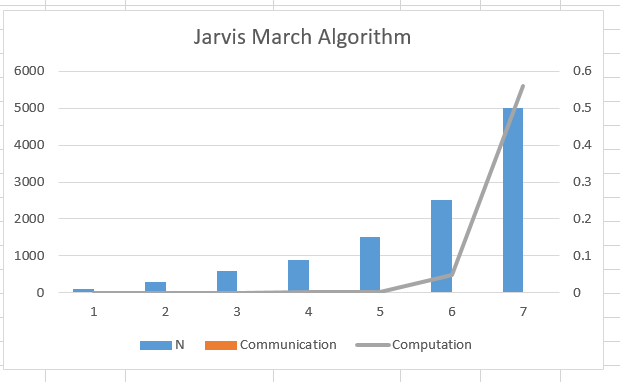
**PERFORMANCE COMPARISON (OPENMP vs MPI):**

* JARVIS MARCH ALGORITHM:

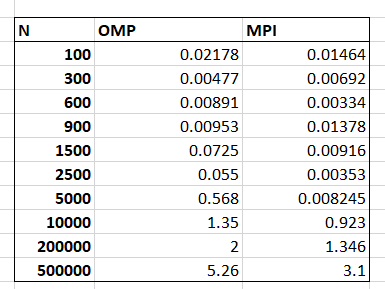


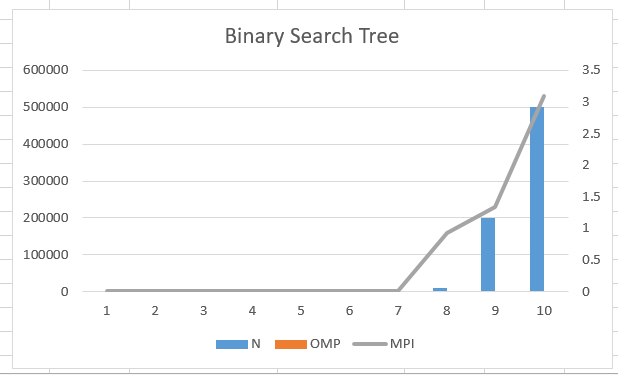


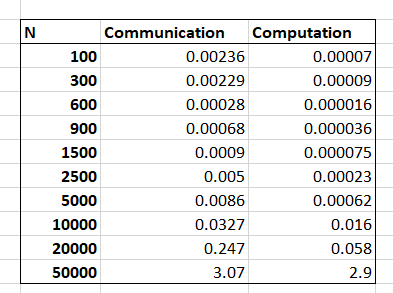


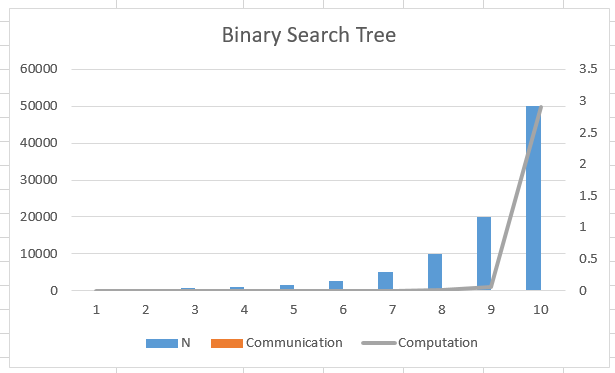


BINARY SEARCH TREE:



**:**





**CONCLUSION:**

In conclusion, we synthesize our findings from the parallelization efforts on the Jarvis March algorithm and Binary Search Tree. By summarizing the advantages and challenges uncovered during the performance comparisons using OpenMP and MPI, we aim to provide a comprehensive understanding of the implications for computational geometry and data structure management in parallel and distributed computing scenarios. The conclusion will also outline potential avenues for future research and applications of parallel computing techniques in similar algorithms and data structures.