

University of Craiova  
Faculty of Automation, Computers and Electronics



# Dijkstra

Parallel and Distributed Algorithms

**Student**

Smarandache Alexandru

**Computers Romanian Language**

Study Group CR 3.2

**Third Year**

March 2021

# Contents

<b>1</b>	<b>Implementations of Dijkstra's algorithm</b>	<b>2</b>
1.1	C++ Sequential . . . . .	2
1.2	Java Sequential(using priority queue) . . . . .	2
1.3	C++ STL parallel . . . . .	2
1.4	C++ OpenMP . . . . .	2
1.5	C++ MPI . . . . .	2
1.6	Java Parallel Streams . . . . .	2
1.7	Java with Threads . . . . .	2
<b>2</b>	<b>Input data generation</b>	<b>2</b>
<b>3</b>	<b>Output data</b>	<b>3</b>
<b>4</b>	<b>Results</b>	<b>4</b>
4.1	C++ Sequential . . . . .	4
4.2	Java Sequential(using priority queue) . . . . .	4
4.3	C++ STL parallel . . . . .	5
4.4	C++ OpenMP . . . . .	5
4.5	C++ MPI . . . . .	6
4.6	Java Parallel Streams . . . . .	6
4.7	Java with Threads . . . . .	7
<b>5</b>	<b>Comparasion</b>	<b>8</b>
5.1	All implementations . . . . .	8
5.2	C++ implementations . . . . .	8
5.3	Java implementations . . . . .	9
5.4	C++ STL Parallel vs Java Streams Parallel . . . . .	9
5.5	C++ Parallel . . . . .	10
<b>6</b>	<b>Bibliography</b>	<b>11</b>

## 1 Implementations of Dijkstra's algorithm

### 1.1 C++ Sequential

### 1.2 Java Sequential(using priority queue)

### 1.3 C++ STL parallel

### 1.4 C++ OpenMP

### 1.5 C++ MPI

### 1.6 Java Parallel Streams

### 1.7 Java with Threads

## 2 Input data generation

To generate input data, a graph is generated based on a given density.

Input format:

noOfNodes sourceNode

c00 c01 ... c0(noOfNodes -1)

c01 c11 ... c1(noOfNodes -1)

...

c(noOfNodes -1)0 c(noOfNodes -1)1 ... c(noOfNodes -1)(noOfNodes -1)

Where:

- **noOfNodes**: the number of nodes
- **sourceNode**: the source node
- **c**: the adjacency matrix

```

/**
 * Generate a random graph.
 *
 * @param noOfNodes The number of nodes.
 * @param density The density of graph. If density is @Constants.MIN_DENSITY, no edge will be generated.
 * If density is @Constants.MAX_DENSITY, edges between all nodes will be generated.
 *
 * @return The generated graph.
 */
public static Graph generateGraph(final int noOfNodes, final int density) {
    List<List<Integer>> adjacencyMatrix = new ArrayList<>(noOfNodes);
    for (int i = 0; i < noOfNodes; i++) {
        List<Integer> row = new ArrayList<>();
        for (int j = 0; j < noOfNodes; j++) {
            final int grade = RandomUtil.generateNumber(Constants.MIN_DENSITY, Constants.MAX_DENSITY);
            if (i == j) {
                row.add(j, element: 0);
            } else {
                row.add(j, grade > density ? Constants.INFINITE :
                    RandomUtil.generateNumber(Constants.MIN_EDGE_COST, Constants.MAX_EDGE_COST)
                );
            }
        }
        adjacencyMatrix.add(i, row);
    }

    return new Graph(adjacencyMatrix);
}

```

### 3 Output data

Each algorithm will have the following output format:

Vertex Distance from source(source node)

0 c0

1 c1

...

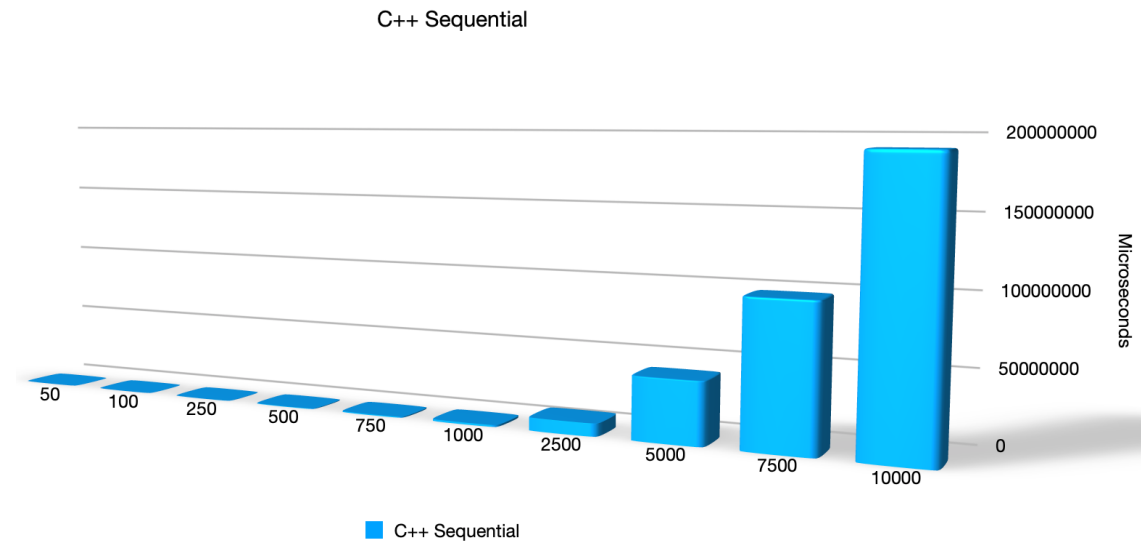
noOfNodes -1 c(noOfNodes - 1)

Where:

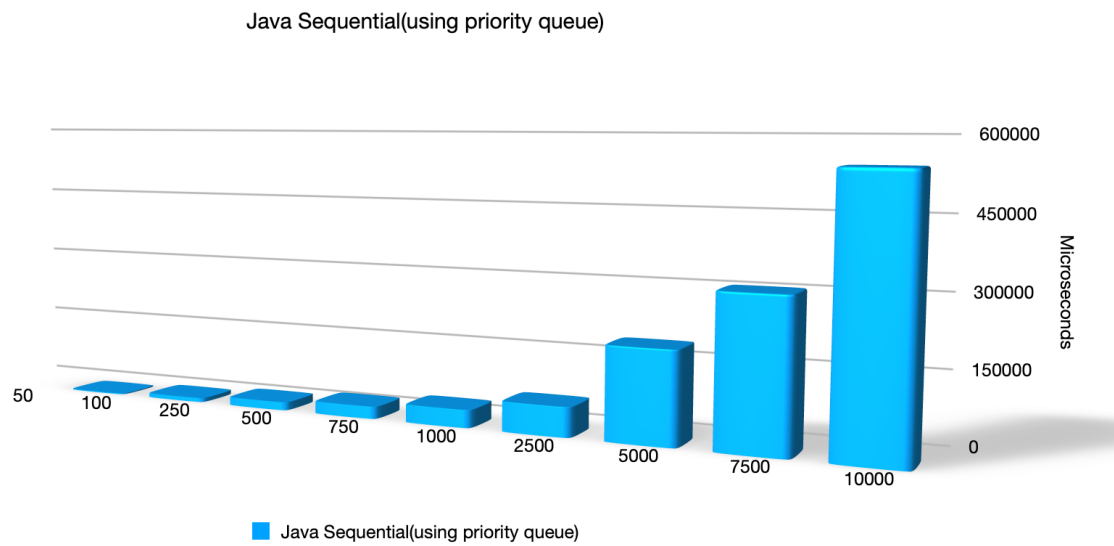
c[i] is the distance between node i and source node.

## 4 Results

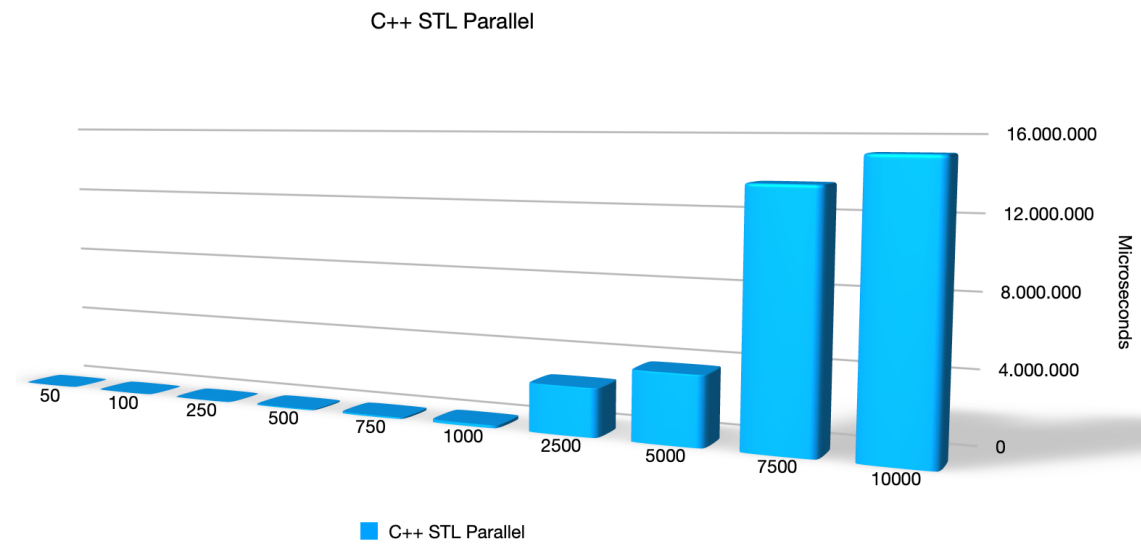
### 4.1 C++ Sequential



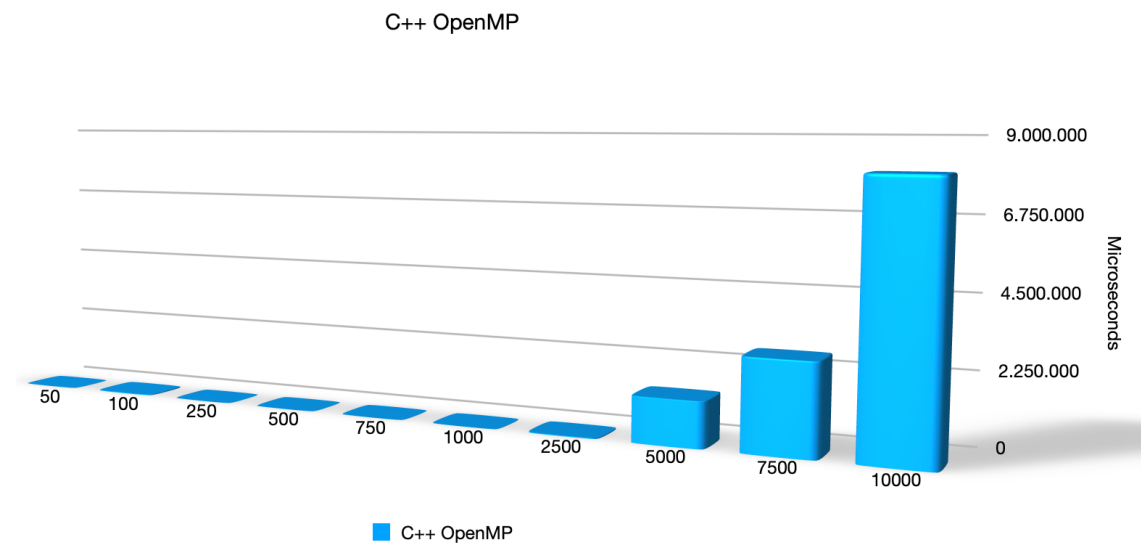
### 4.2 Java Sequential(using priority queue)



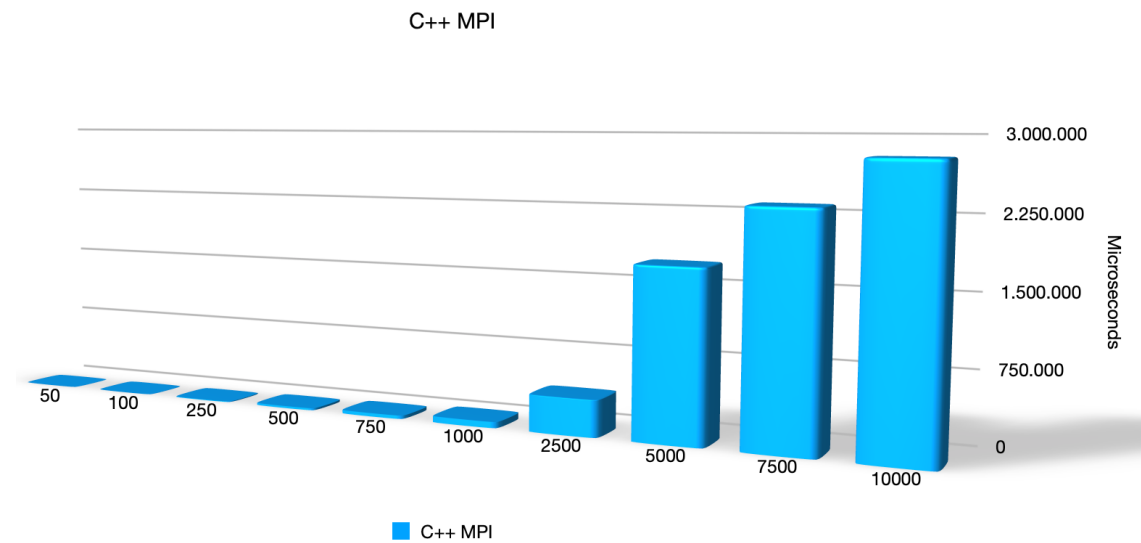
### 4.3 C++ STL parallel



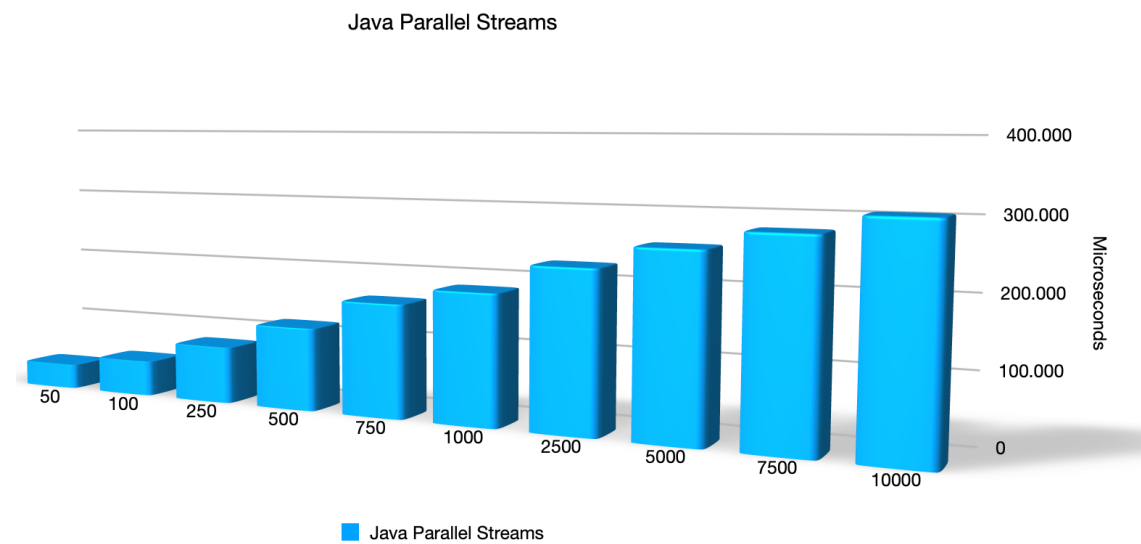
### 4.4 C++ OpenMP



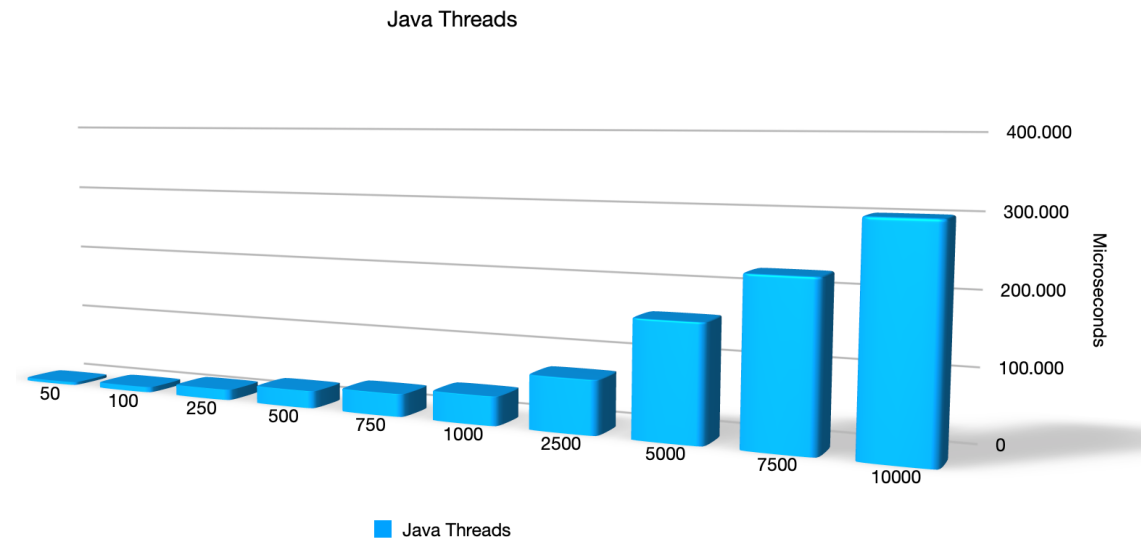
## 4.5 C++ MPI



## 4.6 Java Parallel Streams



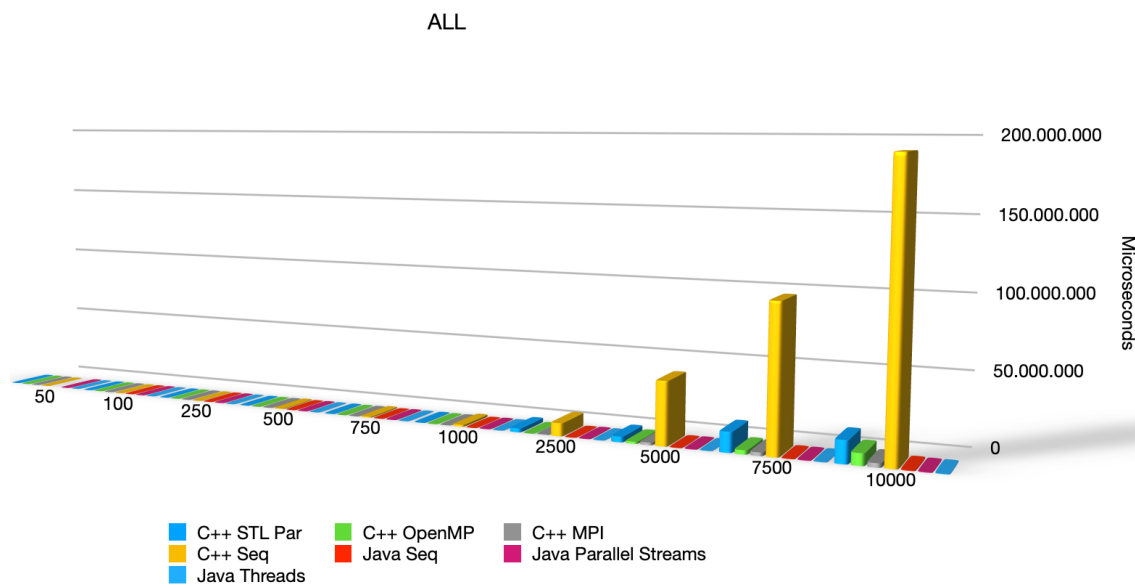
## 4.7 Java with Threads



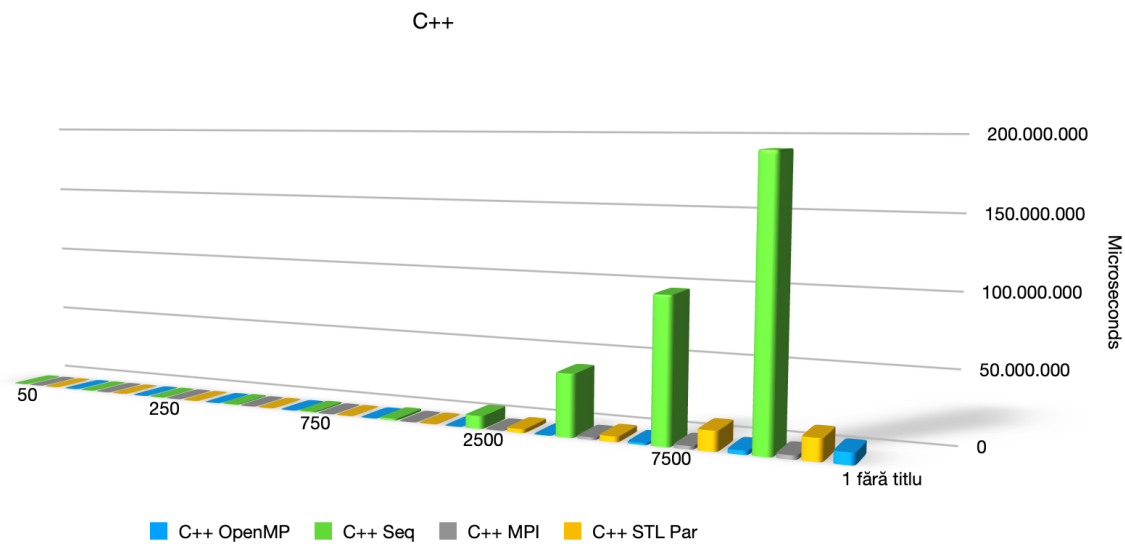


## 5 Comparasion

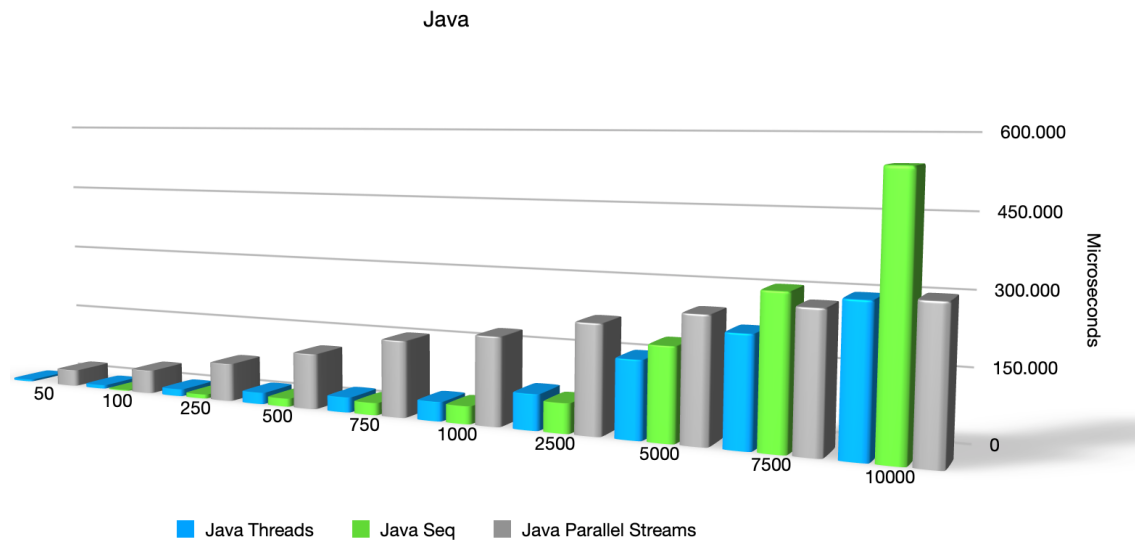
### 5.1 All implementations



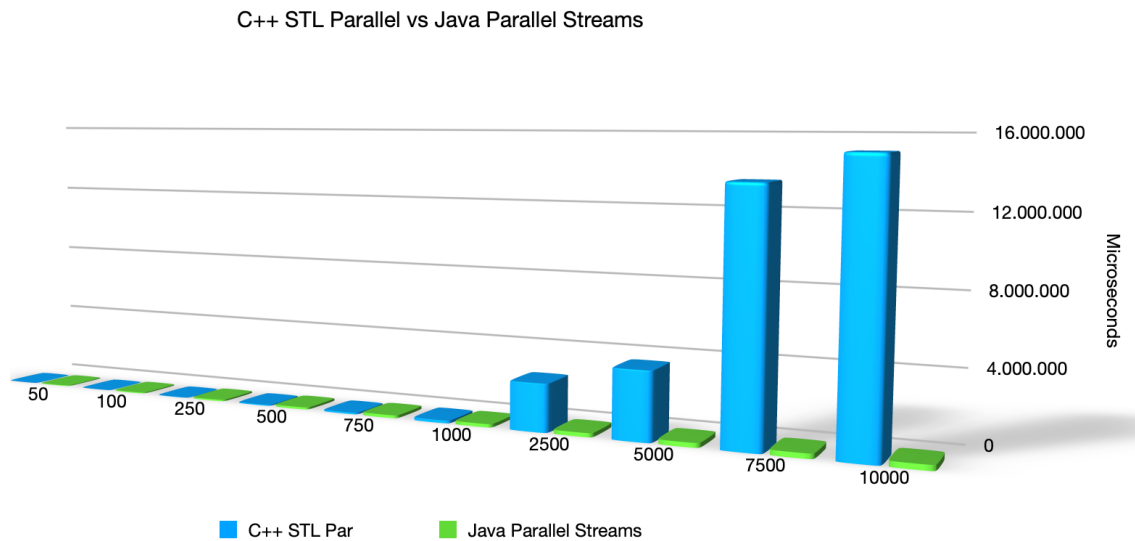
### 5.2 C++ implementations



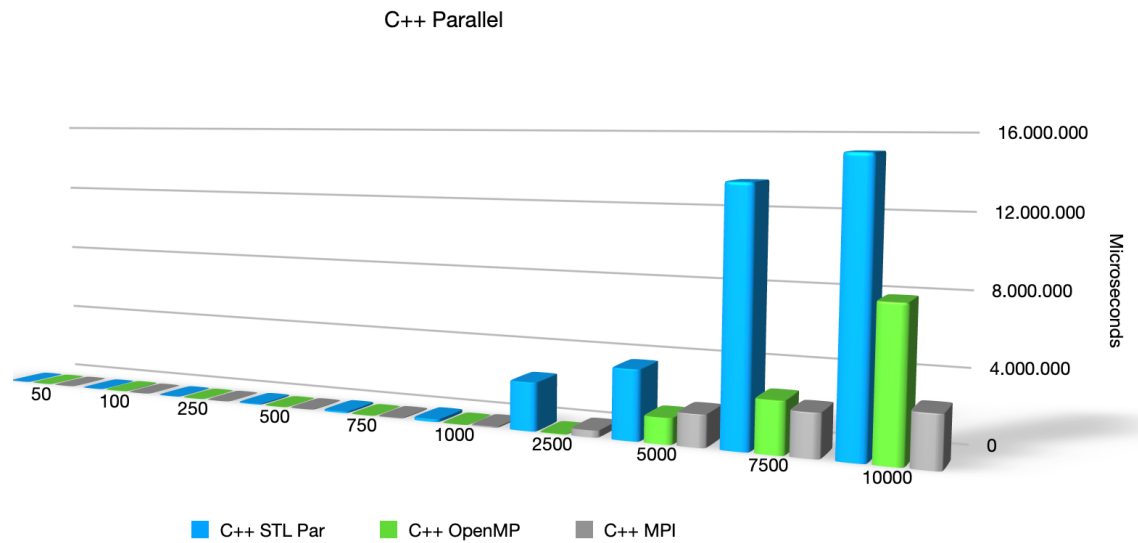
### 5.3 Java implementations



### 5.4 C++ STL Parallel vs Java Streams Parallel



## 5.5 C++ Parallel



### Observation

The environment on which I ran the tests is: Windows 10 Home, 16GB Ram, 512GB SSD, Eclipse for Java programs and Visual Studio 2019 for C++.

## 6 Bibliography

### References

- [1] repository.stcloudstate.ed,[https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1044&context=csit\\_etds](https://repository.stcloudstate.edu/cgi/viewcontent.cgi?article=1044&context=csit_etds), accessed in March 2022.
- [2] geeksforgeeks,<https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/>, accessed in March 2022.