PDP Lab Project – Pugna Bogdan, Rad George-Rares 936/2

Problem:

Find an n-coloring of a graph.

Requirement:

Each student or team of 2 students will take one project. It is ok to take a theme that is not listed below (but check with the lab advisor before starting).

Each project will have 2 implementations: one with "regular" threads or tasks/futures, and one distributed (possibly, but not required, using MPI).

Algorithms:

In order to find the n coloring of the graph we use the Jones-Plassmann method. Which is based on dividing the graph in individual sets of nodes with the criteria that any two nodes are not neighbors. After which, each node from the individual set is colored with the smallest color available globally. This approach is easy to parallelize since each individual set can be computed on a different thread.

The distributed approach is done the same, but this time the individual sets are computed on the worker nodes not on threads.

Sources(http://www.new-npac.org/users/fox/pdftotal/sccs-0666.pdf, https://ireneli.eu/2015/10/26/parallel-graph-coloring-algorithms/)

Synchronization Techniques:

Simple:

```
    Set<Integer> set = getIndependentSet();
    for (Integer node : set) {
        exec.submit(() -> setColor(node));
    }
    independentSet.removeAll(set);
}
exec.shutdown();
```

```
for (Integer node : listOfNodes) {
    Future<Boolean> f = exec.submit(() -> checkNode(node));
    list.add(f);
}
```

MPI:

```
MPI.COMM_WORLD.Send(matrixSize, offset: 0, count: 1, MPI.INT, procToSend, tag: 1);
for (int[] row : adjencyMatrix) {
    MPI.COMM_WORLD.Send(row, offset: 0, row.length, MPI.INT, procToSend, tag: 2);
}
int[] vertex = new int[1];
vertex[0] = v;
MPI.COMM_WORLD.Send(vertex, offset: 0, count: 1, MPI.INT, procToSend, tag: 3);


int[] neighbors = new int[graph.size()];
MPI.COMM_WORLD.Recv(neighbors, offset: 0, graph.size(), MPI.INT, procToSend, tag: 4);
```

Testing:

Size	Time
5	0.0127319 seconds.
10	0.0149332 seconds.
100	0.0318246 seconds.
1000	0.0697162 seconds.