CSE 2046 ANALYSIS OF ALGORITHM

GRAPH COLORING PROBLEM REPORT

07.06.2022

Prepared by

İsmail ÖKSÜZ - 150119516
Emre SAĞIROĞLU - 150119766
Erdi TÜRKAY - 150119853

ABOUT PROJECT

In this project, three different methods were implemented. The results of the three methods can also be output as code.
At the very beginning of the code, the input and output files are assigned. Then the function checkColorOfEdges checks if a node has a neighboring node assigned to the specified color, and will not use that color if it does. Before assigning colors, nodes are sorted from the most edged to the least edged. Then all three methods are processed sequentially.

```
69  function setColor(node) {
70    for (let i = 0; i < colors.length; i++) {
71      if (node.color === "" && checkColorOfEdges(node, colors[i])) {
72        node.color = colors[i];
73        break;
74      }
75    }
76  }
```

Coloring with Greedy Technique

FIRST METHOD: (FirstMethodColorNumber)

The first of the methods we will use to assign colors to vertex is to start from the node with the most neighbors and color them towards the least.

```
// The first method is to start coloring the nodes from the one with the most neighbors
// to the one with the least.
sortNodes.map((node) => setColor(node));

// Get the total number of colors used as a result of the first method and keep the graph info
// We will use this if the first method is the most optimized method.
let firstMethodColorNumber = getColorNumber(sortNodes);
console.log("firstMethodColorNumber: " + firstMethodColorNumber);

let contentFirst = firstMethodColorNumber + "\n";
let str = Nodes.map((e) => e.color);
contentFirst += str.join(" ");
```

Implementation of the First Method

SECOND METHOD: (SecondMethodColorNumber)

In the second method, the BFS Algorithm is used, starting with the node that has the maximum edge

BFS Algorithm →

```
79  function BFS(start) {
80    var listToExplore = [...start.nodes];
81
82    start.visited = true;
83    setColor(start);
84
85    while (listToExplore.length > 0) {
86      var nodeIndex = listToExplore.shift();
87      if (!nodeIndex.visited) {
88        nodeIndex.visited = true;
89        setColor(nodeIndex);
90        nodeIndex.nodes.map((e2) =>
91          !listToExplore.includes(e2) ? listToExplore.push(e2) : null
92        );
93      }
94    }
95  }
```

Implementation of the Second Method

```
144   BFS(sortNodes[0]);
145
146   // Keep the result of the second method.
147   let secondMethodColorNumber = getColorNumber(sortNodes);
148   console.log("secondMethodColorNumber: " + secondMethodColorNumber);
149
150   let contentBFS = secondMethodColorNumber + "\n";
151   let str2 = Nodes.map((e) => e.color);
152   contentBFS += str2.join(" ");
```

THIRD METHOD: (ThirdMethodColorNumber)

The third method colors vertex starting from the node that has the least edge with using the DFS algorithm.

DFS Algorithm

```
163   function depthFirstSearch(currentVertex) {
164     currentVertex.nodes.map((nextVertex) => {
165       if (!nextVertex.visited) {
166         currentVertex.visited = true;
167         depthFirstSearch(nextVertex);
168       }
169     });
170
171     setColor(currentVertex);
172   }
```

```
174   // The third mehtod is applying dfs but starting from the one with the least neighbors.
175   depthFirstSearch(sortNodes[sortNodes.length - 1]);
176
177   // Keep the result of the third method.
178   let thirdMethodColorNumber = getColorNumber(sortNodes);
179   console.log("thirdMethodColorNumber: " + thirdMethodColorNumber);
180
181   let contentDFS = thirdMethodColorNumber + "\n";
182   let str3 = Nodes.map((e) => e.color);
183   contentDFS += str3.join(" ");
184
```

Implementation of the Third Method

CONCLUSION

In conclusion, the results from these 3 different methods are compared and the method with the optimal results is written to the output file.

REFERENCES

-Ömer Korçak

DIVISION OF LABOR

	Code	Report
İsmail ÖKSÜZ	✓	✓
Emre SAĞIROĞLU	✓	✓
Erdi TÜRKAY	✓	✓