COMPSCI 711 - Parallel Computing | Assignment 2

Algorithm

The algorithm chosen to solve the problem utilises BFS to find the eccentricity of a single node, then repeats this for each node in the graph. The minimum of these eccentricities is calculated to be the radius of the graph. It also checks for connectedness, as this is not guaranteed. As it performs BFS, it marks each node as visited, if not all nodes are marked visited at the end of BFS execution, then an exception is thrown.

Sequential Algorithm

```
1  void findRadiusSequential(Graph graph, int numNodes)
2  {
3    vector<int> eccs(numNodes, 0);
4    bool notConnectedFlag = false;
5
6    for (int i = 0; i < graph.V; ++i)
7    {
8      try
9      {
10       int ecc = graph.BFS(i);
11       eccs[i] = ecc;
12     }
13     catch (const char *msg)
14     {
15
16       notConnectedFlag = true;
17       break;
18     }
19   }
20   if (notConnectedFlag)
21   {
22     cout << "None" << endl;
23   }
24   else
25   {
26     cout << *min_element(eccs.begin(), eccs.end()) << endl;
27   }
28 }
```

Parallel Algorithm

```
void findRadiusParallel(Graph graph, int numNodes)
{
  vector<int> eccs(numNodes, 0);
  bool notConnectedFlag = false;

#pragma omp parallel for
  for (int i = 0; i < graph.V; ++i)
  {
    try
    {
      int ecc = graph.BFS(i);
      eccs[i] = ecc;
    }
    catch (const char *msg)
    {
      notConnectedFlag = true;
    }
  }
  if (notConnectedFlag)
  {
    cout << "None" << endl;
  }
  else
  {
    cout << *min_element(eccs.begin(), eccs.end()) << endl;
  }
}
```

Test Data

Design and describe own test cases

Test Cases

- #1-3: The three examples in the assignment brief
- #4: Randomly generated 100 node, 4924 edge, Erdos-Renyi graph
- #5: Randomly generated 300 node, 44822 edge, Erdos - Renyi graph

Reasoning

Test cases #1-3 were chosen to test the correctness of the algorithm, as the expected answer was known for these three. Therefore, these were utilised throughout the development phase to periodically test the algorithm.

Test cases #4 and #5 were needed after the testing phase for performance results, as the original graphs are relatively small (3-5 nodes), and were being processed too quickly. Therefore, graphs test cases 4 and 5 are significantly larger in both number of nodes and edges.

```
n = 300
G = nx.erdos_renyi_graph(n, 0.5, seed=123, directed=True)
for line in nx.generate_adjlist(G):
    print(line)
```

Performance Results

See if expected running times do show the results of parallelism

Compare Sequential results

Algorithm
Sequential
Parallel

Thread Overhead