Q1)

Assuming :

1) in free graph there can be max **inf** vertex possible
2) Adding an edge (a, b) to the graph implies that nodes a and b are also added to the graph
3) **inf** = 100000

```
g = UndirectedGraph(10)
g.addNode(11)
```

```
<class 'Exception'>
Node index cannot exceed number of nodes
```

```
g = UndirectedGraph()
print(g)
```

```
Graph with 0 nodes and 0 edges. Neighbours of the nodes are belows:
```

```
g = UndirectedGraph(5)
print(g)
```

```
Graph with 5 nodes and 0 edges. Neighbours of the nodes are belows:
Node 1: {}
Node 2: {}
Node 3: {}
Node 4: {}
Node 5: {}
```

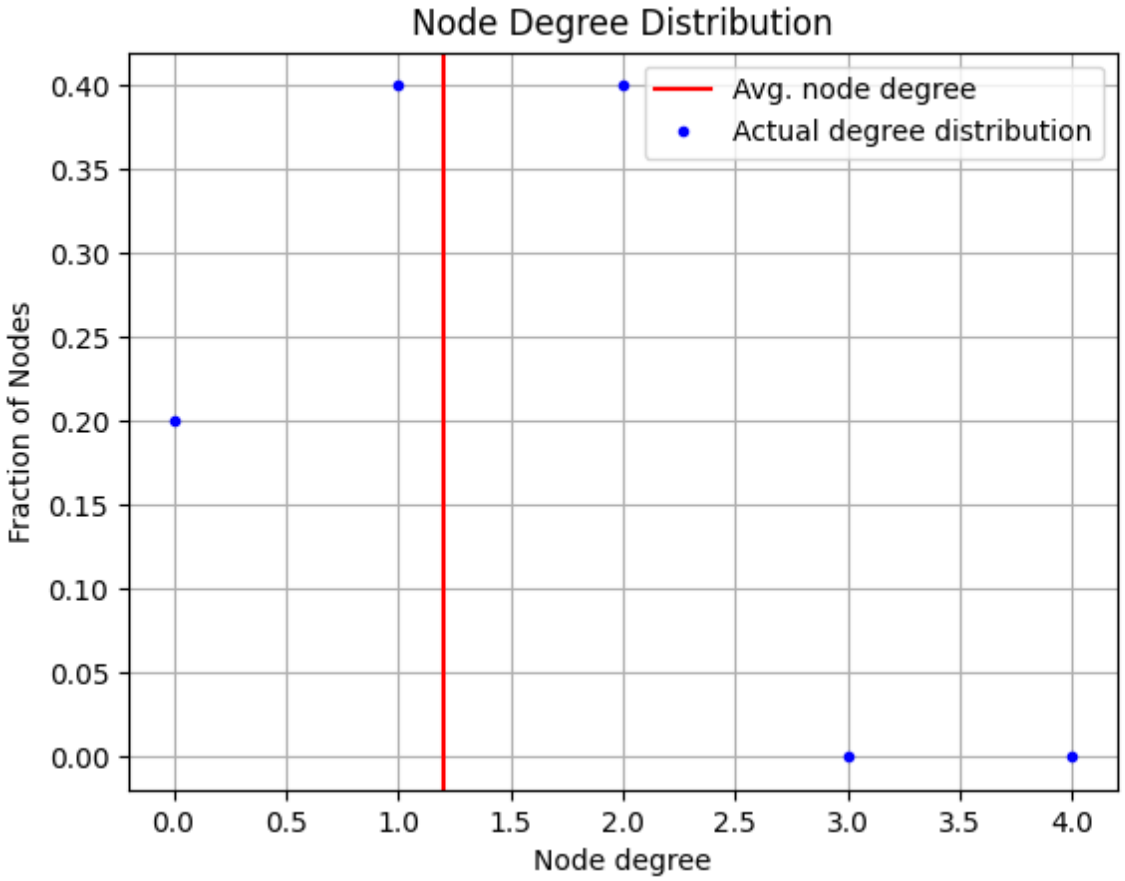
```
g = UndirectedGraph()
g = g + 10
g = g + (11, 12)
print(g)
```

```
Graph with 3 nodes and 1 edges. Neighbours of the nodes are belows:
Node 10: {}
Node 11: {12}
Node 12: {11}
```

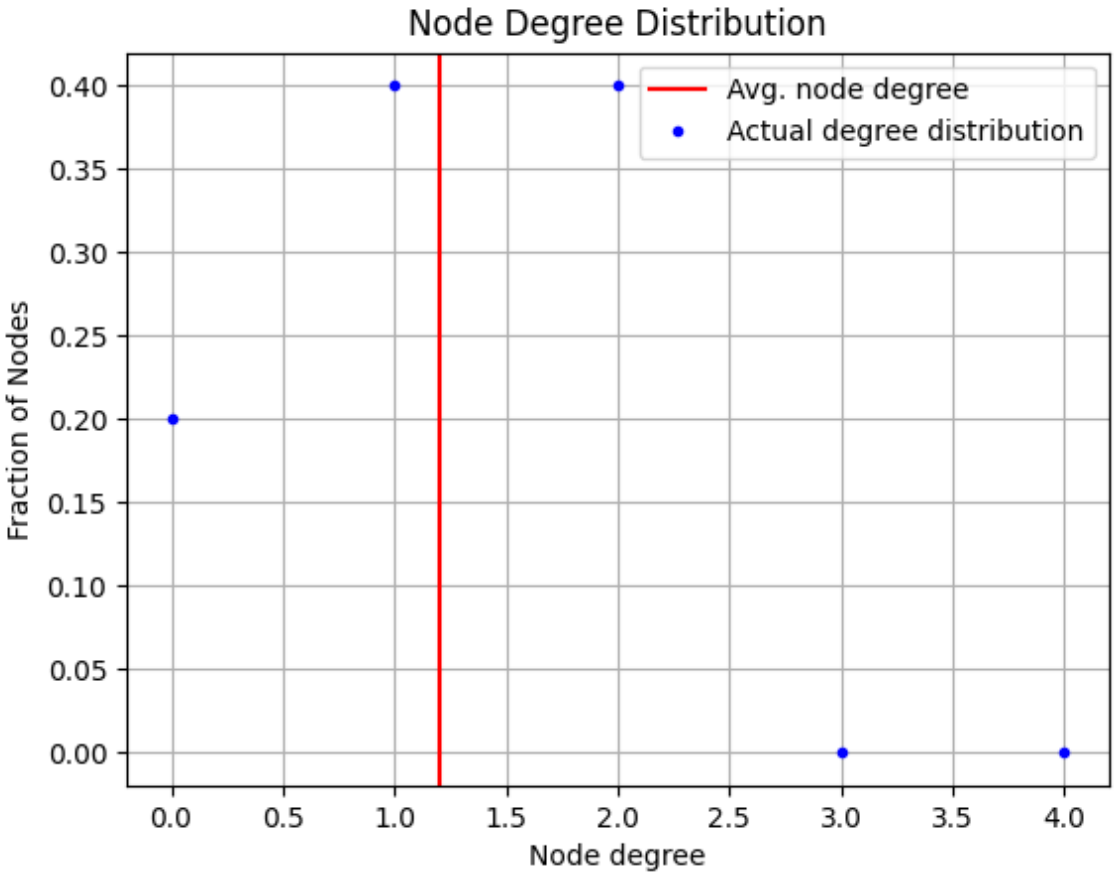
```
g = UndirectedGraph(5)
g = g + (1, 2)
g = g + (3, 4)
g = g + (1, 4)
print(g)
```

```
Graph with 5 nodes and 3 edges. Neighbours of the nodes are belows:
Node 1: {2, 4}
Node 2: {1}
Node 3: {4}
Node 4: {1, 3}
Node 5: {}
```

```
g = UndirectedGraph(5)
g = g + (1, 2)
g = g + (3, 4)
g = g + (1, 4)
g.plotDegDist()
```




```
g = UndirectedGraph()
g = g + 100
g = g + (1, 2)
g = g + (1, 100)
g = g + (100, 3)
g = g + 20
g.plotDegDist()
```

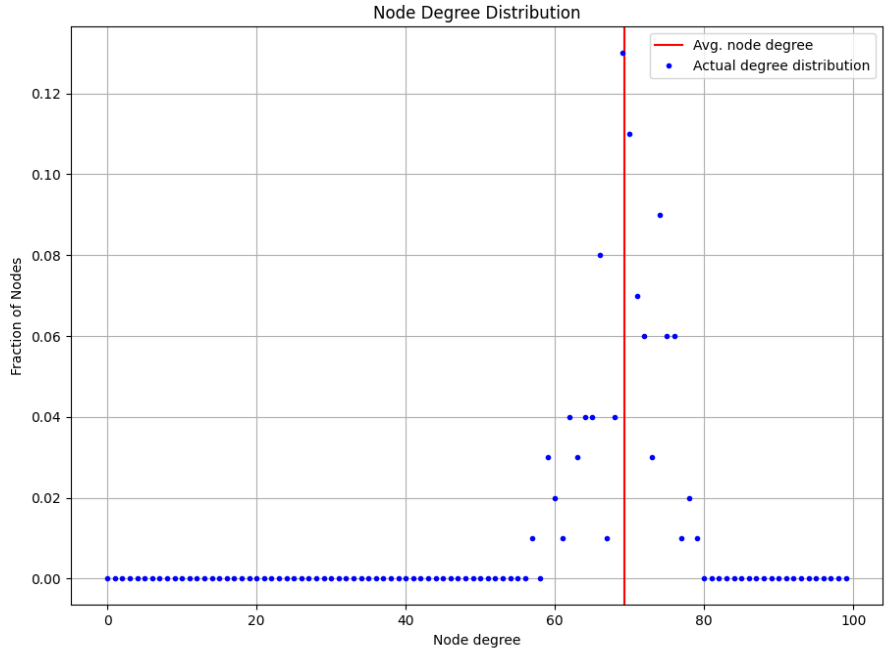


Q2)

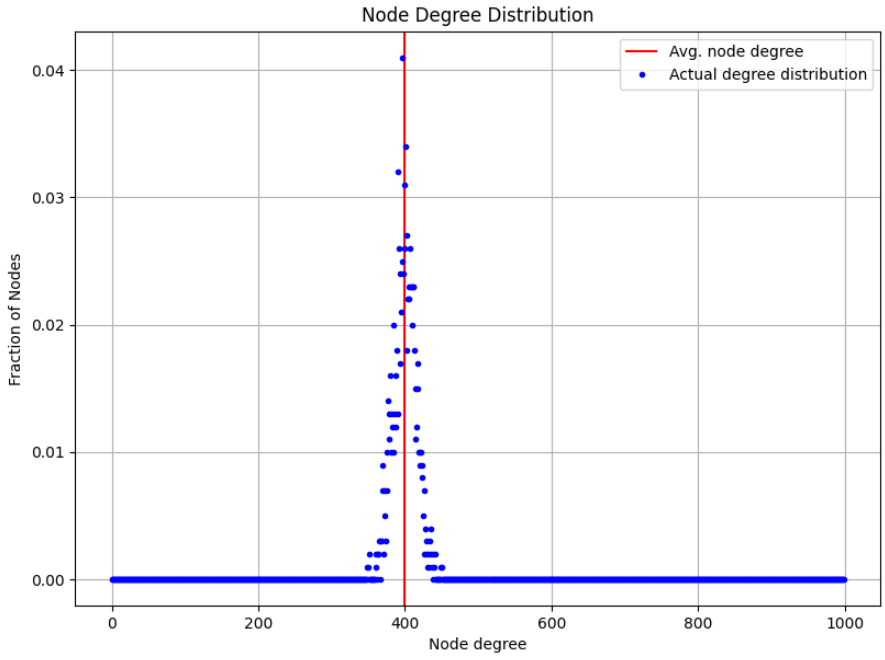
I have imported UndirectedGraph class from q1.py,

To run q2.py, q1.py and q2.py should be in the same directory.

```
g = ERRandomGraph(100)
g.sample(0.7)
g.plotDegDist()
```




```
g = ERRandomGraph(1000)
g.sample(0.4)
g.plotDegDist()
```



Q3)

```
g = UndirectedGraph(5)
g = g + (1, 2)
g = g + (2, 3)
g = g + (3, 4)
g = g + (3, 5)
print(g.isConnected())
```

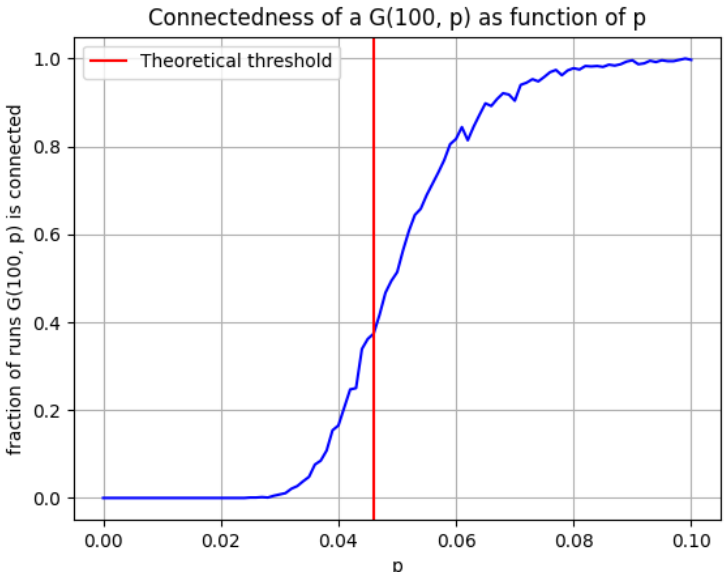
```
True
```

```
g = UndirectedGraph(5)
g = g + (1, 2)
g = g + (2, 3)
g = g + (3, 5)
print(g.isConnected())
print(g)
```

```
False
Graph with 5 nodes and 3 edges. Neighbours of the nodes are belows:
Node 1: {2}
Node 2: {1, 3}
Node 3: {2, 5}
Node 4: {}
Node 5: {3}
```

to verify : Erdos-Renyi random graph G(100, p) is almost surely connected only if p > ln(100) / 100

Time : 70 sec



Q4)

```
g = UndirectedGraph(6)
g = g + (1, 2)
g = g + (3, 4)
g = g + (6, 4)
print(g.oneTwoComponentSizes())
```

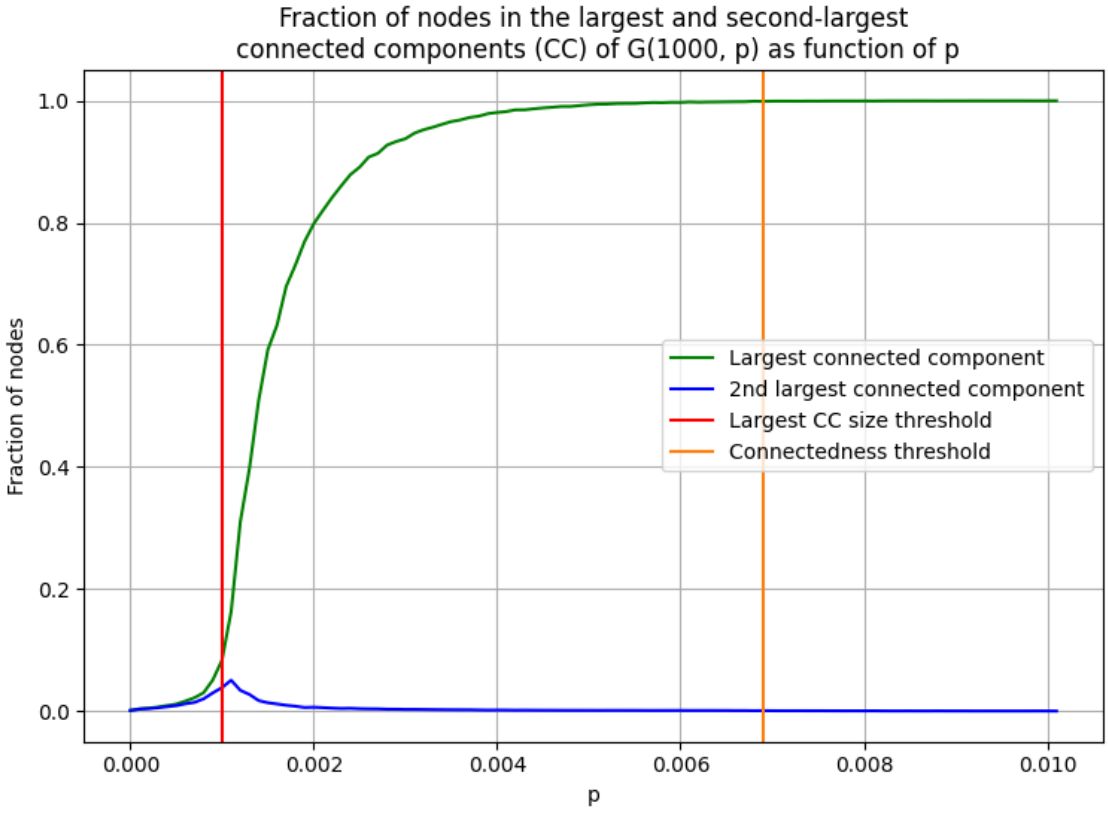
```
[3, 2]
```

```
g = RandomGraph(100)
g.sample(0.01)
print(g.oneTwoComponentSizes())
```

```
[24, 10]
```

to verify : If p < 0.001, the Erdős-Rényi random graph G(1000, p) will almost surely have only small connected components. On the other hand, if p > 0.001, almost surely, there will be a single giant component containing a positive fraction of the vertices.

Time : 140 sec



Q5)

Used networkx inbuilt modules :

grid_2d_graph() : for creating 2d grid.

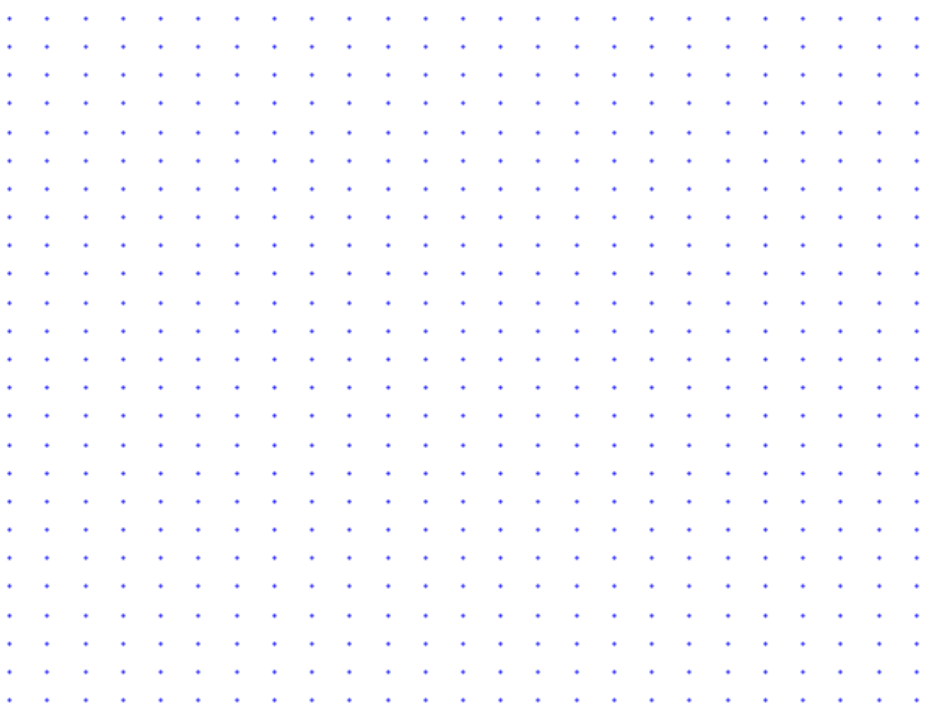
bfs_tree() : for doing bfs in networkx 2d grid graph objects.

shortest_path() : for getting nodes in shortest path between two nodes.

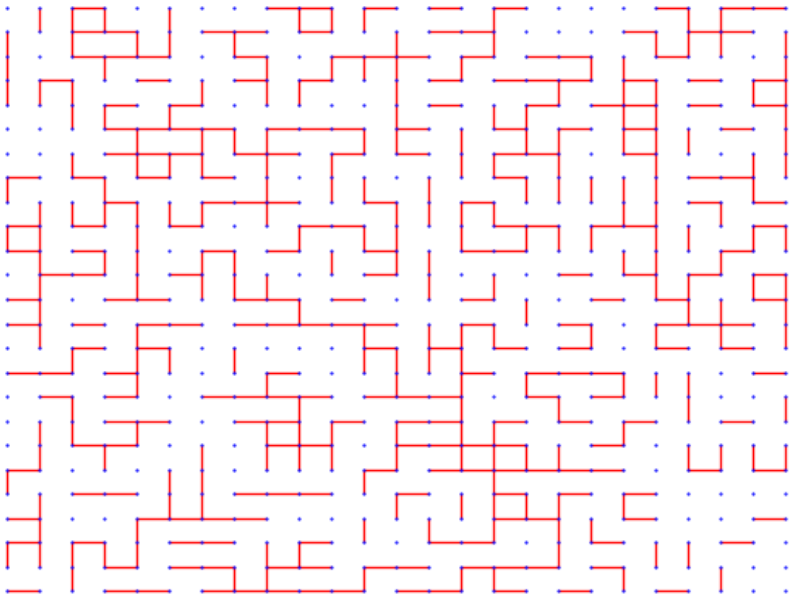
has_path() : for checking whether there exists a path between two nodes.

add_edge() : for adding edge in the 2d grid graph.

```
l = Lattice(25)
l.show()
```

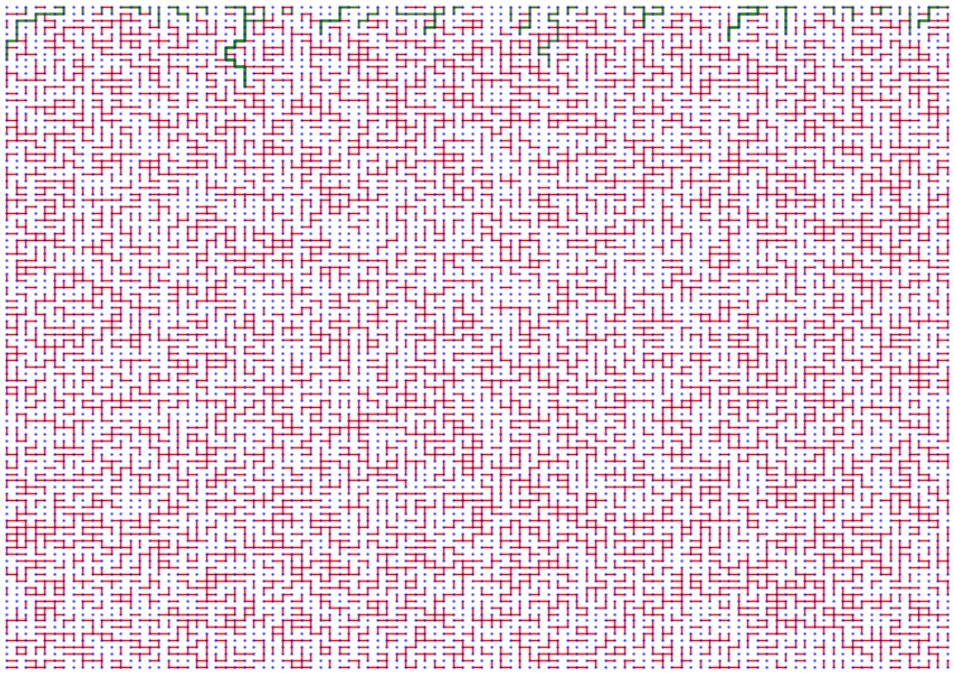


```
l = Lattice(25)
l.percolate(0.4)
l.show()
```



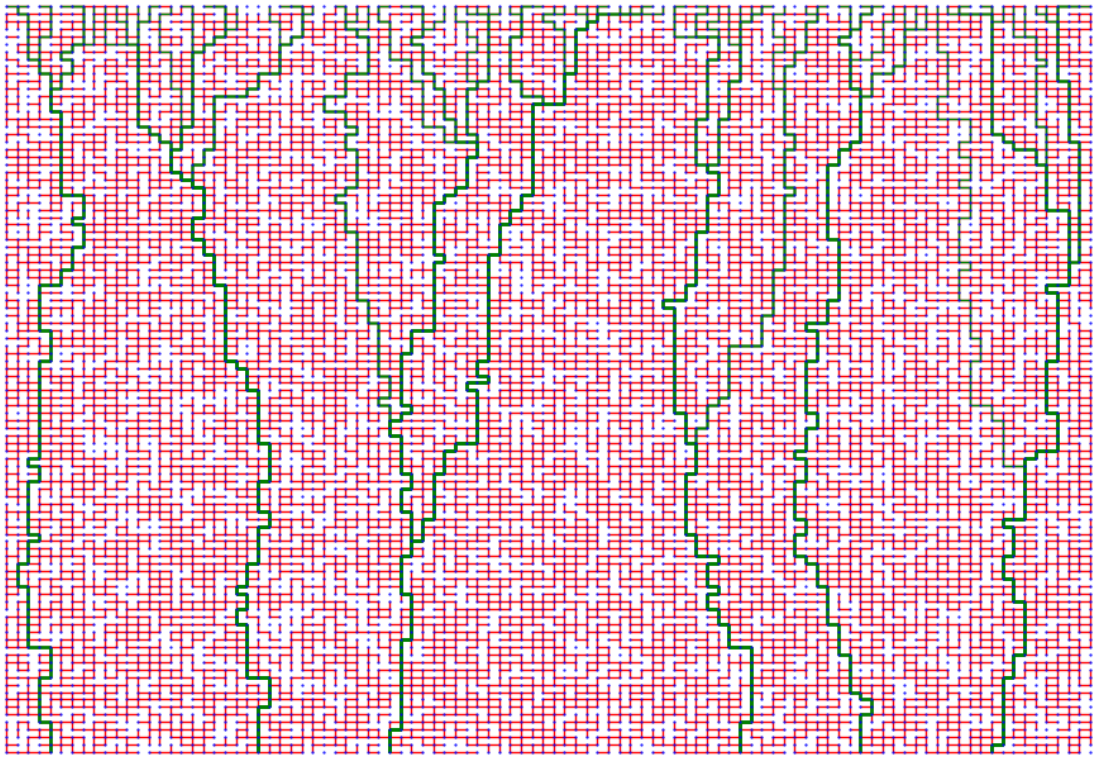
```
l = Lattice(100)
l.percolate(0.4)
l.showPaths()
```

Time : 2 sec



```
l = Lattice(100)
l.percolate(0.7)
l.showPaths()
```

Time : 8 sec



Q6)

to verify : A path exists (almost surely) from the top-most layer to the bottom-most layer of a 100*100 grid graph only if the bond percolation probability exceeds 0.5

Time : 300 sec