Report: Lab 10

Submitted by: Aakash Banik (16i190010)

Question 1:

File: ex1b.py

Subpart a:
Algorithm has been submitted to the TA offline.

Subpart c:

```
Reloaded modules: net1
Enter the starting node : 5
Enter the terminal node : 9
Length of the shortest path from  5 to 9 is : 40
The number of iterations taken to compute the shortest path is : 171
>>> 
```

Figure 1: Output for net1

```
Reloaded modules: net2
Enter the starting node : 99
Enter the terminal node : 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Anaconda2\lib\site-packages\spyderlib\widgets\externalshell\sitecustomize.py", line 714, i
runfile
    execfile(filename, namespace)
  File "C:\Anaconda2\lib\site-packages\spyderlib\widgets\externalshell\sitecustomize.py", line 74, in
execfile
    exec(compile(scripttext, filename, 'exec'), glob, loc)
  File "C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 10/ex1b.py", line 98, in <module>
    dfs(hd,tl,s1,t1)
  File "C:/Users/Aak/Desktop/Sem IV/IE 684/Lab 10/ex1b.py", line 78, in dfs
    for j in nbs[not_visited[-1]]:
KeyError: 99
>>> 
```

Figure 2: Output for net2 [Invalid Input]

```
Reloaded modules: net3
Enter the starting node : 50
Enter the terminal node : 51
Length of the shortest path from  50 to 51 is : 110
The number of iterations taken to compute the shortest path is : 19701
```

Figure 3: Output for net3

```
Reloaded modules: net4
Enter the starting node : 400
Enter the terminal node : 1
Length of the shortest path from  400 to 1 is : 164
The number of iterations taken to compute the shortest path is : 498501
```

Figure 4: Output for net4

```
Reloaded modules: net6
Enter the starting node : 750
Enter the terminal node : 320
Length of the shortest path from  750 to 320 is : 12
The number of iterations taken to compute the shortest path is : 1277601
```

Figure 5: Output for net6

Question 2:

Subpart a:
File: ex2a.py

```
Enter the no. of vertices: 10
Enter p: .5
The Edge Set of the graph is given as:
[0, 1, 0, 1, 0, 0, 0, 0, 1, 0]
[1, 0, 0, 0, 1, 0, 1, 0, 1, 1]
[0, 0, 0, 0, 1, 0, 1, 0, 1, 1]
[1, 0, 0, 0, 0, 0, 1, 0, 1, 1]
[0, 1, 1, 0, 0, 1, 1, 0, 0, 1]
[0, 0, 0, 0, 1, 0, 0, 0, 1, 0]
[0, 1, 1, 1, 1, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 1]
[1, 1, 1, 1, 0, 1, 0, 0, 0, 0]
[0, 1, 1, 1, 1, 0, 0, 1, 0, 0]
>>>
```

Figure 6: A random graph generated by the algorithm in Ques 2

The above output corresponds to an Edge Set. It has N rows and N columns corresponding to the N vertices. The $(i, j)^{th}$ entry is 1 if there is an edge between nodes i and j; otherwise its 0.

Subpart b:
File: ex2b.py

The degree distribution has more or less the same form across all iterations. The slight changes are due to the randomness in the algorithm. The mean, median and mode degree is approximately around 250 which we expected [since the expected no. of edges for a particular node is $(N-1)^{*}p$ which is approximately around 250 in this case.]

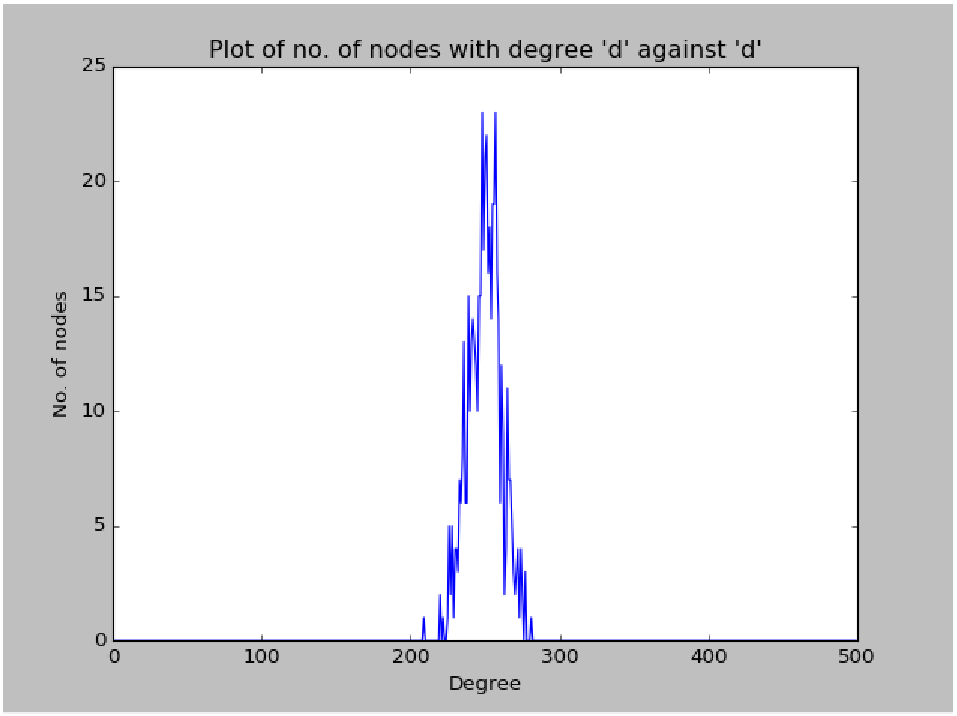

Figure 7: Plot of the degree distribution(iteration 1)

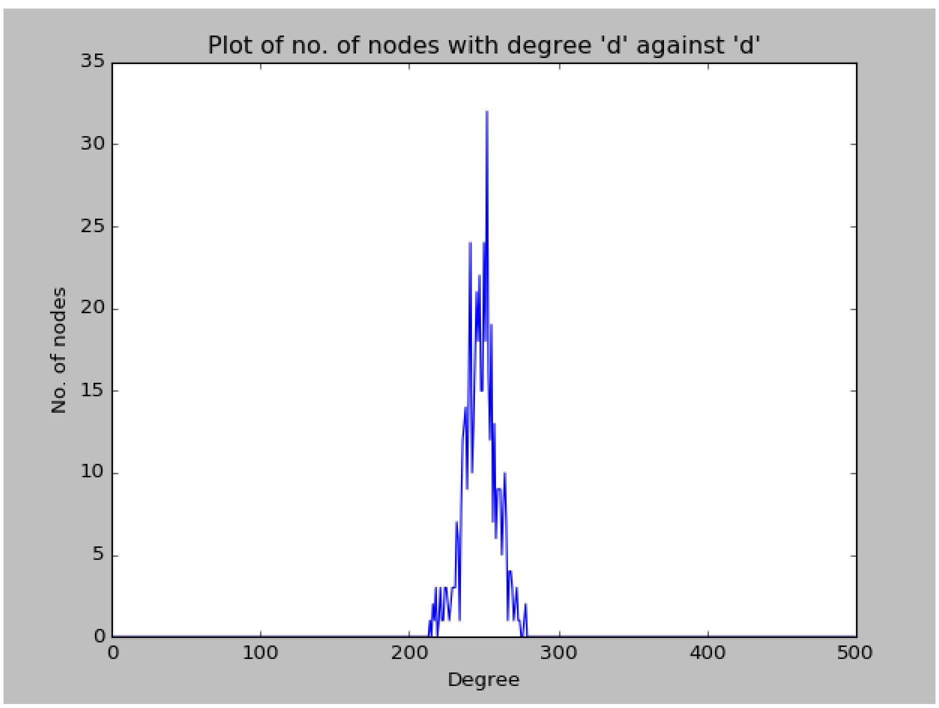

Figure 8: Plot of the degree distribution (iteration 2)

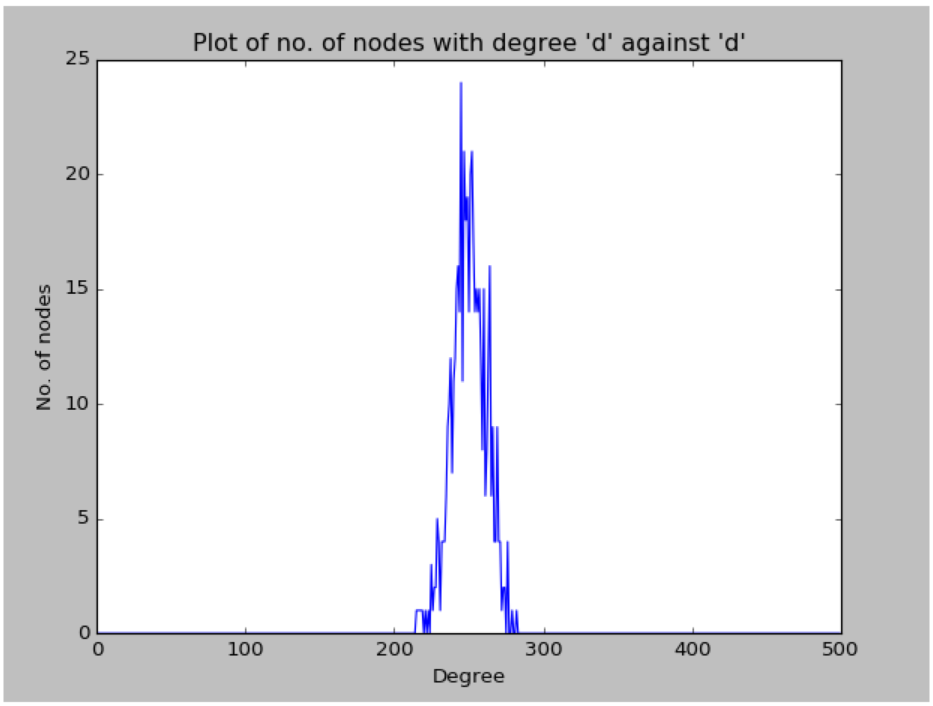

Figure 9: Plot of the degree distribution (iteration 3)

Subpart c:

For conducting this experiment, we ran the code for ex2b.py for different values of N and p. First, to check the dependence of the degree distribution on N, we repeated the experiment with different values of N while keeping p to be constant at p=0.5; Later to check the dependence of the degree distribution on p, we repeated the experiment for different values of p while keeping N to be constant at N=500.

We can see that as N increases, we get a smoother curve for the degree distribution.
Also, we observe that as p increases the distribution gets shifted towards right. This is because the expected degree of each node increases with p.

We can conclude from our observations that the total no. of edges in our graph is directly proportional to N and p. Moreover if p=1, we would have the maximum possible no. of edges in the graph.

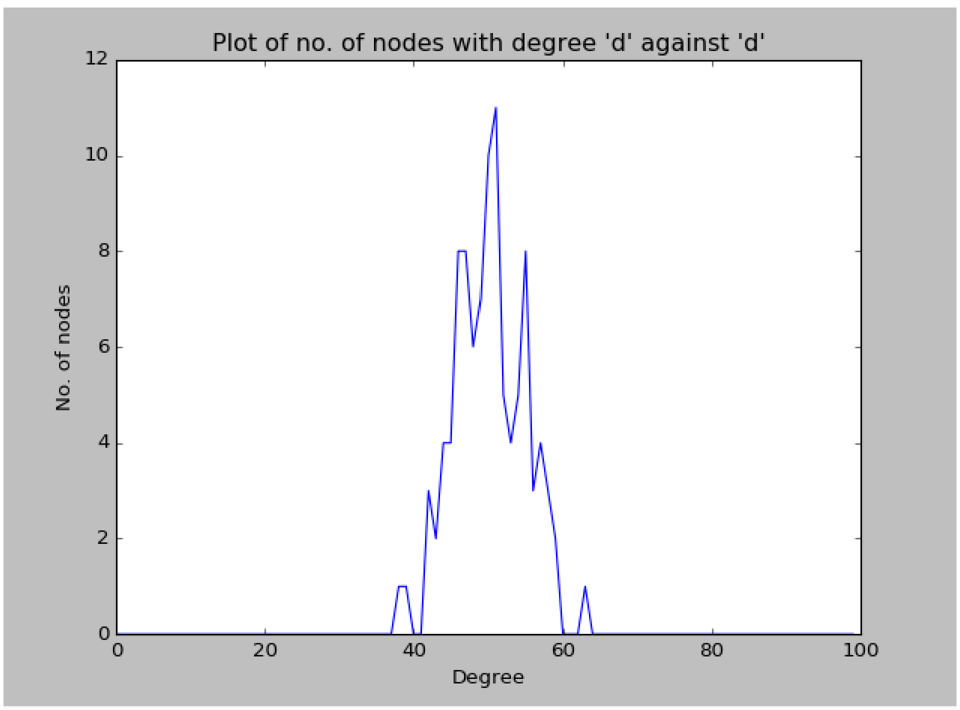

Figure 10: Plot of the degree distribution (N=100, p=0.5)

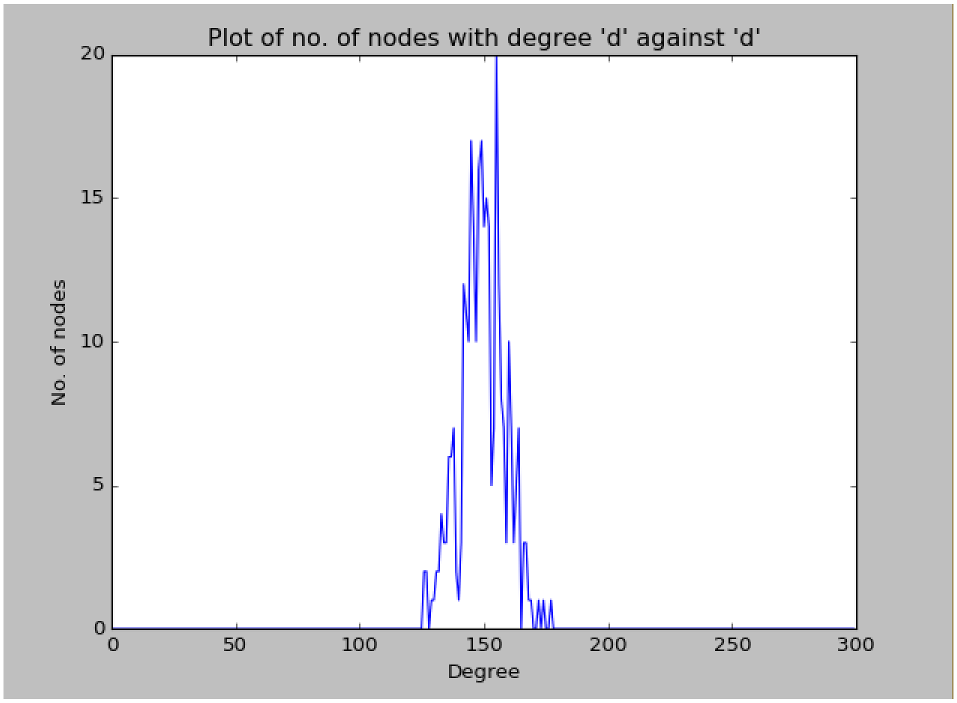

Figure 11: Plot of the degree distribution (N=300, p=0.5)

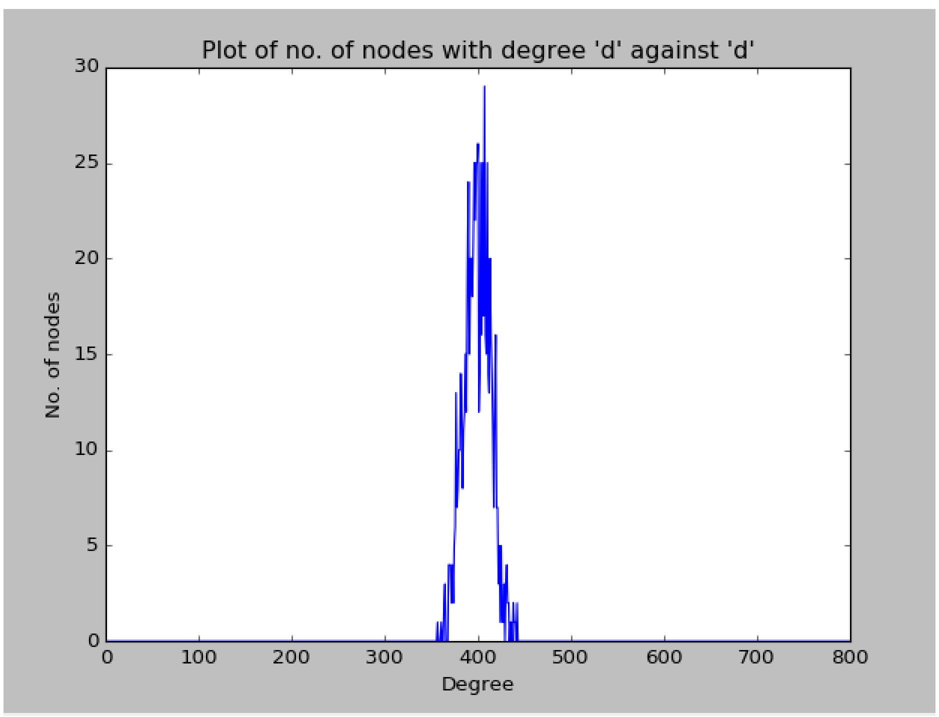

Figure 12: Plot of the degree distribution (N=800, p=0.5)

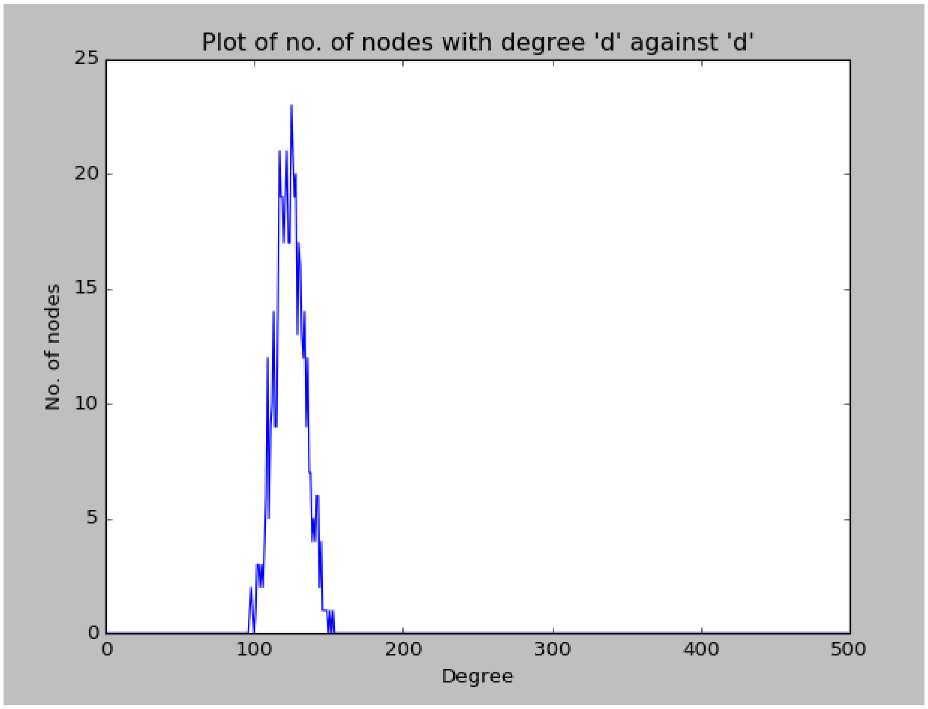

Figure 13: Plot of the degree distribution (N=500, p=0.25)

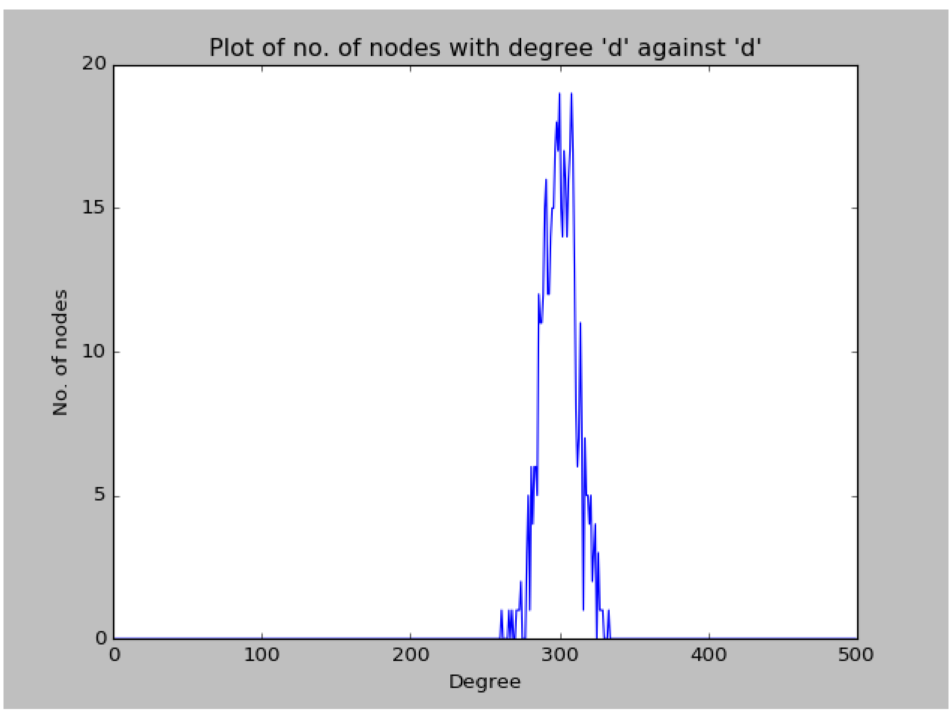

Figure 14: Plot of the degree distribution (N=500, p=0.6)

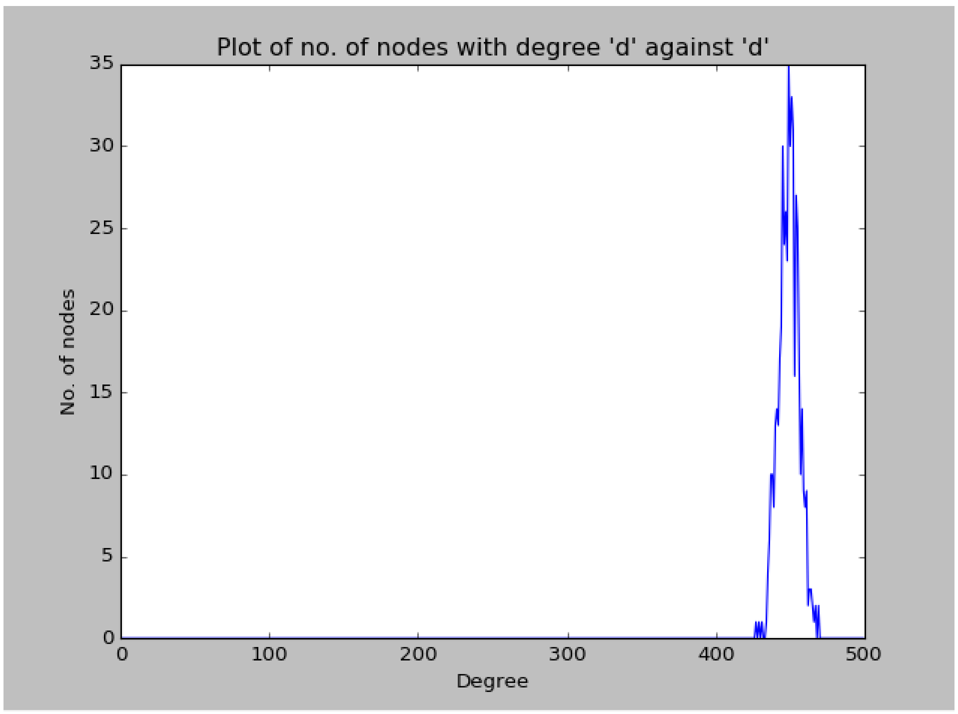

Figure 15: Plot of the degree distribution (N=500, p=0.9)

Question 3:

File: ex3a.py

Subpart a:
The total no. of edges in the graph won't change after the algorithm has finished. Before the algorithm starts, we'd have a total no. of $\frac{NK}{2}$ edges. Since we are just reallocating edges with a certain probability, the total no. of edges in the Graph won't change.

Subpart c:

For conducting this experiment, we ran the code for ex3a.py for different values of N, K and p. First, to check the dependence of the degree distribution on N, we repeated the experiment with different values of N while keeping p and K to be constant; Later to check the dependence of the degree distribution on p, we repeated the experiment for different values of p while keeping N and K to be constant; Lastly to check the dependence of the degree distribution on K, we vary K across iterations while keeping the values of N and p to be constant.

We can conclude that as N increases (with p and K fixed), the frequency of the nodes having a degree of some value around K also increases.
Also, as p increases (with N and K fixed), the frequency of the nodes having a degree of some value around K decreases.
Finally, as K increases (with N and p fixed), expected degree of each node increases.

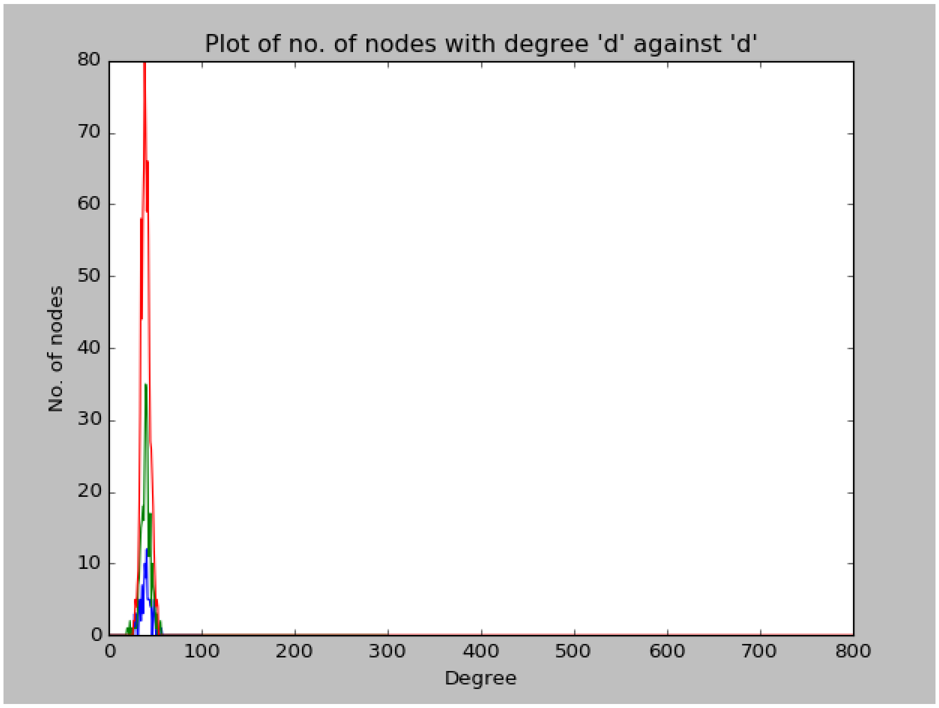

Figure 16: Plot of the degree distribution for fixed p=0.5 and K=40; and varying N=100(Blue),N=300(Green),N=800(Red)

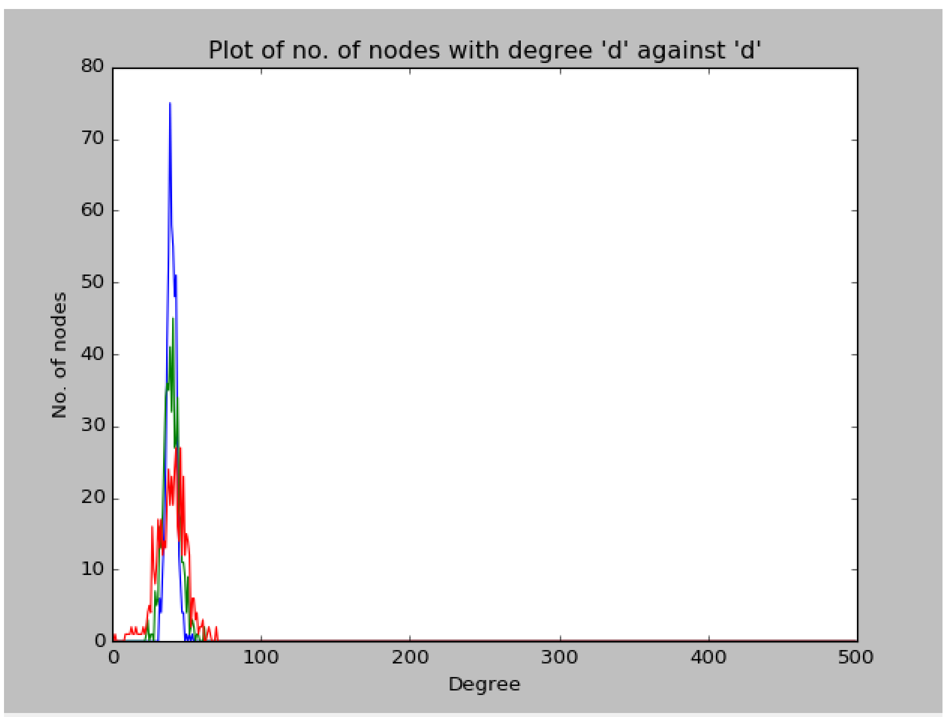

Figure 17: Plot of the degree distribution for fixed N=500 and K=40; and varying p=0.25(Blue),p=0.6(Green),p=0.9(Red)

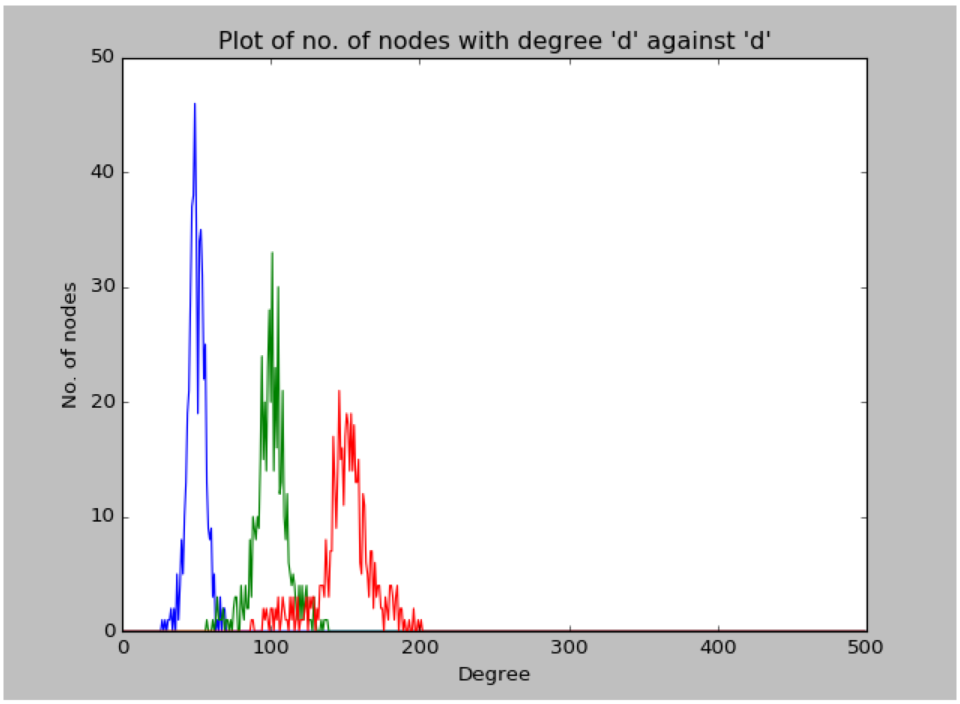

Figure 18: Plot of the degree distribution for fixed N=500 and p=0.5; and varying K=50(Blue),K=100(Green),K=150(Red)