Table of Contents

CS4423-Networks: Week 11 (26+27 March 2025)

Part 1: Watts-Strogatz model

Niall Madden, School of Mathematical and Statistical Sciences
University of Galway

This Jupyter notebook, and PDF and HTML versions, can be found at https://www.niallmadden.ie/2425-CS4423/#Week11

This notebook was adapted by Niall Madden from one developed by Angela Carnevale.

Our usual preamble:

In [1]:
```
import networkx as nx
import numpy as np
opts = { "with_labels": True,  "node_color": "#84003d", "font_color": "white" } # Galw

import random   # some random number generators:random, random_choices
import statistics  # e.g., mean of entries in a list
import math     # for comb (=binomial coef)
import matplotlib.pyplot as plt

np.set_printoptions(precision=2)    # just display arrays to 2 decimal places
np.set_printoptions(suppress=True)
```

Small Worlds, again

Last week, we claimed that **small world networks** tend to share three characteristics:

1. Short **characteristic path length**, which scales like $\ln n$, where n is the number of nodes.
2. Low **transitivity**, meaning that a high proportion of *triads* form *triangles*.
3. A **high clustering coefficient**

We saw that the G_{ER} models tend to have the first property, but not the second or third. Therefore, in this sense, they don't mimicking real world networks very well.

An alternative, developed by Watts and Strogatz in 1998, is to start with some **regular network** that naturally has a **high clustering**, and then to randomly distort its edges, to introduce some **short paths**.

Circle Graphs

Cycle graphs

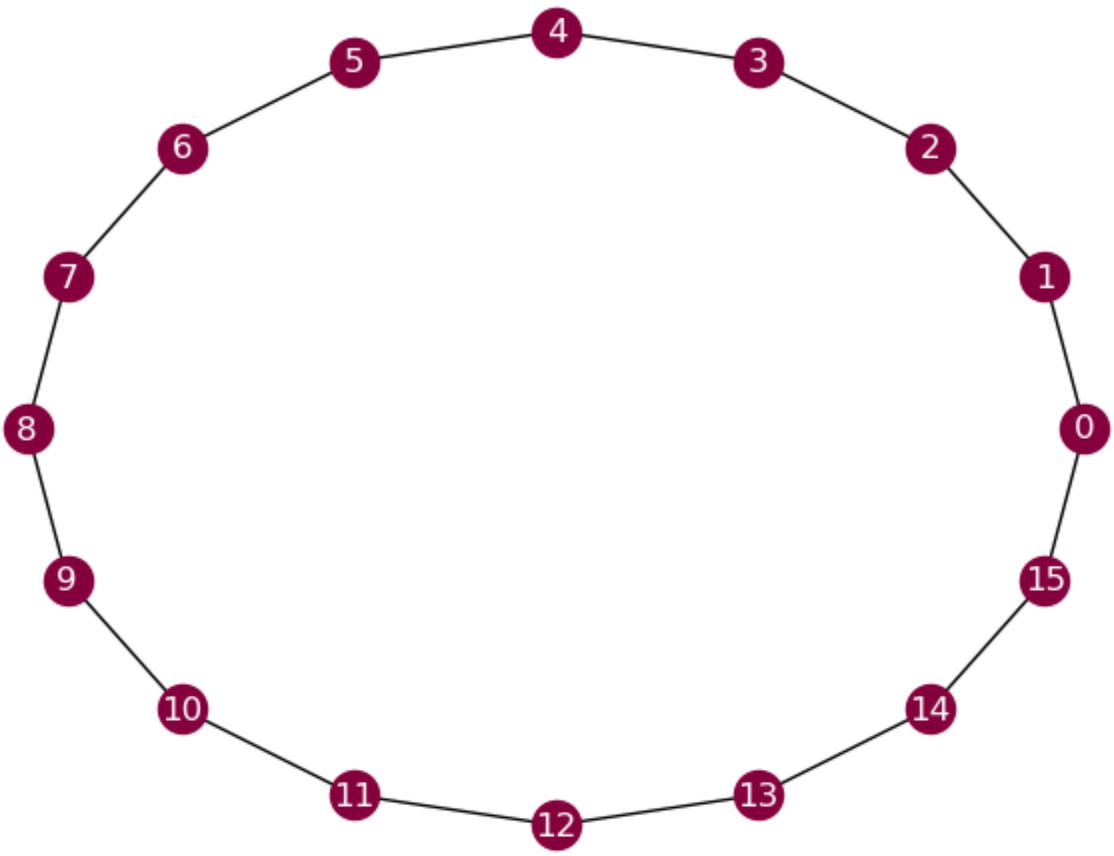
To learn how to create a network that has the right properties, we'll start with one that does not, and then see how we can change to.

So we start with a **cycle graph**.

In [2]:

```
n = 16
G = nx.cycle_graph(n)
nx.draw_circular(G, **opts)
print(f"G has an average path length of L={nx.average_shortest_path_length(G):.3}")
print(f"Its transitivity value is T={nx.transitivity(G)}, and Clustering is C={nx.ave
```

```
G has an average path length of L=4.27
Its transitivity value is T=0, and Clustering is C=0.0
```



We won't dwell on it right now, but the average path length is $\approx n/4$. Let's focus of the transitivity and clustering.

- **Transitivity:** clearly, C_n has many triads, but no triangles.

- **Clustering**: the subgraph induced by the neighbours of any node has no edges.

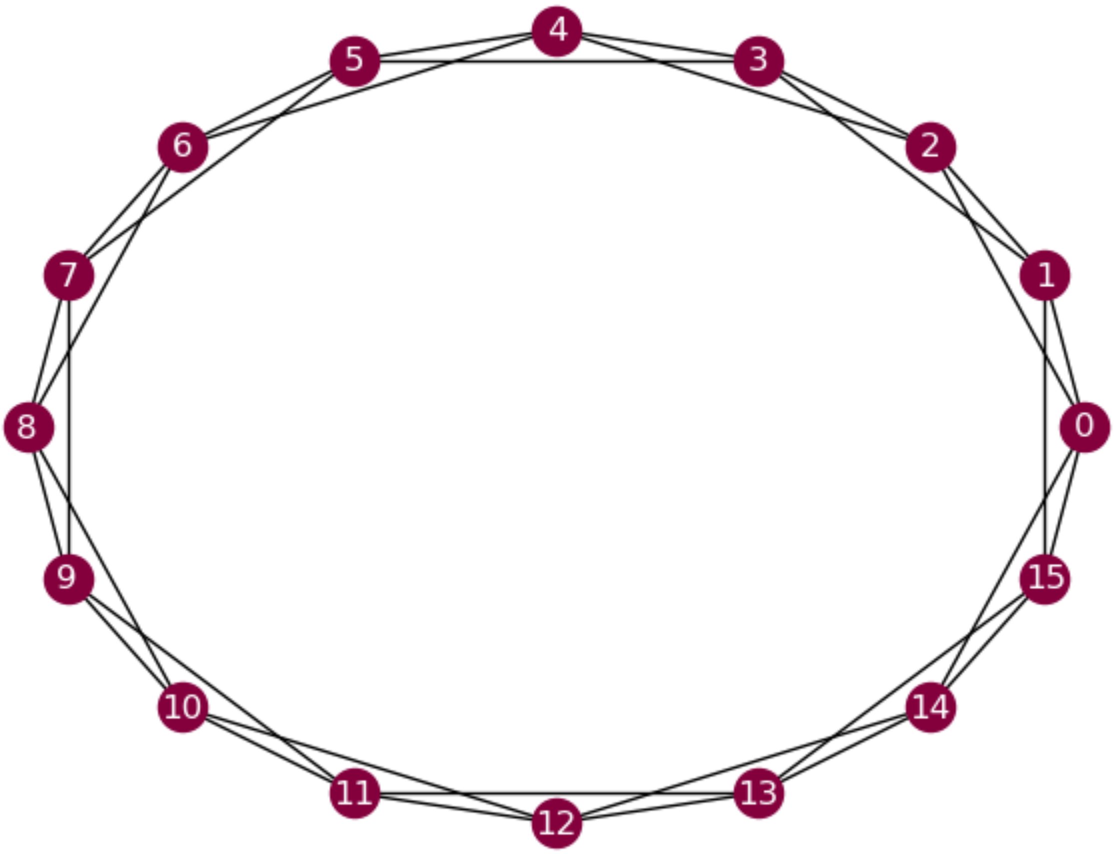
To address this, we could add edges between nodes that have a common neighbour.

Increasing Clustering

Starting with the cycle graph, $G = C_n$, let's add an edge between Node i and Node $i+2$ (mod n).

In [3]:
```
for v in G:
    G.add_edge(v, (v+2) % n)
nx.draw_circular(G, **opts)
print(f"For this graph, G, we have L={nx.average_shortest_path_length(G):.3}, T = {nx
```

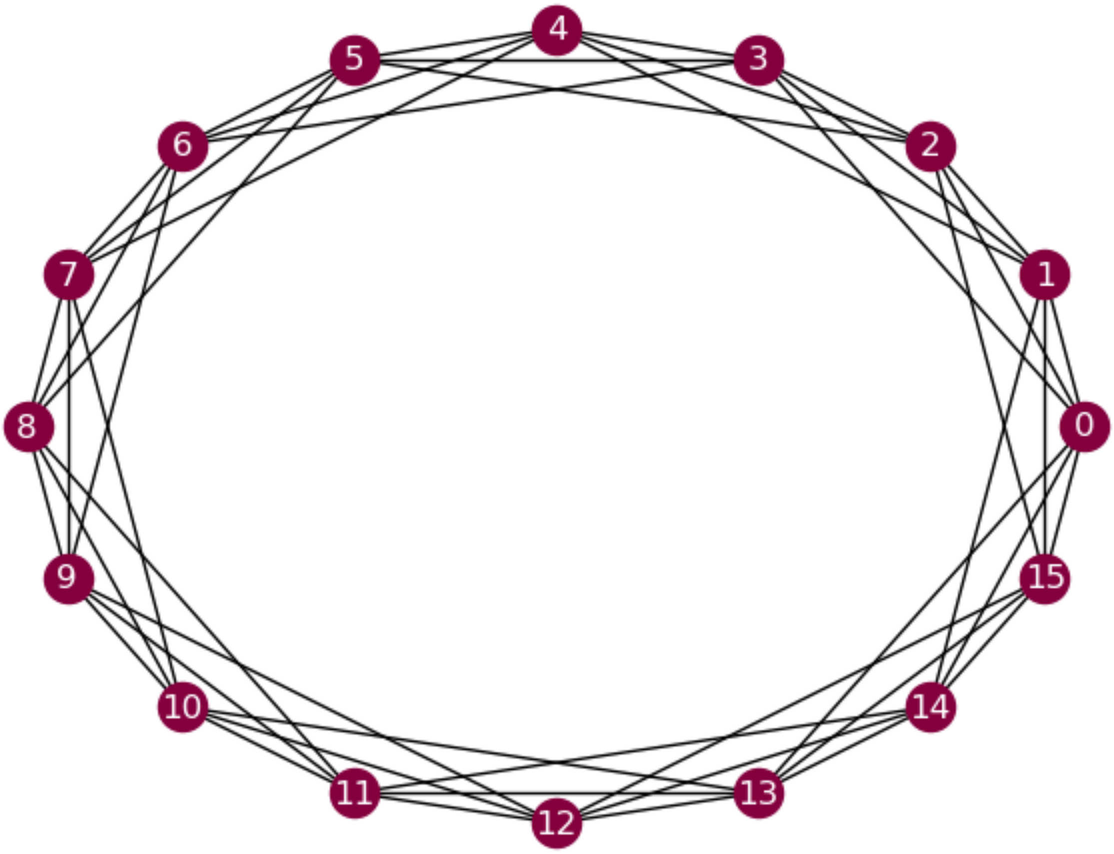
```
For this graph, G, we have L=2.4, T = 0.5, and C = 0.5
```



Looks like we're going in the right direction: L is getting smaller while C (and T) are increasing. Let's keep going by adding an edge between Node i and Node $i+3$ (mod n).

In [4]:
```
for v in G:
    G.add_edge(v, (v+3) % n)
nx.draw_circular(G, **opts)
print(f"For this graph, G, we have L={nx.average_shortest_path_length(G):.3}, and C =
```

```
For this graph, G, we have L=1.8, and C = 0.6
```



Circle Graph: Definition

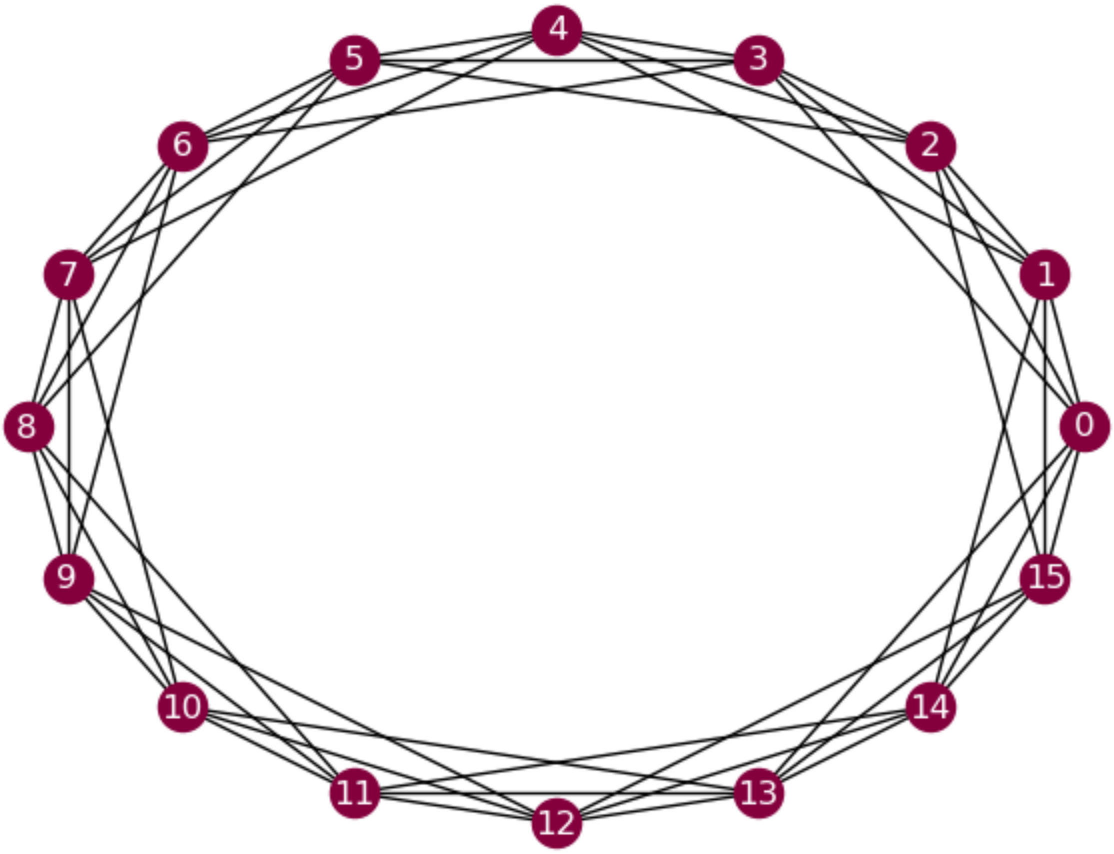
Definition (Circle Graph). For $1 < d < n/2$, an (n, d)-**circle graph** is obtained from a cycle on n vertices by additionally linking each node to all nodes that are not more than d steps away on the cycle.

Here is some code to generated it:

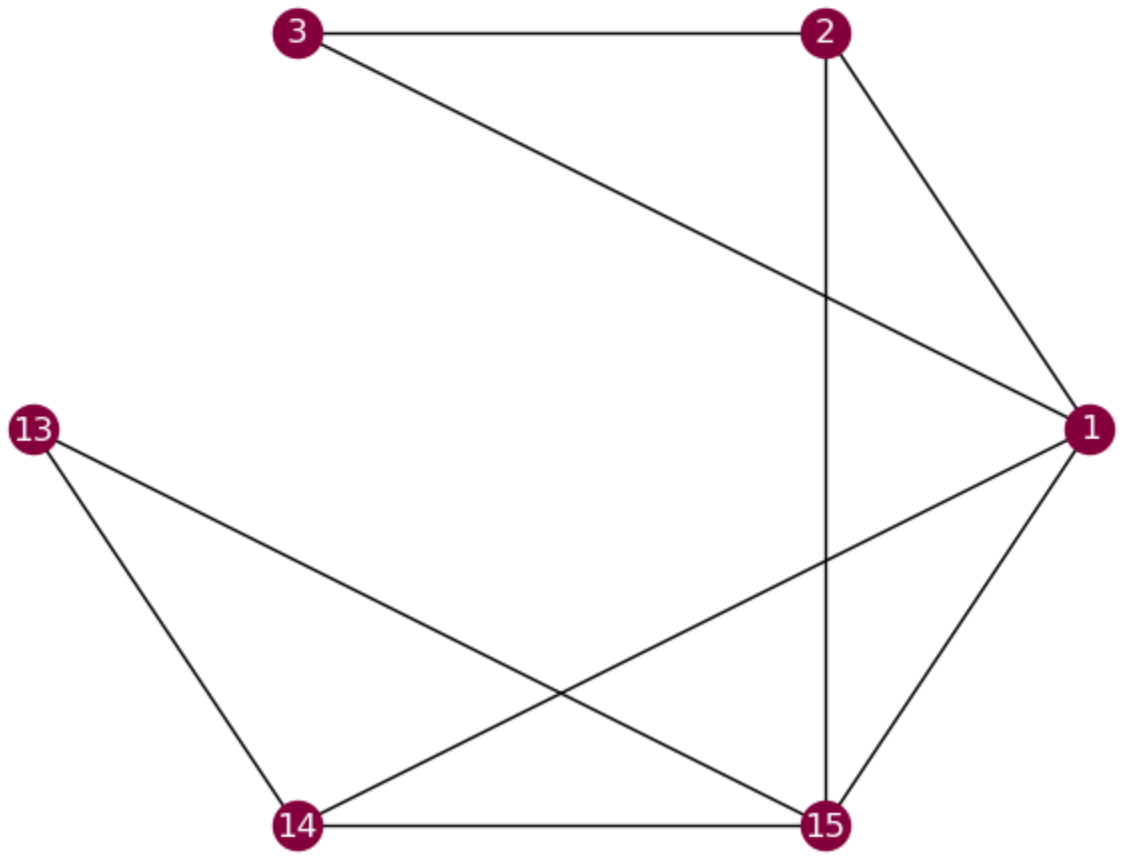
In [5]:
```
def circle_graph(n, d):
    G = nx.cycle_graph(n)
    for v in G:
        for o in range(2, d+1):
            G.add_edge(v, (v+o) % n)
    return G
```

In [6]:
```
G = circle_graph(16, 3)
nx.draw_circular(G, **opts)
CPL = nx.average_shortest_path_length(G)
print(f"For this graph, G, we have L={CPL:.3f}, C={nx.average_clustering(G):.2f}")
```

```
For this graph, G, we have L=1.800, C=0.60
```



In [7]:
```
N = G.neighbors(0)
S = nx.subgraph(G, list(N))
nx.draw_circular(S, **opts)
```



In [8]:
```
S.degree()
```

Out[8]:
```
DegreeView({1: 4, 2: 3, 3: 2, 13: 2, 14: 3, 15: 4})
```

In [9]:
```
S.size()
```

Out[9]:
```
9
```

- An (n, d)-circle graph has n nodes and $m = nd$ edges.
- Each node has degree $\frac{2m}{n} = 2d$.
- The social graph of each node has $\frac{3}{2}d(d-1)$ edges.

- The graph clustering coefficient of an (n, d)-circle graph is **independent of** n, and can be determined as

$$C = \frac{3d-3}{4d-2} \to \frac{3}{4}, \text{ as } d \to \infty.$$

In particular:

d	1	2	3	4	5
C	0	0.5	0.6	0.643	0.667

Characteristic Path Length

- However, things don't work as well when it comes to shortest paths (if we let $n \to \infty$). Indeed, the characteristic path length of an (n, d)-circle graph is approximately

$$L \approx \frac{n}{4d},$$

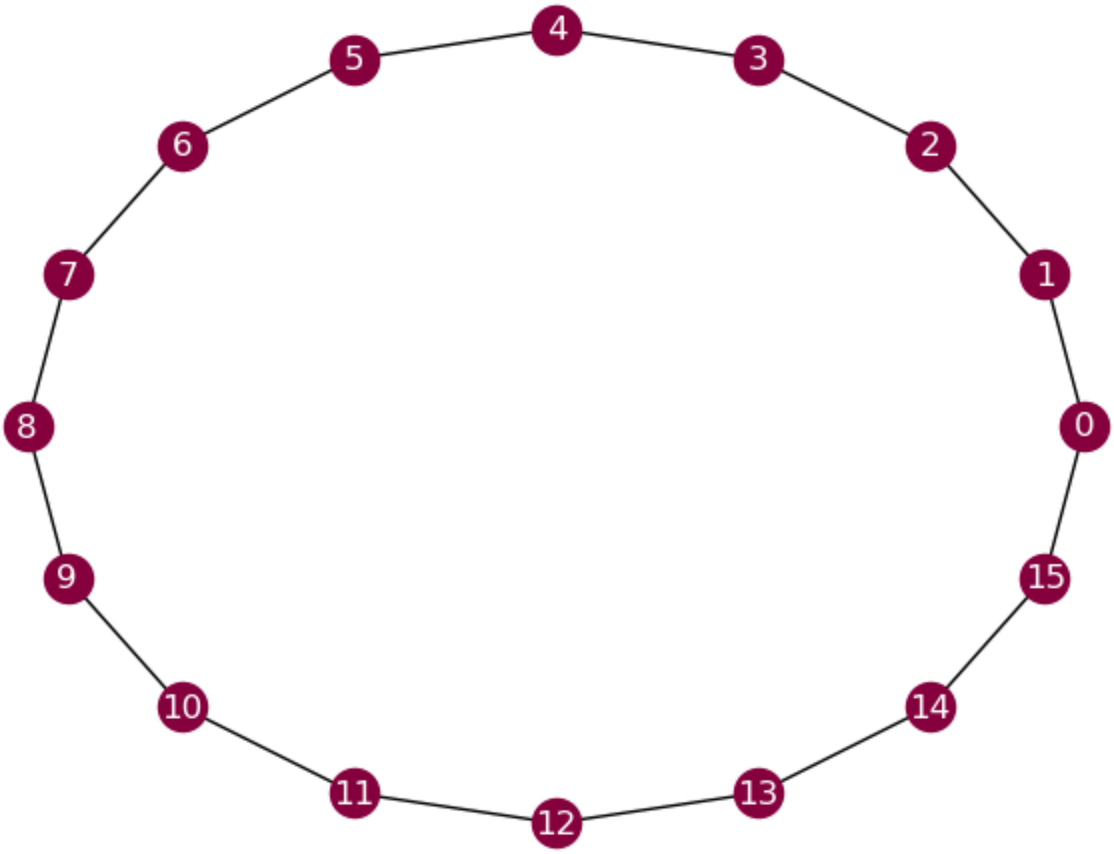
growing linearly with n (for fixed d).

In conclusion, such regular graphs have **high clustering** but **long shortest paths**, hence (n, d)-circle graphs do not exhibit the small world behaviour.

To see how we could reduce the CPL, let's return to the Cycle Graph from earlier

In [10]:
```
n = 16
G = nx.cycle_graph(n)
nx.draw_circular(G, **opts)
print(f"For this G, we have L={nx.average_shortest_path_length(G):.3}, and C={nx.avera
```

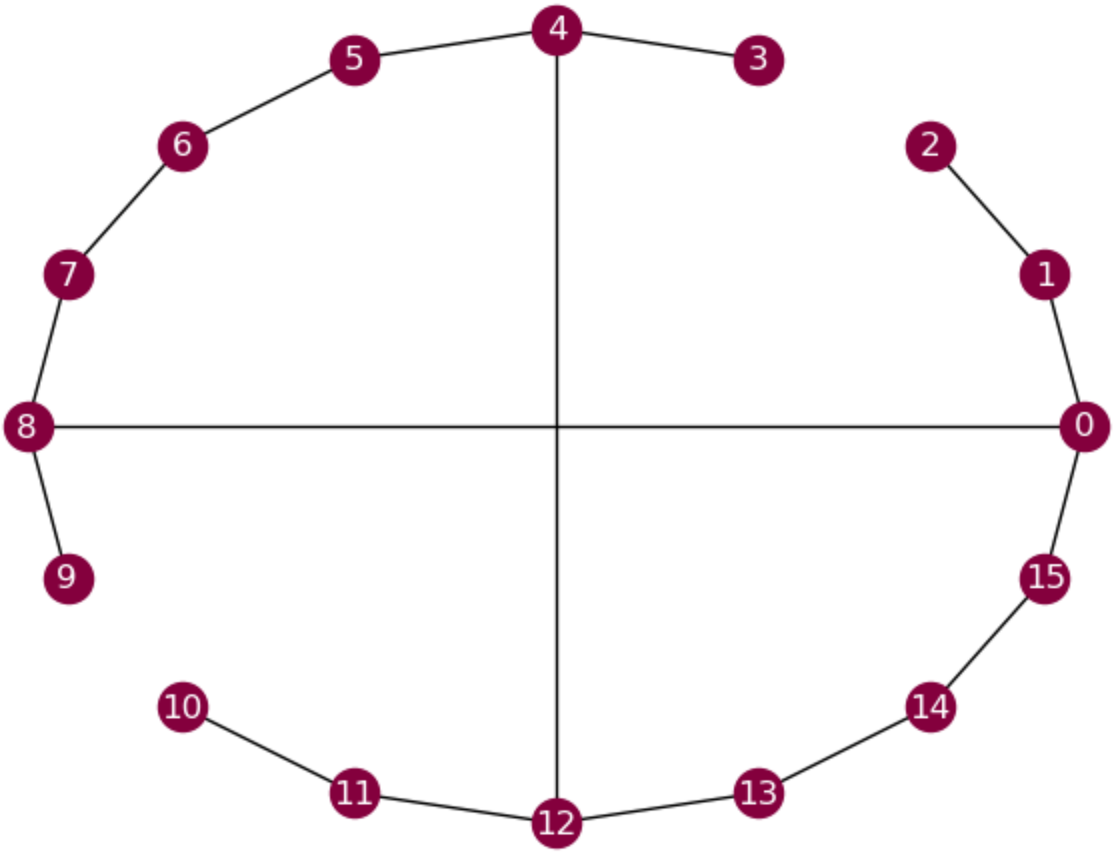
```
For this G, we have L=4.27, and C=0.0
```



Let's "rewire" two edges:

In [11]:
```
G.remove_edges_from( [(2,3), (9,10)])
G.add_edges_from( [(0, 8), (4, 12)] )
nx.draw_circular(G, **opts)
print(f"Now G has L={nx.average_shortest_path_length(G):.3}, and C={nx.average_cluste
```

```
Now G has L=3.62, and C=0.0
```



So we can reduce the CPL, by adding relatively few edges. Finally, we can get a combined solution...

The Watts-Strogatz Model

The following modification of the circle graph was suggested by Duncan J. Watts and Steven Strogatz (1998). The idea is to introduce a probabilistic element to the graph, which results in "shortcuts" (or "teleports") between the nodes and in a shortening of the characteristic path length.

Definition (The WS Model). Let $1 < d < n/2$ and $0 \leq p \leq 1$. An (n, d, p)-WS graph $G = (X, E)$ is constructed from an (n, d)-circle graph $G_0 = (X, E_0)$ by rewiring each of the edges in E_0 with probability p, as follows:

1. visit the nodes $X = \{0, \ldots, n{-}1\}$ in turn ('clockwise').
2. for each node $i \in X$ consider the d edges connecting i to j in a clockwise sense ($j = i + 1, \ldots, i + d$).
3. With probability p, in the edge (i, j) replace j by node $k \in X$ chosen uniformly at random, subject to
   - $k \neq i$, and
   - (i, k) must not be an edge of G already.

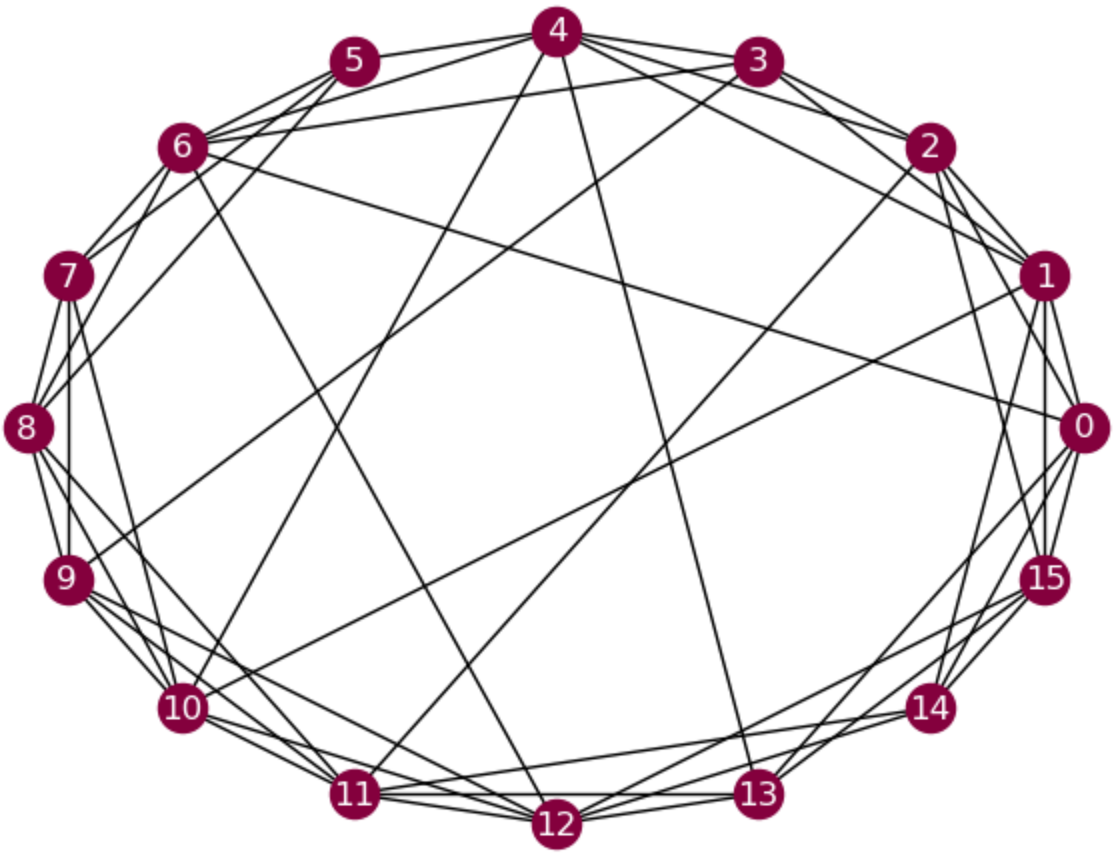
In [12]:

```
import random as rd
def ws_graph(n, d, p):
    G = circle_graph(n, d)
```

```
    for v in G:
        for o in range(1, d+1):
            if rd.random() < p:
                w = rd.randint(0,n-1)  # pick a random node
                if w != v and not G.has_edge(v, w):
                    G.remove_edge(v, (v+o) % n)
                    G.add_edge(v, w)
    return G
```

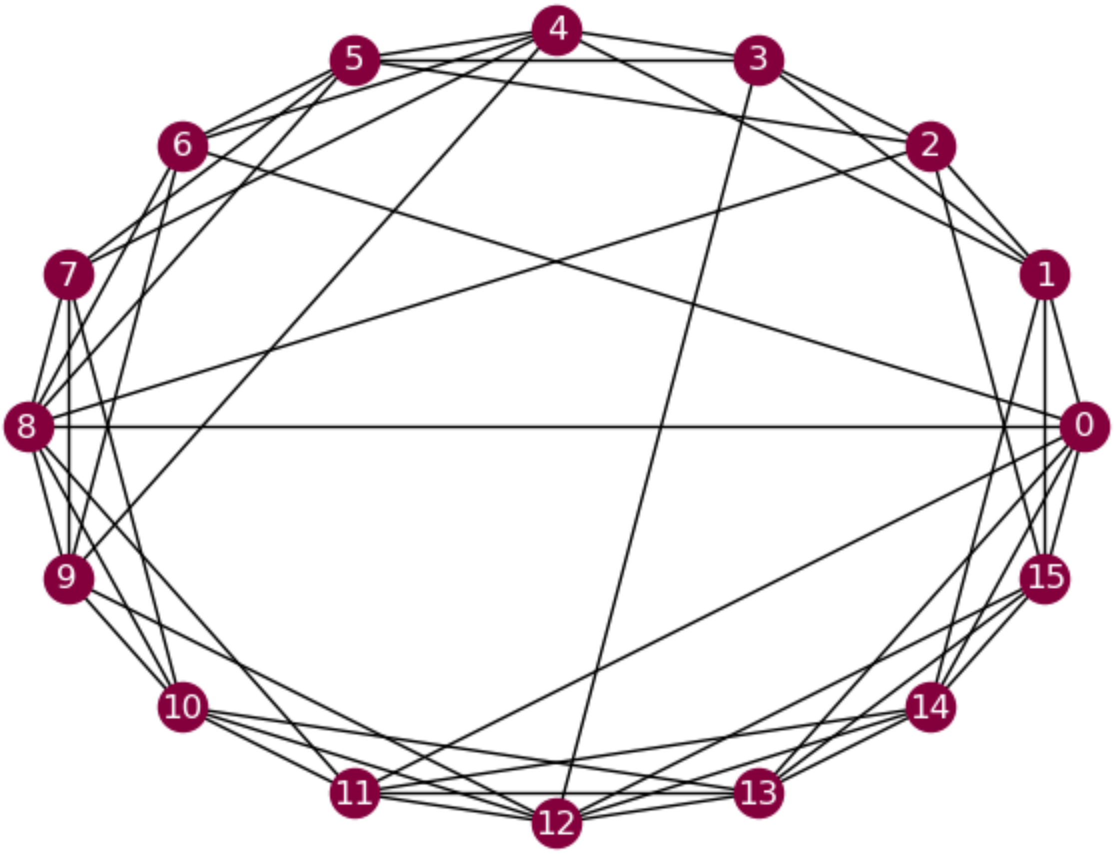
In [13]:
```
n, d = 16, 3
G = ws_graph(n, d, 0.2)
nx.draw_circular(G, **opts)
print(f"G has L={nx.average_shortest_path_length(G):.3}, and C={nx.average_clustering
```

```
G has L=1.67, and C=0.42
```



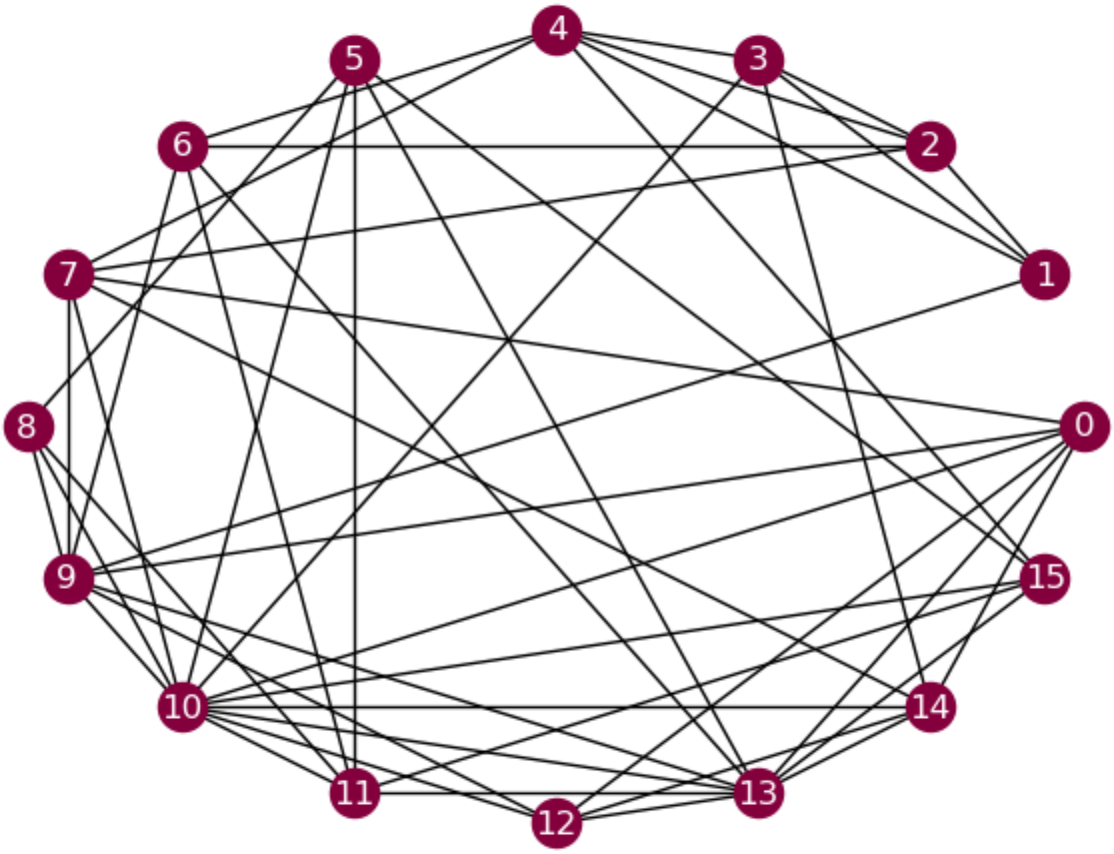
In [14]:
```
n, d = 16, 3
G = ws_graph(n, d, 0.3)
nx.draw_circular(G, **opts)
print(f"G has L={nx.average_shortest_path_length(G):.3}, and C={nx.average_clustering
```

```
G has L=1.66, and C=0.49
```



In [15]:
```
G = ws_graph(n, d, 1)
nx.draw_circular(G, **opts)
print(f"G has L={nx.average_shortest_path_length(G):.3} and C={nx.average_clustering(
```

```
G has L=1.66 and C=0.54
```

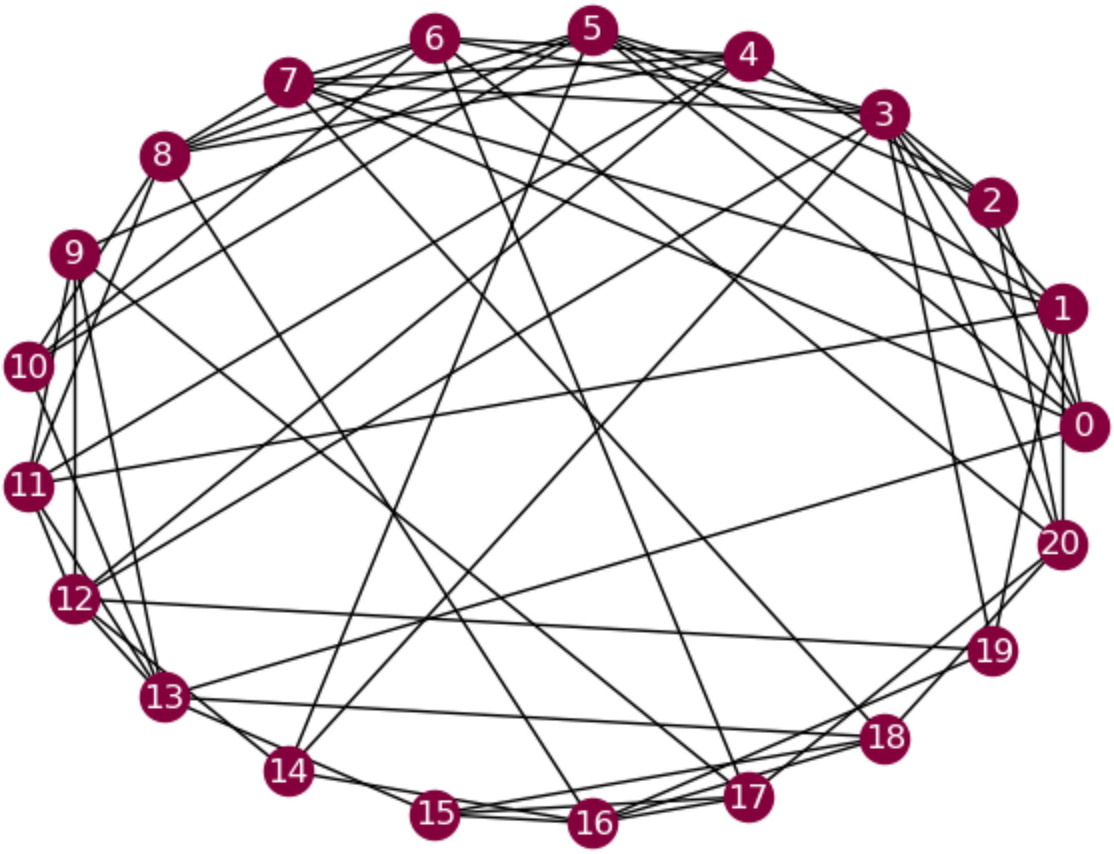


A WS graph with parameters (n, d, p) can be generated with the command:

`nx.watts_strogatz_graph(n, 2*d, p)` .

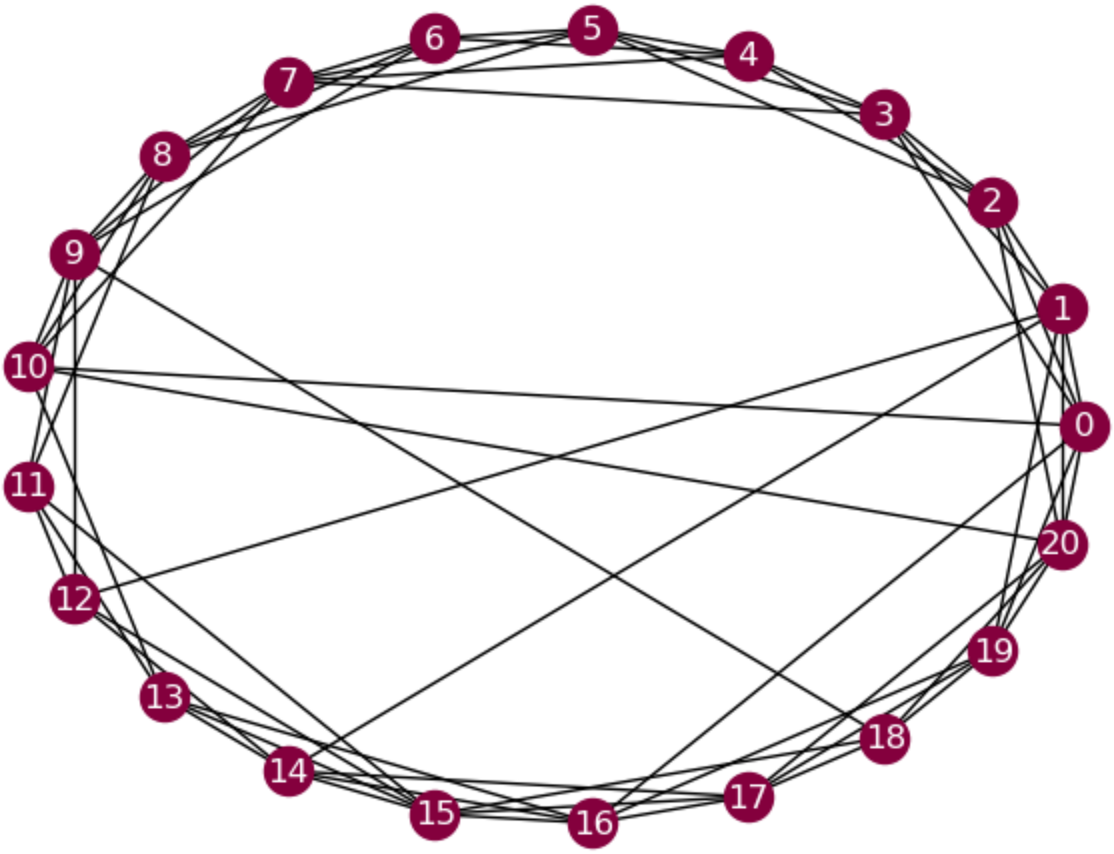
In [16]:
```
n, d = 21, 3
G = nx.watts_strogatz_graph(n, 2*d, 0.5)
nx.draw_circular(G, **opts)
print(f"G has L={nx.average_shortest_path_length(G):.3} and C={nx.average_clustering(
```

```
G has L=1.75 and C=0.31
```



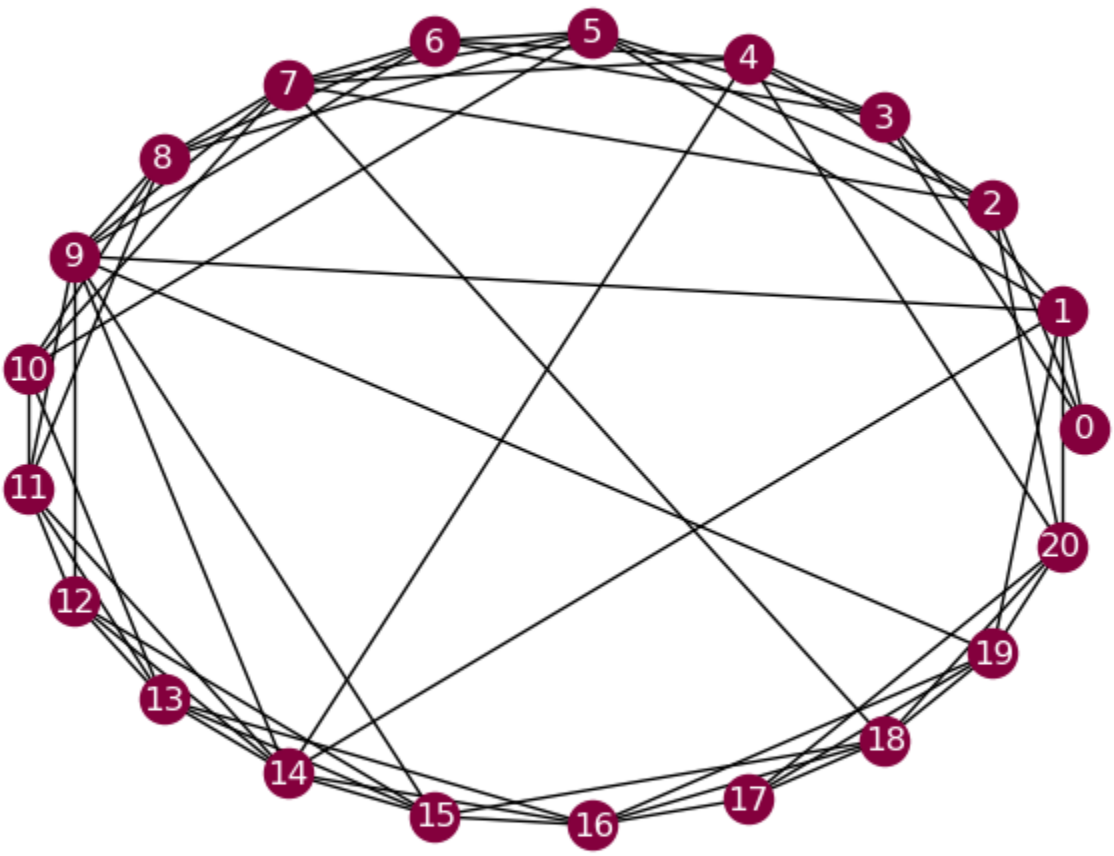
In [17]:
```
G = nx.watts_strogatz_graph(n, 2*d, 0.1)
nx.draw_circular(G, **opts)
print(f"G has L={nx.average_shortest_path_length(G):.3}, and C={nx.average_clustering
```

```
G has L=1.94, and C=0.48
```




```
In [18]: G = nx.watts_strogatz_graph(n, 2*d, 0.2)
         nx.draw_circular(G, **opts)
         print(f"G has L={nx.average_shortest_path_length(G):.3}, C={nx.average_clustering(G):
```

```
G has L=1.89, C=0.45
```



Properties of WS-Graphs

The small-world attributes of a (n, d, p)-WS graph depend on the probability p. The following measurements have been taken for $n = 1000$ and $d = 5$.

p	L	C
0	50.5	0.667
0.01	8.94	0.648
0.05	5.26	0.576
1	3.27	0.00910

Exercises

1. In terms of the parameters, n, d and p, what is the clustering coefficient C of an (n, d, p)-WS graph?
2. In terms of the parameters, n, d and p, what is the average shortest path length L of an (n, d, p)-WS graph?