# Social Network Analysis

Chen Avin
School of Electrical and Computer Engineering

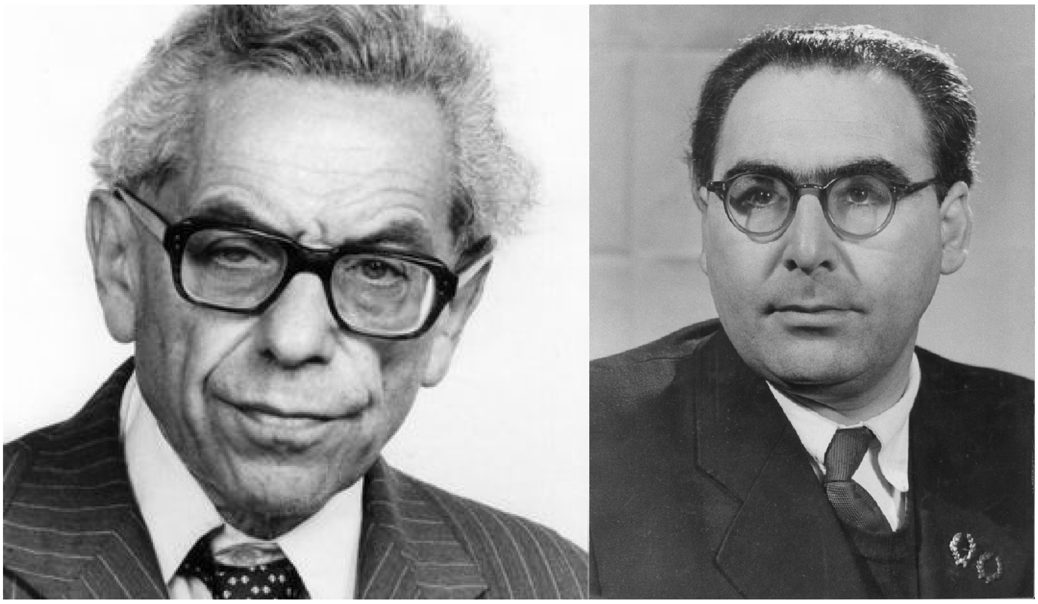
# Unit 8
# Random Graph Models
# Erdos-Reny, Preferential Attachment

(Based on Networks: An Introduction. By M.E.J Newman, and Slides by Lada Adamic)

## Network Models

- Why model?
  - Simple representation of complex network
  - Can derive properties mathematically
  - Predict properties and outcomes
- Check against your data
  - In what ways is your real-world network different from hypothesized model?
  - What insights can be gleaned from this?
  - Understand processes

# Erdös-Rényi Graph model (1960), Gilbert (1959)



https://www.youtube.com/watch?v=fEbYLNyvQy0

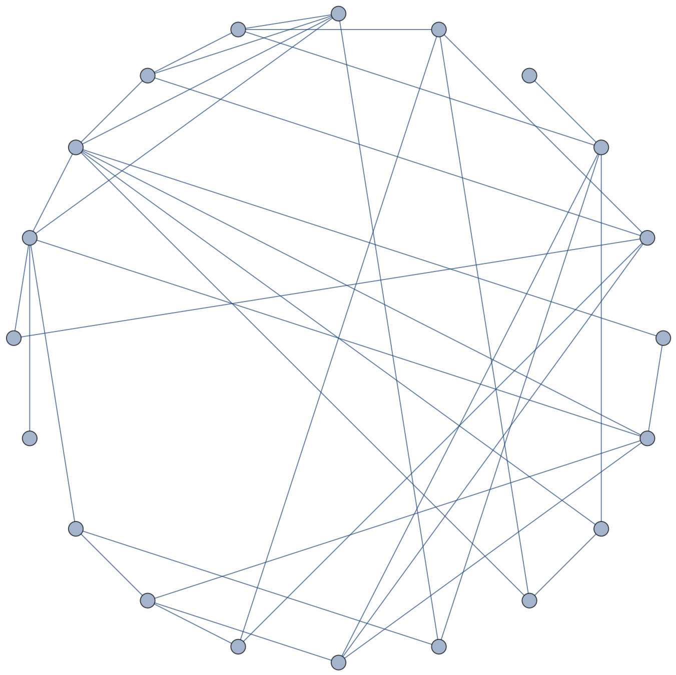
# Simplest random network model

- Also known as Bernoulli random graph
- Assumptions:
  - Nodes connect at random
  - Graph is undirected
- G(n, p)
  - n - number of nodes, p - probability for two nodes to be connected
  - For each potential edge we flip a (biased) coin and add an edge with probability p and don't add with probability (1-p).

In[1101]:=

```
ER = RandomGraph[BernoulliGraphDistribution[20, 2 / 10],
  GraphLayout → "CircularEmbedding"]
```

Out[1101]=



In[1106]:=

```
Binomial[20, 2] 1/5 // N
```

Out[1106]=

```
38.
```

In[1107]:=

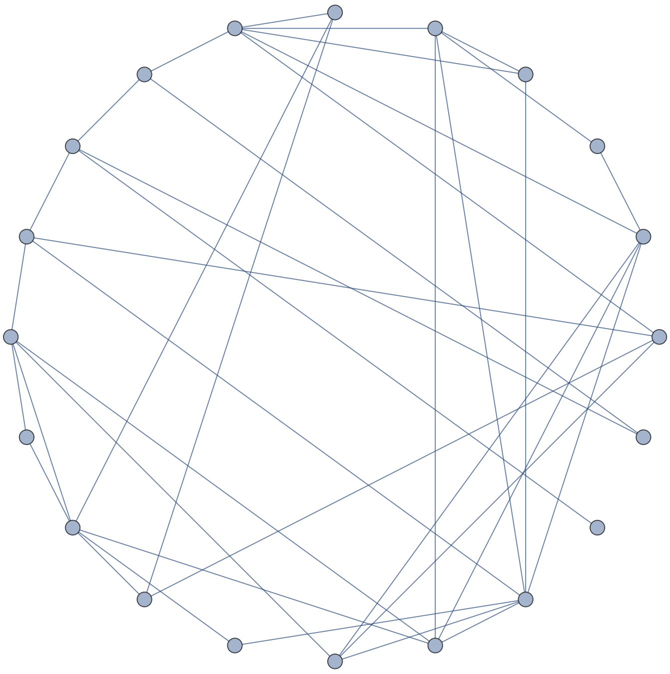
```
EdgeCount[ER]
```

Out[1107]=

```
37
```

- G(n, m)
  - n - number of nodes, m - the number of random (uniformly) edges

In[1110]:=
```
UR = RandomGraph[UniformGraphDistribution[20, Binomial[20, 2] * 1 / 5],
  GraphLayout → "CircularEmbedding"]
```
Out[1110]=



In[1111]:=
```
EdgeCount[UR]
```
Out[1111]=
```
38
```

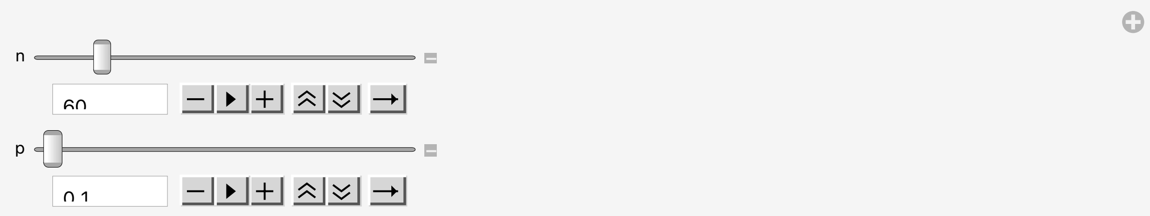
# Question

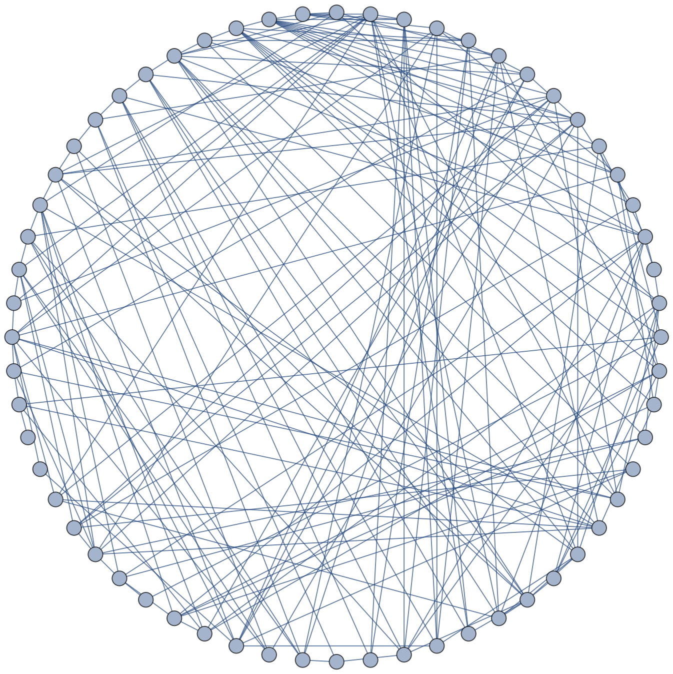
- As the size of the network increases, if you keep p, the probability of any two nodes being connected, the same, what happens to the average degree?
    - a) stays the same
    - b) increases
    - c) decreases

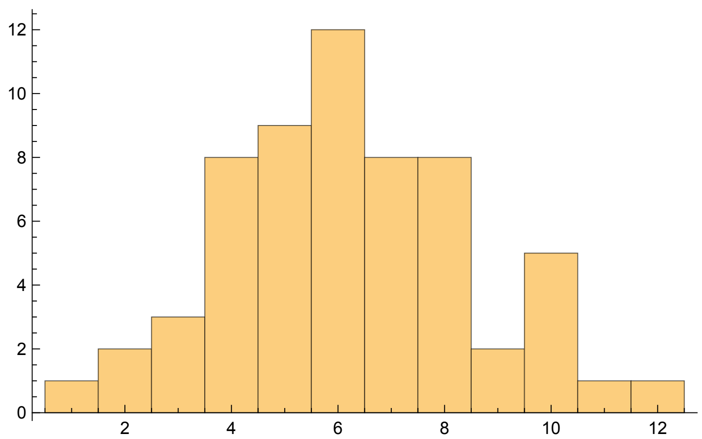
In[1112]:=

```
Manipulate[Row[{gr = RandomGraph[BernoulliGraphDistribution[n, p],
      GraphLayout → "CircularEmbedding", ImageSize → Medium],
    Column[{Histogram[VertexDegree[gr], Max[d = VertexDegree[gr]],
       ImageSize → Medium], "Average Degree:" <> ToString[Mean[d] // N],
      "(n-1)p = " <> ToString[(n - 1) p // N]}]}], {n, 20, 300, 20}, {p, 0.1, 1, 0.1}]
```

Out[1112]=






```
Average Degree:6.23333
(n-1)p = 5.9
```

# How many edges per node?

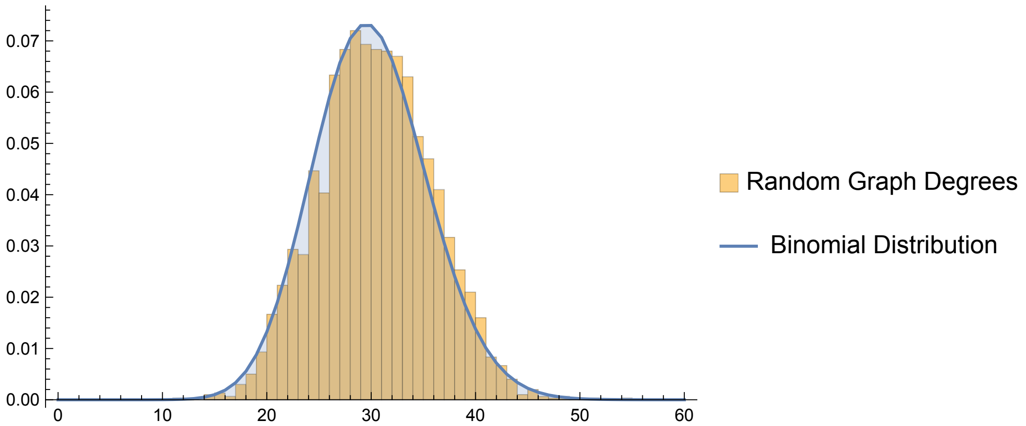
- Each node has (n – 1) tries to get edges
- Each try is a success with probability p
- The binomial distribution gives us the probability that a node has degree k:

$$B(n-1; k; p) = \binom{n-1}{k} p^k (1-p)^{n-k}$$

In[1114]:=

```
Show[
 Histogram[VertexDegree[RandomGraph[BernoulliGraphDistribution[3000, 0.01]]],
  {0, 60, 1}, "Probability", ChartLegends → {"Random Graph Degrees"}],
 DiscretePlot[Evaluate@PDF[BinomialDistribution[3000, 0.01], k],
  {k, 0, 60, 1}, Joined → True, PlotLegends → {"Binomial Distribution"}]]
```

Out[1114]=




Expected Value (Degree):

$$z = (n-1)\,p \approx \text{np}$$

If X is a r.w of the degree then X can be seem as the sum of (n-1) independent trails and

$$X = \sum_{i=1}^{n-1} Y_i \Longrightarrow E[X] = \sum_{i=1}^{n-1} E[Y_i] = (n-1)\,p$$

Variance:

$$\sigma^2 = (n-1)\;p\;(1-p)$$

## Approximation

$$p_k = \binom{n-1}{k} p^k (1-p)^{n-1-k}$$

$$p_k = \frac{z^k e^{-z}}{k!}$$

$$p_k = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(k-z)^2}{2\sigma^2}}$$

Binomial

↓ limit *p* small

Poisson

↓ limit large *n*

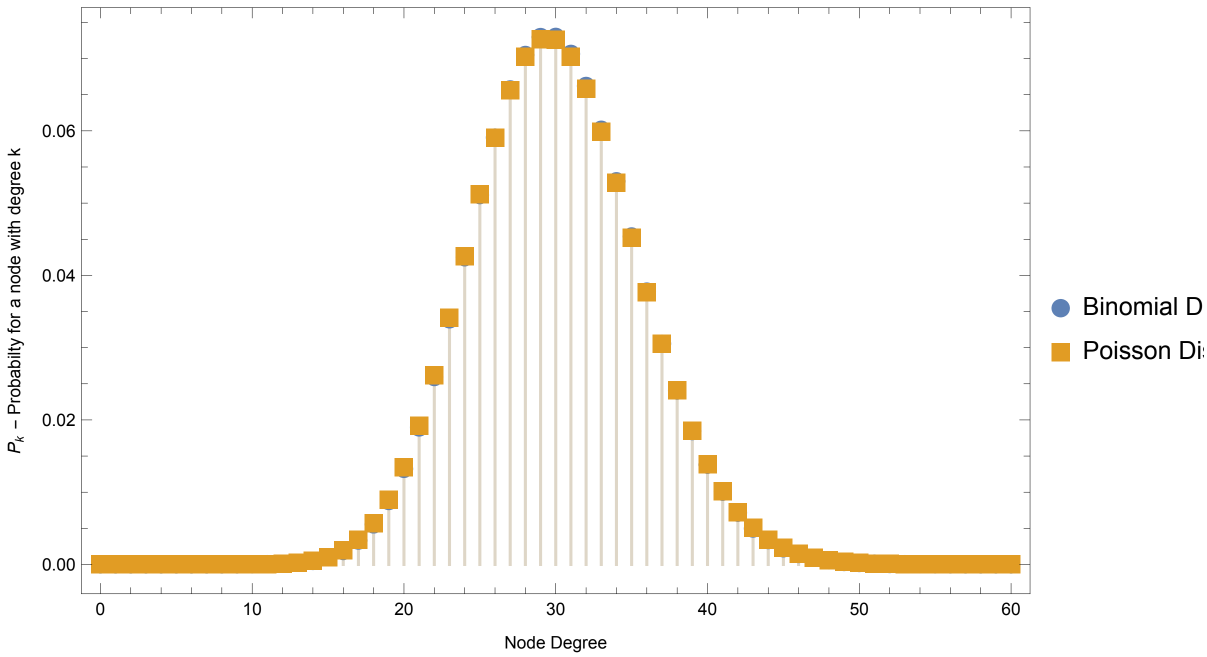
Normal

- Where z is the expected degree z = p(n-1)

In[1115]:=

```
DiscretePlot[Evaluate[{PDF[BinomialDistribution[3000, 0.01], k],
   PDF[PoissonDistribution[(3000 - 1) 0.01], k]}],
 {k, 0, 60}, PlotRange → All, PlotMarkers → {Automatic, 10},
 PlotLegends → {"Binomial Distribution (3000,0.01)",
   "Poisson Distribution (Z)"}, Frame → True,
 FrameLabel → {"Node Degree", "Pk - Probabilty for a node with degree k"},
 ImageSize → Medium]
```

Out[1115]=




# What insights does this yield?

# No hubs!

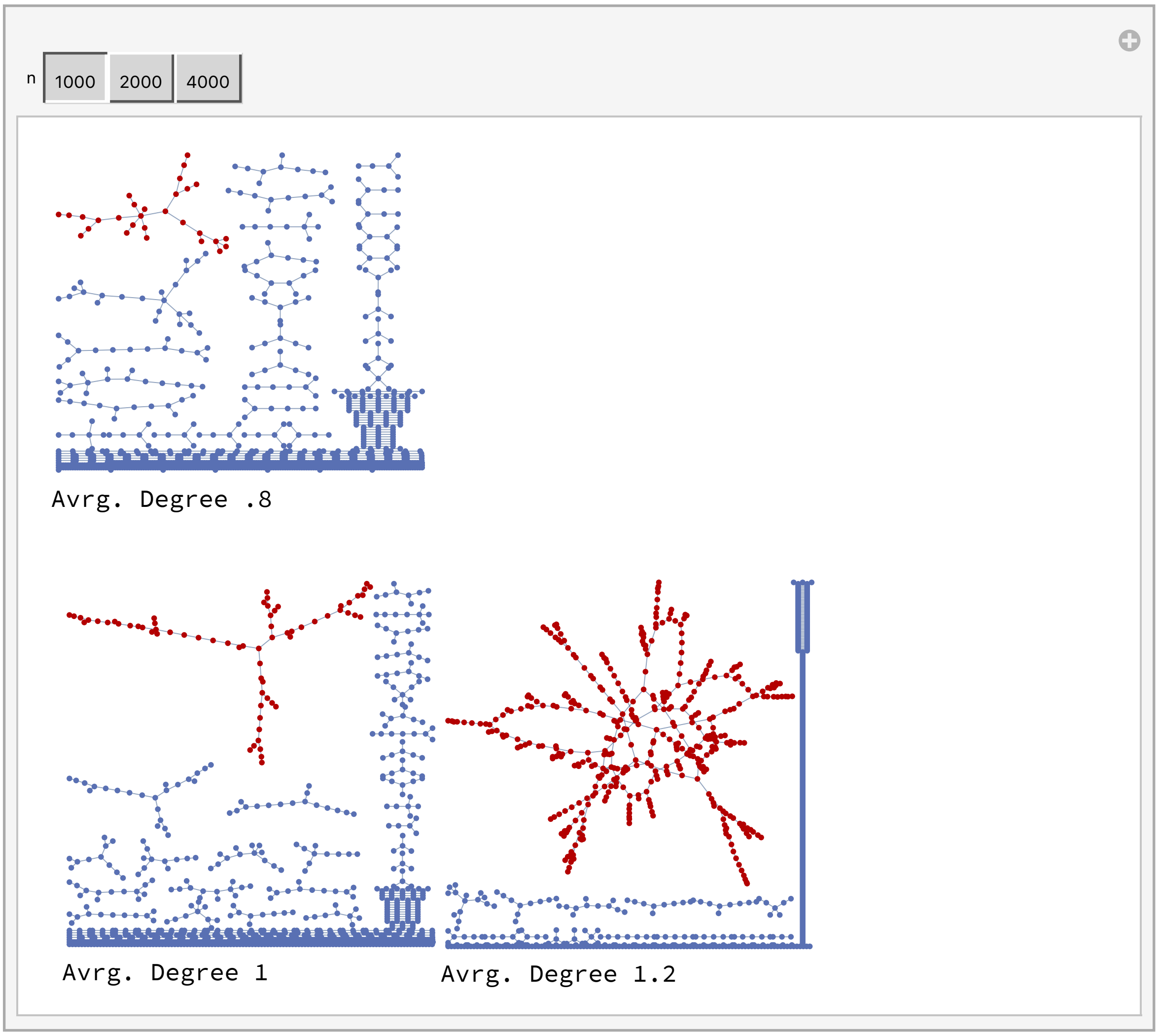
## You don’t expect large hubs in the network

# Emergence of the giant component

In[1116]:=

```
Manipulate[
 Row[{Column[{HighlightGraph[g1 = RandomGraph[BernoulliGraphDistribution[n,
         .8 / (n - 1)], ImageSize → 200], First@ConnectedComponents[g1]],
     "Avrg. Degree .8"}], Column[{HighlightGraph[g2 =
        RandomGraph[BernoulliGraphDistribution[n, 1 / (n - 1)], ImageSize → 200],
      First@ConnectedComponents[g2]], "Avrg. Degree 1"}],
   Column[{HighlightGraph[g3 = RandomGraph[BernoulliGraphDistribution[
         n, 1.2 / (n - 1)], ImageSize → 200], First@ConnectedComponents[g3]],
     "Avrg. Degree 1.2"}]}], {n, {1000, 2000, 4000}}]
```

Out[1116]=

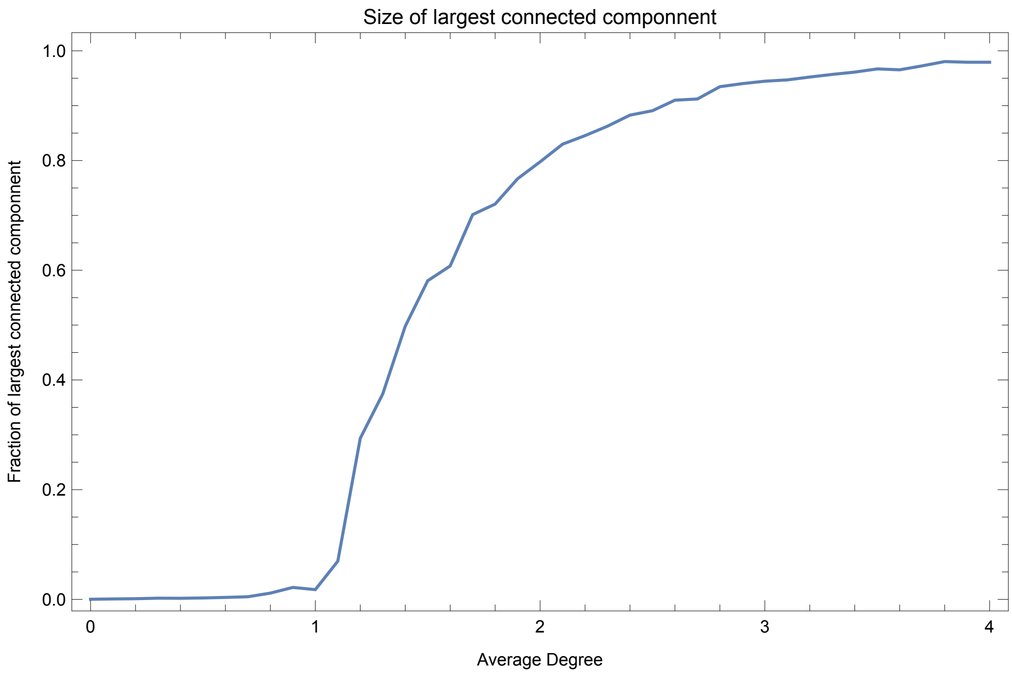



## ■ 5,000 Network

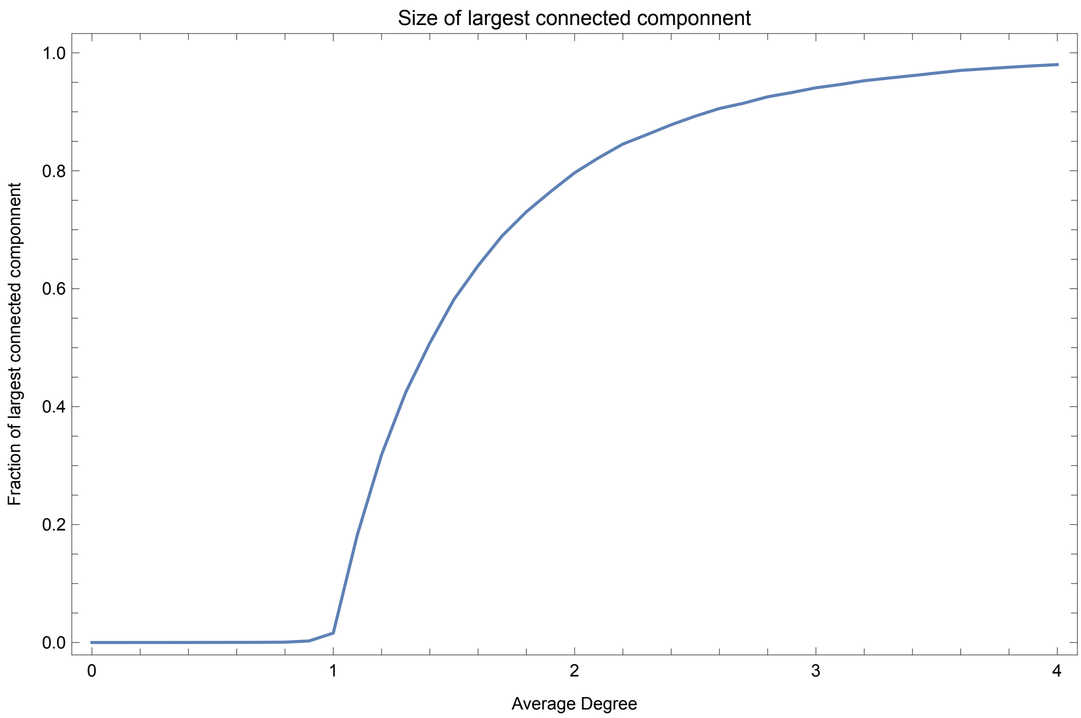
In[1117]:=

```
ListLinePlot[Table[{i, Length@First@ConnectedComponents[
      RandomGraph[BernoulliGraphDistribution[5000, i / (5000 - 1)]]] / 5000},
  {i, 0, 4, 0.1}], Frame → True, FrameLabel → {"Average Degree",
   "Fraction of largest connected componnent"},
 PlotLabel → "Size of largest connected componnent"]
```

Out[1117]=




■ 200 K Network

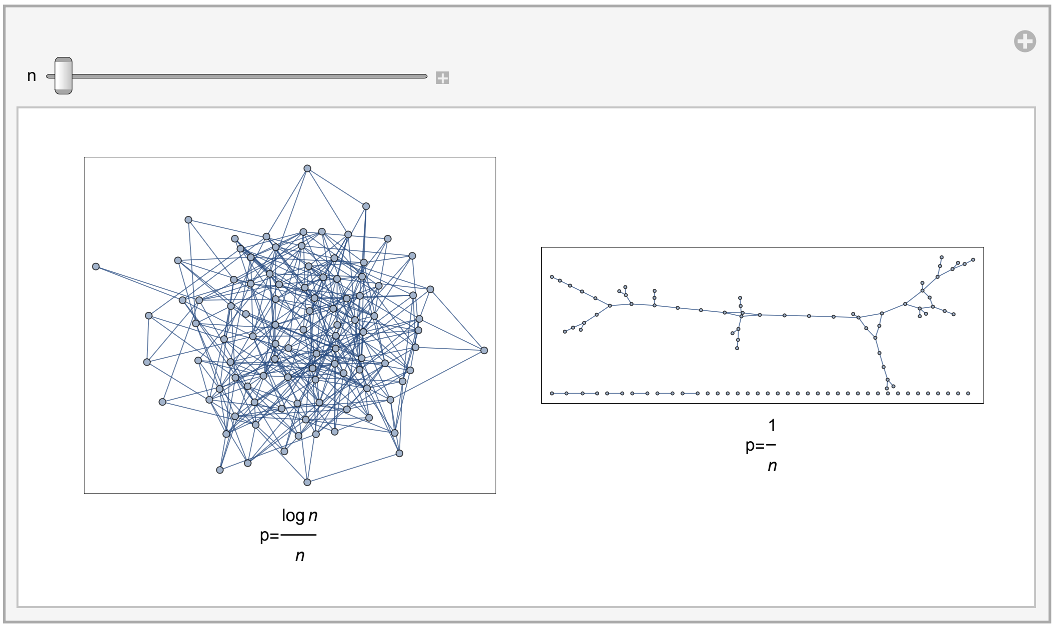



# A Phase Transition!

In[1119]:=

```
Manipulate[
 GraphicsRow[{RandomGraph[BernoulliGraphDistribution[n, Log[2, n] / (n)],
    Frame → True, FrameLabel → "p=log n/n"], RandomGraph[
    BernoulliGraphDistribution[n, 1 / (n)], Frame → True, FrameLabel → "p=1/n"]},
  ImageSize → 500], {{n, 100}, 100, 1000, 100}]
```

Out[1119]=



# Why just one giant component?

If you have 2 large - components each occupying roughly 1/2 of the graph, how long does it typically take for the addition of random edges to join them into one giant component?

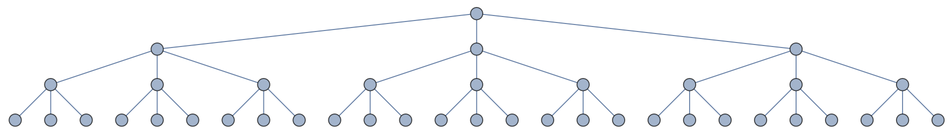
- 1 - 4 edge additions
- 5 - 20 edge additions
- over 20 edge additions

# Average shortest path?

- How many hops on average between each pair of nodes?
- Again, each of your friends has z = avg. degree = (n-1)p friends besides you
- Ignoring loops, the number of people you have at distance l is $n_l = z^l$

$$n_l = z^l$$

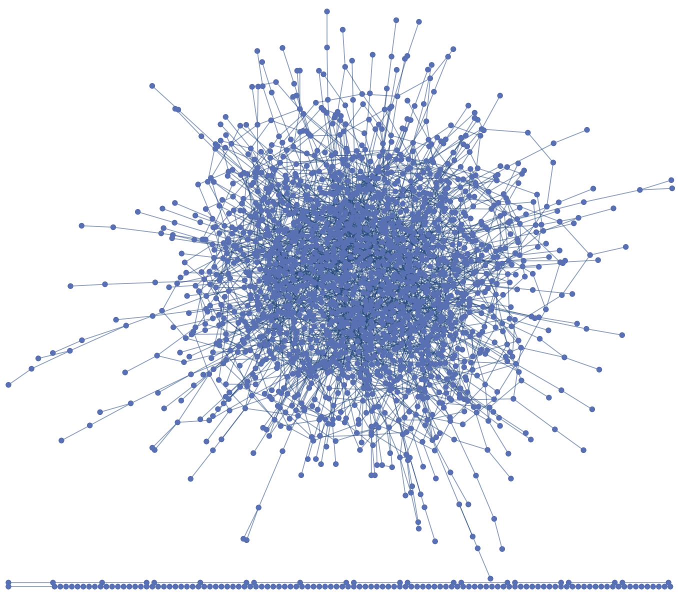
```
KaryTree[40, 3]
```



In[1120]:=

```
ERD = RandomGraph[BernoulliGraphDistribution[2000, 3 / (2000 - 1)]]
```

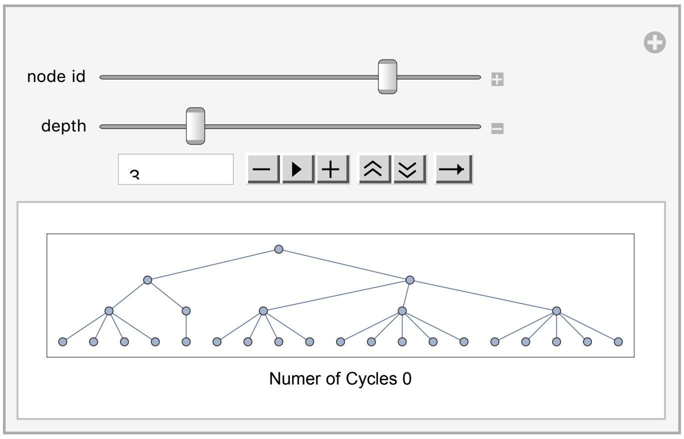
Out[1120]=



In[1121]:=

```
Manipulate[
 TreePlot[g = NeighborhoodGraph[ERD, j, i], ImageSize → 400, Frame → True,
  FrameLabel → "Numer of Cycles " <> ToString[Length[FindCycle[g, i + 3, All]]]],
 {{j, 1, "node id"}, 1, 10, 1}, {{i, 1, "depth"}, 1, 10, 1}]
```

Out[1121]=




- Approximate the Diameter

$$n = n_D = z^D$$

$$D \approx \frac{\text{Log}(n)}{\text{Log}(Z)} \leq \text{Log}(n)$$

If $Z = \text{Log}(n)$ then $D \approx \dfrac{\text{Log}(n)}{\text{Log}(\log n)}$

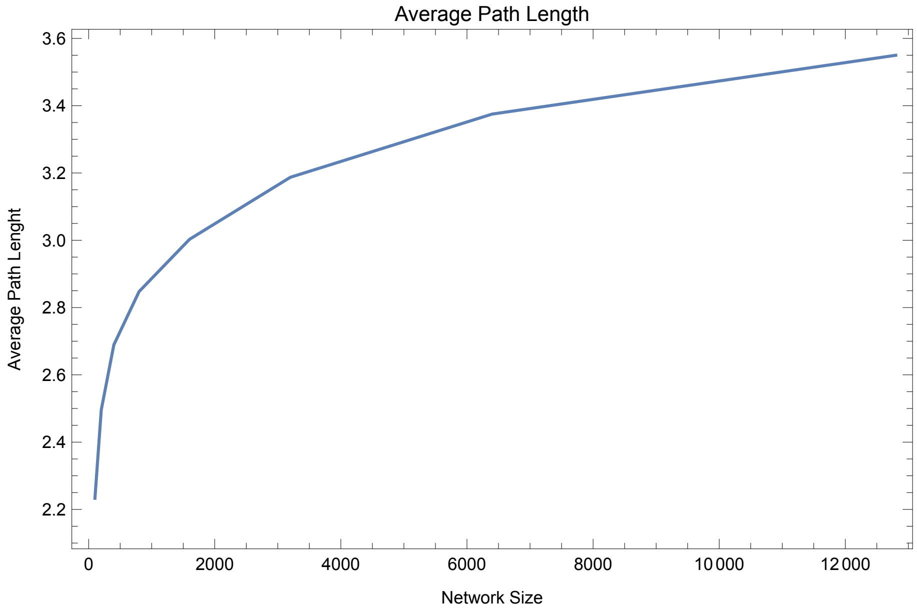
In[1122]:=

```
AvgLength = Table[{100 × 2^i,
   Mean@Flatten@GraphDistanceMatrix[RandomGraph[BernoulliGraphDistribution[
        100 × 2^i, 2 Log[100 × 2^i] / (100 × 2^i - 1)]]]}, {i, 0, 7, 1}];
```

In[1123]:=

```
ListLinePlot[AvgLength, Frame → True,
 FrameLabel → {"Network Size", "Average Path Lenght"},
 PlotLabel → "Average Path Length"]
```

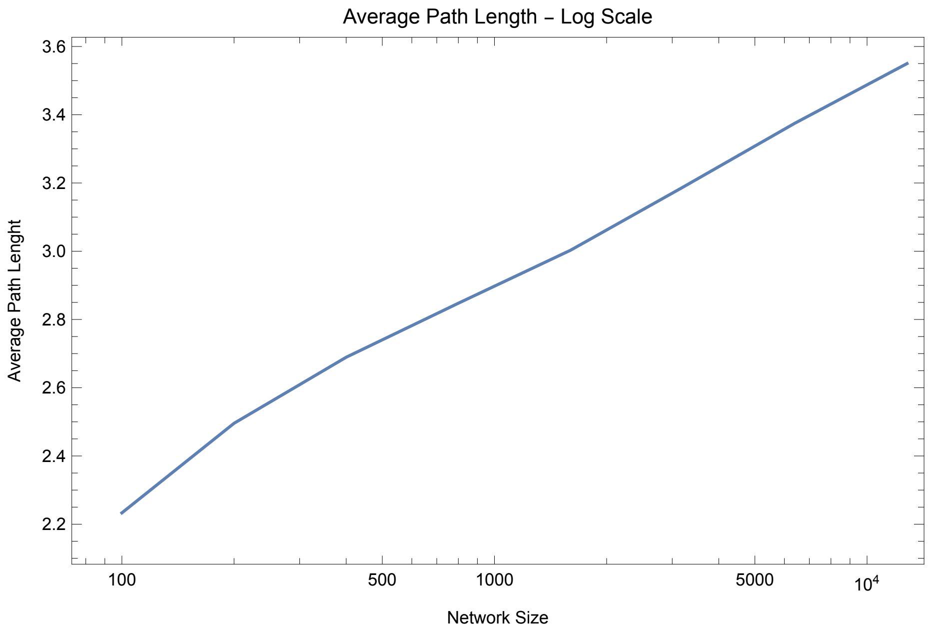
Out[1123]=




In[1124]:=

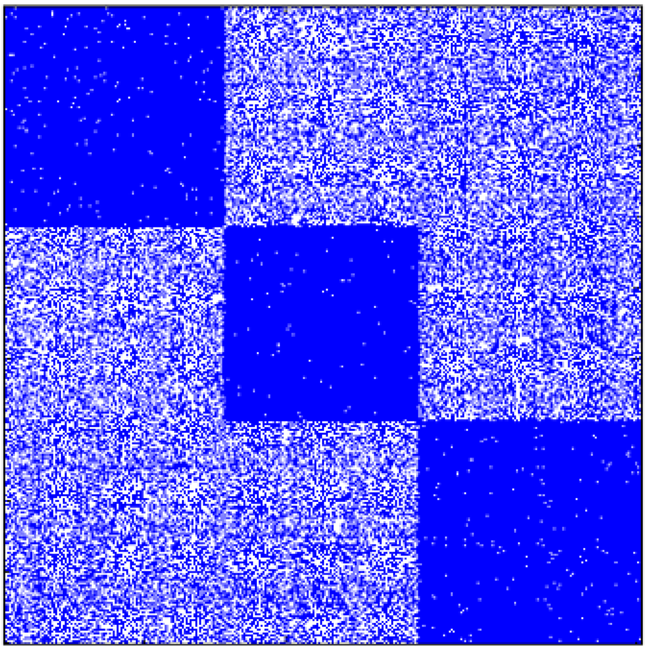
```
ListLogLinearPlot[AvgLength, Frame → True,
 FrameLabel → {"Network Size", "Average Path Lenght"},
 PlotLabel → "Average Path Length - Log Scale", Joined → True]
```

Out[1124]=



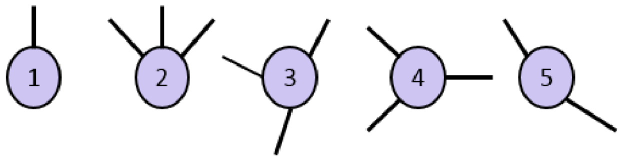
# Block Model

- The stochastic block model takes the following parameters:
  - The number n of vertices;
  - A partition of the vertex set {1, ... ,n} into disjoint subsets $\{C_1, \ldots, C_r\}$, called communities;
  - a symmetric $r \times r$ matrix P of edge probabilities.
- The edge set is then sampled at random as follows: any two vertices in $C_i$ and $C_j$ are connected by an edge with probability $P_{\mathrm{ij}}$.
  - If the probability matrix is a constant, in the sense that $P_{\mathrm{ij}}$=p for all i,j, then the result is the Erdős–Rényi model G(n,p).
  - The planted partition model is the special case that the values of the probability matrix P are a constant p on the diagonal and another constant q off the diagonal.



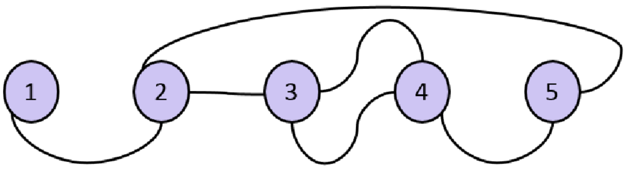
# Random Configuration Model

## 1. Degree Sequence
## 2. Set edges




1 2 2 2 3 3 3 4 4 4 5 5

## 3. Random Matching (here by random order)




1 2 2 5 3 4 3 2 4 5 4 3

In[1131]:=

```
SetDirectory[NotebookDirectory[]];
file = Rest[Import["data/stormofswords.csv"]];
tribes = Import["data/tribes.csv"];
nodes = Flatten[tribes⟦All, 1⟧];
edges = #⟦1⟧ ⟷ #⟦2⟧ & /@ file [[All, {1, 2}]];
ThronesG = Graph[nodes, edges, VertexLabels → Placed["Name", Tooltip]]
```

Out[1136]=

In[1137]:=

```
DS = VertexDegree[ThronesG]
```

Out[1137]=

```
{1, 5, 4, 3, 1, 2, 19, 6, 6, 4, 6, 2, 14, 7, 4, 10, 18, 20, 2, 5, 1, 4, 14, 3, 5, 1, 4, 12,
 3, 8, 5, 2, 4, 4, 12, 4, 4, 3, 1, 6, 4, 24, 6, 2, 18, 4, 26, 2, 6, 1, 6, 1, 1, 9, 5,
 2, 10, 3, 12, 7, 3, 5, 4, 7, 2, 4, 2, 7, 2, 1, 7, 5, 4, 5, 1, 1, 1, 4, 8, 6, 2, 6,
 25, 18, 3, 4, 4, 1, 15, 13, 26, 5, 1, 14, 2, 4, 5, 5, 36, 22, 4, 7, 3, 8, 1, 1, 4}
```

In[1138]:=

```
RandomGraph[DegreeGraphDistribution[DS]]
```

Out[1138]=

# Class Assignment

1. Using Network of Throne data generate G(n,p), G(n,m), Block model, configuration model
2. Compare for graphs: Degree Distribution, Giant component, Average distance, Shift Diagram for Core-Periphery, Community Detedction.
3. Discuss your finding, what (and why) are the differences with the original network.