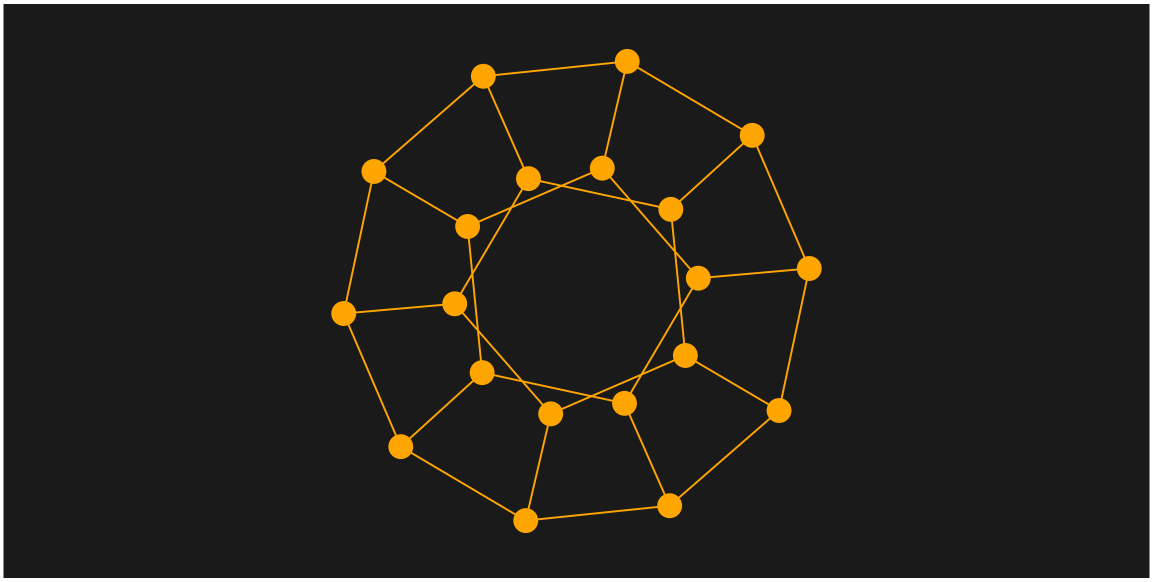
Random Network Generation

Mohammad Ghoraishi - 99100788

We want to generate a network with number of nodes N and average degree λ. I'm using Julia for this purpose and the network creation function is written in t he RandomNetwork.jl file. I have not used any libraries for the purpose of generating the network and the network is stored in an adjacency list which is just an array where each node has a corresponding index and in each index of the array, another array is stored that shows which nodes is said node connected to.
Example: (*Example Source: Karnak Julia Docs*)
[[2, 5, 7],
[1, 3, 9],
[2, 4, 11],
[3, 5, 13],
[1, 4, 15],
[7, 15, 20],
[1, 6, 8],
[7, 9, 16],
[2, 8, 10],
[9, 11, 17],
[3, 10, 12],
[11, 13, 18],
[4, 12, 14],
[13, 15, 19],
[5, 6, 14],
[8, 17, 20],
[10, 16, 18],
[12, 17, 19],
[14, 18, 20],
[6, 16, 19]]

which generates the graph below:



The function used to generate the graphs is called **GenerateNetwork** located in the *RandomNetwork.jl* file. An explanation of said function is brought below: GenerateNetwork(lambda::Float64, N::Integer, UseTrueRandom::Bool)

Generates a random network with average degree lambda and nodes N.

Arguments

- `lambda` = the average degree of all nodes.
- `N` = Number of nodes in the network.
- `AlgorithmType` = if type 1 is selected, vertices will be generated with a probability and multiple vertices between the same nodes wouldn't exist.

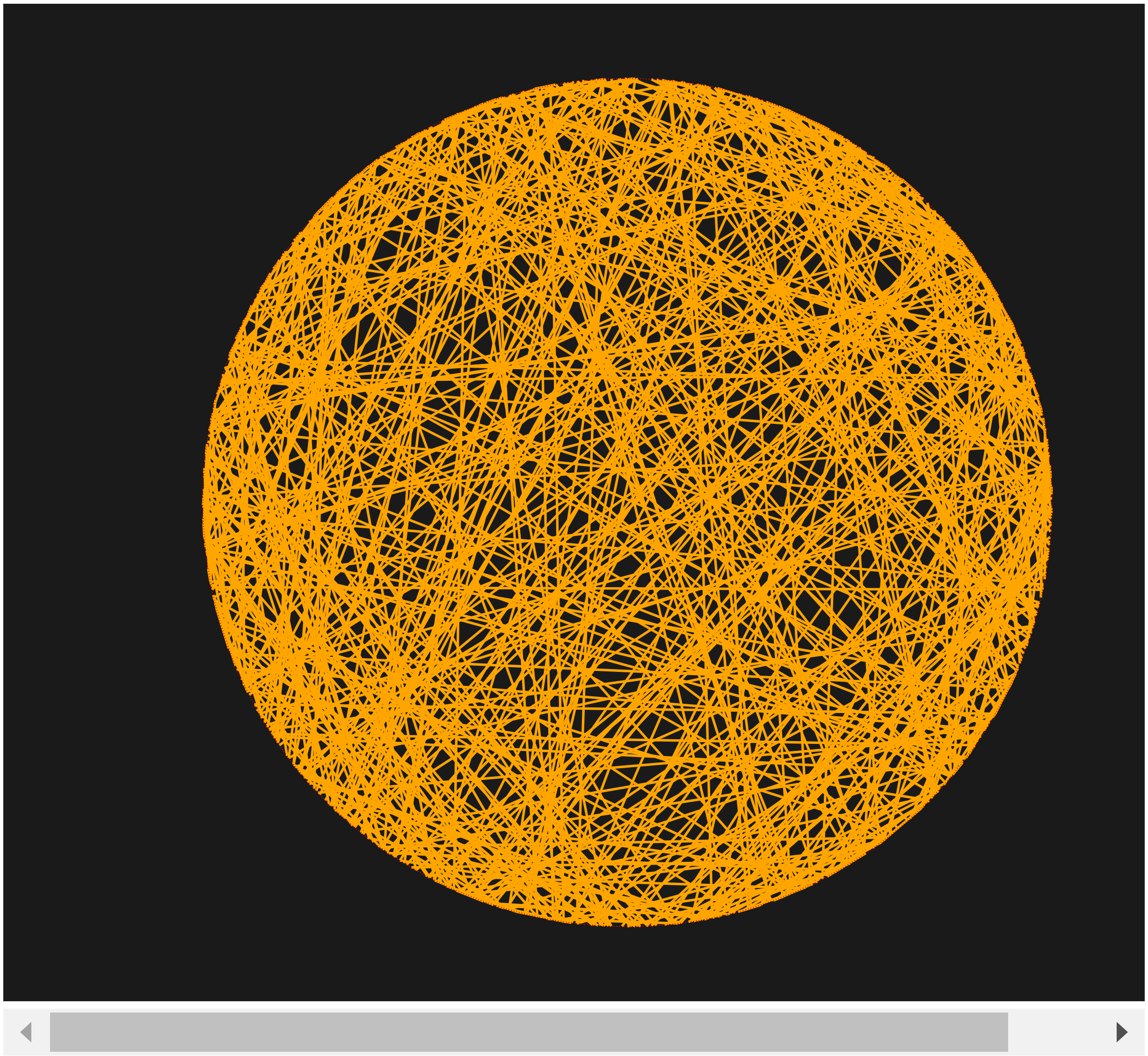
  if type 2 is selected, then an exact number of vertices will be generated and then they would be randomly distributed.

In [ ]:
```
include("RandomNetwork.jl")
using .RandomNetwork
using Karnak, Graphs, Plots, Colors #I'm using karnak and Graphs to draw the graphs
```

Using Algorithm Type one:

In [ ]:
```
#Initial parameters
N = 1000
lambda = 1
Net = RandomNetwork.GenerateNetwork(lambda, N, 1)
g = Graph(N, Net)

@drawsvg begin
    background("grey10")
    sethue("orange")
    drawgraph(g, layout=shell, vertexshapesizes = 0.1, vertexstrokeweights = 0.001,
    end 500 400
```



After $N = 1000$ The plot will almost become a filled in circle so I won't draw that.(I drew the graph using the shell layout because the calculations for the stress layout would take too long).
Just to see that the generator works, here is the adjacency list and a test for λ:

In [ ]:

```
Net
```

```
1000-element Vector{Vector{Int64}}:
 []
 [577, 994]
 [317]
 []
 []
 [79, 220, 605, 794]
 [248]
 []
 [941]
 [241, 343]
 ⋮
 []
 []
 [2, 219, 423, 560, 849, 892, 920]
 []
 [585]
 [682]
 [127]
 []
 [505]
```

In [ ]:

```
global sum = 0
for i in 1:N
    global sum += length(Net[i])
end
ActualLambda = sum / N
println("Actual value for lambda = $ActualLambda")
```

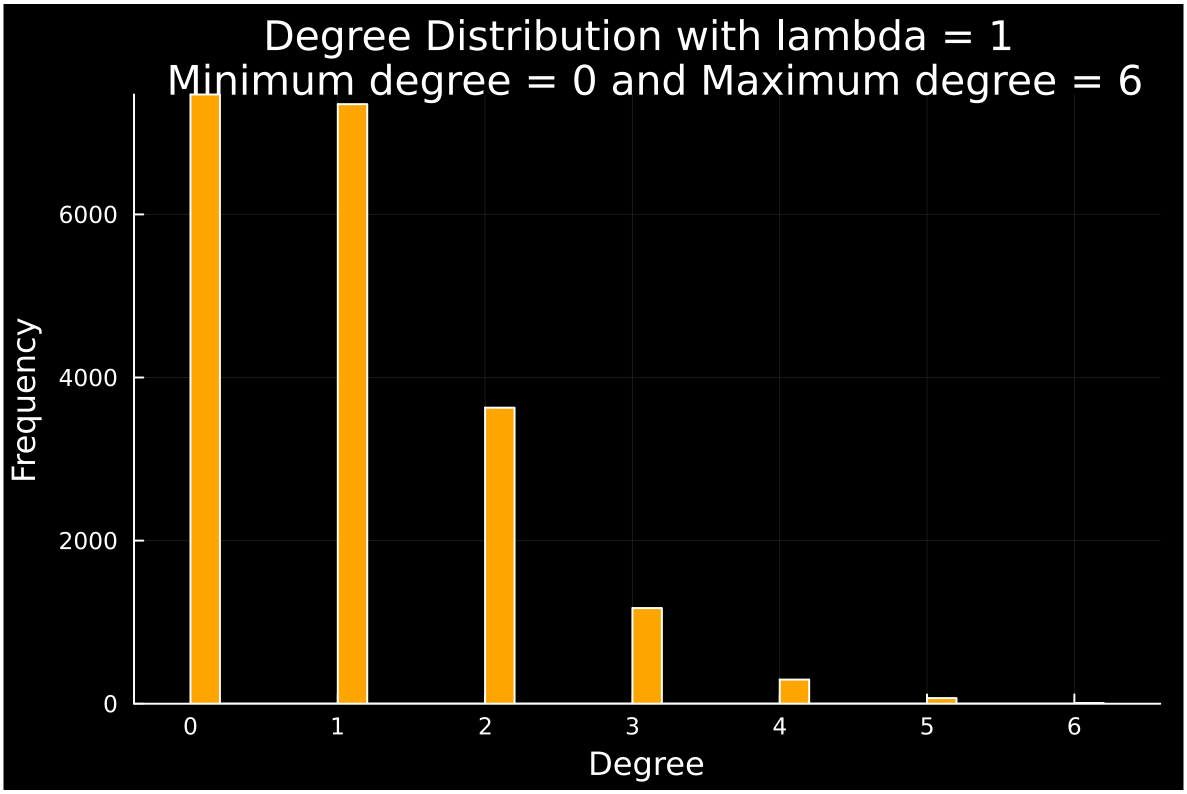
```
Actual value for lambda = 0.978
```

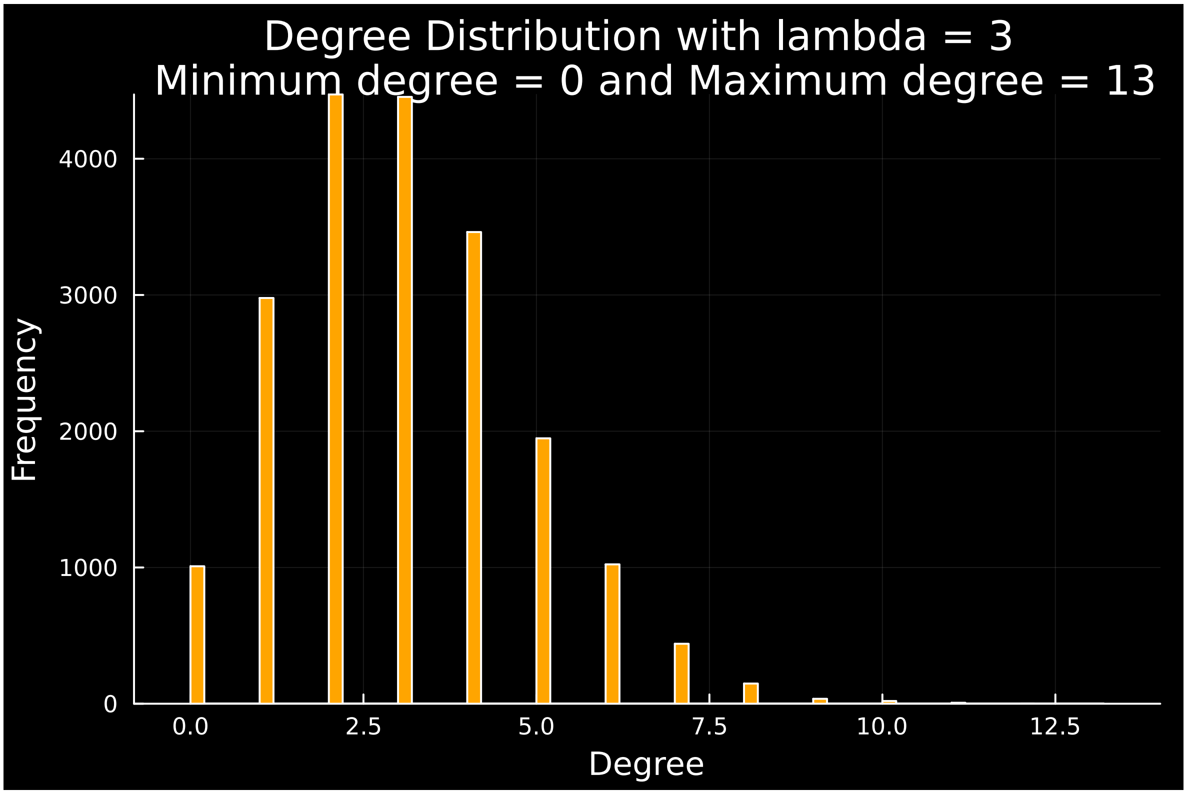
Degree Distribution

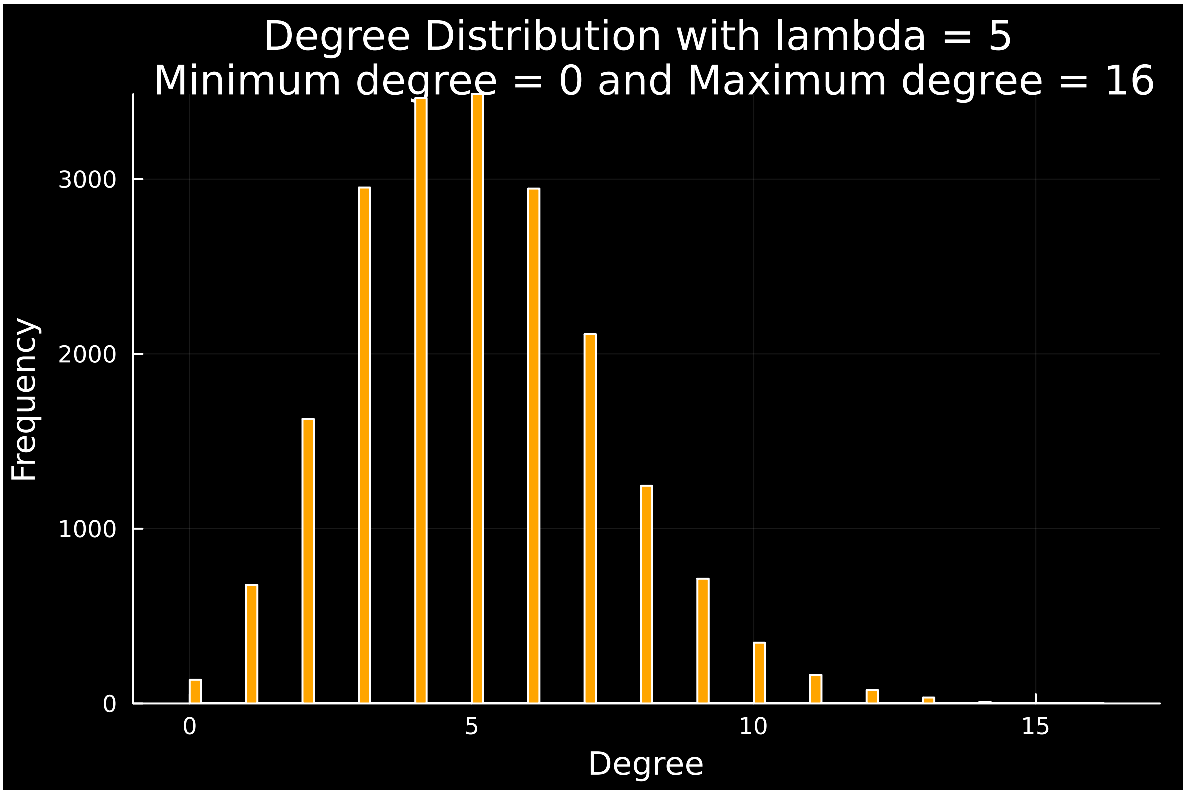
First we generate a network then put their degrees into an array, then we plot the histogram of said array. (The minimum and maximum degrees are available in the plots)

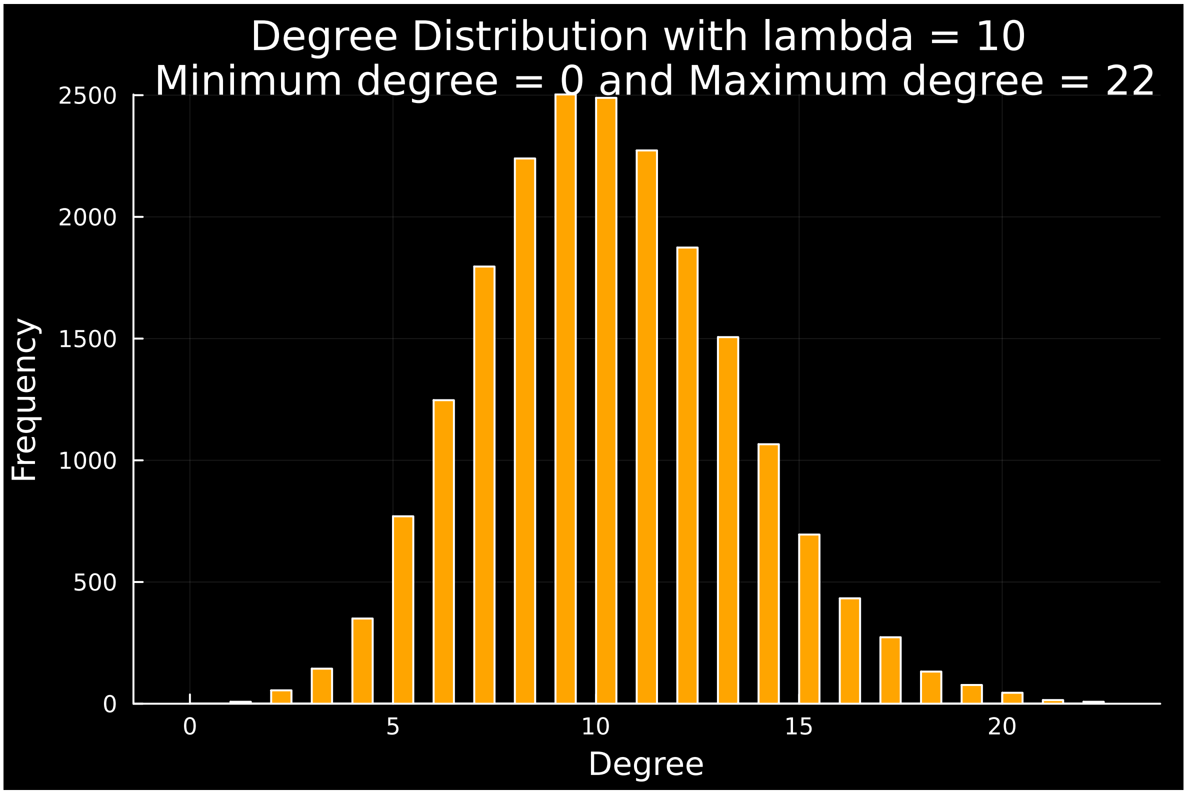
In [ ]:

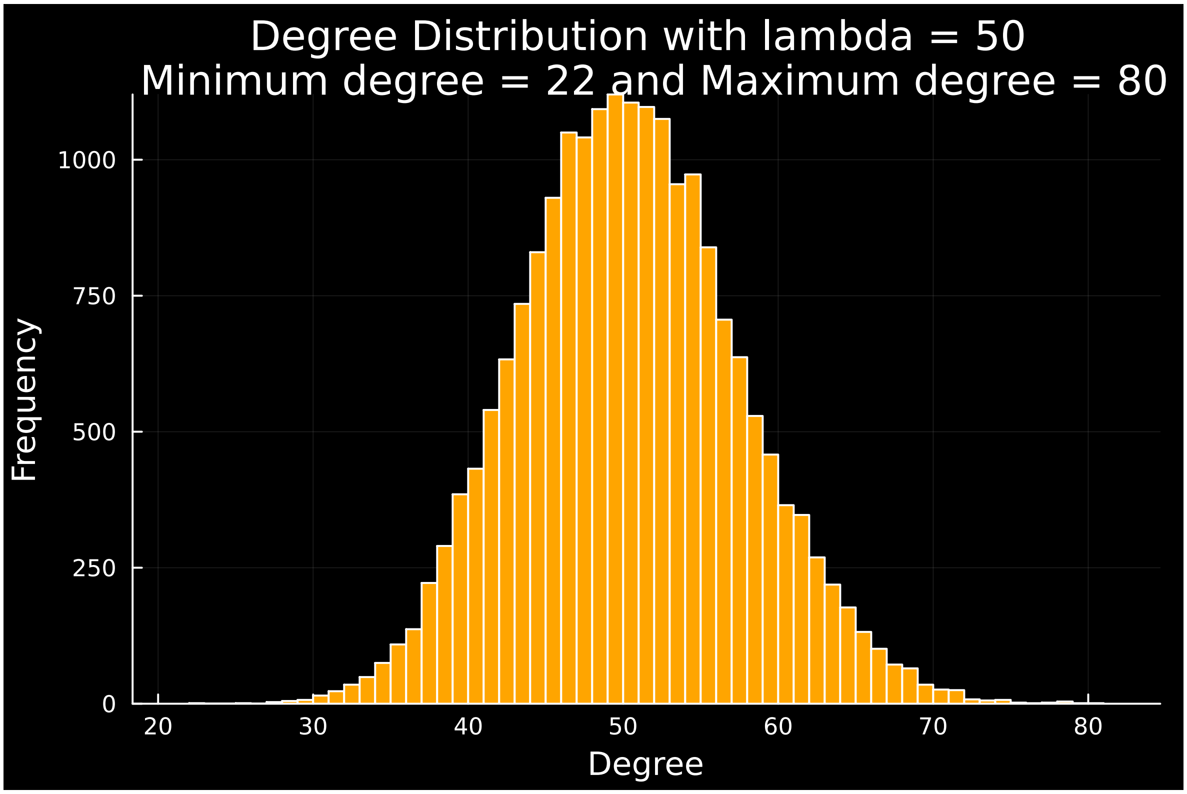
```
#Network Generation
N = 20000
lambdaArr = [1, 3, 5, 10, 50, 100] #Array of all lambdas we want to test
for lambda in lambdaArr
    DegreesArr = Int64[]
    Net = RandomNetwork.GenerateNetwork(lambda, N, 1)
    for j in 1:N
        push!(DegreesArr, length(Net[j]))
    end
    MinimumDegree = minimum(DegreesArr)
    maximumDegree = maximum(DegreesArr)
    display(histogram(DegreesArr, dpi = 300, title = "Degree Distribution with lamb
    , background = :black, legend = false,
     xlabel = "Degree", ylabel = "Frequency", color = :orange))
end
```

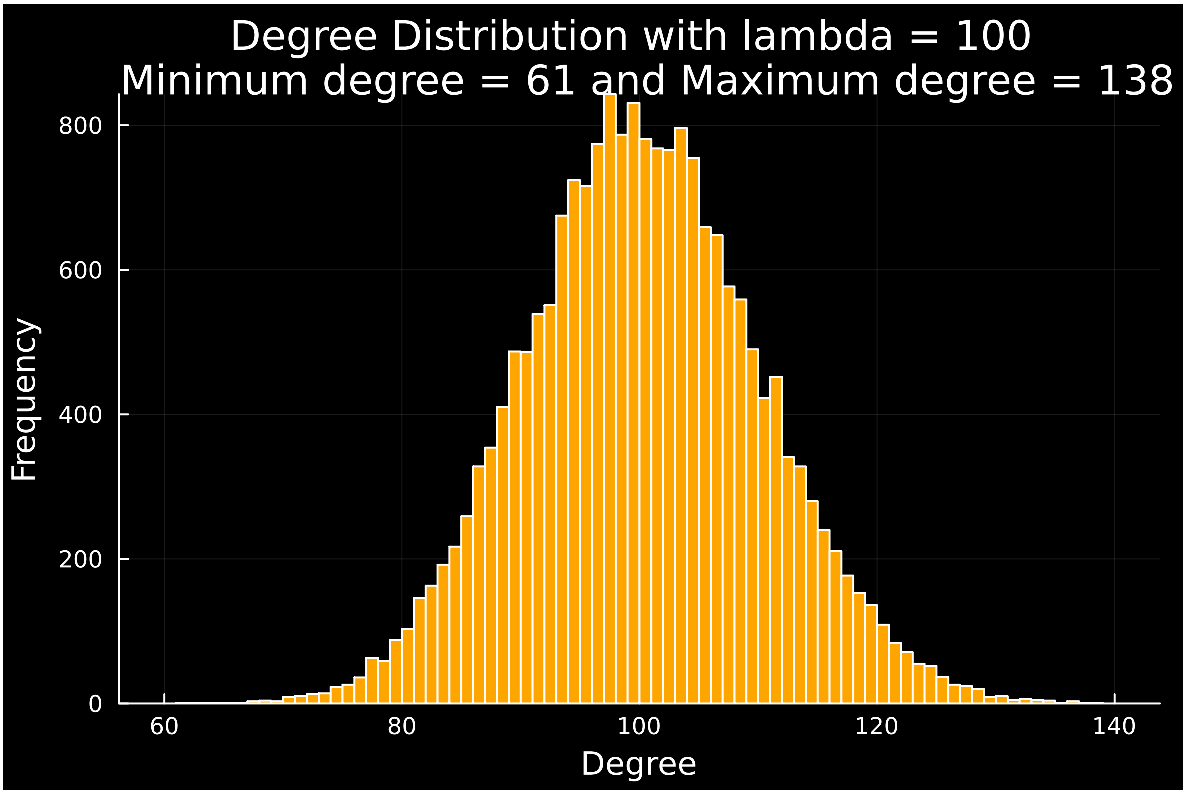

Degree Distribution with lambda = 1
Minimum degree = 0 and Maximum degree = 6
Frequency
0
2000
4000
6000
Degree
0
1
2
3
4
5
6


Degree Distribution with lambda = 3
Minimum degree = 0 and Maximum degree = 13
Frequency
0
1000
2000
3000
4000
Degree
0.0
2.5
5.0
7.5
10.0
12.5


Degree Distribution with lambda = 5
Minimum degree = 0 and Maximum degree = 16
Frequency
0
1000
2000
3000
0
5
10
15
Degree


Degree Distribution with lambda = 10
Minimum degree = 0 and Maximum degree = 22
Frequency
0
500
1000
1500
2000
2500
0
5
10
15
20
Degree



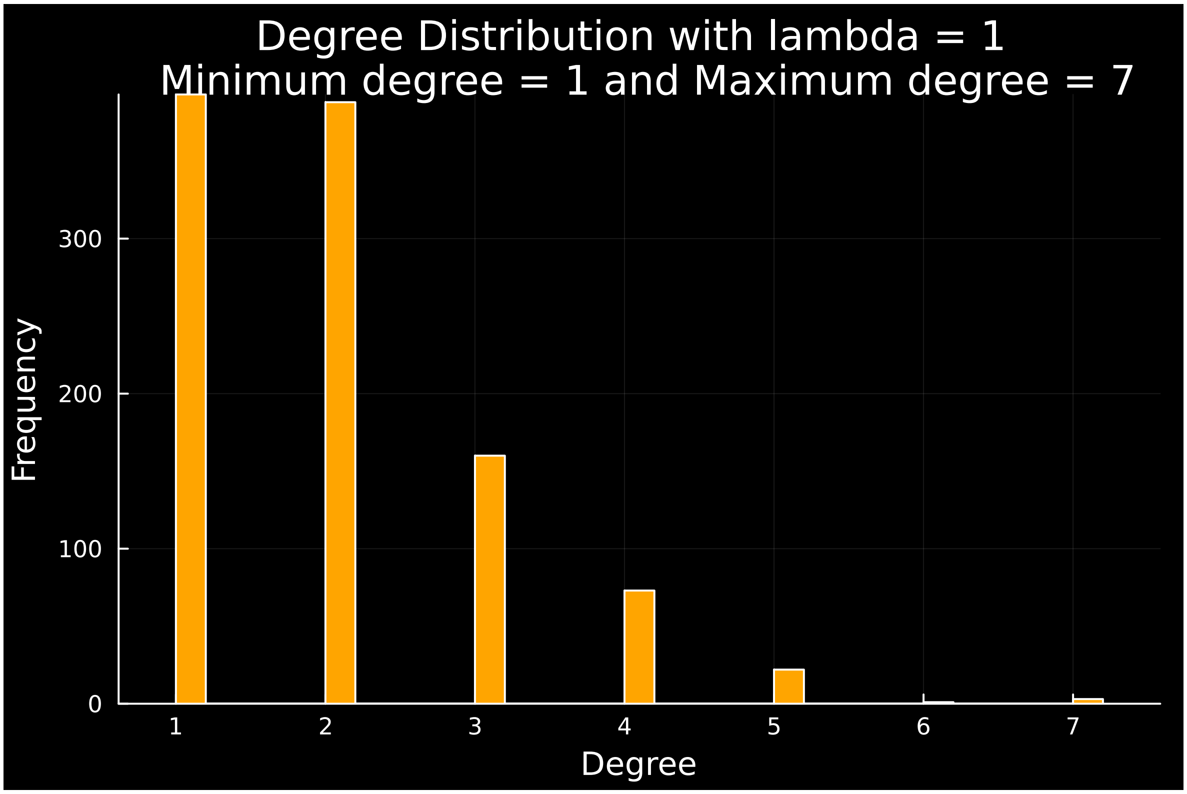


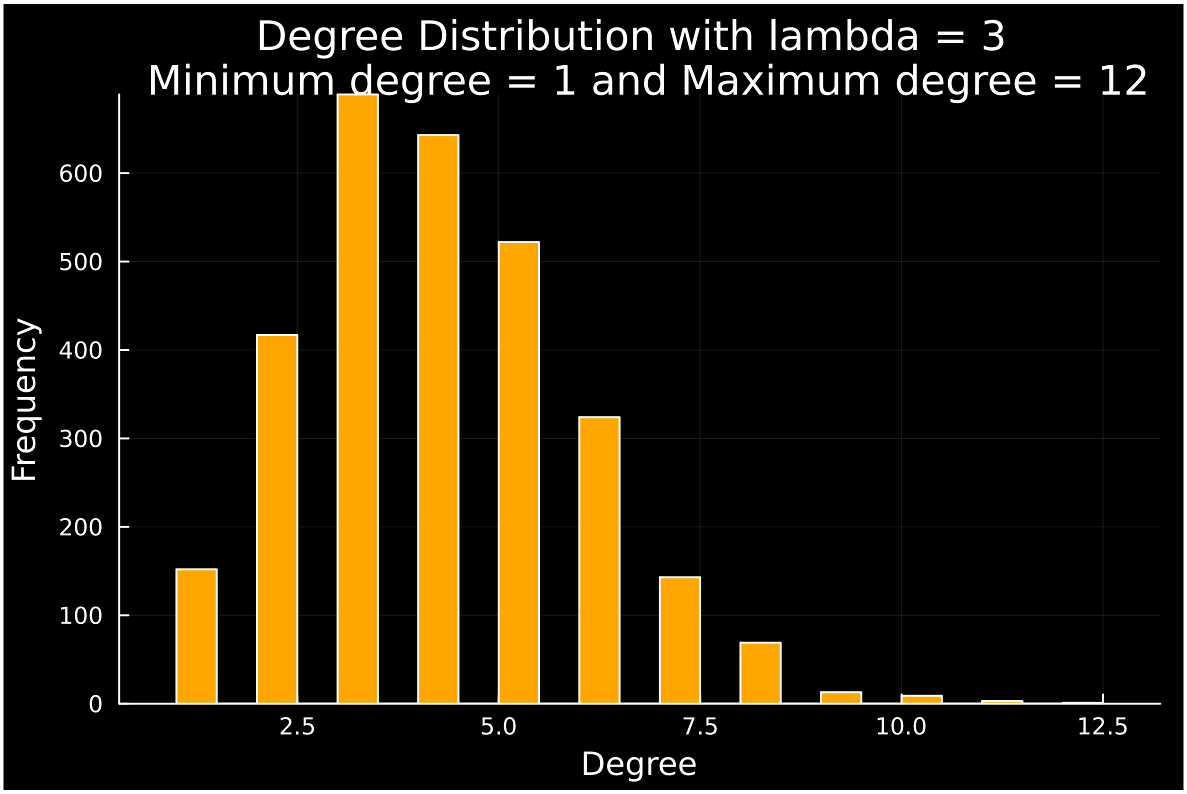
The Distribution seems to look like a normal distribution in higher λ s around the specified λ .

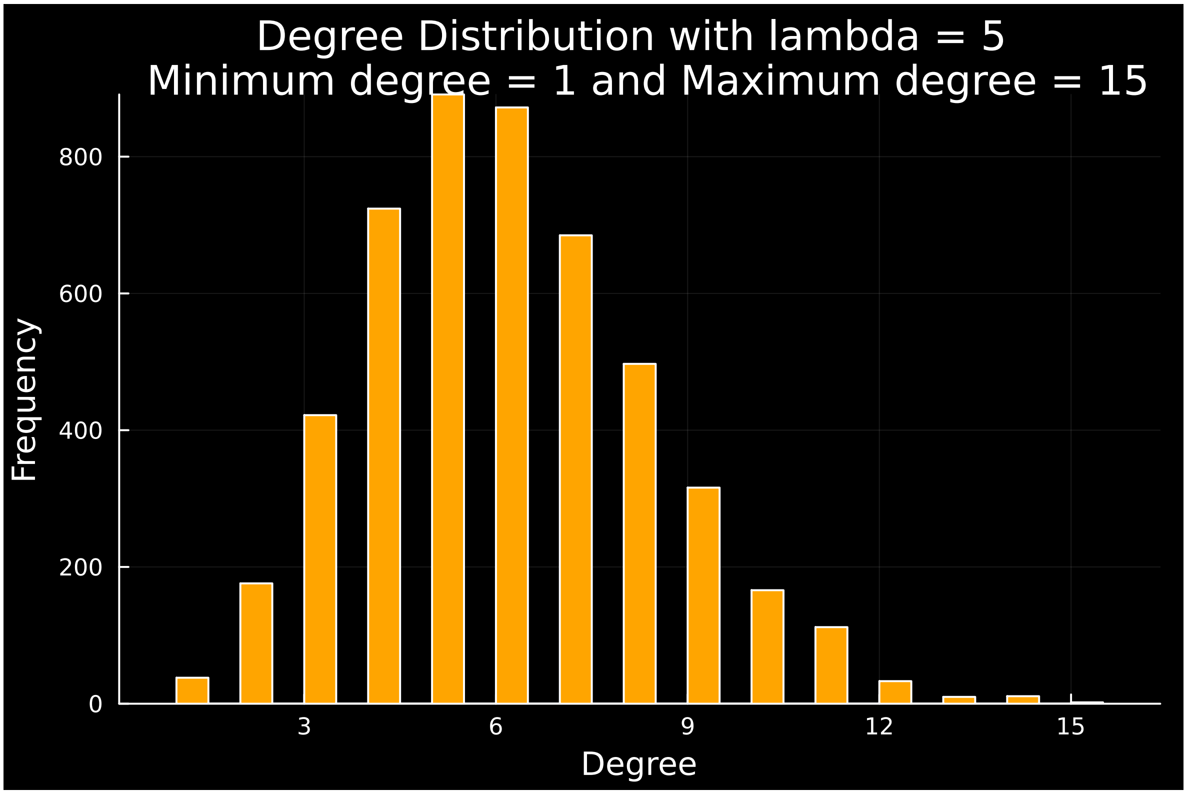
Neighbour Degree Distribution

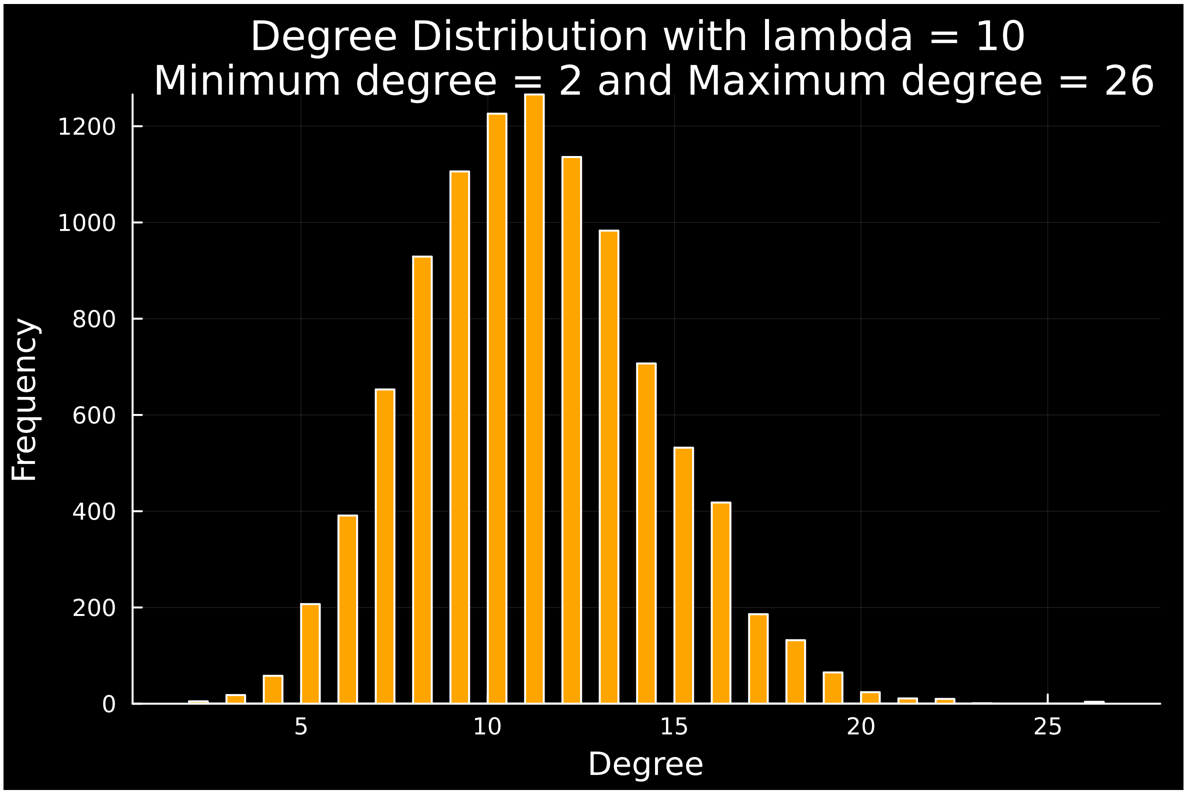
Here we select a number of nodes, then we find their neighbours and store the degrees of these neighbour nodes in a new array. We exclude the nodes that don't have any neighbours. (The minimum and maximum degrees specified in the plots show the minimum and maximum degrees of neighbours of nodes)

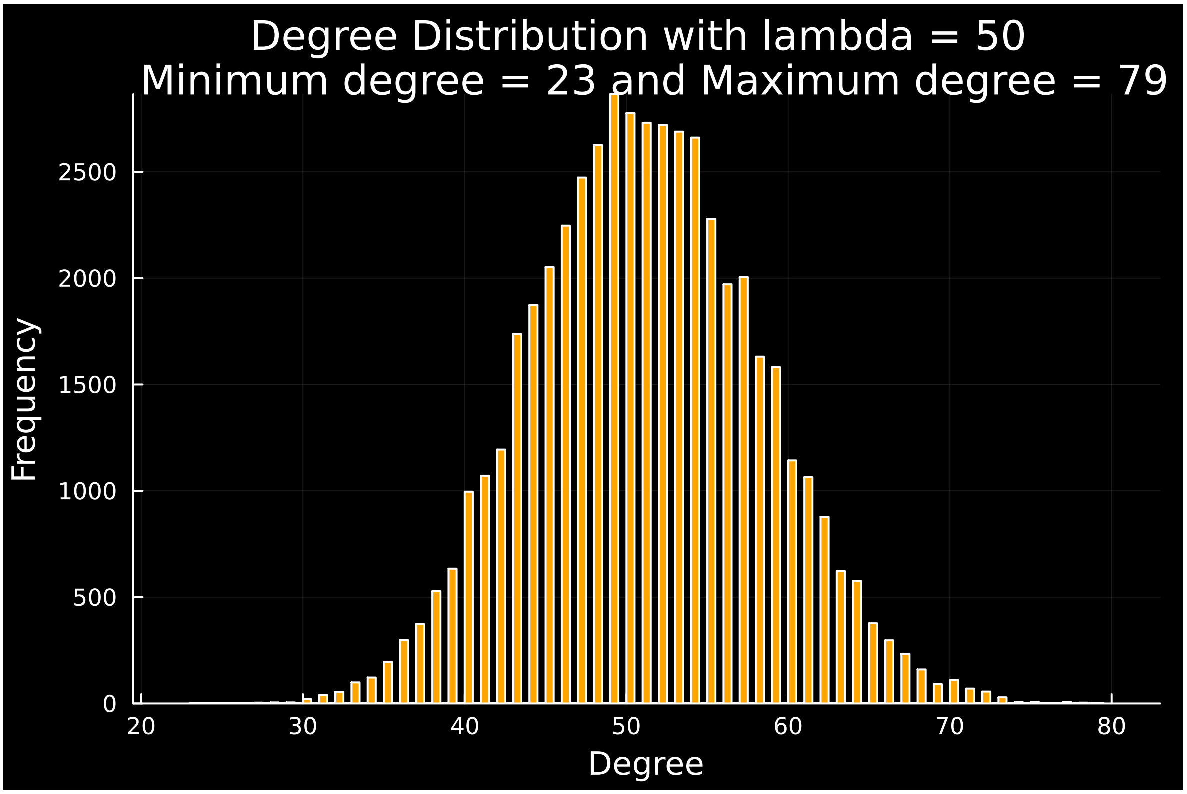
```
In [ ]: N = 20000
        lambdaArr = [1, 3, 5, 10, 50, 100] #Array of all lambdas we want to test
        SampleCount = 1000 #Number of randomly chosen nodes
        for lambda in lambdaArr
            DegreesArr = Int64[]
            Net = RandomNetwork.GenerateNetwork(lambda, N, 1)
            for j in 1:SampleCount
                SelectedNode = rand(1:N)
                NeighbourArr = Net[SelectedNode]
                NeighbourCount = length(NeighbourArr)
                if NeighbourCount != 0
                    for j in NeighbourArr
                        push!(DegreesArr, length(Net[j]))
                    end
                end
            end
            MinimumDegree = minimum(DegreesArr)
            maximumDegree = maximum(DegreesArr)
            display(histogram(DegreesArr, dpi = 300, title = "Degree Distribution with lamb
            , background = :black, legend = false,
             xlabel = "Degree", ylabel = "Frequency", color = :orange))
        end
```

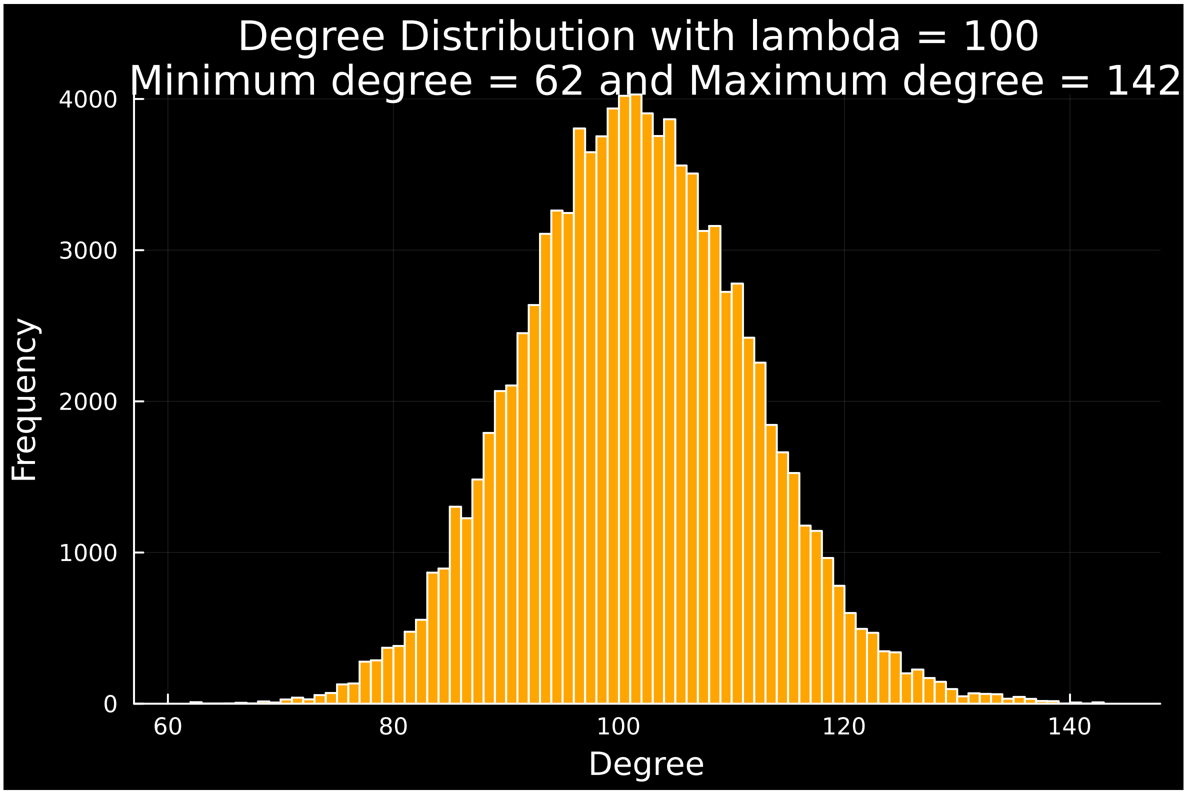



Degree Distribution with lambda = 3
Minimum degree = 1 and Maximum degree = 12
Frequency
0
100
200
300
400
500
600
2.5
5.0
7.5
10.0
12.5
Degree


Degree Distribution with lambda = 5
Minimum degree = 1 and Maximum degree = 15
Frequency
0
200
400
600
800
3
6
9
12
15
Degree


Degree Distribution with lambda = 10
Minimum degree = 2 and Maximum degree = 26
Frequency
0
200
400
600
800
1000
1200
5
10
15
20
25
Degree

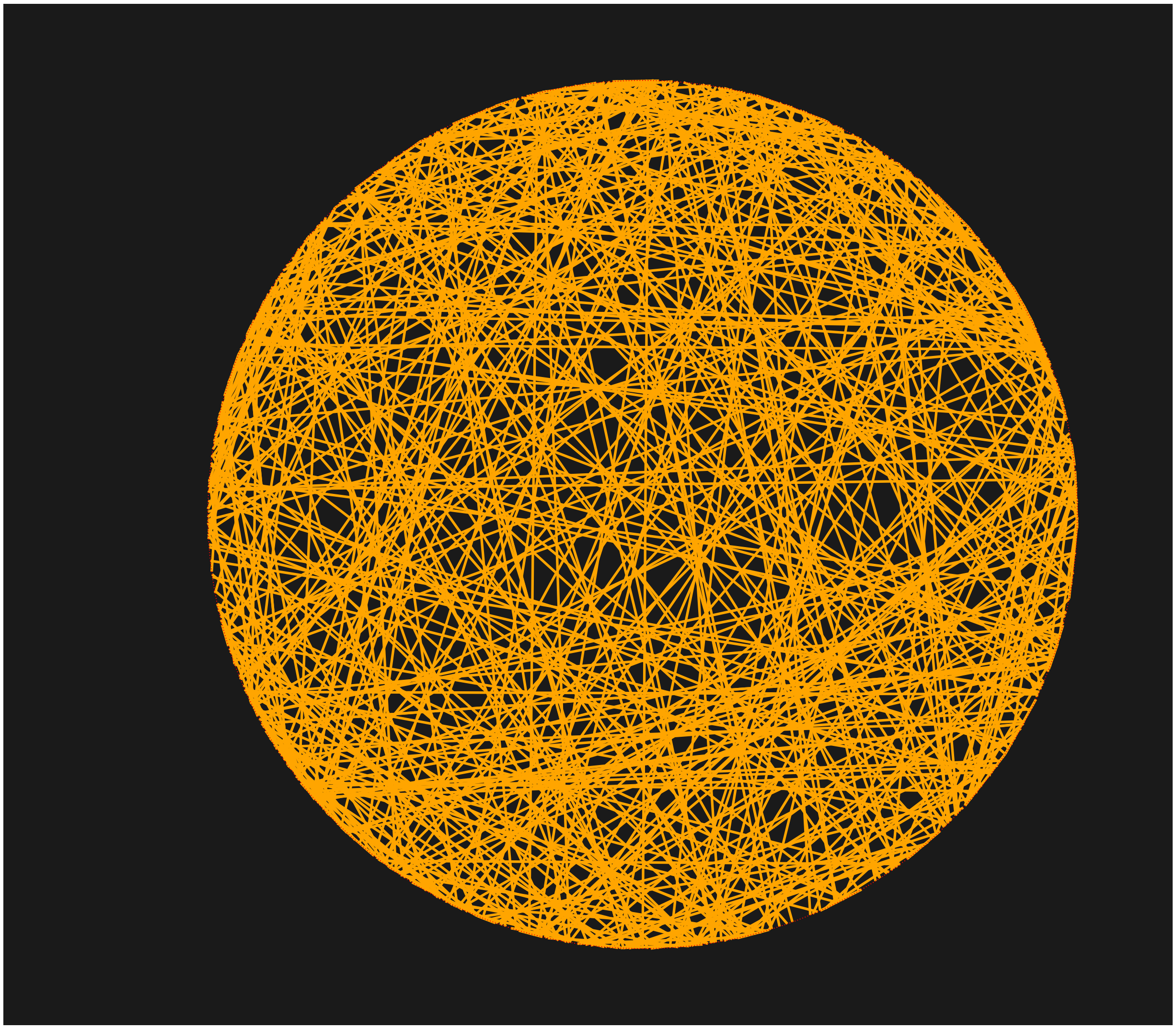

Degree Distribution with lambda = 50
Minimum degree = 23 and Maximum degree = 79
Frequency
0
500
1000
1500
2000
2500
20
30
40
50
60
70
80
Degree



Using Algorithm Type Two

In [ ]:
```
#Initial parameters
N = 1000
lambda = 1
Net = RandomNetwork.GenerateNetwork(lambda, N, 2)
g = Graph(N, Net)

@drawsvg begin
    background("grey10")
    sethue("orange")
    drawgraph(g, layout=shell, vertexshapesizes = 0.1, vertexstrokeweights = 0.001,
    end 500 400
```



In [ ]:

```
Net
```

```
1000-element Vector{Vector{Int64}}:
 [402]
 []
 []
 []
 []
 []
 []
 [80]
 []
 []
 ⋮
 []
 [719, 990]
 [622, 66]
 [186]
 [213, 672]
 []
 [416, 12]
 [319]
 []
```

In [ ]:

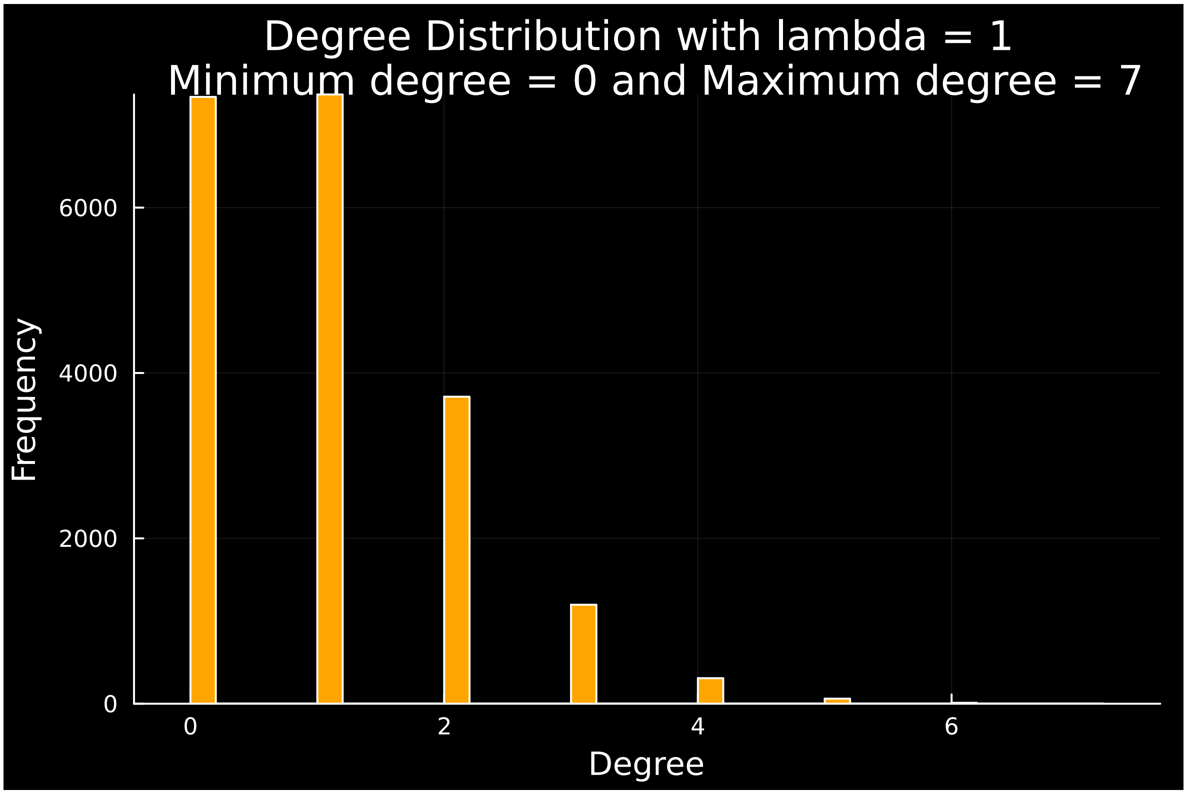
```
global sum = 0
for i in 1:N
    global sum += length(Net[i])
end
ActualLambda = sum / N
println("Actual value for lambda = $ActualLambda")
```

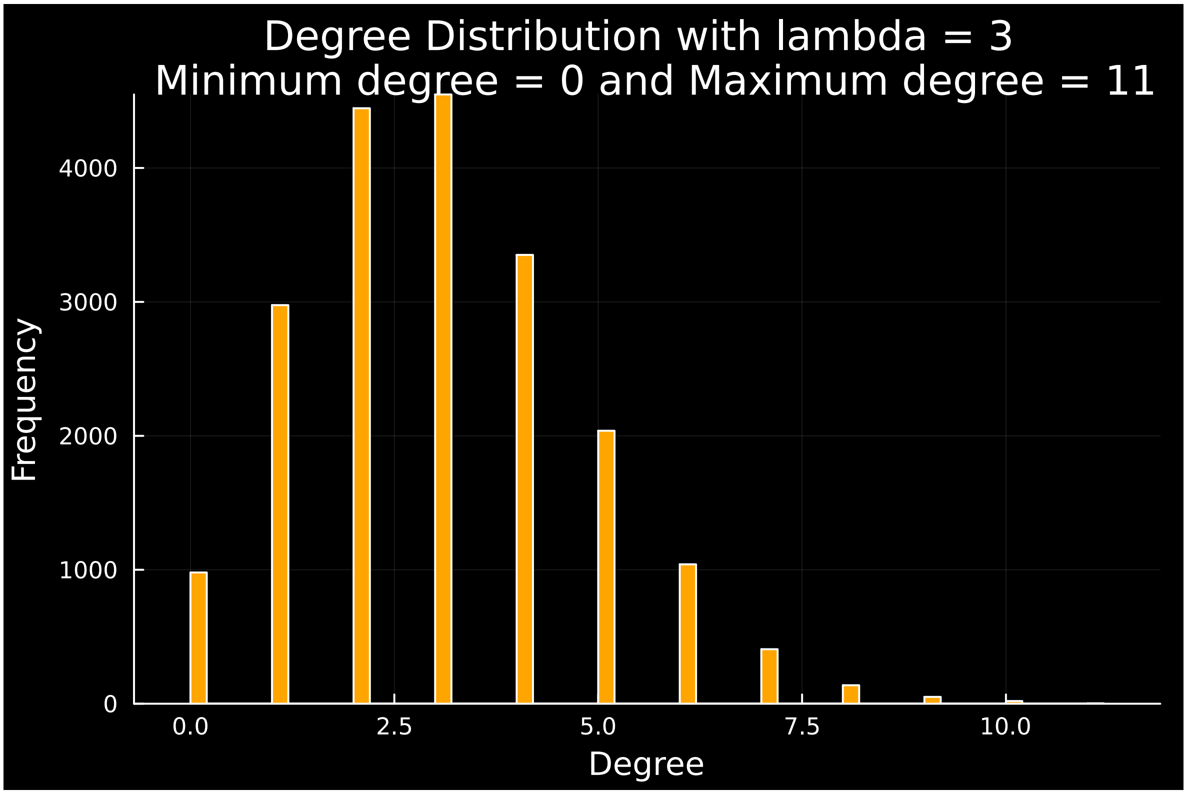
```
Actual value for lambda = 1.0
```

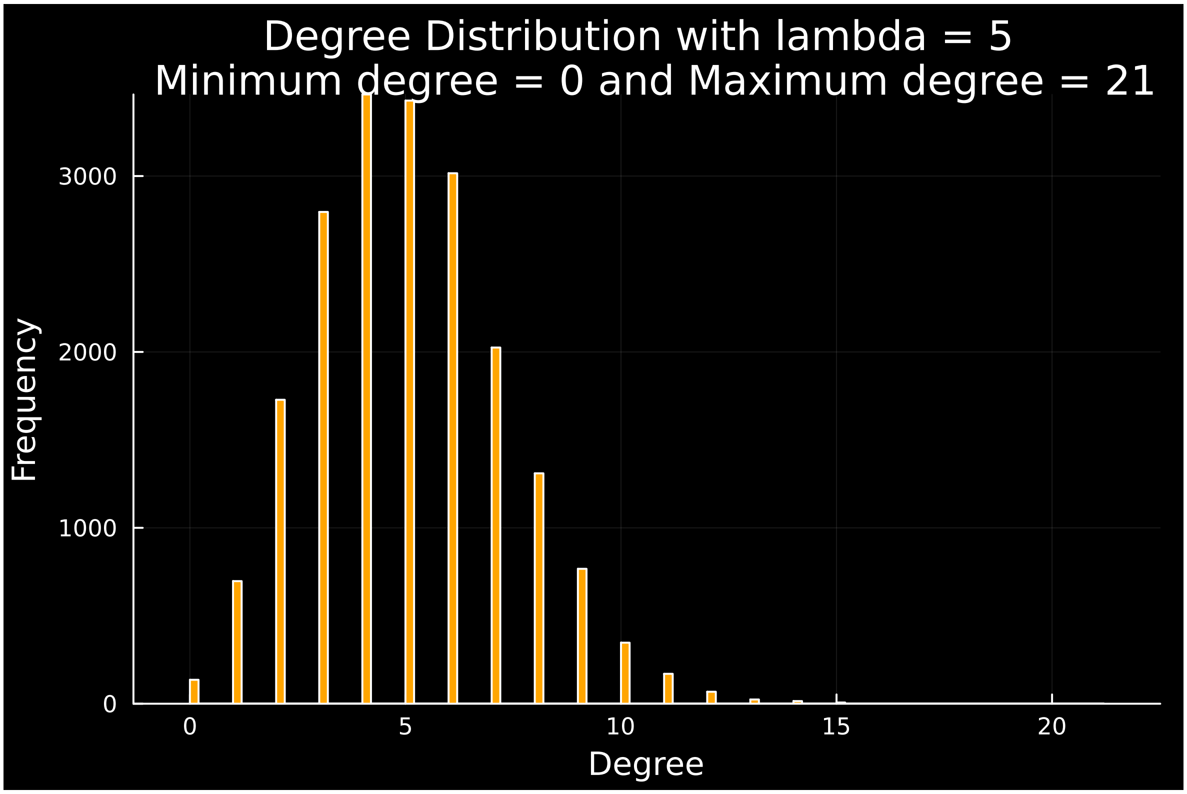
Degree Distribution

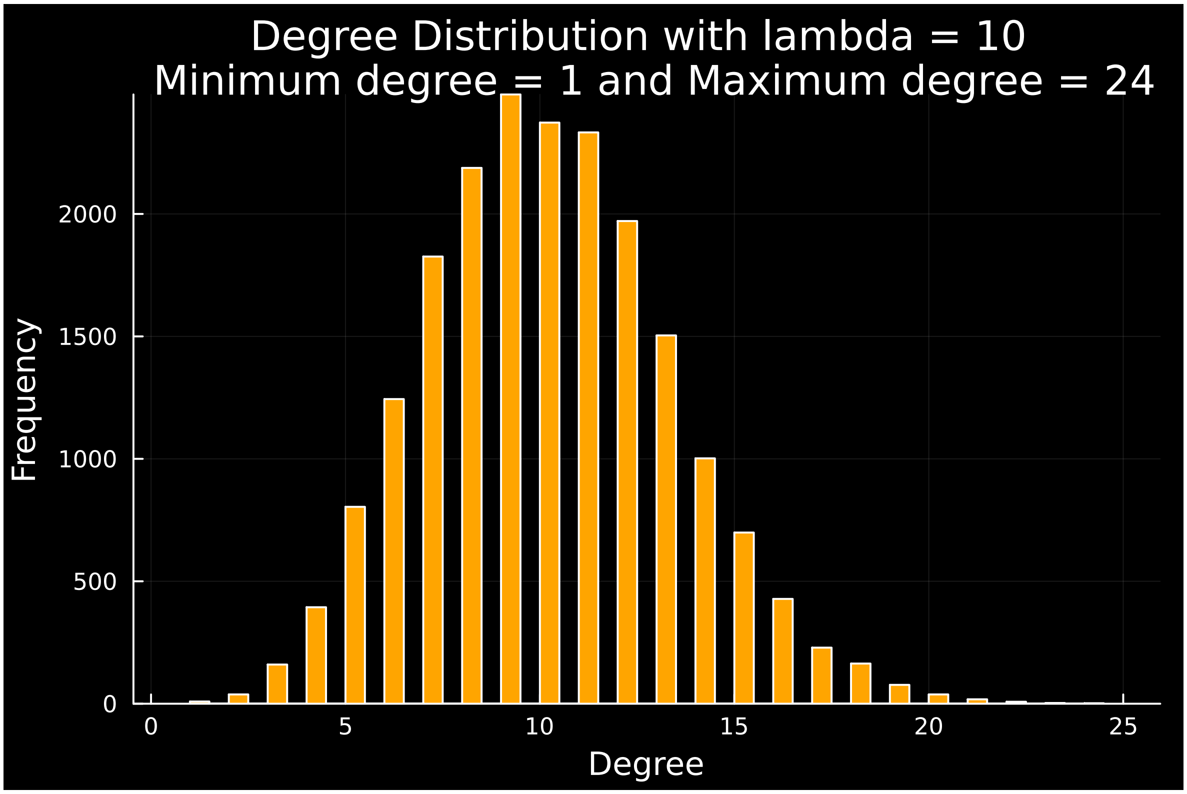
In [ ]:

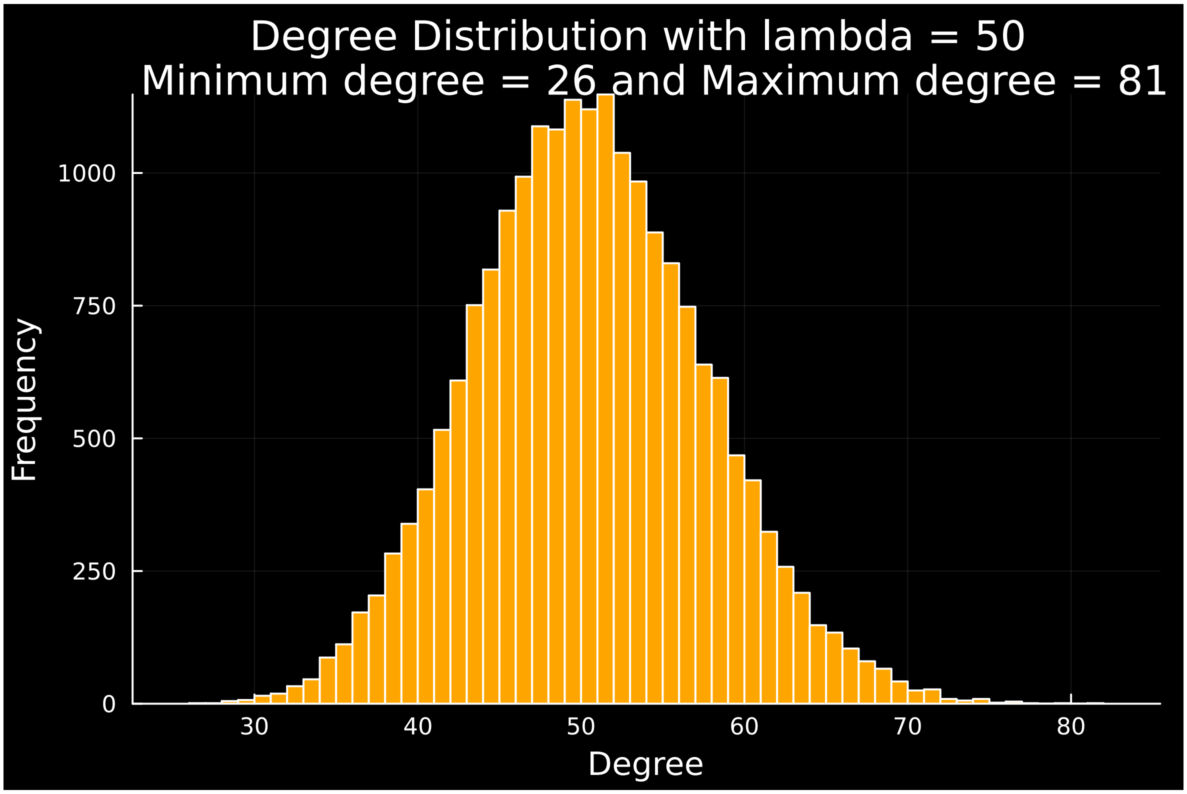
```
#Network Generation
N = 20000
lambdaArr = [1, 3, 5, 10, 50, 100] #Array of all lambdas we want to test
for lambda in lambdaArr
    DegreesArr = Int64[]
    Net = RandomNetwork.GenerateNetwork(lambda, N, 2)
    for j in 1:N
        push!(DegreesArr, length(Net[j]))
    end
    MinimumDegree = minimum(DegreesArr)
    maximumDegree = maximum(DegreesArr)
    display(histogram(DegreesArr, dpi = 300, title = "Degree Distribution with lamb
    , background = :black, legend = false,
     xlabel = "Degree", ylabel = "Frequency", color = :orange))
end
```

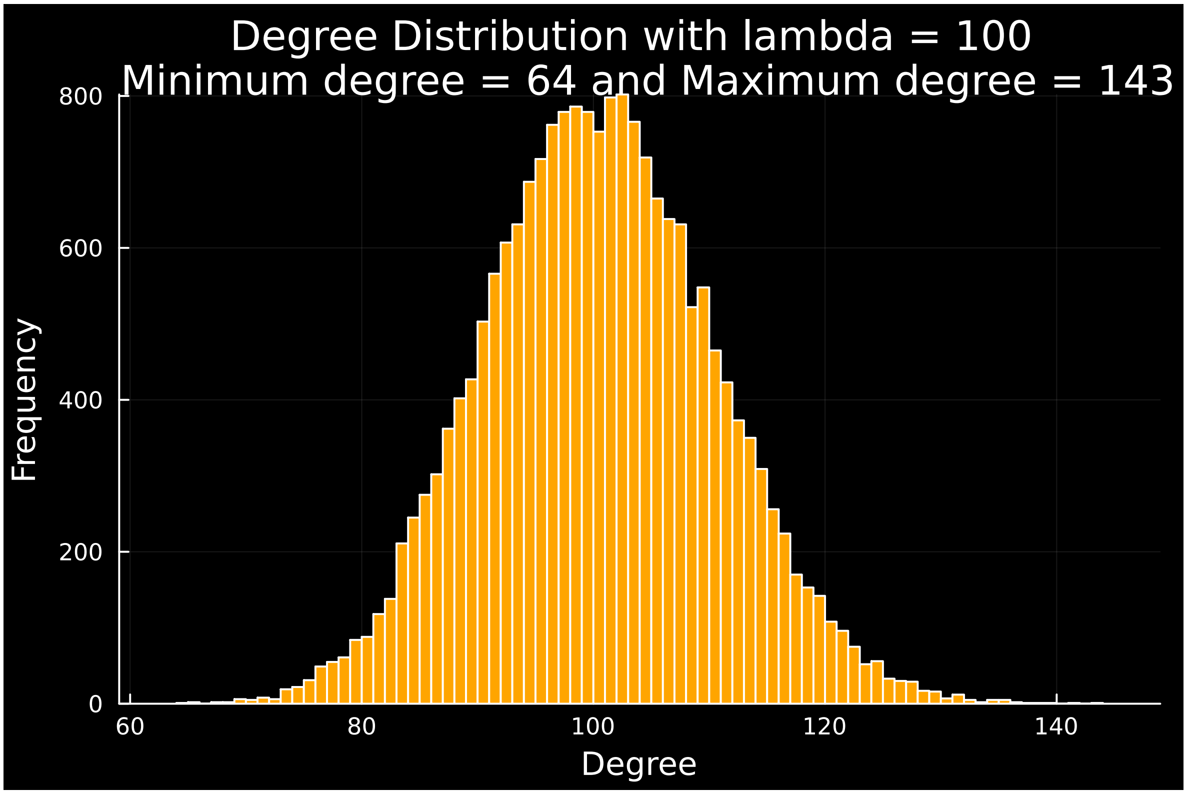

Degree Distribution with lambda = 1
Minimum degree = 0 and Maximum degree = 7
Frequency
0
2000
4000
6000
0
2
4
6
Degree


Degree Distribution with lambda = 3
Minimum degree = 0 and Maximum degree = 11
Frequency
0
1000
2000
3000
4000
0.0
2.5
5.0
7.5
10.0
Degree


Degree Distribution with lambda = 5
Minimum degree = 0 and Maximum degree = 21
Frequency
3000
2000
1000
0
0
5
10
15
20
Degree


Degree Distribution with lambda = 10
Minimum degree = 1 and Maximum degree = 24
Frequency
2000
1500
1000
500
0
0
5
10
15
20
25
Degree



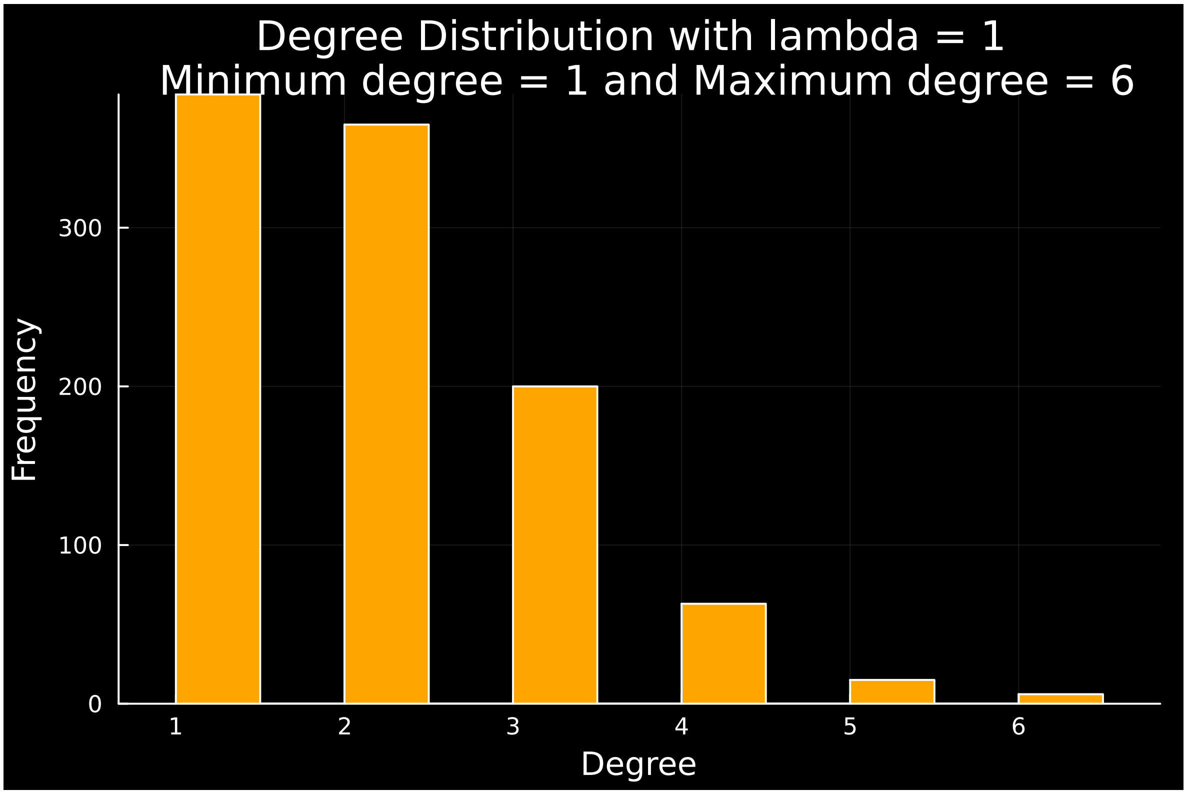


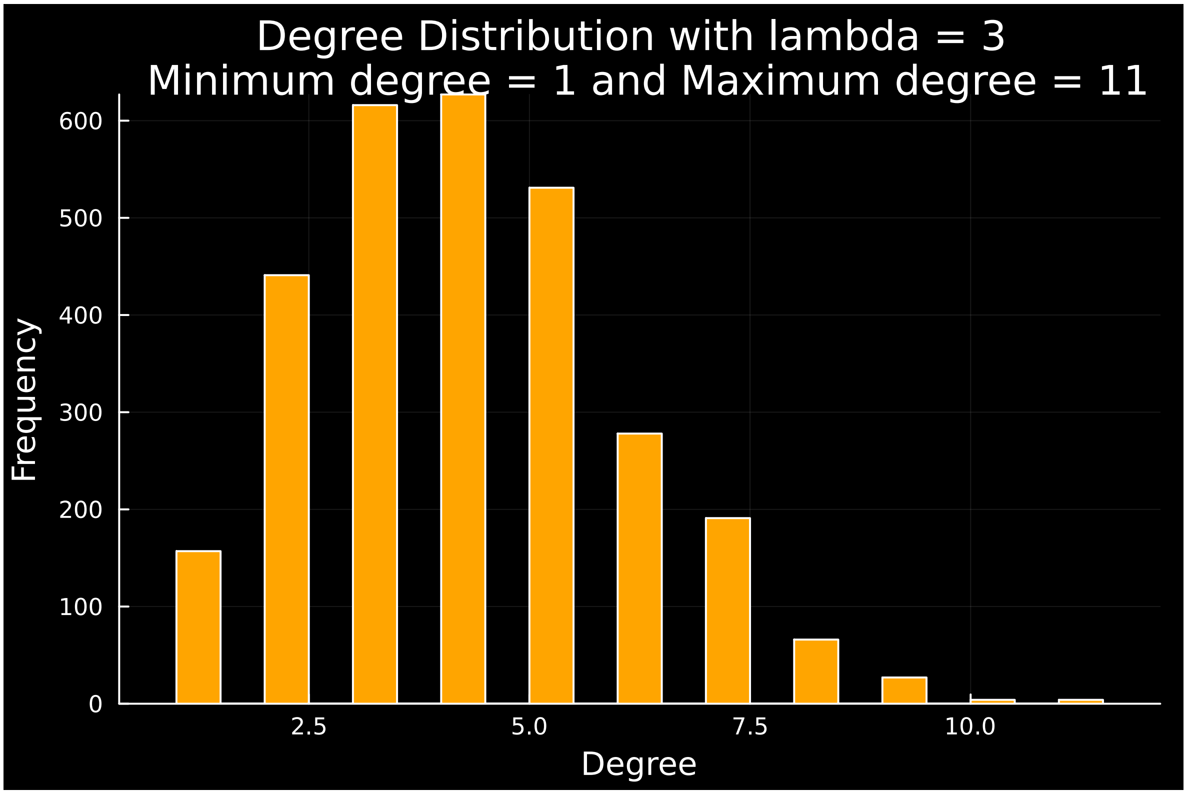
Neighbour Degree Distribution

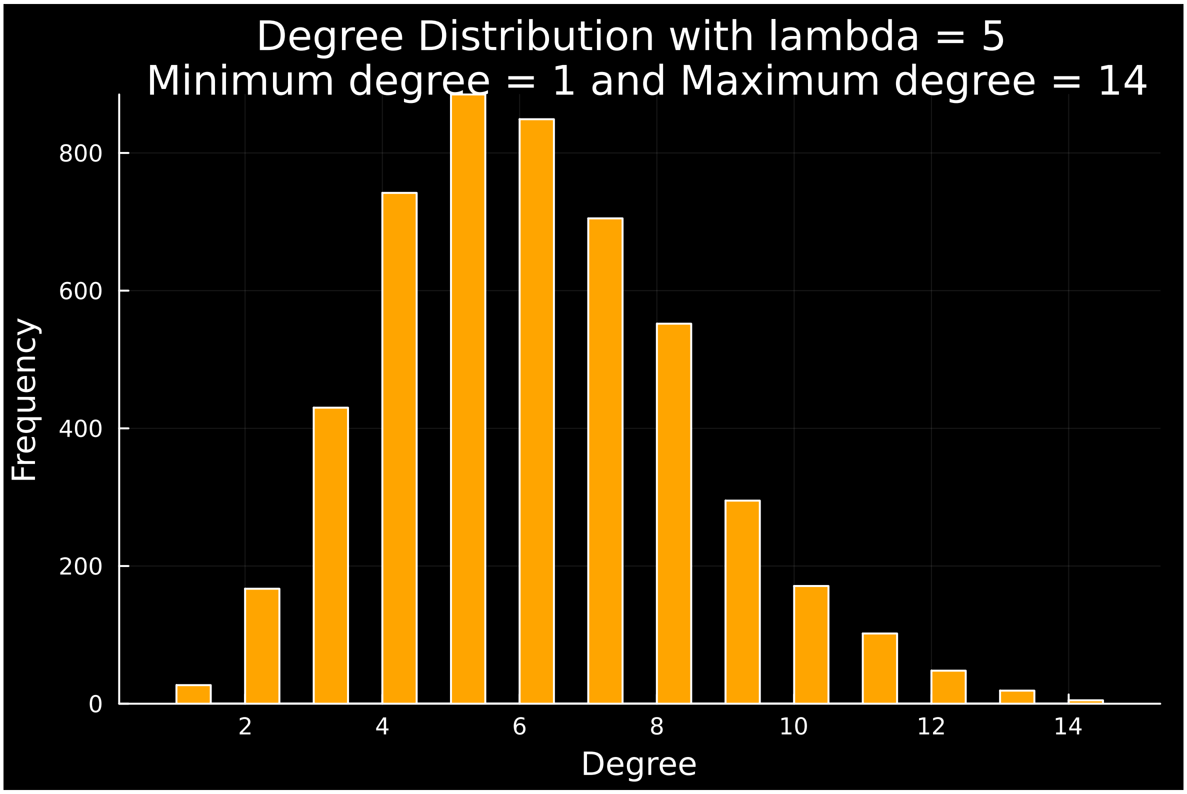
In [ ]:

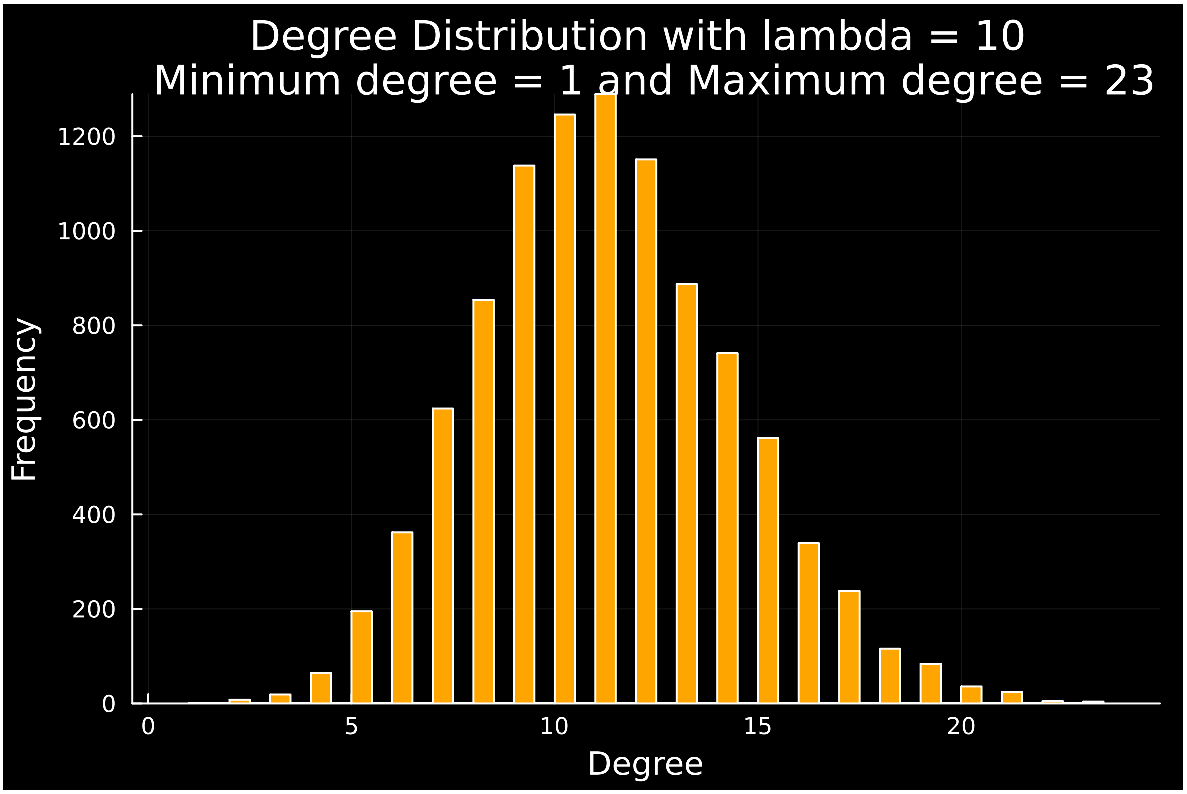
```
N = 20000
lambdaArr = [1, 3, 5, 10, 50, 100] #Array of all lambdas we want to test
SampleCount = 1000 #Number of randomly chosen nodes
for lambda in lambdaArr
```

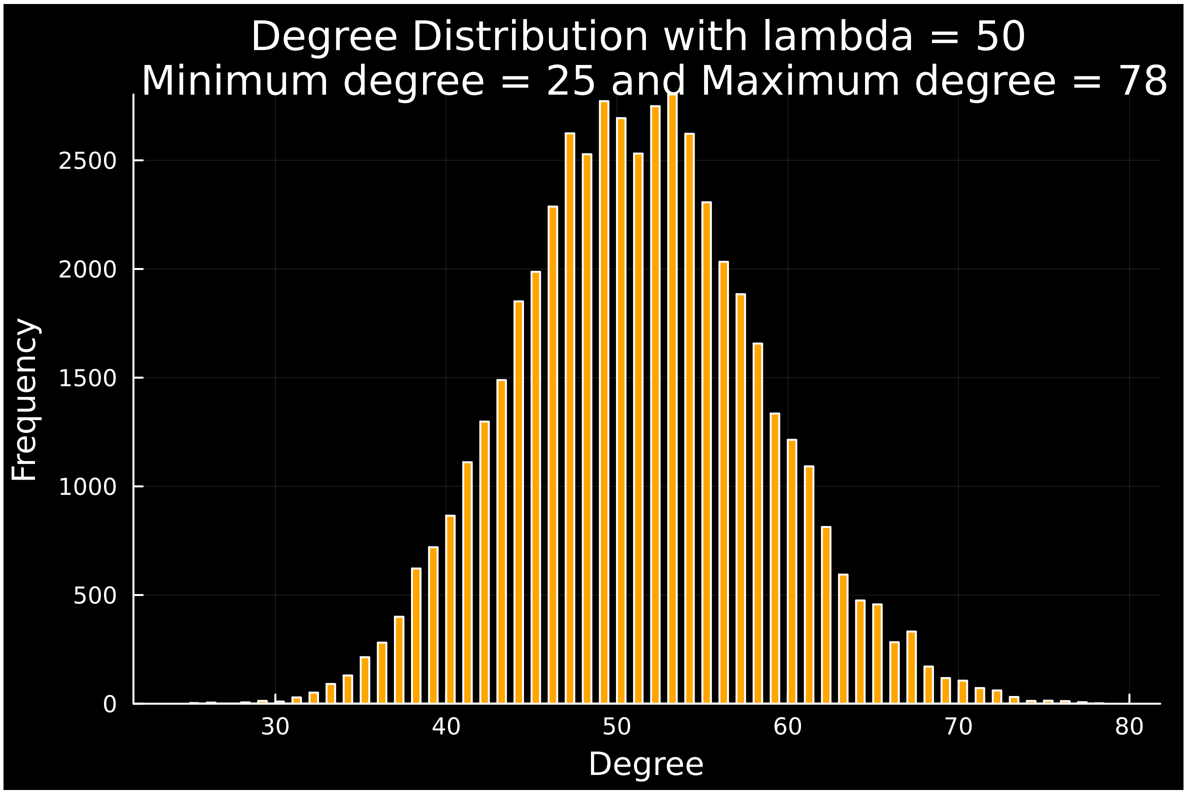
```
    DegreesArr = Int64[]
    Net = RandomNetwork.GenerateNetwork(lambda, N, 2)
    for j in 1:SampleCount
        SelectedNode = rand(1:N)
        NeighbourArr = Net[SelectedNode]
        NeighbourCount = length(NeighbourArr)
        if NeighbourCount != 0
            for j in NeighbourArr
                push!(DegreesArr, length(Net[j]))
            end
        end
    end
    MinimumDegree = minimum(DegreesArr)
    maximumDegree = maximum(DegreesArr)
    display(histogram(DegreesArr, dpi = 300, title = "Degree Distribution with lamb
    , background = :black, legend = false,
     xlabel = "Degree", ylabel = "Frequency", color = :orange))
end
```

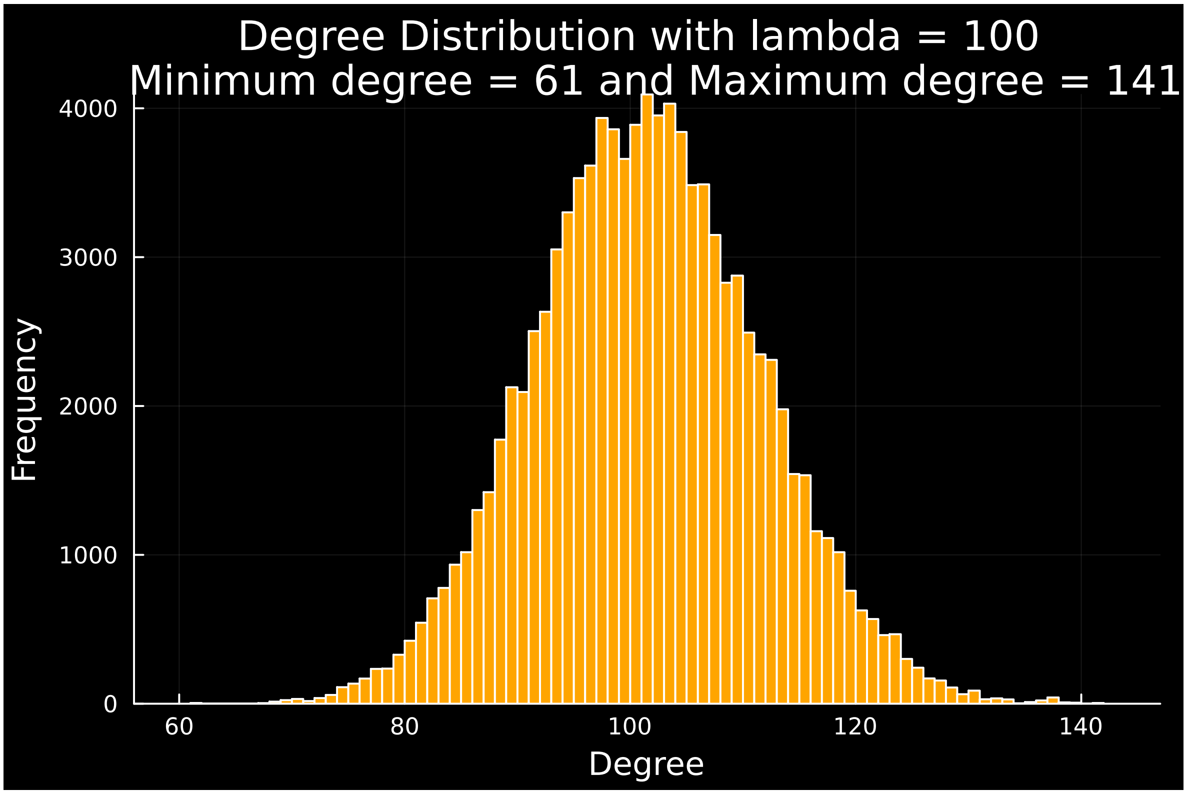



Degree Distribution with lambda = 3
Minimum degree = 1 and Maximum degree = 11
Frequency
0
100
200
300
400
500
600
2.5
5.0
7.5
10.0
Degree


Degree Distribution with lambda = 5
Minimum degree = 1 and Maximum degree = 14
Frequency
0
200
400
600
800
2
4
6
8
10
12
14
Degree


Degree Distribution with lambda = 10
Minimum degree = 1 and Maximum degree = 23
Frequency
Degree
0
200
400
600
800
1000
1200
0
5
10
15
20


Degree Distribution with lambda = 50
Minimum degree = 25 and Maximum degree = 78
Frequency
Degree
0
500
1000
1500
2000
2500
30
40
50
60
70
80


Degree Distribution with lambda = 100
Minimum degree = 61 and Maximum degree = 141
Frequency
0
1000
2000
3000
4000
Degree
60
80
100
120
140