Parcial1- TDA2

April 30, 2022

[53]:
```
import os, sys

from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call
drive.mount("/content/gdrive", force_remount=True).
```

[54]:
```
#load the module in  a  new notebook
import sys
sys.path.append('/content/gdrive/My Drive/social-networks-utils-main')
```

[55]:
```
import sys
sys.path.insert(0,'/content/drive/My Drive/social-networks-utils-main')

from metricas import *
from homofilia import *
from modelos import*
```

Comenzamos por hacer un gráfico del grafo utilizando la librería networkx

[56]:
```
import pandas as pd
```

[57]:
```
import networkx as nx
```

[58]:
```
df = pd.read_csv('World.csv')
G = nx.from_pandas_edgelist(df, 'Origen', 'Destino')
G = nx.to_undirected(G)
```

0.0.1 1) Determinar:

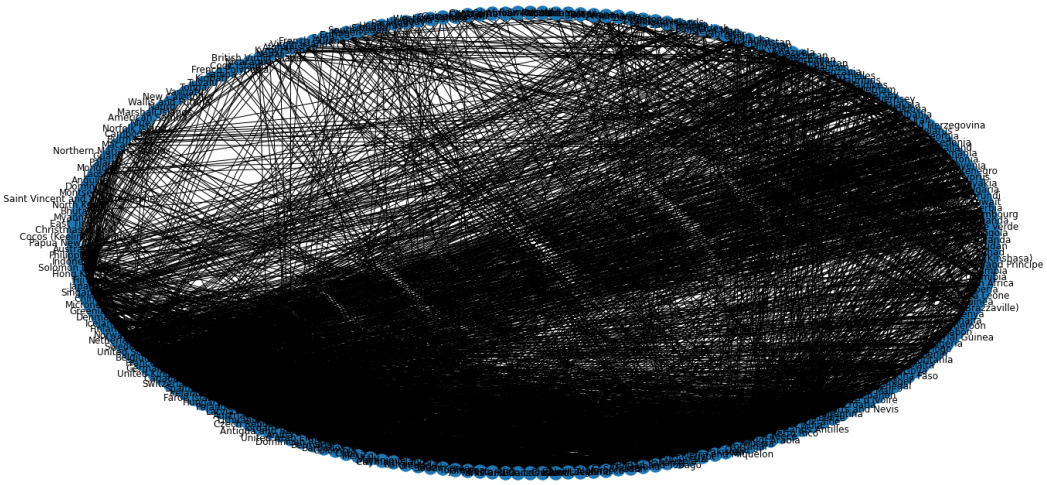
0.0.2 a. El diámetro de la red.

0.0.3 b. El grado promedio de la red.

0.0.4 c. El coeficiente de clustering promedio de la red.

0.0.5 Realizo un gráfico del grafo

[59]:
```
from matplotlib.pyplot import figure
figure(figsize=(20, 10))
nx.draw_shell(G, with_labels=True)
```



[ ]:
```
# El diámetro se puede calcular con la función diameter de Networkx

nx.diameter(G, e=None, usebounds=False)
```

[ ]: 5

[ ]:
```
# El grado promedio se puede calcular con la siguiente función  de Networkx
grado_promedio(G)
```

[ ]: 24.90829694323144

[ ]:
```
# Con la función clustering podemos obtener el valor del coeficiente de␣
↪clustering promedio de la red, quedandonos con el segundo valor de la tupla␣
↪que devuelve la función
```

```
c = clustering(G)
coeficiente_clustering_promedio = c[1]
coeficiente_clustering_promedio
```

[ ]: 0.6601565365859736

0.0.6 2) Para el análisis de Homofilia por algún tipo, voy a considerar la característica de los hemisferios, específicamente el Occidente y el Oriente, dejando de lado los demás. En especial, el interés particular en éstos se debe a que además de la connotación geográfica, podemos además sumar que entre estos dos existe una diferencia cultural. Tenemos por un lado la cultura Occidental y la cultura de Oriente.

0.0.7 Siguiendo con la explicación se tratará a continuación de medir la homofilia de esta característica detallada anteriormente. Tratando de responder la preguntas que sigue.

0.0.8 ¿Existe una tendencia de los paises a establecer una mayor cantidad conexiones/enlaces entre otros que sean de su mismo hemisferio/cultura?

0.0.9

A continuación realizo un manejo del dataframe creado anteriormente correspondiente a los datos proporcionados para lograr añadir el atributo que se va a estudiar.

[ ]:
```
origen = df[['Origen']]
origen.columns = ['pais']
origen
```

[ ]:

	pais
0	Papua New Guinea
1	Papua New Guinea
2	Papua New Guinea
3	Papua New Guinea
4	Papua New Guinea
…	…
2847	Lithuania
2848	Armenia
2849	Eritrea
2850	Yemen
2851	Solomon Islands

[2852 rows x 1 columns]

[ ]:
```
destino = df[['Destino']]
destino.columns = ['pais']
```

```
destino
```

[ ]:
```
                 pais
0           Australia
1         Philippines
2           Indonesia
3     Solomon Islands
4           Hong Kong
…                 …
2847          Georgia
2848          Georgia
2849            Yemen
2850         Djibouti
2851            Nauru

[2852 rows x 1 columns]
```

[ ]:
```
paises = origen.merge(destino,  how='outer').drop_duplicates()
paises
```

[ ]:
```
                         pais
0             Papua New Guinea
10                   Greenland
12                     Iceland
33                      Canada
104                    Algeria
…                          …
48583                 Anguilla
48587               Montserrat
48588              North Korea
48592                  Myanmar
48593  Cocos (Keeling) Islands

[229 rows x 1 columns]
```

[ ]:
```
pip install pycountry-convert
```

[ ]:
```
import pycountry_convert as pc
```

[ ]:
```
def convert(x):

    #convert country name to country code
    try:
        cn_code = pc.country_name_to_country_alpha2(x, cn_name_format =␣
 ↪"default")
    except:
        return "Unk"
```

```
    #convert cn_code to continent code
    try:

        conti_code = pc.country_alpha2_to_continent_code(cn_code)
    except:
        return "Unk"
    return conti_code
```

[ ]:
```
paises['continente'] = paises['pais'].map(lambda x: convert(x))
paises
```

[ ]:
```
                          pais continente
0              Papua New Guinea         OC
10                    Greenland         NA
12                      Iceland         EU
33                       Canada         NA
104                     Algeria         AF
…                             …        …
48583                  Anguilla         NA
48587                Montserrat         NA
48588               North Korea         AS
48592                   Myanmar         AS
48593  Cocos (Keeling) Islands         AS

[229 rows x 2 columns]
```

[ ]:
```
# Determino diccionario de continente
cont_hemis = {'OC': 'Oriental',
              'NA': 'Occidental',
              'EU': 'Oriental',
              'AF': 'Occidental',
              'SA': 'Occidental',
              'AS': 'Oriental',
              'Unk': 'Unk'}
cont_hemis
```

[ ]:
```
{'AF': 'Occidental',
 'AS': 'Oriental',
 'EU': 'Oriental',
 'NA': 'Occidental',
 'OC': 'Oriental',
 'SA': 'Occidental',
 'Unk': 'Unk'}
```

[ ]:
```
paises['p_c'] = (paises['pais'] + paises['continente'].map(lambda x: '_{}'.
 ↪format(x))).map(lambda x: tuple(x.split('_')))
paises
```

[ ]:
```
                         pais continente                          p_c
0             Papua New Guinea         OC         (Papua New Guinea, OC)
10                   Greenland         NA                (Greenland, NA)
12                     Iceland         EU                  (Iceland, EU)
33                      Canada         NA                   (Canada, NA)
104                    Algeria         AF                  (Algeria, AF)
…                          …        …                             …
48583                  Anguilla         NA                (Anguilla, NA)
48587                Montserrat         NA              (Montserrat, NA)
48588               North Korea         AS             (North Korea, AS)
48592                   Myanmar         AS                 (Myanmar, AS)
48593  Cocos (Keeling) Islands         AS  (Cocos (Keeling) Islands, AS)

[229 rows x 3 columns]
```

[ ]:
```
def to_hemis(x):
  if (x[0] == 'Iceland' or x[0] == 'Portugal' or x[0] == 'Spain' or x[0] ==␣
↪'United Kingdom' or x[0] == 'Irlanda' ):
    return 'Occidental'
  else:
    return cont_hemis[x[1]]
```

[ ]:
```
paises['hemisferio'] = paises['p_c'].map(lambda x: to_hemis(x))
paises
```

[ ]:
```
                         pais continente                          p_c  \
0             Papua New Guinea         OC         (Papua New Guinea, OC)
10                   Greenland         NA                (Greenland, NA)
12                     Iceland         EU                  (Iceland, EU)
33                      Canada         NA                   (Canada, NA)
104                    Algeria         AF                  (Algeria, AF)
…                          …        …                             …
48583                  Anguilla         NA                (Anguilla, NA)
48587                Montserrat         NA              (Montserrat, NA)
48588               North Korea         AS             (North Korea, AS)
48592                   Myanmar         AS                 (Myanmar, AS)
48593  Cocos (Keeling) Islands         AS  (Cocos (Keeling) Islands, AS)

       hemisferio
0        Oriental
10     Occidental
12     Occidental
33     Occidental
104    Occidental
…             …
48583  Occidental
48587  Occidental
```

```
48588    Oriental
48592    Oriental
48593    Oriental

[229 rows x 4 columns]
```

[ ]:
```
paises_hemisferios = paises[['pais', 'hemisferio']].set_index('pais').
↪to_dict()['hemisferio']
paises_hemisferios
```

[ ]:
```
{'Afghanistan': 'Oriental',
 'Albania': 'Oriental',
 'Algeria': 'Occidental',
 'American Samoa': 'Oriental',
 'Angola': 'Occidental',
 'Anguilla': 'Occidental',
 'Antigua and Barbuda': 'Occidental',
 'Argentina': 'Occidental',
 'Armenia': 'Oriental',
 'Aruba': 'Occidental',
 'Australia': 'Oriental',
 'Austria': 'Oriental',
 'Azerbaijan': 'Oriental',
 'Bahamas': 'Occidental',
 'Bahrain': 'Oriental',
 'Bangladesh': 'Oriental',
 'Barbados': 'Occidental',
 'Belarus': 'Oriental',
 'Belgium': 'Oriental',
 'Belize': 'Occidental',
 'Benin': 'Occidental',
 'Bermuda': 'Occidental',
 'Bhutan': 'Oriental',
 'Bolivia': 'Occidental',
 'Bosnia and Herzegovina': 'Oriental',
 'Botswana': 'Occidental',
 'Brazil': 'Occidental',
 'British Virgin Islands': 'Occidental',
 'Brunei': 'Oriental',
 'Bulgaria': 'Oriental',
 'Burkina Faso': 'Occidental',
 'Burma': 'Unk',
 'Burundi': 'Occidental',
 'Cambodia': 'Oriental',
 'Cameroon': 'Occidental',
 'Canada': 'Occidental',
 'Cape Verde': 'Occidental',
```

```
'Cayman Islands': 'Occidental',
'Central African Republic': 'Occidental',
'Chad': 'Occidental',
'Chile': 'Occidental',
'China': 'Oriental',
'Christmas Island': 'Oriental',
'Cocos (Keeling) Islands': 'Oriental',
'Colombia': 'Occidental',
'Comoros': 'Occidental',
'Congo (Brazzaville)': 'Unk',
'Congo (Kinshasa)': 'Unk',
'Cook Islands': 'Oriental',
'Costa Rica': 'Occidental',
"Cote d'Ivoire": 'Unk',
'Croatia': 'Oriental',
'Cuba': 'Occidental',
'Cyprus': 'Oriental',
'Czech Republic': 'Oriental',
'Denmark': 'Oriental',
'Djibouti': 'Occidental',
'Dominica': 'Occidental',
'Dominican Republic': 'Occidental',
'East Timor': 'Unk',
'Ecuador': 'Occidental',
'Egypt': 'Occidental',
'El Salvador': 'Occidental',
'Equatorial Guinea': 'Occidental',
'Eritrea': 'Occidental',
'Estonia': 'Oriental',
'Ethiopia': 'Occidental',
'Falkland Islands': 'Occidental',
'Faroe Islands': 'Oriental',
'Fiji': 'Oriental',
'Finland': 'Oriental',
'France': 'Oriental',
'French Guiana': 'Occidental',
'French Polynesia': 'Oriental',
'Gabon': 'Occidental',
'Gambia': 'Occidental',
'Georgia': 'Oriental',
'Germany': 'Oriental',
'Ghana': 'Occidental',
'Gibraltar': 'Oriental',
'Greece': 'Oriental',
'Greenland': 'Occidental',
'Grenada': 'Occidental',
'Guadeloupe': 'Occidental',
```

```
'Guam': 'Oriental',
'Guatemala': 'Occidental',
'Guernsey': 'Oriental',
'Guinea': 'Occidental',
'Guinea-Bissau': 'Occidental',
'Guyana': 'Occidental',
'Haiti': 'Occidental',
'Honduras': 'Occidental',
'Hong Kong': 'Oriental',
'Hungary': 'Oriental',
'Iceland': 'Occidental',
'India': 'Oriental',
'Indonesia': 'Oriental',
'Iran': 'Oriental',
'Iraq': 'Oriental',
'Ireland': 'Oriental',
'Isle of Man': 'Oriental',
'Israel': 'Oriental',
'Italy': 'Oriental',
'Jamaica': 'Occidental',
'Japan': 'Oriental',
'Jersey': 'Oriental',
'Jordan': 'Oriental',
'Kazakhstan': 'Oriental',
'Kenya': 'Occidental',
'Kiribati': 'Oriental',
'Kuwait': 'Oriental',
'Kyrgyzstan': 'Oriental',
'Laos': 'Oriental',
'Latvia': 'Oriental',
'Lebanon': 'Oriental',
'Lesotho': 'Occidental',
'Liberia': 'Occidental',
'Libya': 'Occidental',
'Lithuania': 'Oriental',
'Luxembourg': 'Oriental',
'Macau': 'Oriental',
'Macedonia': 'Oriental',
'Madagascar': 'Occidental',
'Malawi': 'Occidental',
'Malaysia': 'Oriental',
'Maldives': 'Oriental',
'Mali': 'Occidental',
'Malta': 'Oriental',
'Marshall Islands': 'Oriental',
'Martinique': 'Occidental',
'Mauritania': 'Occidental',
```

```
'Mauritius': 'Occidental',
'Mayotte': 'Occidental',
'Mexico': 'Occidental',
'Micronesia': 'Oriental',
'Moldova': 'Oriental',
'Mongolia': 'Oriental',
'Montenegro': 'Oriental',
'Montserrat': 'Occidental',
'Morocco': 'Occidental',
'Mozambique': 'Occidental',
'Myanmar': 'Oriental',
'Namibia': 'Occidental',
'Nauru': 'Oriental',
'Nepal': 'Oriental',
'Netherlands': 'Oriental',
'Netherlands Antilles': 'Unk',
'New Caledonia': 'Oriental',
'New Zealand': 'Oriental',
'Nicaragua': 'Occidental',
'Niger': 'Occidental',
'Nigeria': 'Occidental',
'Niue': 'Oriental',
'Norfolk Island': 'Oriental',
'North Korea': 'Oriental',
'Northern Mariana Islands': 'Oriental',
'Norway': 'Oriental',
'Oman': 'Oriental',
'Pakistan': 'Oriental',
'Palau': 'Oriental',
'Panama': 'Occidental',
'Papua New Guinea': 'Oriental',
'Paraguay': 'Occidental',
'Peru': 'Occidental',
'Philippines': 'Oriental',
'Poland': 'Oriental',
'Portugal': 'Occidental',
'Puerto Rico': 'Occidental',
'Qatar': 'Oriental',
'Reunion': 'Unk',
'Romania': 'Oriental',
'Russia': 'Oriental',
'Rwanda': 'Occidental',
'Saint Helena': 'Unk',
'Saint Kitts and Nevis': 'Occidental',
'Saint Lucia': 'Occidental',
'Saint Pierre and Miquelon': 'Occidental',
'Saint Vincent and the Grenadines': 'Occidental',
```

```
'Samoa': 'Oriental',
'Sao Tome and Principe': 'Occidental',
'Saudi Arabia': 'Oriental',
'Senegal': 'Occidental',
'Serbia': 'Oriental',
'Seychelles': 'Occidental',
'Sierra Leone': 'Occidental',
'Singapore': 'Oriental',
'Slovakia': 'Oriental',
'Slovenia': 'Oriental',
'Solomon Islands': 'Oriental',
'Somalia': 'Occidental',
'South Africa': 'Occidental',
'South Korea': 'Oriental',
'South Sudan': 'Occidental',
'Spain': 'Occidental',
'Sri Lanka': 'Oriental',
'Sudan': 'Occidental',
'Suriname': 'Occidental',
'Swaziland': 'Occidental',
'Sweden': 'Oriental',
'Switzerland': 'Oriental',
'Syria': 'Oriental',
'Taiwan': 'Oriental',
'Tajikistan': 'Oriental',
'Tanzania': 'Occidental',
'Thailand': 'Oriental',
'Togo': 'Occidental',
'Tonga': 'Oriental',
'Trinidad and Tobago': 'Occidental',
'Tunisia': 'Occidental',
'Turkey': 'Oriental',
'Turkmenistan': 'Oriental',
'Turks and Caicos Islands': 'Occidental',
'Tuvalu': 'Oriental',
'Uganda': 'Occidental',
'Ukraine': 'Oriental',
'United Arab Emirates': 'Oriental',
'United Kingdom': 'Occidental',
'United States': 'Occidental',
'Uruguay': 'Occidental',
'Uzbekistan': 'Oriental',
'Vanuatu': 'Oriental',
'Venezuela': 'Occidental',
'Vietnam': 'Oriental',
'Virgin Islands': 'Unk',
'Wallis and Futuna': 'Oriental',
```

```
 'Western Sahara': 'Unk',
 'Yemen': 'Oriental',
 'Zambia': 'Occidental',
 'Zimbabwe': 'Occidental'}
```

[ ]:
```
def mapper(x):
  return paises_hemisferios[x]
```

[ ]:
```
props = proporcion_por_tipo(G, mapper)
props
```

[ ]:
```
{'Occidental': 0.4585152838427948,
 'Oriental': 0.4978165938864629,
 'Unk': 0.043668122270742356}
```

[ ]:
```
# Proporción de tipo Occidental con respecto a la cantidad de nodos totales
p  = props['Occidental']
# Proporción de tipo Oriental con respecto a la cantidad de nodos totales
q = props['Oriental']
```

[ ]:
```
# Probabilidad de encontrar un extremo Occidental y otro Oriental
# Calculamos el ideal sin homofilia
2*p*q
```

[ ]:
```
0.4565130336950096
```

[ ]:
```
### Proporcion cruzan campos
proporcion_cruzan_campo(G, mapper)
```

[ ]:
```
0.27419354838709675
```

[ ]:
```
### Proporcion que cursan campo de tipo Occidental
proporcion_cruzan_campo_de_tipo(G, 'Occidental', mapper)
```

[ ]:
```
0.3593073593073593
```

[ ]:
```
### Proporción que cruzan campo de tipo Oriental
proporcion_cruzan_campo_de_tipo(G, 'Oriental', mapper)
```

[ ]:
```
0.2
```

[ ]:
```
# Proporción de no homofilia
proporcion_cruzan_campo(G, mapper)/(2*p*q)
```

[ ]:
```
0.600625892688711
```

0.0.10 Conclusiones:

Si la fracción de aristas que cruzan géneros es significativamente menor (o mayor) a $2qp$, entonces hay evidencia de Homofilia.

Para la característica propuesta se obtuvo que tenemos una proporción de cruces de campos mucho menor al valor $2pq$. Esto significa una evidencia que soporta la hipótesis de homofilia expuesta.

Es decir que se evidencia homofilia tanta para un lado (occidental) como para el otro (oriental).

Esto nos demuestra que existe evidencia para soportar nuestra hipótesis: Los países occidentales tienden a viajar más hacia países occidentales. Asi como los paises orientales tienden a hacerlo a los países orientales.

0.0.11 Por último, se obtuvo un porcentaje de "sin-homofilia" de 0.060.

0.0.12 3) Determinar los puentes (globales o locales) en dicha red.

[ ]:
```
# Los puentes globales de la red pueden calcularse facilmente con la función
↪bridges de Networkx.
# A continuación una lista de los puentes globales
list(nx.bridges(G))
```

[ ]:
```
[('Fiji', 'Tuvalu'),
 ('United States', 'American Samoa'),
 ('United Kingdom', 'Saint Helena'),
 ('Canada', 'Saint Pierre and Miquelon'),
 ('Antigua and Barbuda', 'Montserrat'),
 ('New Zealand', 'Niue'),
 ('South Africa', 'Lesotho'),
 ('South Africa', 'Swaziland'),
 ('Burma', 'Myanmar')]
```

[ ]:
```
# Los puentes locales de la red pueden calcularse facilmente con la función
↪local_bridges de Networkx.
# A continuación una lista de los puentes locales
list(nx.local_bridges(G))
```

[ ]:
```
[('Papua New Guinea', 'Micronesia', 3),
 ('Fiji', 'Tuvalu', inf),
 ('Micronesia', 'Marshall Islands', 3),
 ('United States', 'American Samoa', inf),
```

```
('United Kingdom', 'Saint Helena', inf),
('Canada', 'Saint Pierre and Miquelon', inf),
('Antigua and Barbuda', 'Montserrat', inf),
('New Zealand', 'Niue', inf),
('South Africa', 'Lesotho', inf),
('South Africa', 'Swaziland', inf),
('Burma', 'Myanmar', inf)]
```

0.0.13 4) a. Determinar un tipo de centralidad que podría ser útil calcular para esta red, justificando.

b. Realizar una representación gráfica de dicha red, considerando la centralidad de los distintos países dada por la métrica del punto a (tamaño de los nodos proporcional a dicha métrica).

0.0.14 Elección de la centralidad

0.0.15 Centralidad por Grado:

Un tipo de centralidad interesante para considerar y la que voy a elegir es la Centralidad por Grado. En esta centralidad los nodos con más conexiones (leáse con mayor cantidad de aristas) son más influyentes e importantes en la red. Particularmente, en la red que se viene trabajando estamos considerandos países y destinos. Es muy intuitivo entonces pensar que si en cierto país (nodo) hay un número elevado de aristas pues entonces dicho país es de relevancia en la red, hay un interés para visitarlo. La razón de esta distinción, no se puede asegurar, podríamos conjeturar que se puede deber a razones de turismo, trabajo, atracciones, etc.

[ ]:
```
import networkx as nx
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

# for Notebook
%matplotlib inline
```

[62]:
```
def draw(G, pos, measures, measure_name):
    figure(figsize=(50, 20), dpi=300)
    #250
    values = list(measures.values())
    x = list(map(lambda x: x*1000,values))
    nodes = nx.draw_networkx_nodes(G, pos, node_size=x, cmap=plt.cm.plasma,
                                   node_color=list(measures.values()),
                                   nodelist=measures.keys())
    nodes.set_norm(mcolors.SymLogNorm(linthresh=0.01, linscale=1, base=10))
    labels = nx.draw_networkx_labels(G, pos)
    #edges = nx.draw_networkx_edges(G, pos)
```

```
    plt.title(measure_name)
    plt.colorbar(nodes)
    plt.axis('off')


    plt.show()
```

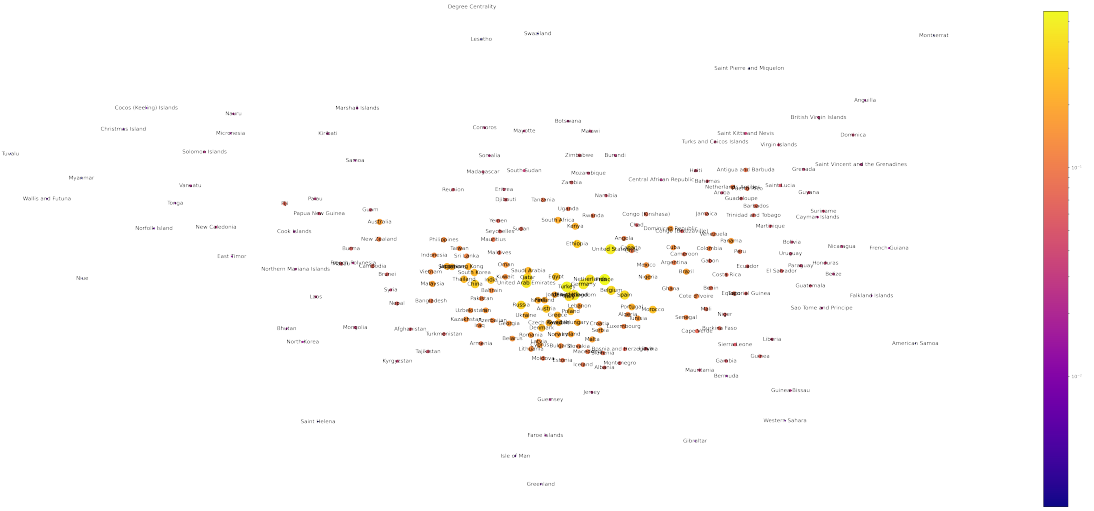
[63]:
```
pos = nx.spring_layout(G, seed=675)
```

0.0.16 En el siguiente gráfico se grafica como queda la representación de los países más centrales. Observese que los países con mayor centralidad tienen un tamaño de nodo más grande que el resto.

0.0.17 A su vez también se tiene el criterio de colores, donde el amarillo simboliza la mayor centralidad y el color violeta los menos centrales.

0.0.18 Aclaración: se habla de mayor o menor centralidad en relación a la centralidad elegida: Centralidad de grado.

[64]:
```
draw(G, pos, nx.degree_centrality(G), 'Degree Centrality')
```



0.0.19 Aquí se puede observar una lista ordenada con los países de mayor a menor centralidad según la centralidad de grado.

[65]:
```
central = nx.degree_centrality(G)

sorted(central.items(), key=lambda item: -item[1])
```

[65]:
```
[('France', 0.5614035087719298),
 ('Turkey', 0.5087719298245614),
 ('United Kingdom', 0.4956140350877193),
 ('United States', 0.4649122807017544),
 ('Germany', 0.4649122807017544),
 ('United Arab Emirates', 0.44298245614035087),
 ('Netherlands', 0.40789473684210525),
 ('Spain', 0.39035087719298245),
 ('Belgium', 0.381578947368421),
 ('Qatar', 0.381578947368421),
 ('Italy', 0.37280701754385964),
 ('Russia', 0.3421052631578947),
 ('China', 0.3289473684210526),
 ('Switzerland', 0.32456140350877194),
 ('Canada', 0.3157894736842105),
 ('Ethiopia', 0.3070175438596491),
 ('Austria', 0.293859649122807),
 ('Egypt', 0.2850877192982456),
 ('India', 0.2763157894736842),
 ('Morocco', 0.27192982456140347),
 ('Poland', 0.2587719298245614),
 ('Thailand', 0.2543859649122807),
 ('Denmark', 0.24561403508771928),
 ('South Korea', 0.24122807017543857),
 ('Saudi Arabia', 0.24122807017543857),
 ('Greece', 0.2324561403508772),
 ('Kenya', 0.22368421052631576),
 ('Hong Kong', 0.21929824561403508),
 ('Czech Republic', 0.21929824561403508),
 ('Portugal', 0.2149122807017544),
 ('Ukraine', 0.2149122807017544),
 ('Japan', 0.21052631578947367),
 ('Sweden', 0.21052631578947367),
 ('Hungary', 0.21052631578947367),
 ('Israel', 0.20614035087719296),
 ('South Africa', 0.20614035087719296),
 ('Singapore', 0.20175438596491227),
 ('Finland', 0.20175438596491227),
 ('Jordan', 0.18859649122807015),
 ('Norway', 0.18421052631578946),
```

```
('Ireland', 0.18421052631578946),
('Brazil', 0.17543859649122806),
('Malta', 0.17543859649122806),
('Nigeria', 0.17543859649122806),
('Malaysia', 0.17543859649122806),
('Australia', 0.16666666666666666),
('Panama', 0.16228070175438597),
('Lebanon', 0.16228070175438597),
('Serbia', 0.16228070175438597),
('Cyprus', 0.16228070175438597),
('Romania', 0.15789473684210525),
('Oman', 0.15789473684210525),
('Dominican Republic', 0.15350877192982454),
('Tunisia', 0.15350877192982454),
('Georgia', 0.15350877192982454),
('Iran', 0.15350877192982454),
('Latvia', 0.14912280701754385),
('Belarus', 0.14912280701754385),
('Mexico', 0.14473684210526316),
('Kuwait', 0.14473684210526316),
('Kazakhstan', 0.14473684210526316),
('Cuba', 0.14035087719298245),
('Lithuania', 0.13596491228070173),
('Croatia', 0.13596491228070173),
('Bahrain', 0.13596491228070173),
('Philippines', 0.13157894736842105),
('Bulgaria', 0.13157894736842105),
('Vietnam', 0.13157894736842105),
('Uzbekistan', 0.13157894736842105),
('Indonesia', 0.12719298245614036),
('Algeria', 0.12719298245614036),
('Taiwan', 0.12719298245614036),
('Ghana', 0.12719298245614036),
('Azerbaijan', 0.12719298245614036),
('Iraq', 0.12719298245614036),
('Senegal', 0.12280701754385964),
("Cote d'Ivoire", 0.11842105263157894),
('Sri Lanka', 0.11842105263157894),
('Colombia', 0.11403508771929824),
('Pakistan', 0.11403508771929824),
('Luxembourg', 0.11403508771929824),
('New Zealand', 0.10964912280701754),
('Angola', 0.10964912280701754),
('Slovakia', 0.10964912280701754),
('Netherlands Antilles', 0.10526315789473684),
('Argentina', 0.10526315789473684),
('Venezuela', 0.10526315789473684),
```

```
('Chile', 0.10087719298245613),
('Macedonia', 0.10087719298245613),
('Bangladesh', 0.10087719298245613),
('Iceland', 0.09649122807017543),
('Rwanda', 0.09649122807017543),
('Estonia', 0.09649122807017543),
('Moldova', 0.09649122807017543),
('Mauritius', 0.09649122807017543),
('Antigua and Barbuda', 0.09210526315789473),
('Peru', 0.09210526315789473),
('Tanzania', 0.09210526315789473),
('Armenia', 0.09210526315789473),
('Puerto Rico', 0.08771929824561403),
('Cameroon', 0.08771929824561403),
('Uganda', 0.08771929824561403),
('Mali', 0.08333333333333333),
('Benin', 0.08333333333333333),
('Togo', 0.08333333333333333),
('Congo (Kinshasa)', 0.08333333333333333),
('Slovenia', 0.08333333333333333),
('Maldives', 0.08333333333333333),
('Barbados', 0.07894736842105263),
('Jamaica', 0.07894736842105263),
('Fiji', 0.07456140350877193),
('Costa Rica', 0.07456140350877193),
('Burkina Faso', 0.07456140350877193),
('Gabon', 0.07456140350877193),
('Sudan', 0.07456140350877193),
('Cape Verde', 0.07456140350877193),
('Cambodia', 0.07456140350877193),
('Trinidad and Tobago', 0.07017543859649122),
('Bosnia and Herzegovina', 0.07017543859649122),
('Yemen', 0.07017543859649122),
('Albania', 0.06578947368421052),
('Seychelles', 0.06578947368421052),
('Brunei', 0.06578947368421052),
('Nepal', 0.06578947368421052),
('Burma', 0.06578947368421052),
('El Salvador', 0.06140350877192982),
('Niger', 0.06140350877192982),
('Equatorial Guinea', 0.06140350877192982),
('Congo (Brazzaville)', 0.06140350877192982),
('Guinea', 0.06140350877192982),
('Montenegro', 0.06140350877192982),
('Turkmenistan', 0.06140350877192982),
('Ecuador', 0.06140350877192982),
('Zambia', 0.06140350877192982),
```

```
('Guadeloupe', 0.05701754385964912),
('Libya', 0.05701754385964912),
('Chad', 0.05701754385964912),
('Djibouti', 0.05701754385964912),
('Macau', 0.05701754385964912),
('Sierra Leone', 0.05263157894736842),
('Gambia', 0.05263157894736842),
('Afghanistan', 0.05263157894736842),
('Martinique', 0.04824561403508772),
('Bahamas', 0.04824561403508772),
('Saint Lucia', 0.04824561403508772),
('Namibia', 0.04824561403508772),
('Tajikistan', 0.04824561403508772),
('Zimbabwe', 0.04824561403508772),
('Reunion', 0.04824561403508772),
('Guam', 0.04824561403508772),
('Papua New Guinea', 0.043859649122807015),
('Haiti', 0.043859649122807015),
('Guatemala', 0.043859649122807015),
('Eritrea', 0.043859649122807015),
('Aruba', 0.039473684210526314),
('Guyana', 0.039473684210526314),
('Mauritania', 0.039473684210526314),
('Liberia', 0.039473684210526314),
('Suriname', 0.039473684210526314),
('Madagascar', 0.039473684210526314),
('Mozambique', 0.039473684210526314),
('Bolivia', 0.039473684210526314),
('Uruguay', 0.039473684210526314),
('Paraguay', 0.039473684210526314),
('Honduras', 0.039473684210526314),
('Syria', 0.039473684210526314),
('Virgin Islands', 0.039473684210526314),
('Dominica', 0.039473684210526314),
('Grenada', 0.03508771929824561),
('Turks and Caicos Islands', 0.03508771929824561),
('Saint Kitts and Nevis', 0.03508771929824561),
('Malawi', 0.03508771929824561),
('South Sudan', 0.03508771929824561),
('Kyrgyzstan', 0.03508771929824561),
('Mongolia', 0.03508771929824561),
('Belize', 0.03070175438596491),
('Burundi', 0.03070175438596491),
('Jersey', 0.03070175438596491),
('Comoros', 0.03070175438596491),
('Mayotte', 0.03070175438596491),
('British Virgin Islands', 0.03070175438596491),
```

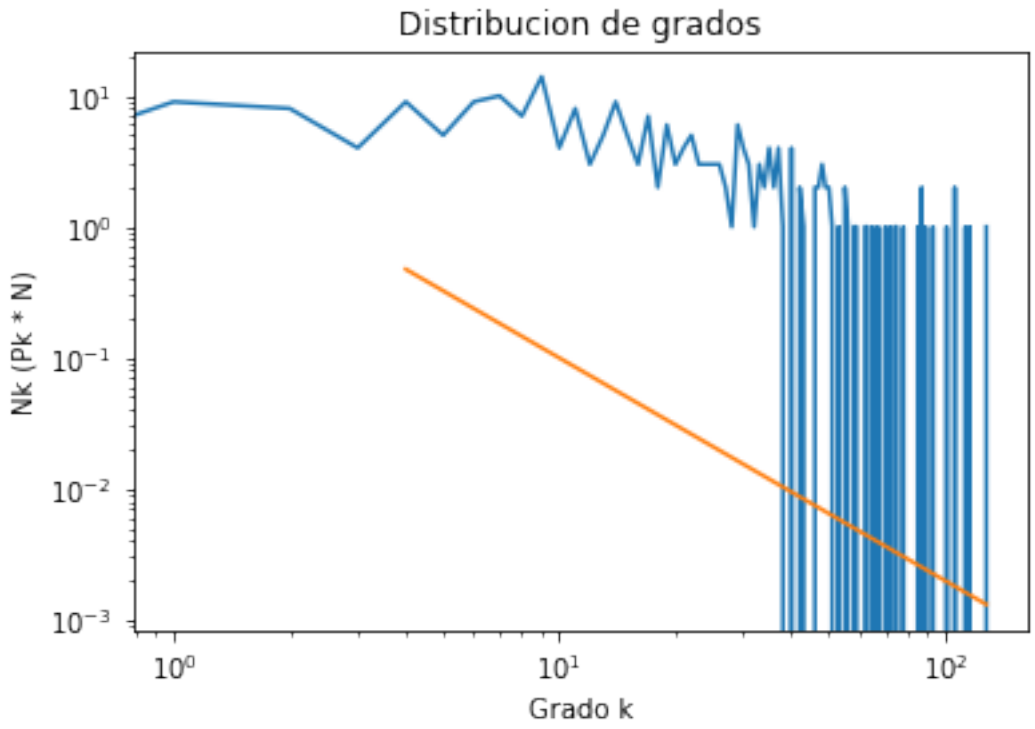
```
('Nicaragua', 0.03070175438596491),
('New Caledonia', 0.03070175438596491),
('Palau', 0.03070175438596491),
('Laos', 0.03070175438596491),
('Solomon Islands', 0.02631578947368421),
('Cayman Islands', 0.02631578947368421),
('Sao Tome and Principe', 0.02631578947368421),
('Botswana', 0.02631578947368421),
('Central African Republic', 0.02631578947368421),
('Guinea-Bissau', 0.02631578947368421),
('Somalia', 0.02631578947368421),
('French Polynesia', 0.02631578947368421),
('Saint Vincent and the Grenadines', 0.02631578947368421),
('Guernsey', 0.021929824561403508),
('Kiribati', 0.021929824561403508),
('Vanuatu', 0.021929824561403508),
('Nauru', 0.021929824561403508),
('Northern Mariana Islands', 0.021929824561403508),
('Micronesia', 0.017543859649122806),
('Faroe Islands', 0.017543859649122806),
('French Guiana', 0.017543859649122806),
('Cook Islands', 0.017543859649122806),
('Samoa', 0.017543859649122806),
('Marshall Islands', 0.017543859649122806),
('Anguilla', 0.017543859649122806),
('North Korea', 0.017543859649122806),
('Bhutan', 0.017543859649122806),
('Bermuda', 0.013157894736842105),
('Falkland Islands', 0.013157894736842105),
('Tonga', 0.013157894736842105),
('East Timor', 0.013157894736842105),
('Greenland', 0.008771929824561403),
('Gibraltar', 0.008771929824561403),
('Isle of Man', 0.008771929824561403),
('Western Sahara', 0.008771929824561403),
('Wallis and Futuna', 0.008771929824561403),
('Norfolk Island', 0.008771929824561403),
('Christmas Island', 0.008771929824561403),
('Cocos (Keeling) Islands', 0.008771929824561403),
('Saint Pierre and Miquelon', 0.0043859649122807015),
('Saint Helena', 0.0043859649122807015),
('Lesotho', 0.0043859649122807015),
('Swaziland', 0.0043859649122807015),
('Tuvalu', 0.0043859649122807015),
('American Samoa', 0.0043859649122807015),
('Niue', 0.0043859649122807015),
('Montserrat', 0.0043859649122807015),
```

```
('Myanmar', 0.0043859649122807015)]
```

0.0.20 5. a) Obtener una simulación de un modelado de Erdös-Rényi que corresponda a los parámetros de esta red.

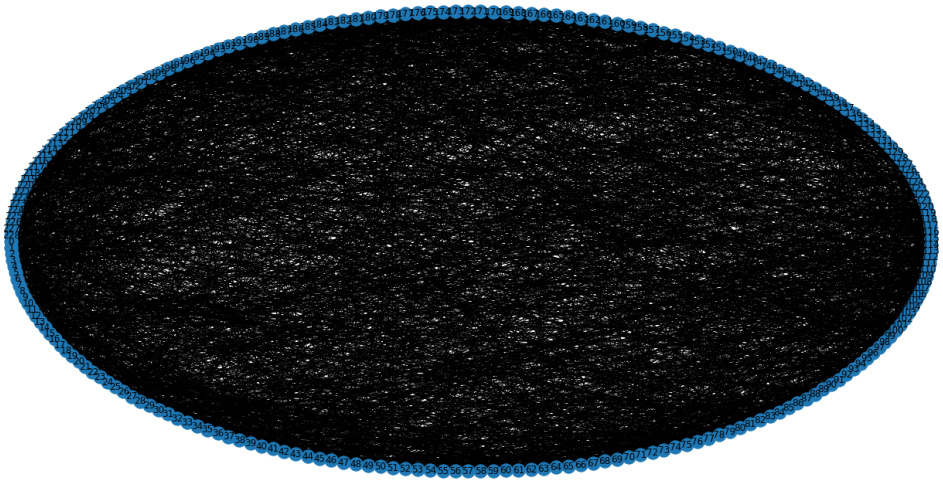
[66]:
```
# Utilizo la función graficar_distribuciones para obtener el alfa
graficar_distribuciones((distribucion_grados(G)))
# Con esto decimos que alfa: 2.7
```

```
Alfa:  2.7
```



[67]:
```
e_r = erdos_renyi(cant = len(G.nodes),k = grado_promedio(G))
```

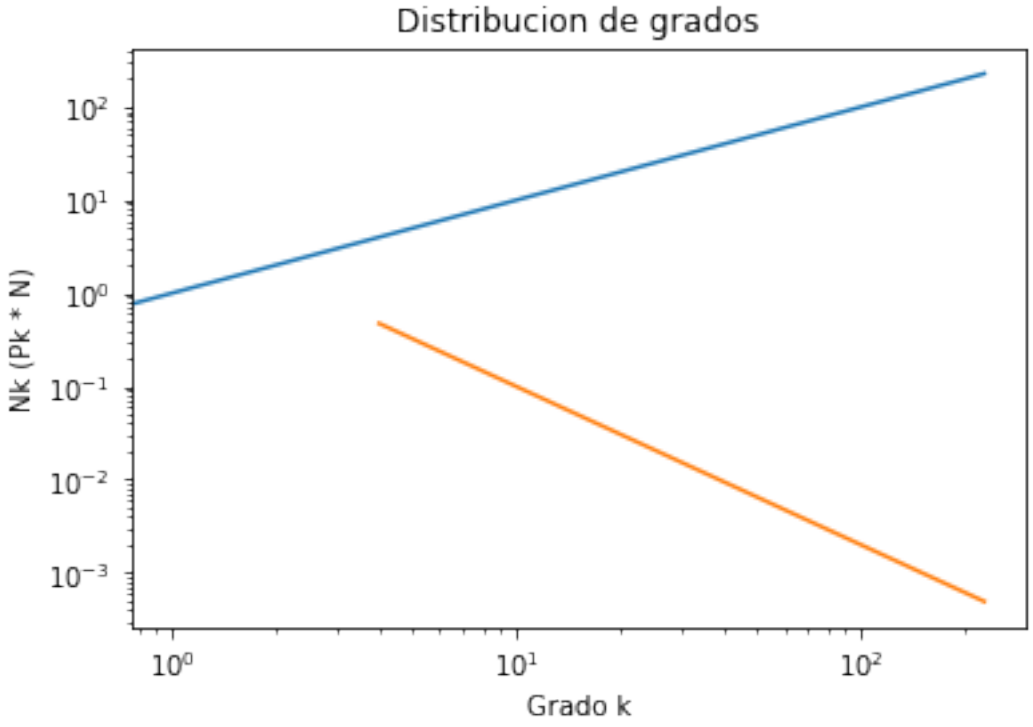
[68]:
```
# Grafico el grafo obtenido
figure(figsize=(20, 10))
nx.draw_shell(e_r, with_labels=True)
```



Análisis de la simulación

[69]:
```
#Analisis
graficar_distribuciones(e_r)
```

```
Alfa:  2.7
```



A mayor grado de nodo mayor cantidad de nodos que tienen dicho grado. Es proporcional la cantidad de nodos que tienen un grado y el grado mismo.

[70]:
```
# Diametro
# El diámetro se puede calcular con la función diameter de Networkx

nx.diameter(e_r, e=None, usebounds=False)
```

[70]:
```
3
```

Las redes reales suelen tener un diámetro pequeño. Este valor es representativo de una red real

[71]:
```
# El grado promedio se puede calcular con la siguiente función  de Networkx
grado_promedio(e_r)
```

[71]:
```
25.004366812227076
```

Similar al grafo original que estuvimos analizando. Es decir que tiene sentido para una red real.

[72]:
```
# Con la función clustering podemos obtener el valor del coeficiente de␣
 ↪clustering promedio de la red, quedandonos con el segundo valor de la tupla␣
 ↪que devuelve la función
c = clustering(e_r)
coeficiente_clustering_promedio = c[1]
coeficiente_clustering_promedio
```

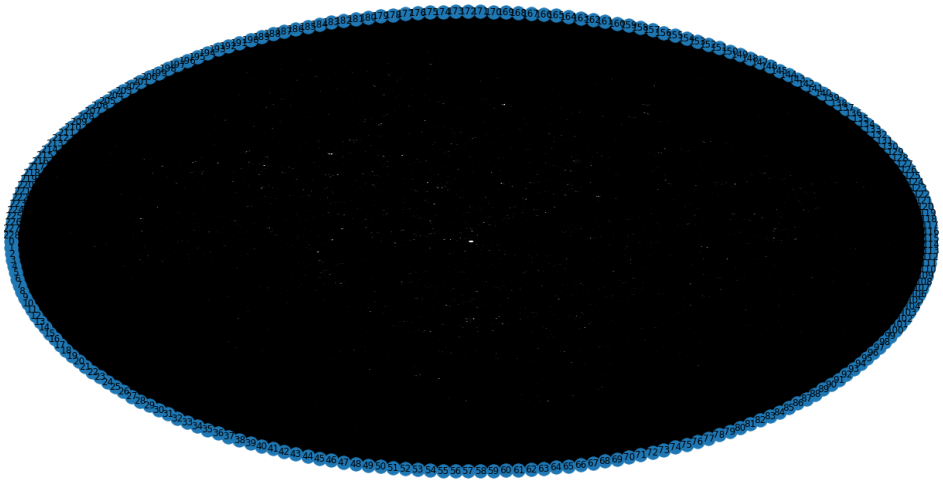
[72]:
```
0.11045181508840052
```

Los coeficientes de clustering de Erdös-Rényi suelen ser muy pequeños, este valor es esperable.

0.0.21 5. b) Obtener una simulación de un modelado de Preferential Attachment (ley de potencias) que corresponda a los parámetros de esta red.

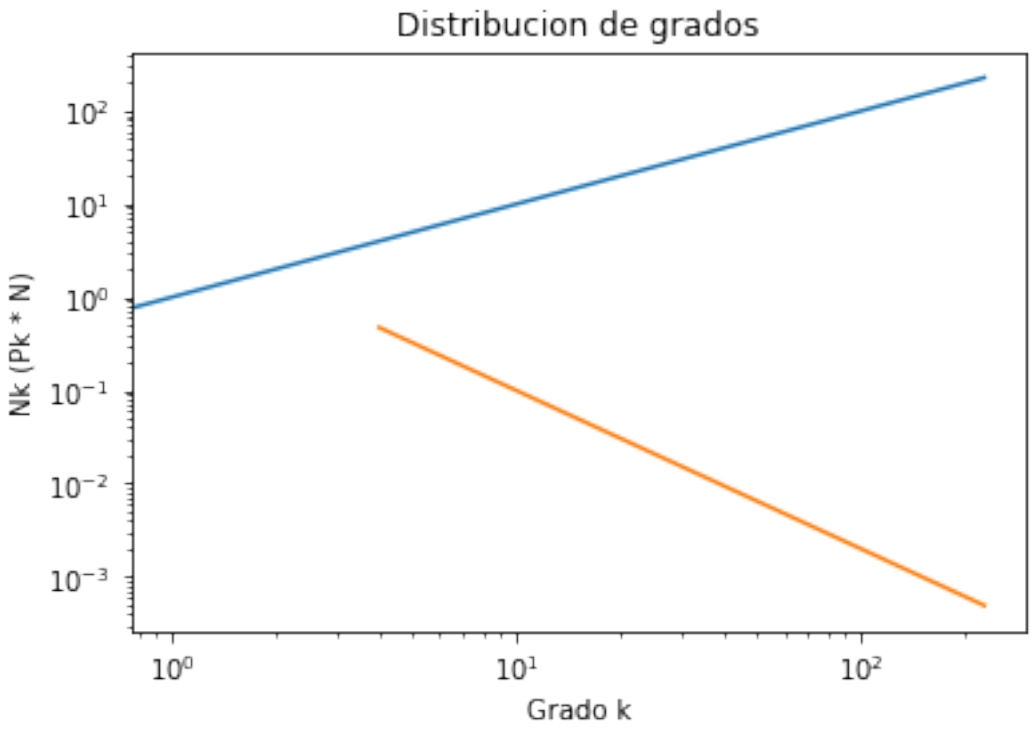
[73]:
```
p_a = preferential_attachment(dirigido = False,alfa = 2.7  , cant = len(G.
 ↪nodes), k = grado_promedio(G))
```

[74]:
```
figure(figsize=(20, 10))
nx.draw_shell(p_a, with_labels=True)
```



[75]:
```
#Analisis
graficar_distribuciones(p_a)
```

```
Alfa:  2.7
```



Mismo caso que el anterior, a mayor grado, mayor cantidad de nodos tienen ese grado.

[76]:
```
# Diametro
# El diámetro se puede calcular con la función diameter de Networkx

nx.diameter(p_a, e=None, usebounds=False)
```

[76]: 2

Las redes reales suelen tener un diámetro pequeño. Este valor es representativo de una red real

[77]:
```
# El grado promedio se puede calcular con la siguiente función  de Networkx
grado_promedio(p_a)
```

[77]: 59.06550218340611

No es representativo de una red real, es un valor extremadamente grande. Considerando que en nuestra real el grado promedio nos dio cerca de 24.

[78]:
```
# Con la función clustering podemos obtener el valor del coeficiente de␣
 ↪clustering promedio de la red, quedandonos con el segundo valor de la tupla␣
 ↪que devuelve la función
c = clustering(p_a)
coeficiente_clustering_promedio = c[1]
coeficiente_clustering_promedio
```

[78]: 0.2622961231675504

Es un poco menor que el anterior pero sigue siendo bajo en comparación con el que da el de la red real analizada (0.66).

[91]:
```
#Utilizo el siguiente código para generar un pdf con la resolucion.

!wget -nc https://raw.githubusercontent.com/brpy/colab-pdf/master/colab_pdf.py
from colab_pdf import colab_pdf
colab_pdf('Parcial1- TDA2.ipynb')
```

```
File ‘colab_pdf.py’ already there; not retrieving.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.


WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab
Notebooks/Parcial1- TDA2.ipynb to pdf
[NbConvertApp] Support files will be in Parcial1- TDA2_files/
[NbConvertApp] Making directory ./Parcial1- TDA2_files
```

```
[NbConvertApp] Making directory ./Parcial1- TDA2_files
[NbConvertApp] Making directory ./Parcial1- TDA2_files
[NbConvertApp] Making directory ./Parcial1- TDA2_files
[NbConvertApp] Making directory ./Parcial1- TDA2_files
[NbConvertApp] Making directory ./Parcial1- TDA2_files
[NbConvertApp] Making directory ./Parcial1- TDA2_files
[NbConvertApp] Writing 87973 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 2866637 bytes to /content/drive/My Drive/Parcial1-
TDA2.pdf
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

[91]: 'File ready to be Downloaded and Saved to Drive'