```
Jupyter 42_Code4 Last Checkpoint: 8 minutes ago (autosaved)
                                                       Help
                                                                                                                                        Python 3 O
                                             Widgets
 File
        Edit
              View
                      Insert
                              Cell
                                     Kernel
                                                                                                                           Not Trusted
                     ↑ ↓ | ► Run | ■ | C | → | Code
B + % < 6 </p>
                                                            ~
       In [1]:
                 1 # Flickr
                 2 import networkx as nx
                 3 G = nx.Graph()
                Preprocessing
                 1 file1 = open(r'C:\Users\DELL\Downloads\Compressed\flickr_UsrResTag\flickr-stream_U-R-T', 'r', encoding="utf8")
                 2 Lines = file1.readlines()
                 3 print(len(Lines))
                112900000
       In [3]:
                 1 from tqdm import tqdm
                 2 node1_List = []
                 3 node2_List = []
                 4 time_List = []
                 5 # Strips the newline character
                 6 for line in tqdm(range(0, 1000000)):
                        time, node1, node2, _ = Lines[line].split("\t")
                        node1_List.append(node1)
                 9
                        node2_List.append(node2)
                        time_List.append(time)
                                                                                 | | | | | | | 1000000/1000000 [00:01<00:00, 837175.40it/s]
       In [ ]: 1
      In [10]:
                 1 for edge in tqdm(range(len(node1_List))):
                       node1=node1_List[edge]
                        node2=node2_List[edge]
                        G.add_edge(node1,node2)
                 5 Degree_count_from={}
                 6 Degree_count_to={}
                 7 Degree_from={}
                 8 Degree_to={}
                 9 node1 List set=set(node1 List)
                10 node2_List_set=set(node2_List)
                11
                 12 for x in node1_List_set:
                # if(x not in Degree_count_from):
                14 Degree_from[x]=G.degree(x)
                 15
                     # else:
                     # print("hi")
                17
                     # break
                18
                 19 for y in node2_List_set:
                     # if(x not in Degree_count_to):
                          Degree_to[y]=G.degree(y)
                 21
                 22
                                  1000000/1000000 [11:59<00:00, 1388.94it/s]
                Degree Distribution : Users
                 1 list_of_degrees=list(Degree_from.values())
      In [12]:
                 2 from_degree_counts={}
                 3 for deg in list_of_degrees:
                        if(deg in from_degree_counts):
                            from_degree_counts[deg]+=1
                 6
                        else:
                            from_degree_counts[deg]=1
                 8 max_from_degree_freq=max(from_degree_counts.values())
                 10
                11 from_degree_counts1=from_degree_counts.copy()
                13 from_distis=dict((sorted(from_degree_counts1.items())))
                14 from_max_degree_count=max(from_distis.values())
                 15 # total_nodes=sum(distis.values())
                16 # from_distis.update((x, round(y/from_max_degree_count,2)) for x, y in from_distis.items())
                17 # x_axis=list(from_distis.keys())[1:len(from_distis.keys())]
                18 # y_axis=list(from_distis.values())[1:len(from_distis.values())]
                20 x_axis=list(from_distis.keys())
                21 y_axis=list(from_distis.values())
                23 import matplotlib.pyplot as plt
                24 import matplotlib
                25 plt.loglog(x_axis, y_axis,"o")
                26 # plt.xscale("log")
                27 # plt.yscale("log")
                28 plt.xlabel('degree')
                 30 plt.ylabel('number of nodes having degree')
                31
                32 plt.title('Degree Distribution of Users')
                 33 plt.legend()
                 34 # plt.show()
                35 fig = matplotlib.pyplot.gcf()
                No handles with labels found to put in legend.
                                               Degree Distribution of Users
                   10^{3}
               number of nodes having degree
                   10°
                                            10<sup>1</sup>
                                                                10<sup>2</sup>
                        10°
                                                                                    10^{3}
                                                        degree
                Degree Distribution : Tags
      In [13]:
                  1 list_of_degrees=list(Degree_to.values())
                 2 to_degree_counts={}
                 3 for deg in list_of_degrees:
                        if(deg in to_degree_counts):
                            to_degree_counts[deg]+=1
                        else:
                            to_degree_counts[deg]=1
                    max_to_degree_freq=max(to_degree_counts.values())
                 10
                11 to_degree_counts1=to_degree_counts.copy()
                13 to_distis=dict((sorted(to_degree_counts1.items())))
                14 to_max_degree_count=max(to_distis.values())
                15 # total_nodes=sum(distis.values())
                16 | # to_distis.update((x, round(y/to_max_degree_count,2)) for x, y in to_distis.items())
                17 # x_axis=list(to_distis.keys())[1:len(to_distis.keys())]
                18 # y_axis=list(to_distis.values())[1:len(to_distis.values())]
                20 x_axis1=list(to_distis.keys())
                21 y_axis1=list(to_distis.values())
                 22
                 23 import matplotlib.pyplot as plt
                24 import matplotlib
                25 # plt.scatter(x_axis1, y_axis1, label = "fraction")
                26 plt.loglog(x_axis1, y_axis1,"o" )
                27 # plt.xscale("log")
                28 # plt.yscale("log")
                29 plt.xlabel('degree')
                31 plt.ylabel('number of nodes having degree')
                 32
                33 plt.title('Degree Distribution of Tags')
                 34 plt.legend()
                35 # plt.show()
                36 fig = matplotlib.pyplot.gcf()
                No handles with labels found to put in legend.
                                                Degree Distribution of Tags
                   105
                 number of nodes having degree
                   10^{3}
                   10^{1}
                   10°
                                            10^{1}
                                                                10^{2}
                                                                                    10^{3}
                        10°
                                                        degree
      In [14]:
                 1 node1_degree_dict = {}
                 2 node2_degree_dict = {}
                 3 total_degree = 0
                 4 node time dict = {}
                 5 for i in tqdm(range(0, len(node2_List))):
                        node1 = node1_List[i]
                        node2 = node2_List[i]
                        if node1 not in node1_degree_dict:
                            node1_degree_dict[node1] = 0
                  9
                        node1_degree_dict[node1] += 1
                 10
                11
                        if node2 not in node2_degree_dict:
                12
                            node2_degree_dict[node2] = 0
                13
                        node2_degree_dict[node2] += 1
                14
                15
                        total_degree += 2
                 16
                                                                                         1000000/1000000 [00:00<00:00, 1104945.92it/s]
      In [15]:
                 1 probability dict = {}
                 2 for key, value in node1_degree_dict.items():
                        probability_dict[key] = value/total_degree
      In [16]:
                 1 print(len(node1_degree_dict))
                 print(len(probability_dict))
                13101
                13101
                Network Evolution : Probability vs Destination Degree
                 1 import matplotlib.pyplot as plt
      In [17]:
                 2 import matplotlib
                 4 plt.loglog(list(probability_dict.values()), list(node1_degree_dict.values()), "o")
                 5 # plt.scatter(list(probability_dict.values()), list(movie_degree_dict.values()), label = "scaled frequencies")
                 6 # plt.xscale("log")
                 7 # plt.yscale("log")
                 8 plt.xlabel('probability')
                10 plt.ylabel('degree')
                12 plt.title('Evolution')
                13 plt.legend()
                14 # plt.show()
                15 fig = matplotlib.pyplot.gcf()
                16 fig.set_size_inches(10, 7)
                No handles with labels found to put in legend.
                                                       Evolution
                   10^{4}
                   10^{3}
                 eg 10²
                   10^{1}
                   10°
                                                                          10^{-3}
                                                                                          10^{-2}
                                            10-5
```

10-6

 $10^{-4}$ 

probability

Logout