Zadání úlohy Random Graphs - 2rg

Cíl úlohy:

- Získat schopnost odhadnout vlastnosti komplexní sítě.
- Naučit se generovat syntetické sítě se zadanými vlastnostmi.
- Naučit se posuzovat základní charakteristiku komplexní sítě, kterou je distribuce stupňů uzlu.

Zadání:

1. Vygenerujte charakteristické komplexní sítě

   a) pravidelný graf,

   b) řídký Erdos-Renyi graf,

   c) Watts-Strogatzův graf,

   d) Barabasi-Albertové graf.

   Pro každý typ grafu vygenerujte jeden graf s 25 uzly a druhý s 10000 uzly. Zvolte vhodné další potřebné parametry.

2. Pro každý graf vygenerujte jeho vhodnou vizualizaci.
3. Pro každý graf určete histrogram četností stupňů uzlů a proveďte diskusi, zda se výsledek shoduje s teoretickou distribucí.

Doporučení:

1. Doporučujeme použít jazyk Python a knihovny NetworkX a Matplotlib
2. K vytvoření a nastavení cesty pro generované diagramy doporučujeme použít funkce

```
def SetOutPN(subFolder):
  #=====================
  # Open a necessary output infrastructure
  outPN = subFolder
  if not os.path.exists(outPN):
    os.makedirs(outPN)
  return outPN
```

3. K zobrazeni histogramu doporučujeme vyjít z funkce

```
def Histogram(sPlotIndex, x = None, xLabel = '', yLabel = '', num_bins = 100, log = False,
              normed = 0, colorLabel = None, alpha = None, **kwargs):

#=============================================================================================
  fig = plt.figure()
  fig.set_size_inches(4, 3) # width and height in inches
  ax = plt.subplot(sPlotIndex)
  n, bins, patches = plt.hist(x, num_bins, normed=normed, log = log, facecolor='green',
alpha=0.5)
  if log:
    plt.xscale('log')
    plt.ylim(ymin=0.5)
  if xLabel: plt.xlabel(xLabel)
  if yLabel: plt.ylabel(yLabel)
```

4. Po uložení diagramu doporučujeme resetovat knihovnu matplotlib pomocí funkce

```
def PltClear():
  #============
  plt.clf()
  fig = plt.gcf()
  plt.close(fig)
  plt.close('all')
```

5. Před zapsáním diagram do souboru doporučujeme použít funkci

```
plt.tight_layout()
```

6. Pro zápis diagram do souboru doporučujeme použít funkci

```
plt.savefig
```

7. Pro generování grafů doporučujeme použít následující funkce knihovny NetworkX:

```
nx.random_regular_graph
nx.erdos_renyi_graph
nx.watts_strogatz_graph
nx.barabasi_albert_graph
```

8. Pro získání seznamu všech hodnot stupňů uzlu v daném grafu `myGraph` doporučuje využít konstrukce

```
list(nx.degree(myGraph).values())
```

9. Log-Log histogram lze vygenerovat pomocí následující funkce

```
  def Histogram(self, sPlotIndex, x = None, xLabel = '', yLabel = 'Count', bins = 100,
                log = False,
                density = 0, logBins = False, logX = False, logY = False,  isGrid = False,
                colorLabel = None, title = None, **kwargs):
                #facecolor = 'green', alpha = 0.5,
    if type(sPlotIndex) is tuple:
      ax = plt.subplot(*sPlotIndex)
    else:
      ax = plt.subplot(sPlotIndex)
    if logBins:
      minV = min([v for v in x if v > 0])
      bins = np.logspace(np.log10(minV/2), np.log10(np.max(x)), num=bins)
    n, bins, patches = ax.hist(x, bins = bins, density=density, log = log, **kwargs)
```

```
    if logX:   ax.set_xscale('log')
    if logY:   ax.set_yscale('log', nonposy='clip')#, nonposy='clip'
    if xLabel: ax.set_xlabel(xLabel)
    if yLabel: ax.set_ylabel(yLabel)
    if title:  ax.set_title(title)
    if isGrid: ax.grid(isGrid)
    return ax
```