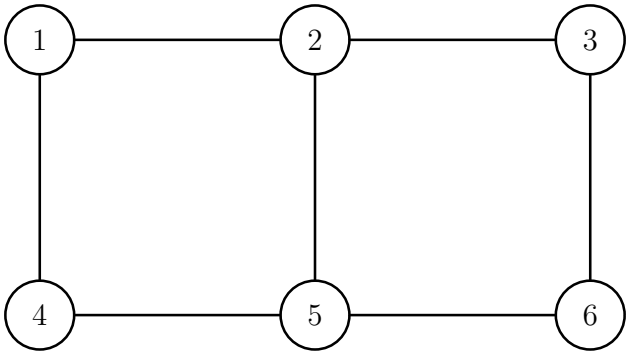
Homework 3 (Due March 29)

CS7253: Graph Algorithms
Kennesaw State University
Spring 2024

In this project, we consider distributions over spanning trees.

Consider the following graph:



In total, this graph has 15 different spanning trees:

- either edge $(2, 5)$ is part of the spanning tree, or it is not.
- if edge $(2, 5)$ is not part of the spanning tree, then the graph consists of a single cycle of 6 nodes and 6 edges. If we delete any single edge, the cycle will become a spanning tree. Hence, there are $\binom{6}{1} = 6$ different spanning trees that do not use the edge $(2, 5)$.
- if edge $(2, 5)$ is part of a spanning tree, then the (left) sub-tree over nodes 1,2,4,5 must be a spanning tree, and the (right) sub-tree over nodes 2,3,5,6 must also be a spanning tree. Note that we can pick any spanning tree on the left, and any spanning tree on the right and combine them, to form a spanning tree over all nodes. The left sub-graph and right-sub graph are both cycles. Since we assumed that edge $(2, 5)$ is included in the spanning tree, then we pick one out of three edges on each side to remove. This gives us $3 \cdot 3 = 9$ different spanning trees that use the edge $(2, 5)$.

In total, this graph has $6 + 9 = 15$ spanning trees. Out of these 15 spanning trees, 9 of them have the edge $(2, 5)$ and 6 of them do not have the edge $(2, 5)$. Hence, if we were to pick one of these spanning trees at random (with equal probability), then the probability of picking a tree that has the edge $(2, 5)$ is $\frac{9}{15} = \frac{3}{5}$.

Let G be an undirected graph, and let t be the number of spanning trees in G. If we assume a uniform distribution over spanning trees, then the probability of picking a specific spanning tree of G, from the set of all spanning trees of G is $\frac{1}{t}$. Let $G_{(i,j)}$ be the graph where edge (i, j) has been removed from G, and let $t_{(i,j)}$ be the number of spanning trees in $G_{(i,j)}$. That is, $t_{(i,j)}$ is the number of spanning trees of G that does not contain edge (i, j). Moreover, $t - t_{(i,j)}$ is the number of spanning trees of G containing the edge (i, j).

The edge-appearance probability of an edge (i, j) is the probability that it appears in a spanning tree of G, if we select a spanning tree of G at random. The edge-appearance probability of edge (i, j) is thus

$$\frac{t - t_{(i,j)}}{t}.$$

In this project, you are to implement three functions:

```
graph_with_edge_pr_one(n,edge)
graph_with_edge_pr_one_over_n(n,edge)
graph_with_edge_pr_one_over_k(n,k,edge)
```

defined in the file `graphs.py`. The first function, given an integer $n \geq 3$ and an edge (i, j) where $0 \leq i, j < n$, should return a list of edges (a graph) where the edge-apperance probability of edge (i, j) is 1. The second function, given an integer $n \geq 3$ and an edge (i, j), should return a graph where the edge-apperance probability of edge (i, j) is $\frac{n-1}{n}$. The third function, given an integer $n \geq 3$ another integer $3 \leq k \leq n$, and an edge (i, j), should return a graph where the edge-apperance probability of edge (i, j) is $\frac{k-1}{k}$.

Code for counting the number of spanning trees of a graph, and for computing edge-appearance probabilities, has been included in the file `count.py`. This includes the functions:

```
count_spanning_trees(edge_list)
edge_probability(edge_list,edge)
```

Given a graph specified by a list of edges `edge_list`, the `count_spanning_trees` function returns the number of spanning trees of the given graph. The `edge_probability` returns, given a list of edges and an edge from this list, the edge-appearance probability of the edge. These functions require the python module `numpy` to be installed, which can be installed, e.g., through pycharm (if you use pycharm), or via pip.

The file `main.py` is an example program that includes a few tests.

Turn in: your modified version of `graphs.py` onto the course website under `Assignments` and `Homework 3`. Turn in `graphs.py` **only**. Your implementation of the three functions should not require any changes outside of `graphs.py`. Assignments are due Friday, March 29 by 11:59pm. Please start early in case you encounter any unexpected difficulties.

Testing and Grading: The homework assignment includes tests in the file `test_public.py`. These are `pytest` tests, and should not be modified. If you want to run these tests, you can install `pytest` and then run the command:

```
pytest test_public.py
```

to run all tests. If you code is implemented correctly, you should see 5 passing tests:

```
============================= test session starts ==============================
platform linux -- Python 3.8.18, pytest-7.4.0, pluggy-1.0.0
rootdir: /kennesaw/cs7253/spring24/hw3/
plugins: anyio-3.5.0
collected 5 items

test_public.py .....                                                     [100%]

============================== 5 passed in 0.13s ===============================
```

Your homework will be graded based on a suite of *private* tests, which may or may not include the public tests provided with the assignment. If your script passes all private tests, your score will be 100 out of 100. If your script does not run, you will fail all tests, and receive a score of zero. Do not import any modules beyond those that are already imported for you in the homework. Missing imports will cause tests to fail. New modules will not be installed on the testing system. Any identical code submissions will also receive a score of zero.

Included files:

- `homework03.pdf`: this document

- `graphs.py`: the script you implement and turn in
- `count.py`: the script containing the `count_spanning_trees` and `edge_probability` functions
- `main.py`: an example program that you can run
- `tests.py`: the script containing the `pytest` tests

Hint: Use the debugger.