Graphs, Graph Algorithms, and Optimization
Problem set 3

W 24/25

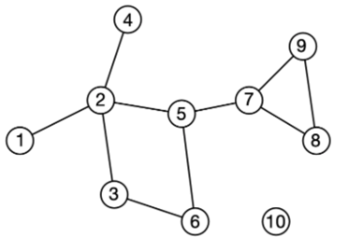
Include all your reasoning steps, but only the neccesary ones. Do not use built-in python functions that already solve the problems for you.

Total points (10 + 7 Bonus)

Paper and pen

(1) Imagine that your social graph/network has a subgraph/subnetwork where 14 of your friends including you are all friends with each other. What is such a subgraph/subnetwork called formally? How many edges are contained in this subnetwork? **(2pts)**

(2)



(i) Write down the adjacency matrix and the edgelist. **(2pts)**

(ii) Draw the degree distribution of network above by hand. 17.5: Degree Distribution - Mathematics LibreTexts **(2pts)**

(iii) Find the number of d=3 paths between 2 and 3. Which node pair has the most d=3 paths? **(2pts)** Hint: which graph representation is the best for this? PS: You can use code for this part, but include the photo of your code, and the output.

(3) Consider a bipartite network with N_1 and N_2 nodes in the two sets. (i) What is the maximum number of edges the network can have? **(1pt)**

(ii) Find an expression for how many edges cannot occur compared with a non-bipartite network of size $N = N_1 + N_2$? **(2pts)**

Computational part

(4) Implement an adjusted version of the Havel-Hakimi algorithm in the provided Python notebook. This version should generate all possible non-isomorphic graphs corresponding to a given sequence of integers, or return an empty list if the sequence is not graphical. The notebook can be found in the "Home Projects" folder under the file Exercises - Home project 1 **(4pts)**

Input: A sequence of integers representing a degree sequence.
Output: A list of all possible graphs corresponding to the graphical sequence or an empty list if no valid graph exists.

For your submission, include:

a) The implementation code in the prescribed places in the provided notebook.

b) A PDF export of the whole python notebook that includes:

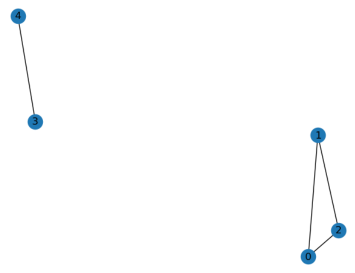
- Tests run on these input sequences :
  - [2,2,2,1,1]
  - [3,3,3,3,3,3]
  - [3,3,3]
- Images of the graphs generated for each sequence (if they exist).

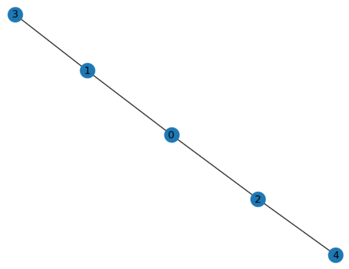
The following code in the notebook will output all of these once you implement all the algorithms in the notebook marked with #TODO.

```
test_havel_hakimi_2() ...
```

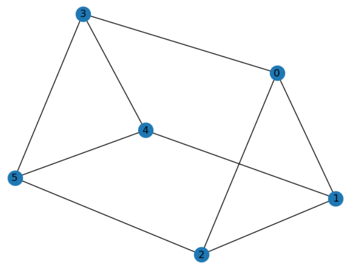
Make sure to include informal analysis on the worst time complexity of your algorithm in the notebook. **(2 pts)**

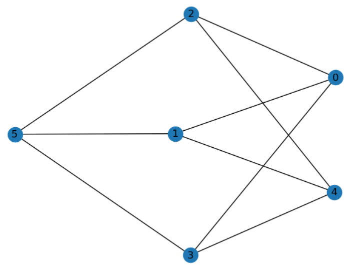
Tests: Sequence [2,2,2,1,1] will output 2 graphs





Sequence [3,3,3,3,3,3] will output 2 graphs





Sequence [3,3,3] will output 0 graphs.