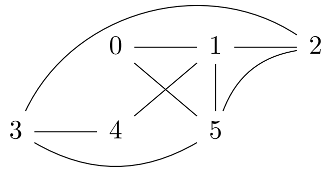
Exercises Problem Set 1

Meta-Algorithm: Before you begin designing an algorithm or a proof it is important to ensure you have a good understanding the problem. Ensure you've read the description carefully and attempt to settle every question or doubt. A great strategy is to try to solve a small instance of the problem before attempting a full solution, i.e. try very small n (2 or 3 can be enough) to devise a toy-version of the problem that you can solve quickly to develop an intuition.

Question 1. What is an algorithm and what are its characterstics? Write down a list of features common to any algorithm. What is the difference between an algorithm and pseudocode?

Recall from the lecture two common ways of representing a graph: an *adjacency list*, a dictionary where vertex are keys with a list of neighbouring vertices as value; *adjacency matrix*, a 2D array where the value of $array[i][j]$ indicates whether or not there is an edge between vertex i and vertex j, with a 1 or 0 respectively.

Question 2. Determine the adjacency matrix and the adjacency list for the following undirected graph.



Question 3. Given an adjacency list representation, provide a pseudocode algorithm that answers whether a given undirected graph contains at most two vertices with an odd number of neighbours. If the answer is yes then output True, otherwise False.

Question 4. **a.** Provided an adjacency list representation, provide a pseudocode algorithm that determines the total number of edges over a given undirected graph.
b. Idem **a.**, but for an adjacency matrix representation.
c. Which, if any, representation is most efficient? Motivate your answer

Question 5. Given a **directed** graph, which has an edge from vertex a to vertex b but not edge from b to a.
a) Give a pseudocode algorithm that inverts this edge. Use the adjacency matrix.
b) Idem, using the adjacency list.
c) Suppose you were to design an algorithm which inverts all edges of the graph, which representation would you prefer? Motivate your answer.

Question 6. Solve the exercises in this week's programming work sheet.