Part 1

The program `test_graphs.py` demonstrates functions to build, modify, and display graphs using an adjacency list representation, implemented in the class `graph_AL.py`. Your task consists of implementing the same functions using an adjacency matrix representation and an edge list representation.

1. Complete the implementation of the following functions inside the `graph_AM.py` class and test them using the `test_graphs.py` program:
   (a) `insert_edge`
   (b) `delete_edge`
   (c) `display`

2. Complete the implementation of the following functions inside the `graph_EL.py` class and test them using the `test_graphs.py` program:
   (a) `insert_edge`
   (b) `delete_edge`
   (c) `display`

3. Add the following functions to each of the classes `graph_AL.py`, `graph_AM.py` and `graph_EL.py`:
   (a) `as_AL( )`, that returns an adjacency list representation of the graph
   (b) `as_AM( )`, that returns an adjacency matrix representation of the graph
   (c) `as_EL( )`, that returns an edge list representation of the graph

4. Use the function `as_AL( )` in `graph_AM.py` and `graph_EL.py` to implement the `draw()` function.

Part 2

You have a fox, a chicken and a sack of grain. You must cross a river with only one of them at a time. If you leave the fox with the chicken he will eat it; if you leave the chicken with the grain he will eat it. How can you get all three across safely? For part 2, you will implement a solution to this problem using a graph search algorithm.

Let’s represent the state of the world with four bits \( \langle b_0, b_1, b_2, b_3 \rangle \). Bit \( b_0 \) represents the location of the fox, \( b_1 \) represents the location of the chicken, \( b_2 \) represents the location of the grain, and \( b_3 \) represents your location. We will assume you are going from the left side to the right side of the river, thus for a particular bit \( b_i \), \( b_i = 0 \) means the corresponding entity (fox, chicken, grain, or person) is on the left side of the river and \( b_i = 1 \) means the entity is on the right side. We can represent states by nodes in a graph \( G = (V, E) \). Thus \( V = \{ \langle 0,0,0,0 \rangle, \langle 0,0,0,1 \rangle, \ldots, \langle 1,1,1,0 \rangle, \langle 1,1,1,1 \rangle \} \), or you could use the decimal representations of the bit vector and thus \( V = \{ 0,1, \ldots, 14, 15 \} \). Legal transitions between legal states will be represented by undirected
edges in that graph. A transition is legal if the person moves from one side of the river to the other and is accompanied by zero or one other entities. A state is not legal if the fox can eat the chicken or the chicken can eat the grain, as explained above, otherwise it is legal. Thus \((0, 0, 0, 0), (0, 1, 0, 1) \in E\), corresponding to the person crossing the river with the chicken, while \((0, 0, 0, 0), (0, 0, 1, 1) \notin E\), since \((0, 0, 1, 1)\) is not a legal state (the fox eats the chicken).

We can now find the solution to the problem by finding a path going from state \((0, 0, 0, 0)\) to state \((1, 1, 1, 1)\) in the graph. Your task is to implement six different solutions to the problem the three graph representations (adjacency list, adjacency matrix and edge list) and two search algorithms (breadth-first search and depth-first search).

As usual, write a report describing your work. Show sample outputs of your program, including the drawing of the graph from part 2 and the sequence of steps required to solve the puzzle.