MODULE 8 PROJECT: GRAPHS

CLO 3. Develop problem solving skills by implementing data structures.

CLO 5. Design and analyze composite data structures.

Data Structure Implementation

1. Implement `AdjacencyMatrix`.

   Files to Modify: AdjacencyMatrix.py.

   Directions: Finish the implementation of the following methods from the `AdjacencyMatrix` class.

   - `add_edge(i, j)` - adds edge (i, j) to the adjacency matrix. Raises `IndexError` if `i` or `j` are not in the range 0 to $n - 1$, where n is the number of vertices.
   - `remove_edge(i, j)` - removes edge (i, j) from the adjacency matrix, if it exists. Returns `True` if the edge was removed, `False` if the edge did not exist in the adjacency matrix and thus was not removed. Raises `IndexError` if `i` or `j` are not in the range 0 to $n - 1$, where n is the number of vertices.
   - `has_edge(i, j)` - returns `True` if the edge (i, j) exists in the adjacency matrix, `False` otherwise.
   - `out_edges(k)` - returns an `ArrayList` containing the indices j of the nodes that have an edge with node `k`, where `k` is the source, i.e., edges (k, j) exist in the adjacency matrix.
   - `in_edges(j)` - returns an `ArrayList` containing the indices i of the nodes that have an edge with node `j`, where `j` is the target, i.e., edges (i, j) exist in the adjacency matrix.
   - `bfs(i)` - returns an `ArrayList` of node indices as the nodes are traversed using Breadth-First Search, starting at node `i`.
   - `dfs(i)` - returns an `ArrayList` of node indices as the nodes are traversed using Depth-First Search, starting at node `i`.

2. Implement `AdjacencyList`.

   Files to Modify: AdjacencyList.py.

   Directions: Finish the implementation of the following methods from the `AdjacencyList` class.

   - `add_edge(i, j)` - adds edge (i, j) to the adjacency list. Raises `IndexError` if `i` or `j` are not in the range 0 to $n - 1$, where n is the number of vertices. ***NOTE: Be careful to NOT have any duplicate edges!***
   - `remove_edge(i, j)` - removes edge (i, j) from the adjacency list, if it exists. Returns `True` if the edge was removed, `False` if the edge did not exist in the adjacency list and thus was not removed. Raises `IndexError` if `i` or `j` are not in the range 0 to $n - 1$, where n is the number of vertices.
   - `has_edge(i, j)` - returns `True` if the edge (i, j) exists in the adjacency list, `False` otherwise.
   - `out_edges(k)` - returns an `ArrayList` containing the indices j of the nodes that have an edge with node `k`, where `k` is the source, i.e., edges (k, j) exist in the adjacency list. Raises `IndexError` if `i` or `j` are not in the range 0 to $n - 1$, where n is the number of vertices.
   - `in_edges(j)` - returns an `ArrayList` containing the indices i of the nodes that have an edge with node `k`, where `j` is the target, i.e., edges (i, j) exist in the adjacency list.
   - `bfsir)` - returns an `ArrayList` of node indices as the nodes are traversed using Breadth-First Search, starting at node `i`.
   - `dfs(i)` - returns an `ArrayList` of node indices as the nodes are traversed using Depth-First Search, starting at node `i`.

SUBMISSION PROCESS

Submit to CodePost

- `AdjacencyMatrix.py`
- `AdjacencyList.py`