For this coursework, I chose to use an adjacency list to represent the directed graph, as it is a simple and efficient data structure for storing sparse graphs. The adjacency list allows us to quickly find all the neighbours of a given vertex and add or remove edges from the graph.

I implemented the sink elimination algorithm to determine if the given graph is acyclic. This algorithm works by repeatedly removing sinks (vertices with no outgoing edges) from the graph until it is empty. If the graph has no sink, it is not acyclic. Otherwise, if the graph is empty, it is acyclic.

To remove sinks from the graph, I implemented a function that searches the adjacency list for vertices with no outgoing edges and removes them from the graph.

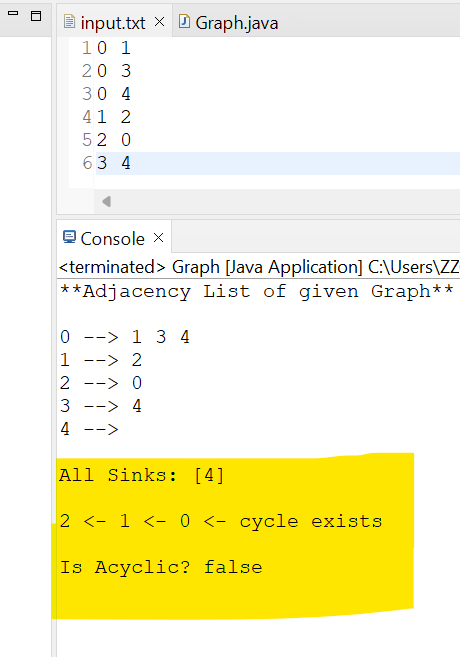
To detect cycles in the graph, I implemented a depth-first search (DFS) function that searches for cycles starting from each vertex in the graph. If a cycle is found, the function returns true, indicating that the graph is not acyclic.

In terms of performance, the adjacency list data structure has a time complexity of O (V + E) for most operations, where V is the number of vertices and E is the number of edges in the graph. This means that the time complexity of the sink elimination algorithm is also O (V + E) since it involves iterating over the vertices and edges of the graph. The DFS function has a time complexity of O (V + E) as well since it also involves iterating over the vertices and edges of the graph. Overall, the performance of the implemented algorithm is satisfactory for small to medium-sized graphs but may not scale well for very large graphs with many vertices and edges. In such cases, it may be more efficient to use a different algorithm, such as topological sorting, which has a time complexity of O (V + E) for acyclic graphs and O (V \* E) for general graphs.

As an example, consider the following small acyclic graph: Using the sink elimination algorithm, we can determine that this graph is acyclic by removing the sinks 1, 2, and 3 in that order. The time complexity of this algorithm for this graph is O (5 + 6) = O (11). On the other hand, using topological sorting, we can determine that this graph is acyclic by sorting the vertices in topological order. The time complexity of topological sorting for this graph is O (5 + 6) = O (11). In both cases, the time complexity is linear with respect to the number of vertices and edges in the graph, which is satisfactory for small graphs. However, for very large graphs with many vertices and edges, topological sorting may be more efficient, since it has a lower time complexity in the worst case. In conclusion, the adjacency list data structure and the sink elimination algorithm are suitable for determining if small to medium-sized directed graphs are acyclic. For very large graphs, a more efficient algorithm such as topological sorting may be preferred.

**Example 1**

**Graph Output**



**Example 2**

**Graph Output**

