1 Graph Representations

Write the graph above as an adjacency matrix, then as an adjacency list. What would be different if the graph were undirected instead?

2 Topological Sorting

Give a valid topological sort of the graph above. For your reference, some orderings of the graph are provided below.

DFS preorder: ABCFDE (G)
DFS postorder: FCBEDA (G)
BFS: ABDCEF (G)

There are two requirements that a graph must satisfy in order for there to be a valid topological sorting of the graph. What are they?

Extra: Why does the method to compute a topological sort work?
3 Graph Algorithm Design

3.1 An undirected graph is said to be bipartite if all of its vertices can be divided into two disjoint sets $U$ and $V$ such that every edge connects an item in $U$ to an item in $V$. For example below, the graph on the left is bipartite, whereas on the graph on the right is not. Provide an algorithm which determines whether or not a graph is bipartite. What is the runtime of your algorithm?

3.2 Consider the following implementation of DFS, which contains a crucial error:

- create the fringe, which is an empty Stack
- push the start vertex onto the fringe and mark it
- while the fringe is not empty:
  - pop a vertex off the fringe and visit it
  - for each neighbor of the vertex:
    - if neighbor not marked:
      - push neighbor onto the fringe
      - mark neighbor

Give an example of a graph where this algorithm may not traverse in DFS order.

3.3 *Extra:* Provide an algorithm that finds the shortest cycle (in terms of the number of edges used) in a directed graph in $O(EV)$ time and $O(E)$ space, assuming $E > V$. 