Algorithms & Data Structures
SI3 - Polytech Nice-Sophia - Edition 2018

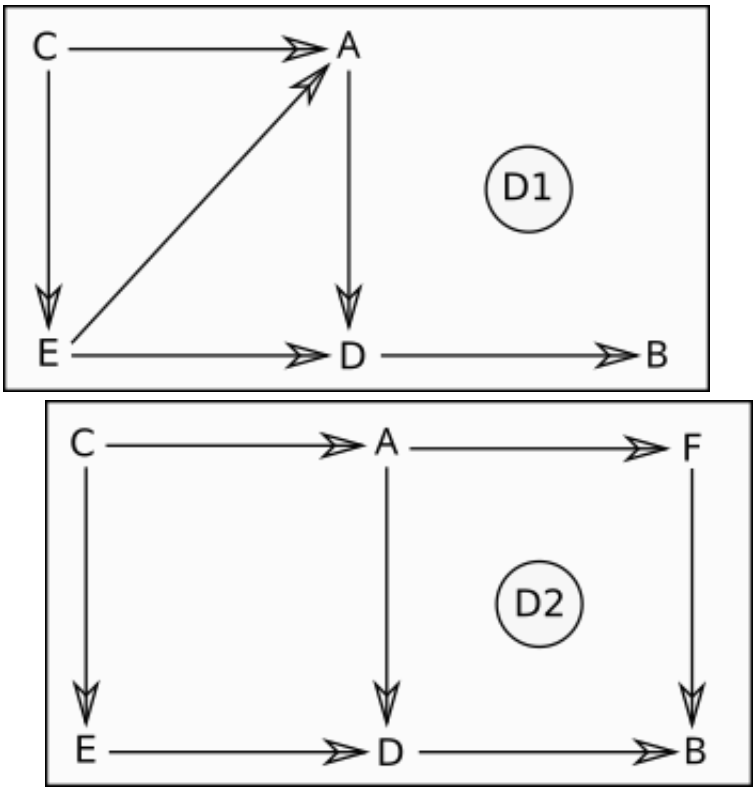
Lab #8: Topological sorting

This assignment will give you practice about topological sorting algorithms. For all parts you are to use the provided material:

- lab8.py: a testing module. This file provides function skeletons you have to complete plus some interactive code you can use to test them
- we use the same `Graph` class as in previous lab

Part 1: topological sorting

Given the following digraphs `D1` and `D2`:

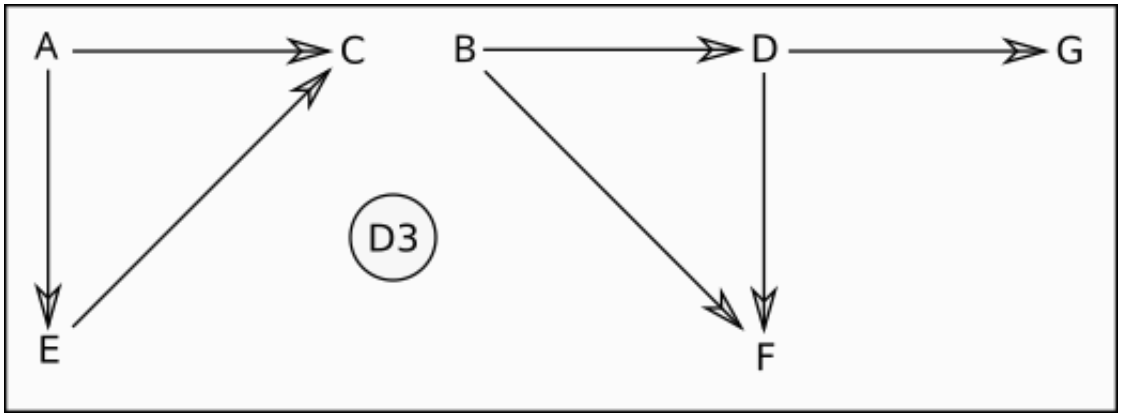


give all the possible topological orders for `D1` and `D2`.

Part 2: unique topological sorting

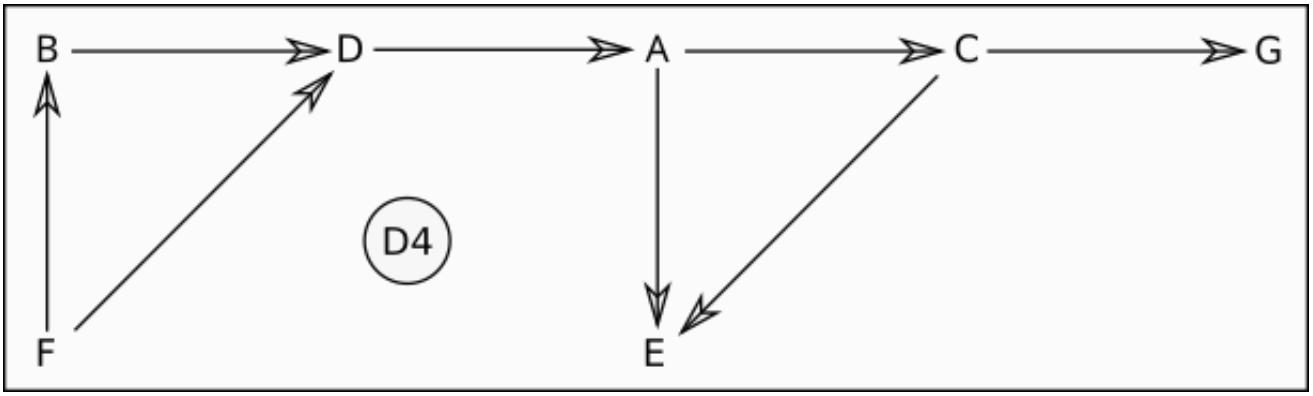
For the graph `D2` from part 1, what is the minimum number of edges to add to the graph so that there is only one topological order? Give an example of such edges.

Explain why it seems obvious that graph `D3` below has more than one topological order, and give a minimum set of edges to add to `D3` so that there is only one topological order for that graph:



Part 3: topological sort, version 1

In this part, you are to design the `topo_1` function to implement the *topological sort* algorithm reviewed in class. Your implementation should be such that when you compute the topological sort of a given digraph, the digraph is not changed by this computation. This means that the statement *"decrease the in-degree of vertex u"* must be replaced by some (simple) programming trick because you are not able to decrease the in-degree of a vertex in a digraph. In that algorithm you must use a *queue* to store vertices of the digraph. While the algorithm described in the lesson is simply displaying the vertices in the ordering found by the topological sort, you should make the `topo_1` function to return a list of vertices such that traversing that list gives the vertices in the topological order. For example, given the graph D4:



one topological order of D4 is:

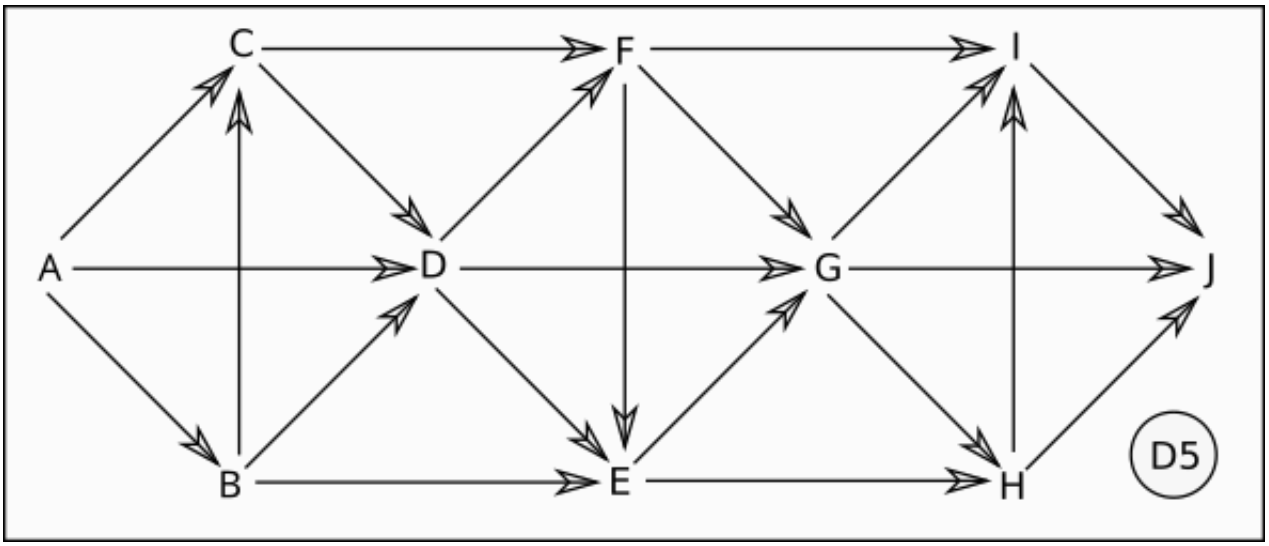
```
[F, B, D, A, C, G, E]
```

Part 4: topological sort, version 2

In this part, you are to design the `topo_2` function to implement another version of the *topological sort* . This version should be based on the generic algorithm Depth-First-Search (DFS). To design this new version you have to review the solutions of lab #7 and adapt them to perform a topological sort.

Part 5: property for uniqueness

Give a general property for a directed graph to have only one topological order. Design the function `topo_unique` to check if a digraph has a unique topological ordering. For example, the following digraph D5 has a unique topological order:



which is:

```
[A, B, C, D, F, E, G, H, I, J]
```

although graphs D2, D3 and D4 from previous parts have not.