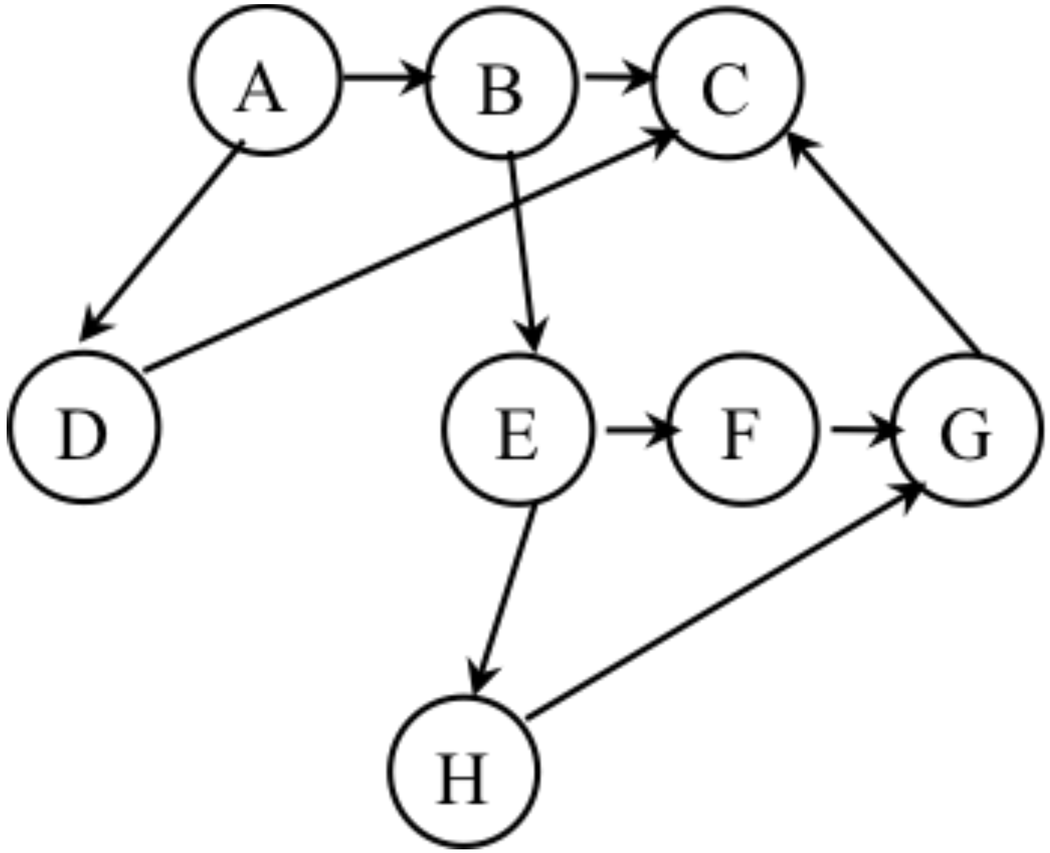
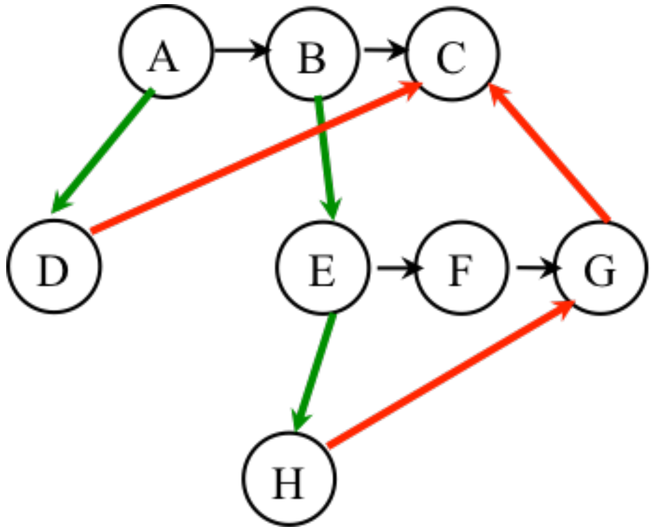
Worksheet: Reverse Engineering a Parallel Program from a Computation Graph (CG)

Write a parallel program that generates exactly the same ordering constraints as the computation graph shown. The program should be written in pseudocode using finish and async annotations. The CG nodes should be clearly identified as statements in the program e.g., as method calls A(), B(), etc. Since the CG edges are not labeled as spawn, continue, or join, you can make whatever assumptions you choose about the edges when writing your program. The only requirement is that the ordering constraints in your program coincide with those in the graph. Submit solution in Canvas.



One Possible Solution to Worksheet (Reverse Engineering a Computation Graph)



Observations:

- Any node with out-degree > 1 must be an async (must have an outgoing spawn edge)
- Any node with in-degree > 1 must be an end-finish (must have an incoming join edge
- Adding or removing transitive edges does not impact ordering constraints

```
1.A();
2.finish { // F1
3.   async D();
4.   B();
5.   E();
6.   finish { // F2
7.     async H();
8.     F();
9.   } // F2
10. G();
11.} // F1
12.C();
```