Consider the Parallel Spanning Tree algorithm discussed in the last lecture (and shown below in slide 18). Assume that the isolated construct is implemented using a Transactional Memory mechanism. Outline a parallel execution scenario for the input graph below that could lead to a conflict between isolated constructs.
Parallel Spanning Tree Algorithm

1. class V {
2.     V [] neighbors; // adjacency list for input graph
3.     V parent; // output value of parent in spanning tree
4.     boolean tryLabeling(final V n) {
5.         return isolatedWithReturn(() -> {
6.             if (parent == null) parent = n;
7.             return parent == n; // return true if n became parent
8.         });
9.     } // tryLabeling
10.    void compute() {
11.        for (int i=0; i<neighbors.length; i++) {
12.            final V child = neighbors[i];
13.            if (child.tryLabeling(this))
14.                async(() -> { child.compute(); }); // escaping async
15.        } // compute
16.    } // class V
17.} // class V
18.
19.    root.parent = root; // Use self-cycle to identify root
20.    finish(() -> { root.compute(); });
21.    . . .