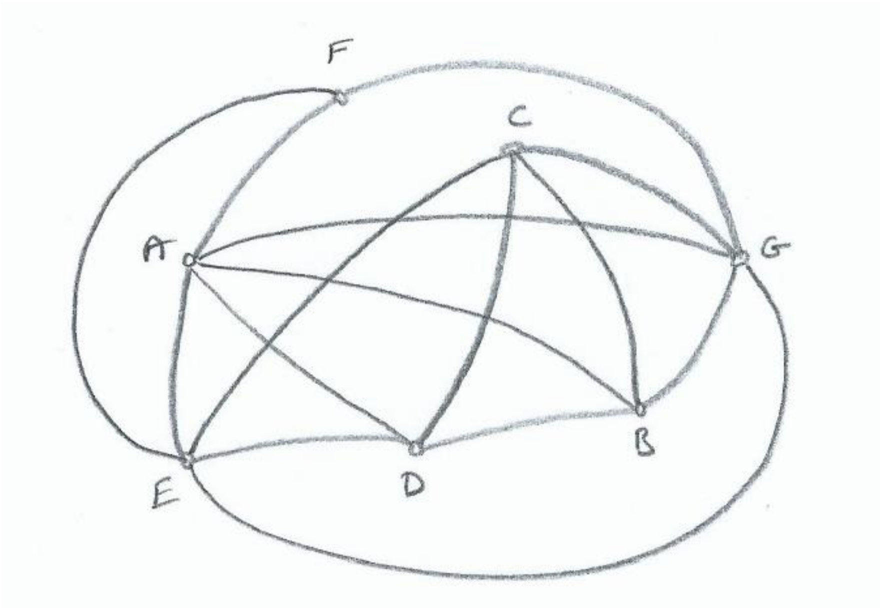
Planarity Algorithm (5 pages; 30/6/20)

(1) A graph is planar if it can be redrawn (if necessary), so that its arcs do not cross.

If a graph contains a Hamiltonian cycle (ie a cycle that passes through all nodes exactly once), then the Planarity Algorithm enables us to establish whether or not the graph is planar.

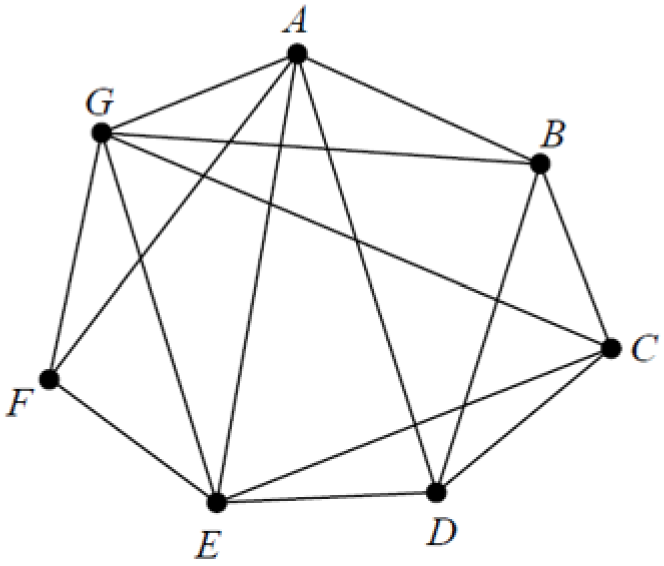


(2) **Step 1**

Referring to the diagram above, we first of all find a Hamiltonian cycle; for example ABCDEFGA.

(3) **Step 2**

We then draw the polygon ABCDEFGA (ie with the nodes of the Hamiltonian cycle in the correct order), and add in the remaining arcs of the original graph, in such a way that they lie within the polygon (see diagram below).



(4) The idea behind the algorithm is that, if the graph is planar, then it will be possible to divide the interior arcs into two groups, such that the members of each group do not cross each other. Then the arcs of one of the two groups (it doesn't matter which) can be redrawn outside the polygon, and we will have obtained a graph that is isomorphic to the original graph, with no arcs crossing.

Note that if two interior arcs cross then they will also cross if they are redrawn outside the polygon, and if they do not cross inside the polygon, then they will also not cross outside the polygon.

As an alternative to completing the algorithm then, we could group the interior arcs as follows (by inspection):

AF, AE, AD, BD

GB, GC, GE, EC

[Obviously this wouldn't be an option if an exam question specifically asked for the Planarity Algorithm to be applied.]

(5) **Step 3**

Create a list of the interior arcs (in any order).

For example: AD, AE, AF, BD, BG, CE, CG, EG

Give the arc AD the label I_1, to indicate that we are allocating it to the group of arcs that will stay inside the polygon (I will suffice for exam questions, but the subscript helps to understand the method). [Any non-standard notation used in an exam answer would need to be defined.]

(6) **Step 4**

Of the remaining arcs in the list, we now consider those that cross AD. They will have to be allocated to the outside group, as AD belongs to the inside group. These arcs are BG, CE & CG.

We need to see whether any of these arcs cross each other. If this is the case, then it will not be possible to divide the interior arcs into two groups, such that the members of each group do not cross each other, and the graph is established to be non-planar.

BG, CE & CG do not cross each other, and so they can be labelled O_1 (or just O).

(7) **Step 5**

The remaining arcs in the list are AE, AF, BD & EG, and of these AE, AF, BD cross one or more arcs in O_1. This means that they will have to be allocated to the inside group (note that we don't have to worry about them crossing the I_1 arc (AD), as any such arcs have already been removed from the list in Step 4).

As AE, AF, BD don't cross each other (otherwise the graph would be established as non-planar), they can be labelled I_2.

(8) **Step 6**

The only remaining arc in the list is EG, and this crosses one of the arcs in I_2 (namely AF). And so EG is labelled O_2 (once again, it won't cross any arcs in O_1, as such arcs were removed from the list in Step 5).

Thus we have successfully divided the arcs in our list into two groups: the inside arcs (AD, AE, AF & BD) and the outside arcs (BG, CE, CG, EG) - though they could of course be grouped the other way round. And so we have established that the graph is planar.

(9) If the arc AF had not existed, then the arc EG is found not to cross any of the arcs in I_2 (as well as any of the arcs in I_1 or O_1). This means that it could be allocated to either the inside or the outside group, but for simplicity the algorithm takes us back to the start, and treats arc EG in the same way as it treated arc AD, giving it an I label. (Had there been more arcs in the list, the algorithm would then have continued.)

(10) The best check that the algorithm has been applied correctly is to redraw the graph with arcs placed in their allocated groups, to demonstrate that no arcs cross.

(11) **Exercise**: Apply the algorithm to the same example, but starting with the arc EG.

Solution

I_1: EG

O_1: AF

I_2: BG & CG

O_2: AE, AD & BD

I_3: CE