Exercises: Bipartite Graphs

Questions

For all questions, assume we are talking about undirected graphs.

1) Suppose you have a connected graph and you do depth first search (DFS) from an arbitrarily chosen vertex. This defines a DFS tree. There may be edges in the graph that do not belong to the DFS tree. I claimed in the lecture that non-tree edges always join a vertex to its ancestor (or descendent). Why?

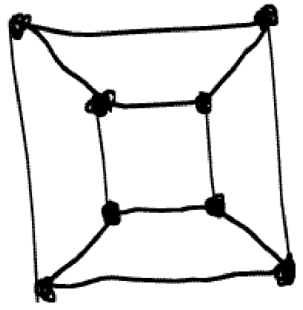
2) Show that a graph is bipartite if and only if does not have an odd cycle. (Note the "if and only if". The proof needs to go both ways.)

3) In class I showed how to test if a connected undirected graph is bipartite. Is the "connectedness" condition necessary? What happens if the graph is not connected?

4) You could use breadth first search (BFS) instead of DFS to test if a connected graph is bipartite. Run BFS and color the edges in the BFS tree red and black, same as in the DFS case, namely even levels red and odd levels black. Again, we only need to test the bipartite condition on **non-tree edges**, since the edges in the BFS tree automatically satisfy the bipartite property. That is, we test if the vertices of each non-tree edge have opposite color.

   More generally, what can we say about the levels of non-tree edges in a BFS tree? That is, if we don't label the vertices red and black but instead we label the vertices with levels, how can we characterize when a graph is not bipartite, from looking that the non-tree edges? Note that the ancestor-descendent property that we saw in DFS does not hold for BFS.

5) Draw at least three different perfect matches for the following graph.



6) Suppose there are n men and n women. How many different instances can there be of the stable marriage problem, that is, how many different sets of rankings are possible?

7) Run through at least one example here (http://mathsite.math.berkeley.edu/smp/smp.html) to make sure you understand the Gale-Shapley algorithm for solving the stable marriage problem (SMP). See also Q9 below.

8) We showed that the Gale-Shapley algorithm yields the match { (α, bestvalid(α)) : for all α in A }. Define worstvalid(β) to be the α in A that β prefers the least, among all valid edges (α, β). Show that the Gale-Shapley algorithm finds the worst valid matching for each of the elements of B, that is, it finds the matching { (worstvalid(β), β) : for all β in B }.

   Hint: Suppose the opposite. That is, there is some β that gets matched to an α that is different than worstvalid(β). Can you derive a contradiction?

9) [added March 3] Consider an instance of the "stable marriage" problem in which, for all α in A, α 's first choice is β if and only if β 's first choice is α. In this case, there is only one stable matching. Why?

10) [added March 3] Given the preferences shown here, use the Gale-Shapley algorithm to find a stable matching.

A	A's preferences		B	B's preferences
$\alpha1$	$\beta1$ $\beta2$ $\beta3$		$\beta1$	$\alpha3$ $\alpha1$ $\alpha2$
$\alpha2$	$\beta1$ $\beta2$ $\beta3$		$\beta2$	$\alpha1$ $\alpha3$ $\alpha2$
$\alpha3$	$\beta1$ $\beta2$ $\beta3$		$\beta3$	$\alpha3$ $\alpha2$ $\alpha1$

Answers

1) Starting from an arbitrary vertex, run DFS to get a DFS tree. Suppose that there are two vertices v and w in the graph that are connected by a non-tree edge in the DFS tree, and that the vertices do not share a common path to the root. Let v be the vertex visited first in the DFS tree. But wait, if there is an edge (v,w) in the graph, then w must be reachable from v (maybe via multiple paths), so either w is a child of v in the DFS tree or else w is a descendent of v. DFS won't return from v until all vertices that are reachable from v have been reached.

2) In class I sketched why a bipartite graph cannot have an odd cycle, that is, all cycles are even. Color the vertices of the cycle red and black, starting at red (and alternating). If its an odd cycle, then you'll end up with two red vertices connected by an edge. The graph cannot be bipartite in that case because, if you just look at the vertices and edges in that cycle, then there is no way to partition those vertices such that the definition of bipartite graph is satisfied.

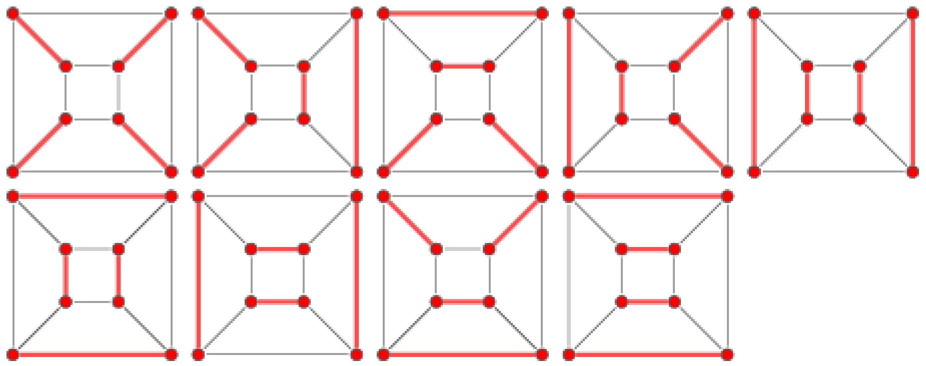
   Let's go in the other direction. Suppose we have graph that doesn't have any odd cycles. Is such a graph always bipartite? Yes. Pick a vertex in the graph and do a depth first search, and color the vertices red (even depth) and black (odd depth). This gives you a depth first search tree, plus a set of non-tree edges that connect vertices in an ancestor-descendent relationship. That's it. That's all the edges. Any simple cycle (non-repeating vertices or edges) in the graph must be defined by a path from ancestor to descendent, plus a single non-tree edge from descendent to ancestor. The path from ancestor to descendent must have an odd number of edges (since we are supposing the cycle is even) and so the label of the ancestor must be different from that of the descendent. E.g. red-black-red-black is three edges. Thus, any non-tree edge has vertices with different colors. That's all we need to show.

3) No, the "connectedness" condition is not necessary. If the graph is not connected, then just run the test on the connected components. The test needs to hold on each of the connected components. Why? If the connected components are bipartite then the whole graph is too since, for connected component k, you would have Ak and Bk, and so you can see the whole graph is bipartite by grouping {Ak} and {Bk}. Similarly, if one of the connected components is not bipartite, then it should be obvious that the whole graph cannot be bipartite.

4) Edges in the graph are either of the form (parent, child) or else the vertices in the edge are siblings, i.e. they have the same parent. Why? Think of how BFS works. It starts at an arbitrary vertex and then visits all nodes that can be reached by a path of length 1. Then it visits all vertices that can be reached by a path of length 2. (The mechanism for doing so uses a queue as you learned in COMP 250 and as I reviewed in the lecture 7 before Dijkstra.) If there is an edge (v,w) in the graph, then assume without loss of generality that BFS reaches v before w. If the edge (v,w) is also in the BFS tree, the obviously v is the parent of w. But if the edge (v,w) is a non-tree edge, then all we can say is that v and w are either in the same layer or else w is one layer deeper than v.

In particular, if there is a non-tree edge between two vertices in the same layer, then these vertices have the same color. Note that there must be an odd cycle. Why? Consider the path from each of the two vertices to their first common ancestor. These paths are the same length, and there is an edge between the two vertices, which gives an odd cycle. By similar reasoning, if the two vertices are in neighboring layers, then they have opposite colors and they belong to an even cycle in the graph.

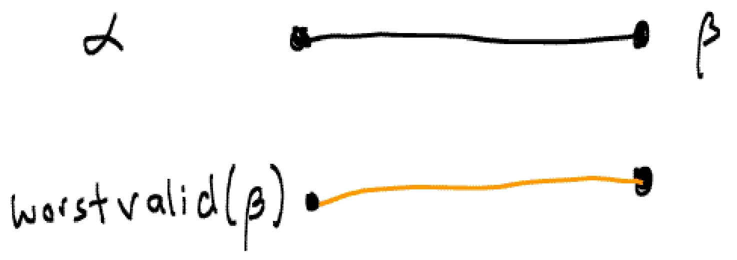
5)



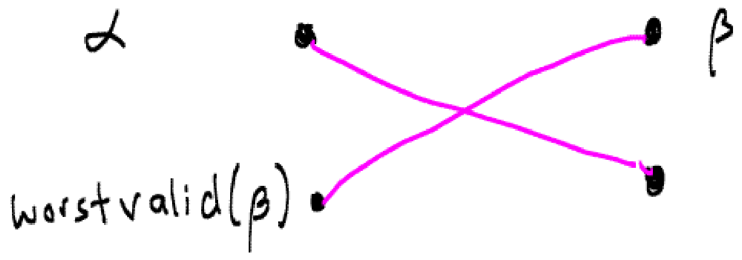
6) Each man has n! possible rankings of the women. There are n men and there are no constraints on the rankings of the different men. So there are (n !)^n possible rankings for the men (of the women). The same holds for the women. There are (n !)^n * (n !)^n = (n !)^2n rankings in total. For large n, this is a huge number.

7) No solution given here. See URL in question.

8) Suppose that the (stable) matching found by the algorithm is such that some β gets matched to some α that is different than worstvalid(β). Note from the claim proven in class that β = bestvalid(α).



Next, since (worstvalid(β), β) is valid, there must be another stable matching (not found by the algorithm) that has that edge (worstvalid(β), β) :



But if this second matching were a stable matching, then (by definition of a stable matching) it could not be the case that α and β prefer each other over their partners i.e. the pink lines. But this contradicts what we know about the first matching, namely that β does prefer α over worstvalid(β), by the definition of worstvalid(β), and that α does prefer β over any other valid match because β = bestvalid(α).

[If you are reading this line and you've understood the proof, then congratulations ! You are probably asking yourself, would you be expected you to reproduce that proof in an exam situation? Answer: no. But now that you understand it, so you'll be ready for the (more reasonable) question that I will ask you.

9) By definition, a matching is unstable if there exists an α in A and a β in B that prefer each other over their present partners. Take any matching for which some α in A doesn't get its first choice. Let the first choice of α be β. Then β is not getting its first choice either, since β 's first choice is α. But then this matching is not stable. Thus, for any stable matching, every α gets its first choice, and so every β (by the constraints given in this question) gets its first choice too.

10) The answer is (α1, β2), (α2, β3), (α3, β1).