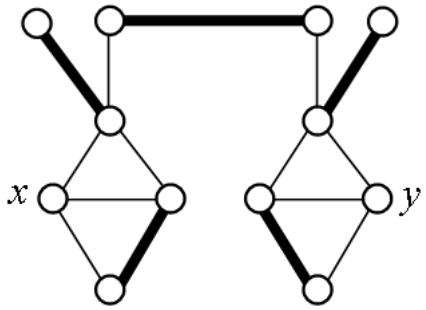
Chapter 9. Section 1

Augmenting Paths in Non-Bipartite Graphs

The theorem given in Section 3, Chapter 8, says that an M-augmenting path will improve a matching and that this is the *only* way to improve a matching: no augmenting path, no bigger matching.

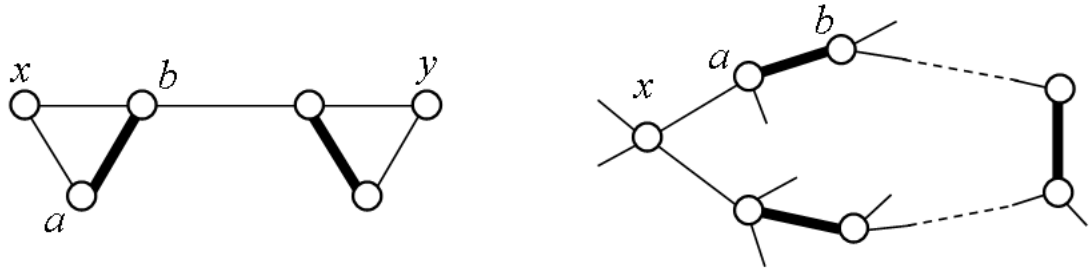
So you might think: Simple! Just check if there is an M-augmenting path exists; if there is, use it; if there is not, declare M to be maximum.

The difficulty is, it is not so easy to decide how to look efficiently for M-augmenting paths. Consider the graph below: it has 12 vertices and the matching M shown has only 5 edges, so a further improvement is worth looking for. Indeed, there are 2 M-unsaturated vertices, x and y, which is just what is required to begin and end an M-augmenting path. But no such path exists: there is no perfect matching in this graph.



Suppose, instead of 2 M-unsaturated vertices there were many? Maybe 1 in every 10 vertices is M-unsaturated: we must check each pair to see if an M-augmenting path joins them! That is about $n^2/100 = O(n^2)$ pairs, if $|V(G)| = n$; for each pair we might check about $|E(G)| = O(n^2)$ edges. Eventually we find an M-augmenting path: this improves the matching by just one edge; now we must repeat the whole process. This is beginning to suggest an algorithm with a running time of $O(n^2 \times n^2 \times n) = O(n^5)$.

Can we not apply our tree-growing algorithm approach to build a tree from an M-unsaturated vertex, adding edges which alternate between not-M and M: i.e. an *M-alternating tree*? Not so easy! In the graph below left there *is* an M-augmenting path from x to y. But if we branch out from x to look for it, starting with the non-M edges xa and xb, we are immediately stuck — no M edge is available to extend our tree.



Rather than branching out we could try following one forward path at a time. Again there are difficulties: in the graph above right we might follow a long M-alternating path (e.g. the one starting x, a, b, only to find we have returned to our starting vertex!

In fact, there is a famous algorithm, due to Jack Edmonds, which solves non-bipartite matching in $O(n^4)$ time; and further improvements are possible. However, we now observe that the key ingredient in the above 'difficult' graphs is the presence of odd-length cycles. If we are guaranteed that all cycles are even length (so that is, bipartite graphs) then our tree-growing method becomes effective.