

Input: Undirected unweighted Graph G .

Output: Set of edges with no two edges incident on the same vertex.

Bipartite Matching is a special case that you studied last semester and more recently using flows. Some ideas from the bipartite case carry over.

An important notion there was that of an augmenting path with respect to a matching M . This is simply a path $u_0, u_1, \dots, u_{2k+1}$ in G where edge $(u_{2i}, u_{2i+1}) \notin M$, edge $(u_{2i-1}, u_{2i}) \in M$, and u_0, u_{2k+1} do not have any edge from M incident on them.

Theorem 1 *A matching M is maximum in a graph G iff there is no augmenting path with respect to M .*

Proof: ($\Rightarrow$) If there is an augmenting path $u_0, \dots, u_{2k+1}$, then we move edges (u_{2i}, u_{2i+1}) out of M , and edges (u_{2i-1}, u_{2i}) into M . This movement is called augmentation. At any of $u_1, \dots, u_{2k}$ exactly edge in the matching was incident before the move, and also after the move. At u_0, u_{2k+1} , no matching edges were incident, and after the move 1 matching edge is incident. The number of incident edges on other vertices has not changed as a result of the move. Hence, M continues to remain a matching. But we removed k edges and added $k + 1$. Hence M could not have been a maximum matching before the move.

($\Leftarrow$): Suppose there is no augmenting path for M and yet M is not maximum, but another M' is. Consider the symmetric difference $M'' = M \oplus M'$, which is the set containing edges which are either in M or in M' but not in both. Note that M'' contains more edges from M' than M , since equal number from both drop out because they are common.

M'' has degree at most 2, since it contains edges from degree 1 graphs, M, M' . Thus it must be made of paths and cycles. If there is any cycle, then the alternate edges on the cycles must be from M, M' . So there are equal number of edges from M, M' in the cycles. Somewhere the extra edges from M' must show up. So there must be a path P in which there are more edges from M' . Even on this, the edges from M, M' must alternate (since M, M' were degree 1 graphs). Thus P must start and end with an edge of M' . But these are edges which are not in M and the starting and ending vertices of

P are not incident to any edges of M . Hence P is an augmenting path for M . ■

Thus a natural strategy for finding maximum matchings is to look for augmenting paths, and use them to grow the matching. When there is no augmenting path, the matching we have is maximum, as the above lemma assures us. In case of bipartite graphs, a BFS or DFS from every unmatched vertex is enough to find an augmenting path if one exists. Unfortunately, natural implementations of BFS or DFS may not find augmenting paths in the general case. As an example, consider a graph on vertices $1, \dots, 10$, in which we have edges $(i, i+1)$ for $i = 1, \dots, 9$ and also an edge $(3, 7)$. Suppose we currently have the matching $M = \{(2, 3), (4, 5), (6, 7), (8, 9)\}$. Clearly, the path $1, 2, \dots, 10$ is augmenting. However, a search for augmenting paths may proceed by including edges $(1, 2), (2, 3), (3, 7), (7, 6), (6, 5), (5, 4)$ after which the search will backtrack to 7 and stop. From 7 it will not continue in the direction $(7, 8)$ because the previous edge $(3, 7)$ is not in the matching, and we need a matching edge next which $(7, 8)$ is also not.

It may be said that the problem arises due to the presence of the odd cycle $3, 4, 5, 6, 7, 3$. Indeed, if the short cut edge produced an even cycle, say it was from 3 to 8 (instead of to 7), the above search would find an augmenting path (either the path $1, 2, 3, 4, 5, 6, 7, 8, 9, 10$ or the path $1, 2, 3, 8, 9, 10$).

How to deal with odd cycles indeed turns out to be a key problem, which we solve next. Based on it, an algorithm can be constructed.

Suppose we have a matching M and an odd cycle B consisting of the vertices $u_0, \dots, u_{2k}$ in which only the edges u_{2j-1}, u_{2j} for all j belong to M . Then the cycle is customarily called a blossom with respect to M . The vertex u_0 which has only non matching edges from B incident on it is called the base. The origin of this term will become clear shortly. The key idea is to shrink a blossom into a point and recursively solve for the remaining graph. The solution for the remaining graph can then be extended to the original graph.

Lemma 1 *Let M be a matching in G and B a blossom $u_0, \dots, u_{2k}$, with base u_0 . Suppose that no edge of M is incident on u_0 even from the edges of M outside of B . Let G' be obtained by contracting B into a single vertex b . Let M' be the matching induced in G' by M . Then M is optimal for G , iff M' is optimal for G' .*

Proof: Suppose M is not optimal. Then there must exist an augmenting path P for M in G . If P misses B , then it must be present in G' as well and must miss b . Thus it will be an augmenting path for M' as well. So assume that P does intersect B . P cannot have both endpoints in B since B only has one unmatched vertex. So it must start at some vertex v outside B . Now, after contracting B the P must pass through the unmatched vertex b . Thus the portion v to b itself forms an augmenting path for M' . Thus M' is not optimal.

Suppose that there is an augmenting path P' for M' . If this path misses the contracted vertex, it is an augmenting path for G . So assume that P' touches the vertex b . Since b was an unmatched vertex, it must be an endpoint of P' . Say the other endpoint of P' is v . In G' suppose P' enters b from a vertex u . In G , suppose u is adjacent to vertex u_{2i} . Then the portion from v to u together with the path $u_{2i}, u_{2i-1}, \dots, u_0$ forms an augmenting path. If u is adjacent to u_{2i+1} , then v to u followed by the path $u_{2i+1}, u_{2i+2}, \dots, u_{2k}, u_0$ is augmenting. ■

1 Algorithm

We will use M to denote the matching under construction. Initially $M = \phi$. The algorithm repeatedly calls the following procedure *FAP* (find augmenting path) and augments M , until in some call no path is found.

In each execution of *FAP*, we grow a forest F . Initially, F consists of only all unmatched vertices. The vertices of F are classified depending upon their distance from the unmatched vertices as being either even or odd. As *FAP* executes, F grows, maintaining the invariant that every odd vertex has a unique child to which it connects by a matching edge, and every even vertex can have many children to which it connects using only non-matching edges.

Procedure *FAP*(G, M) {

F = set of unmatched vertices in G .

While there exists an even vertex $u \in F$ with an unexplored edge (u, v) , do:

1. If v is not in F then include (u, v) in F . Also include $(v, w) \in M$ in F . Since v is not unmatched, such a (v, w) must exist.

2. If v is an odd vertex in F , then consider (u, v) to have been explored, but otherwise do nothing.
3. If v is an even vertex in F and u, v are in different subtrees in F with roots r, s , then we have discovered an augmenting path $r - u - v - s$. Return this path.
4. If v is an even vertex in F and u, v are in the same subtree of F then we must have discovered an odd cycle. Let x be the common ancestor of u, v and r the root of the subtree. Vertex x must be even, since it has two children. The path $r - x$ is the stem, and the cycle $x - u - v - x$ is a blossom B with x the base¹. Let M_1 denote the set of edges on the path $r - x$ that are in M . Let M_2 denote the remaining edges. Now we set $M' = M \cup M_2 - M_1$. Clearly M' is a matching for G and $|M'| = |M|$. In M' , x has now become unmatched. Thus the stem $r - x$ together with the blossom B satisfy the premise of Lemma 1. So we shrink B , as a result of which we get a new graph G'' and the induced matching M'' obtained from M' . We recursively call $\text{FPA}(G'', M'')$.
 - (a) If this returns a path P , then as per Lemma 1 we can construct a path P' which is augmenting for G, M' . Now an exercise shows that you can get a path P'' which is augmenting for G, M which is returned.
 - (b) If no path is returned, then we know that M'' must be maximum for G' and hence M, M' for G . In this case we return null.

End while.

Return null.

}

It only remains to prove that if FAP returns null from the last statement of the algorithm, the matching M is indeed optimal. To see this, consider the forest F at the time of return. Let p, q denote the number of odd and even nodes in F . The size of the matching as restricted to F is p , since every odd vertex has a matching edge to its child. Note now that all the edges from G incident on F in fact have an odd vertex as one of the endpoints. Since there are no edges in G incident only on even vertices, the even vertices must get matched only to odd vertices. Thus it follows that there can be at

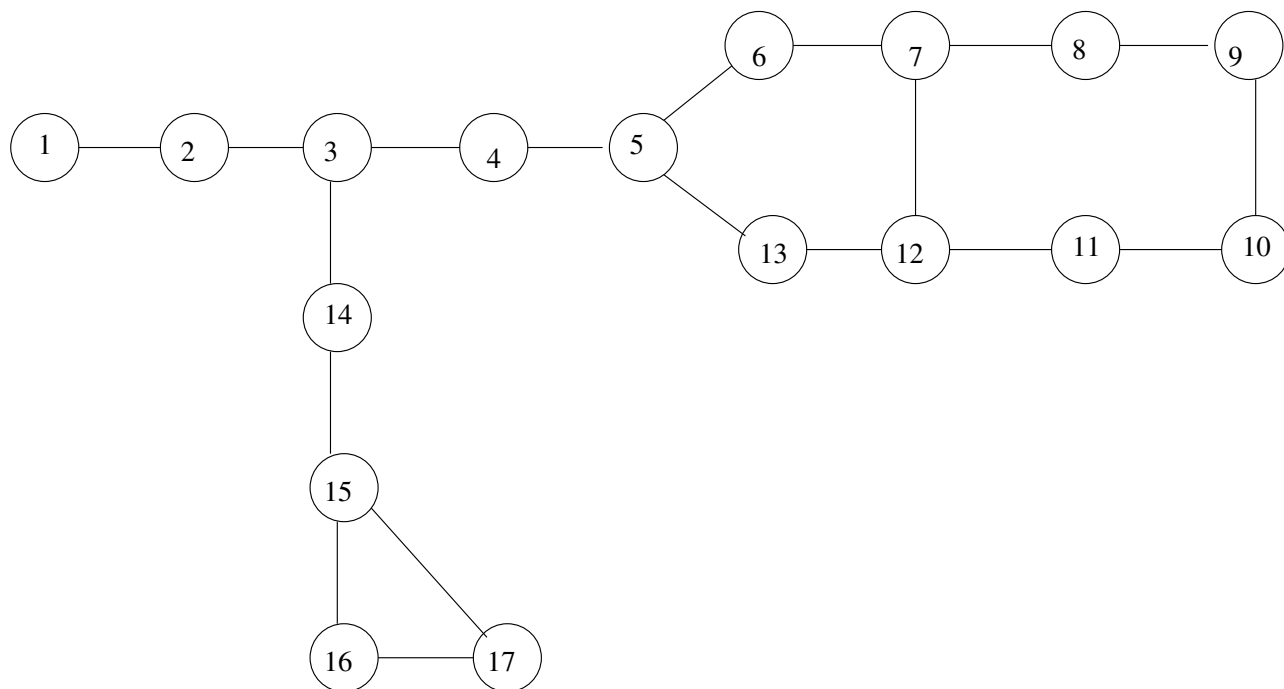
¹With some imagination, you can now see the origin of the terms stem and blossom.

most p even vertices that can be matched, and at least $q - p$ must remain unmatched, no matter what matching we use. But, in fact p of the q vertices are matched, and $q - p$ are unmatched. Thus M is maximum.

The time taken is easy to estimate. We can find at most n augmenting paths. Further, between the consecutive discoveries of augmenting paths, we may contract some blossoms. But in each contraction, the number of vertices reduces. Hence there can be at most n consecutive contractions between consecutive augmentations. Each execution of FAP, ignoring the recursion, either causes an augmentation, or a contraction. Thus there can be at most n^2 such executions. Each execution, ignoring the recursion, takes time $O(m)$. Hence the total time is $O(mn^2)$.

Exercises

1. Execute the algorithm above to find a maximum matching for the graph shown. To make the execution deterministic, examine vertices in increasing order. Show the smallest graph, i.e. after all possible contractions, produced during the execution.



2. In this you are to implement the procedure required in step 4(a) of the algorithm. Specifically, let M' be a matching in a graph G . Let $s_0, s_1, \dots, s_{2j}$ be a stem, and $u_0, \dots, u_{2k}$ be its blossom, with $s_{2j} = u_0$ the base. Of these edges, only edges (s_{2i}, s_{2i+1}) and edges (u_{2i-1}, u_{2i}) are in M' for all i . Suppose there exists an augmenting path P' which starts at the base. Consider now a matching M which is same as M' , except that it includes edges (s_{2i-1}, s_{2i}) and excludes edges (s_{2i}, s_{2i+1}) for all i . Show how to find a path P'' for M .

Suggested strategy: Draw the picture of G and M' . Show P' in it, but note that you only know it starts at u_0 . Let v be the other endpoint of P' . Argue that v cannot lie on the blossom. Starting from v let P' touch the stem first at some s_x . What are the acceptable values for x ? Now draw on your picture the matching M (use pencil + pen combination to keep the diagram understandable) and see if you can spot an augmenting path for it.

3. In a graph $G = (V, E)$, $V' \subseteq V$ is said to be a vertex cover if every edge is incident on some vertex in V' . By minimum vertex cover we mean a vertex cover of the smallest possible size.
 - (a) Show that in any graph (not necessarily bipartite) the size of the minimum vertex cover is no smaller than the size of the maximum matching.
 - (b) Prove Konig's Theorem: In any bipartite graph, the size a maximum matching equals the size of a minimum vertex cover. Hint: The proof at the end of Section 1 contains a proof of this theorem!
4. In the proof at the end of Section 1 we glossed over some details. Is it possible that the contraction might lead to self loops or parallel edges? Show that these can happen, but do not affect the final result.