6 Maximum Matching

Consider an undirected graph $G = (V, E)$.

Definition: A *matching*, M, of G is a subset of the edges E, such that no vertex in V is incident to more that one edge in M.
Intuitively we can say that no two edges in M have a common vertex.

Maximal Matching: A matching M is said to be *maximal* if M is not properly contained in any other matching.
Formally, $M \not\subset M'$ for any matching M' of G.
Intuitively, this is equivalent to saying that a matching is *maximal* if we cannot add any edge to the existing set.

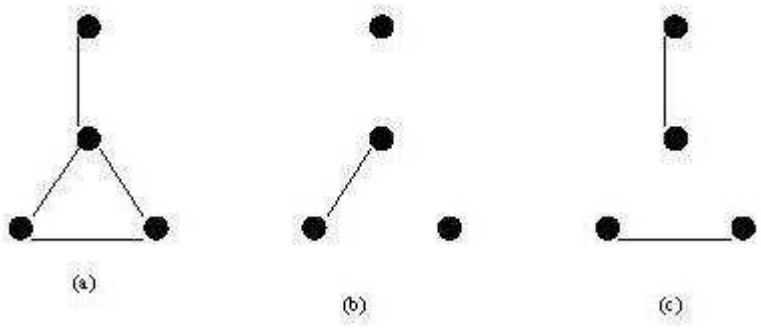
Maximum Matching: A matching M is said to be *Maximum* if for any other matching M', $|M| \geq |M'|$.
$|M|$ is the maximum sized matching.

Theorem 1 *If a matching M is maximum $\Rightarrow$ M is maximal*

Proof: Suppose M is not maximal

$\Rightarrow$ $\exists M'$ such that $M \subset M'$
$\Rightarrow$ $|M| < |M'|$
$\Rightarrow$ M is not *maximum*
Therefore we have a contradiction.

The converse of the above is not true. We see this using the counter example below:



(a) is the original graph.
(b) is a *maximal* matching but not the *maximum* matching (c)

Maximal matching for a given graph can be found by the simple greedy algorithn below:
Maximal Matching(G, V, E)
1. $M = \phi$
2.While(no more edges can be added)
    2.1 Select an edge,e,which does not have any vertex in common with edges in M
    2.2 $M = M \cup e$
3. return$M$

6.1 Augmenting Paths

We now look for an algorithm to give us the maximum matching.
Definition: Matched Vertex: Given a matching M, a vertex, v is said to be matched if there is an edge $e \in M$ which is incident on v.

Definition:Augmenting Path: Given a graph, $G = (V, E)$ and a matching $M \subseteq E$ a path P is called an augmenting path for M if:
1. The two end points of P are unmatched by M.
2. The edges of P alternate between edges $\in M$ and edges $\notin M$.

Theorem 2 (Berge's Theorem) *A matching M is maximum iff it has no augmenting path.*

Proof : We first prove the forward implication, M is maximum $\Rightarrow$ M has no augmenting path.

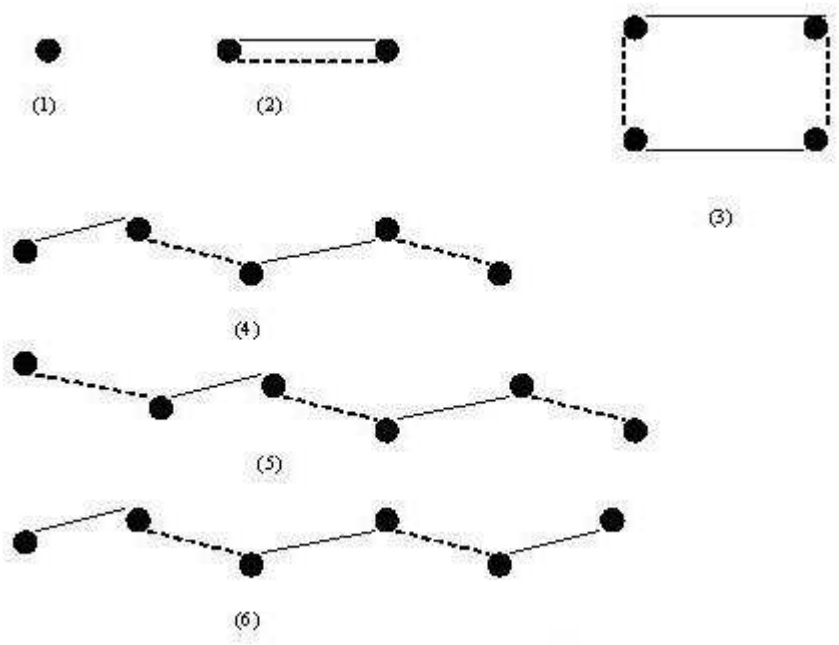
If M has an augmenting path,P, then switch the edges along the path P from in-to-out of M and vice-versa. (We will represent this switching operation as $M \oplus P$).
This gives a matching with one more edge. Thus, M could not have been maximum.

Now we prove the reverse implication, no augmenting path $\Rightarrow$ M is maximum
By contrapositive, $\equiv M$ is not maximum $\Rightarrow$ M has an augmenting path

Let M' be a matching such that $|M'| > |M|$
Consider the "sub-graph" of G induced by $M \cup M'$, including the edges that appear in $M \cap M'$ twice.
Let H be this multi-graph.
$\forall$ vertex $v \in H$, $deg_H(v) \leq 2$ (Because any vertex has atmost 2 edges one from M and one from M' incident on it.)
Therefore, any connected component of H must be either a path or a cycle. The cycles preent in this graph must be of even length because otherwise two edges of the same component M or M' will be adjacent to each other and this will violate the assumption that the original set M (or M') was a matching.
There are 6 different possiblities for the components of H. which are listed below (and shown in the figure):
The components above can be classifies as follows:

1. An isoated vertex.
2. A 2-cycle
3. A larger alternating cycle of even length.
4. An even length path
5. An odd length path starting (and ending) with a vertex in M'.
6. An odd length path starting (and ending) with a vertex in M.

The solid edges represent the edges from M and the dashed edges represent the edge from M'



Now, we know that $|M'| > |M|$ so there must be more dashed edges than solid edges.
Of the above components only 5 and 6 have different number of dashed and solid edges (different no. of edge from M and M').Also 5 is the component which has more edges from M' than from M.
For $|M'| > |M|$ atleast one component of type 5 must be present. (Infact more components of type 5 than 6 but we can do with the weaker statement above).

However, even if one such component is present we have an augmenting path w.r.t. M. Thus we have shown that if M is not maximum then there is an augmenting path w.r.t. M.

Note: The maximum matching for a graph need not be unique.

Using the above result we can write the ouline for an algorithm to find the maximum matching.

Maximum Matching (G, M)
1. $M \leftarrow \phi$
2. While ($\exists$ an augmenting path P)
   2.1 $M \leftarrow M \oplus p$
3. return M

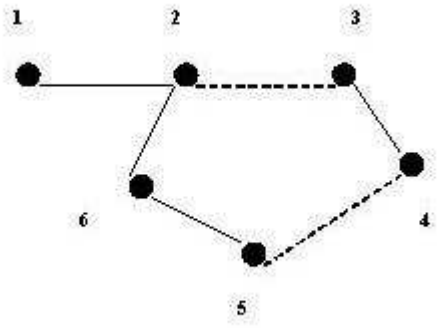
For the above algorithm we need an algorithm to find an augmenting path. In the sections to follow we will write a rough outline for such an algorithm. This will lead us to two cases which need separate handling.
1. The problem for bipartite graphs.
2. The problem for a general graph.

In the subsequent sections we will handle those problem individually

6.2 Intuitive idea for finding the Maximum Matching in a graph

In this section we look at a very simple idea to obtain a maximum matching in a graph G. As we see later the algorithm does not work in the general case. The idea, however, can be modified to treat cases separately.

Idea: With an unmatched vertex as the root start a DFS. Keep track of the distance of any node from the root as we perform the DFS. If the current node is at an odd distance use an edge $\in M$ to continue the DFS else use an edge $\notin M$. If at any point we find an unmatched vertex adjacent to a vertex at an even distance we have an augmenting path.
Intuitively, the idea is good because starts with an unmatched vertex and alternates between edges in M and not present in M.
This approach however does not guarentee that an augmenting path will be found if there exists one. Consider the following graph. Lets start the DFS with 6 as the root node.
If we take the following path for the DFS: 6-5-4-3-2-1 we find an augmenting path.
However, we miss the augmenting path if we take the following order for the DFS: 6-2-3-4-5.



The problem case for this method is the presence of odd cycles in a graph. In an odd cycle there will be two unmatched edges that share a vertex. If the DFS is started from that vertex and the cycle is traveresed the "wrong " way we will miss the augmenting path.
To curcumvent this problem, in the next section we assume that G has no odd cycles i.e. G is bipartite. The case for a general graph will be considered in the subsequent section.

6.3 Maximum Matching for a bipartite graphs

We assume $G = (V, E)$ has no odd cycle i.e. G is bipartite. Then we can divide V into two partitions,L and R such that $\forall(u, v) \in E$,$u \in L \wedge v \in R$.

Then the previous algorithm can be modified as:
Bipartite Matching(G, M)
1. Start DFS at a vertex in L.
2. If current vertex is in L
follow an edge,$e \in M$
else
follow an edge, $e \notin M$

If at any point we find an unmatched verted $\in R$ then augmenting path is found.

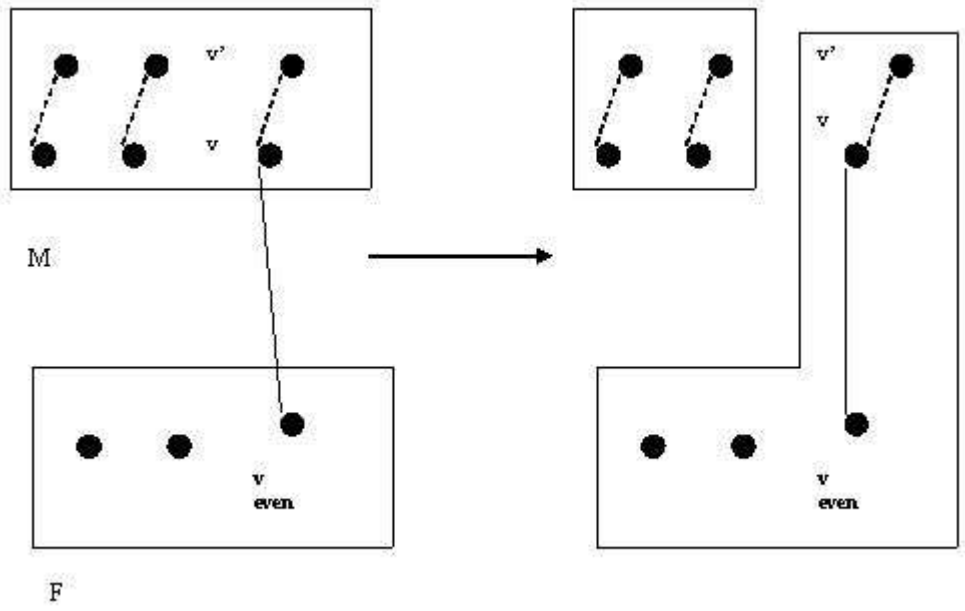
Analysis: $|V| = n, |E| = m$.
Time = (no. of iterations) $\times$ (time per iteration)

$= \frac{n}{2}.m$
$= O(mn)$

For efficiency reasons we improve on the algorithm presented previously.
Let $U \subseteq V$ be the set of unmatched vertices in G for a given matching M. Initially we label all the vertices in U as "even". We can regard U as a forest, F_u. We grow this forest, with every vertex of U in a component, as follows:

1. Find an edge connecting an "even" vertex in F_u to a matched vertex v outside F_u. Let v' be the neighbour of V in M.
2. $F = F \cup (u, v) \cup (v, v')$. Label v as "odd" and v' as "even".

This can be seen more clearly in the figure below.



As can be seen this graph grows as an alternating cycle.
Ultimately, we will come to a stage where we cannot add anything more to any component. At this stage if there are edges present between two "even" edges we have an augmenting path. Also these two edges must belong to different components of the forest as G has no odd cycles.
Summing up we have the following algorithm for finding a maximum matching.

Forest Growing Matching(G)
1. Randomly selected any matching, M.
2. do
   2.1 Grow forest as above, labelling the vertices "even" / "odd"
   2.2 Find all such even - even edges to obtain a maximally disjoint set of augmenting paths $(P_1, P_2, \ldots, P_k)$.
   2.3 $M \leftarrow M \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_k$
   while($\exists$ an augmenting path)

Analysis We need to run atmost $O(\sqrt{n})$ iterations and each iteration takes $O(m)$ time.
Thus the running time of this algorithm is $O(m\sqrt{n})$.

6.4 Maximum Matching for general graphs

Definition: Blossom: A blossom is defined as a cycle of odd length ($2k + 1$ edges) with k matching edges.

Lemma 1 (Cycle Shrinking Lemma):[1] *Let M be a matching of G and B be a blossom. Further, assume that B is vertex-disjointed from (i.e. has no vertex in common) with the rest of M. Consider the graph G' obtained by contracting B to a single vertex. Then the matching M' of G' induced by M is maximum in G' iff M is maximum in G.*

Proof: M is maximum $\Rightarrow$ M' is maximum
Suppose M' is not maximum in G'. Then there is an augmenting path P' w.r.t. M' in G'. Suppose P' does not intersect the blossom,B, in G, then P' is also an augmenting path in G and hence M is not maximum.
So P' must intersect B in G. In particular, the contracted blossom, B', must be an end point of P' in G' since B is vertex-disjointed from M (and so otherwise there will be two consecutive edges in $P' \notin M'$).
Let P' meet B at the vertex v, and let u be the unmatched vertex in the blossom [2].
Let P'' be the path from v to u in the blossom that begins with the matching edge incident to v. Then, $P = P' \cup P''$ is an augmenting path in G and so, again M is not a maximum matching.

[1] The definition and some parts of the proof are borrowed from Sanosh Vempala's notes
[2] In a blossom (vertex-disjointed) there will be exactly one unmatched vertex

Thus, if M is maximum then M' is maximum.

Now consider the converse, M' is maximum $\Rightarrow M$ is maximum.
If M is not maximum then by Berge's theorem there must be an augmenting path P in G.
Following a similar argument as above we assume that P intersects the blossom B.
Since B contains only one unmatched vertex atleast one end-point of P must lie outside B. Let w be this point.
Consider the path P' obtained by starting at w and following P' until it intersects the blossom. P' is an augmenting path in G' and thus M' could not have been maximum.
Hence, we have the proof of the lemma.

To find an augmenting path in a general graph we detect and contract any blossom we find with the knowledge that by the cycle shrinking lemma the maximum matching in the shrunk graph will give us the maximum matching in the original graph.
To obtain a maximum matching we use the same forest growing algorithm as for the bipartite graphs. After growing the forest we consider the neighbourhood of the "even" vertices. The following cases come up:
1. If two "even" vertices from different components are adjacent, then there is an augmenting path between the roots of the components.
2. If two "even" vertices from the same component are adjacent then there is a blossom ,B, that is present. Let P be the path connecting B to the root. We get an equivalent sized matching from $M \leftarrow M \oplus P$. After this B is vertex disjointed with M and we can apply the cycle-shrinking lemma to it. So we shrink B and obtain a new graph G' and we start all over again on G'.
3. If every "even" vertex is adjacent only to "odd" vertices we already have a maximum matching. To see this, let p be the odd vertices and q be the "even" vertices. Then there are $(q-p)$ components in the forest. Any mathcing will have atleast $q-p$ unmatched vertices and so the matching we have must be maximum.

Thus summing it all up we have an algorith for maximum matching for a general graph.

Maximum Matching(G)
1. Randomly selected any matching, M.

2. do
    2.1 Grow forest as described, labelling the vertices "even" / "odd"
    2.2 if (there is a blossom in the graph)
        shrink the blossom to obtain a new graph G'
        continue
    else
        Find all such even - even edges to obtain a maximally disjoint set of augmenting paths $(P_1, P_2, \ldots, P_k)$.
    2.3 $M \leftarrow M \oplus P_1 \oplus P_2 \oplus \cdots \oplus P_k$
  while($\exists$ a blossom $\vee$ $\exists$ an augmenting path)
3. Expand all blossoms to obtain the maximum matching in the original graph

Analysis: We can have$O(n)$ augmenting paths so there are $O(n)$ iterations of the loop in the forest growing algortihm.
For every augmenting path we find we can have $O(n)$ blossoms to shrink.
At each level it takes $O(m)$ work to identify a blossom or an augmenting path.
Therefore, the work done to reduce the graph and find the maximum matching = $O(mn^2)$.
Expanding any blossom can be done in $0(m)$ time so the total work done expanding = $O(mn^2)$.
Therefore the entire algorithm takes time = $O(mn^2)$.

6.5 Historical Signifi cance

In his paper describing the above algorithm, Edmonds defined the notion of polynomial-running time for an efficient algorithm.