CMSC 651 Advanced Algorithms	Lecture 7
Lecturer: Samir Khuller	Th., Feb. 26, 1998

Original notes by Michael Tan.

1 Matchings

Read: Chapter 9 [Tarjan].

In this lecture we will examine the problem of finding a maximum matching in bipartite graphs.

Given a graph $G = (V, E)$, a **matching** M is a subset of the edges such that no two edges in M share an endpoint. The problem is similar to finding an independent set of edges. In the maximum matching problem we wish to maximize $|M|$.

With respect to a given matching, a **matched edge** is an edge included in the matching. A **free edge** is an edge which does not appear in the matching. Likewise, a **matched vertex** is a vertex which is an endpoint of a matched edge. A **free vertex** is a vertex that is not the endpoint of any matched edge.

A **bipartite** graph $G = (U, V, E)$ has $E \subseteq U \times V$. For bipartite graphs, we can think of the matching problem in the following terms. Given a list of boys and girls, and a list of all marriage compatible pairs (a pair is a boy and a girl), a matching is some subset of the compatibility list in which each boy or girl gets at most one partner. In these terms, $E = \{$ all marriage compatible pairs $\}$, $U = \{$ the boys$\}$, $V = \{$ the girls$\}$, and $M = \{$ some potential pairing preventing polygamy$\}$.

A **perfect matching** is one in which all vertices are matched. In bipartite graphs, we must have $|V| = |U|$ in order for a perfect matching to possibly exist. When a bipartite graph has a perfect matching in it, the following theorem holds:

1.1 Hall's Theorem

Theorem 1.1 (Hall's Theorem) *Given a bipartite graph $G = (U, V, E)$ where $|U| = |V|, \forall S \subseteq U, |N(S)| \geq |S|$ (where N(S) is the set of vertices which are neighbors of S) iff G has a perfect matching.*

Proof:

($\leftarrow$) In a perfect matching, all elements of S will have at least a total of $|S|$ neighbors since every element will have a partner. ($\rightarrow$) We give this proof after the presentation of the algorithm, for the sake of clarity. $\Box$

For non-bipartite graphs, this theorem does not work. Tutte proved the following (you can read the Graph Theory book by Bondy and Murty for a proof) theorem for establishing the conditions for the existence of a perfect matching in a non-bipartite graph.

Theorem 1.2 (Tutte's Theorem) *A graph G has a perfect matching if and only if $\forall S \subseteq V, o(G-S) \leq |S|$. ($o(G-S)$ is the* **number** *of connected components in the graph $G-S$ that have an odd number of vertices.)*

Before proceeding with the algorithm, we need to define more terms.

An **alternating path** is a path (edges) which begins at a free vertex and whose edges alternate between matched and unmatched edges.

An **augmenting path** is an alternating path which starts and ends with unmatched edges (and thus starts and ends with free vertices).

The matching algorithm will attempt to increase the size of a matching by finding an augmenting path. By inverting the edges of the path (matched becomes unmatched and vice versa), we increase the size of a matching by exactly one.

If we have a matching M and an augmenting path P (with respect to M), then $M \oplus P = (M \cup P) - (M \cap P)$ is a matching of size $|M| + 1$.

1.2 Berge's Theorem

Theorem 1.3 (Berge's Theorem) *M is a maximum matching iff there are no augmenting paths with respect to M.*

Proof:
($\rightarrow$) Trivial. ($\leftarrow$) Let us prove the contrapositive. Assume M is not a maximum matching. Then there exists some maximum matching M' and $|M'| > |M|$. Consider $M \oplus M'$. All of the following will be true of the graph $M \oplus M'$:

1. The highest degree of any node is two.
2. The graph is a collection of cycles and paths.
3. The cycles must be of even length, half of which are from M and half of which are from M'.
4. Given these first three facts (and since $|M'| > |M|$), there must be some path with more M' edges than M edges.

This fourth fact describes an augmenting path (with respect to M). This path begins and ends with M' edges, which implies that the path begins and ends with free nodes (i.e., free in M). $\square$

Armed with this theorem, we can outline a primitive algorithm to solve the maximum matching problem. It should be noted that Edmonds (1965) was the first one to show how to find an augmenting path in an arbitrary graph (with respect to the current matching) in polynomial time. This is a landmark paper that also defined the notion of polynomial time computability.

Simple Matching Algorithm [Edmonds]:

Step 1: Start with $M = \emptyset$.

Step 2: Search for an augmenting path.

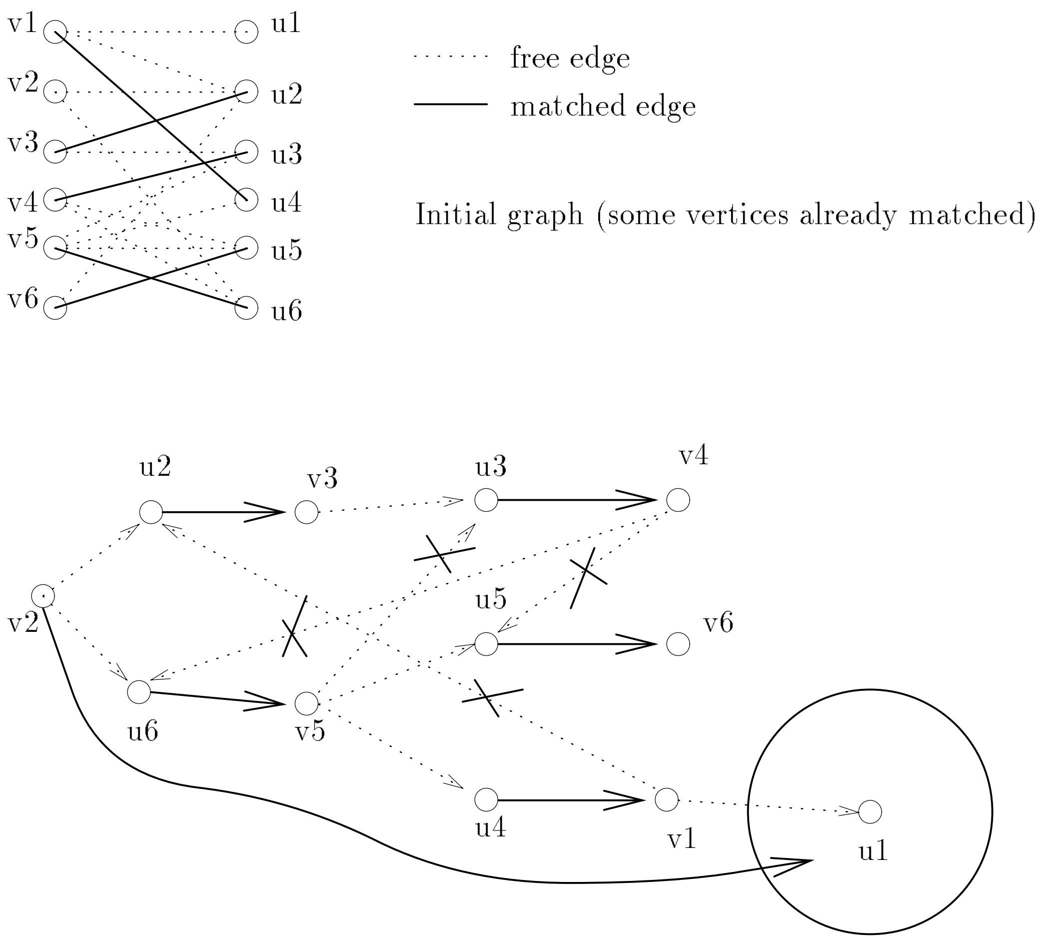
Step 3: Increase M by 1 (using the augmenting path).

Step 4: Go to 2.

We will discuss the search for an augmenting path in bipartite graphs via a simple example (see Fig. 1).

The upper bound on the number of iterations is $O(|V|)$ (the size of a matching). The time to find an augmenting path is O($|E|$) (use BFS). This gives us a total time of O($|V||E|$).

In the next lecture, we will learn the Hopcroft-Karp O($\sqrt{|V|}|E|$) algorithm for maximum matching on a bipartite graph. This algorithm was discovered in the early seventies. In 1981, Micali-Vazirani extended this algorithm to general graphs. This algorithm is quite complicated, but has the same running time as the Hopcroft-Karp method. It is also worth pointing out that both Micali and Vazirani were first year graduate students at the time.



BFS tree used to find an augmenting path from v2 to u1

Figure 1: Sample execution of Simple Matching Algorithm.