9.1 Matching

A **matching** M in a graph G is a set of non-loop edges with no shared endpoints. Vertices incident to M are **saturated**; vertices not incident to M are **unsaturated**. A **perfect matching** is a matching that saturates all $v \in V(G)$. A **maximal matching** is a matching that can't be extended with the addition of an edge. A **maximum matching** is a matching that is the maximum size over all possible matchings on G.

Given a matching M on G, an M**-alternating path** is a path that alternates between edges from G in M and edges not in M. An M-alternating path whose endpoint vertices are unsaturated by M is an M**-augmenting path**. **Berge's Theorem** states that a matching M of G is a maximum matching if and only if G has no M-augmenting path.

The **symmetric difference** between two graphs G and H, written as $G\Delta H$, is the subgraph of $G \cup H$ whose edges are the edges that appear in only one of G and H. The symmetric difference between two matchings contains either paths or cycles. We can use this idea of symmetric difference to prove Berge's Theorem.

Hall's Theorem states that an X, Y-bipartite graph G has a matching that saturates X if and only if $|N(S)| \geq |S|$ for all possible $S \subseteq X$. **Hall's Condition** implies $\forall S \subseteq X, |N(S)| \geq |S|$ for X to be saturated. We can therefore show that a bipartite graph has no matching saturating X if we identify a subset $S \subseteq X$ where $|N(S)| < |S|$.

We can use Hall's theorem to show that all k-regular bipartite graphs have a perfect matching.

9.2 Maximum Bipartite Matching

In unweighted bipartite graphs, we can iteratively increase the size of an initial matching M by finding augmenting paths. If an augmenting path can't be found, we know via **Berge's Theorem** that we have a maximum match. The **Augmenting Path Algorithm** is below. For unweighted shortest paths, we can simply use breadth-first search.

```
procedure MatchBipartite(X, Y-bigraph G)
    M ← ∅                                      ▷ M initially empty
    do
        P ← AugPathAlg(G, M)                   ▷ New augmented path found with M, G
        M ← MΔP                                ▷ Symmetric difference between M, P
    while P ≠ ∅
    return M
```

As we'll see next class, things get a little trickier when we allow odd cycles as in general graphs. We'll need to modify our algorithm to account for them.

procedure AugPathAlg(X, Y-bigraph G and matching $M = (V_M, E_M)$)

$G' \leftarrow G$

Orient $G' : \forall e \in E_M : e(x_i, y_j) = e(y_j \rightarrow x_i); \forall e \notin E_M : e(x_i, y_j) = e(x_i \rightarrow y_j)$

Add vertex s to G' with edges $\forall x_i \in X, x_i \notin V_M : (s \rightarrow x_i)$

Add vertex t to G' with edges $\forall y_j \in Y, y_j \notin V_M : (y_j \rightarrow t)$

$P \leftarrow$ ShortestPathBFS(G', s, t) ▷ Use BFS to find shortest path from s to t

return $P - \{e(s, x_i), e(y_j, t)\}$ ▷ Return path without added edges