10.1 Matching

A **matching** M in a graph G is a set of non-loop edges with no shared endpoints. I.e., $\forall v \in V(G)$, v shows up at most as one endpoint for some edge $e \in M$. Vertices incident to some $e \in M$ are **saturated**; vertices not incident to some $e \in M$ are **unsaturated**. A **perfect matching** is a matching that saturates all $v \in V(G)$. A **maximal matching** is a matching that cannot 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 both 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. Observe how to we can swap the matched$\leftrightarrow$unmatched edges on an M-augmenting path to increase the size of a match.

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 (i.e., 'exclusive or'). 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.

10.2 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 (BFS).

As the algorithm demonstrates, we orient the edges of $G_{X,Y}$ such that matched edges are directed from some $x \in X$ to some $y \in Y$ and unmatched edges are directed in the opposite way. We then add a vertex s that has a directed edge *to* all unsaturated vertices in X and a vertex t that has edges *from* all unsaturated vertices in Y. A directed BFS from root s will then follow M-alternating paths on G. If the BFS is able to reach t, that implies that there is an M-augmenting path from some vertex in X to some vertex in Y.

procedure MATCHBIPARTITE(X, Y-bigraph G)
  $M \leftarrow \emptyset$ ▷ M initially empty
  do
    $P \leftarrow$ AugPathAlg(G, M) ▷ New augmented path found with M, G
    $M \leftarrow M \Delta P$ ▷ Symmetric difference between M, P
  while $P \neq \emptyset$
  return M

On this path, we can swap matched↔unmatched edges to increase the size of the match on G.

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

As we'll see next class, things get a little trickier when we allow odd cycles, as in general graphs. We would need to modify our algorithm to account for them, though the main idea of searching for augmenting paths is the same.

10.3 Mo' Matching

There are several other variations of the matching problem. Generally, matching is considered an optimization problem, where we attempt to match as many edges in a graph as possible. We refer to this problem as **maximum cardinality matching**. As with our spanning tree and shortest paths problems, we can also consider edge-weighted graphs. The **maximum weight matching** problem seeks to maximize the sum of the edge weights in the matched set.

Another matching problem variant is the **stable matching** problem. Instead of attempting to optimize for weights, stable matching optimizes matches between two bipartite sets, where each member of each set has an ordered preference for potential matches to the opposite set. In this problem, an **unstable pair** is a pair of vertices x, a that are not currently matched, but both vertices have a higher preference for match (x, a) over their current match partner. A **stable match** is a match between the two bipartite sets where there exists no possible stable pair.

The **Gale-Shapley Proposal Algorithm** is a greedy algorithm that solves for a stable match given two 'bipartite' sets and list of preferences of matches for all vertices. The

algorithm is generally given in the context of the **stable marriage** problem, which is a synonymous term for the stable matching problem. The problem is described as there being a set of n men and n women, where the men have ordered preferences for each woman and the women have ordered preferences for each man. The algorithm iteratively has men propose and women reject all but one proposal, with a final output having no unstable pairs. The algorithm is given below.

---

procedure GaleShapley(n men and n women and preference lists for each)
  while not done do
    Each man proposes to their highest preference woman
    who has not yet rejected them
    if each woman gets exactly 1 proposal **then**
      return these proposals as a stable match
    else
      Women with 2 or more proposals reject all proposals
      except their highest preference
