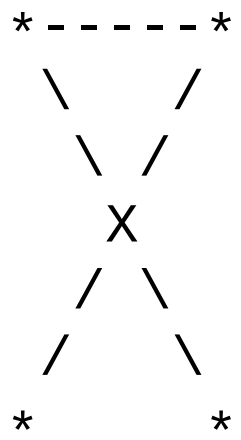
We saw last week that the greedy algorithm can fail to find the maximum-weight **matching** in an arbitrary graph. In fact it can fail for the simpler problem of finding a *maximum cardinality* matching in a **bipartite** graph:



If we take the top edge first, we will miss the larger matching formed by the other two edges. Thus the set of valid partial matchings on a bipartite graph *need not be* a matroid.

Even this special case can be interesting in practice. Suppose the left set consists of employees, the right set consists of jobs, and there is an edge whenever an employee is qualified to perform a given job. The maximal matching is a feasible assignment that gets as many jobs as possible done.

Though not a matroid, the set of partial matchings on a bipartite graph is the *intersection* of two matroids – the matroid of sets that share no point on the left, and the matroid of sets that share no point on the right. (Why are each of these sets matroids?)

Later in this lecture we'll look at the problem of finding a maximum cardinality set in the intersection of two matroids. But now we'll look at the bipartite matching problem, which is simpler and demonstrates the key ideas.

**Solving the Bipartite Matching Problem:**

Let $(U, V, E)$ be a bipartite graph and let $M$ be a matching.

A **free vertex** for $M$ is one that touches no edge of $M$.

An **augmenting path** for $M$ is a sequence of edges that starts at a free vertex, ends at a free vertex, and alternates between edges of $E \setminus M$ and edges of $M$. It must have odd length.

**The Bipartite Matching Algorithm:**

- Set $M$ to $\emptyset$.
- While there is an augmenting path $P$, replace $M$ by $M \oplus P$.
- When there is no augmenting path, return $M$.

It should be clear that taking the symmetric difference with an augmenting path increases the size of $M$ by one. We must show that it preserves the property of being a matching, and that we can find an augmenting path unless $M$ is maximum.

**The Augmenting Path Operation Preserves Matchings:**

A matching is precisely a graph where no node has degree more than one. Given our augmenting path $P$, we are removing the edges in $P \cap M$ and adding those in $P \setminus M$. This adds one to the degree of the two endpoints of $P$, but they were free vertices before and so now have degree one. The interior points of $P$ had degree one in $M$ before, from the edge in $P \cap M$, and have degree one after, from the edge in $P \setminus M$. So the resulting graph is still a matching.

**An Augmenting Path Exists:**

Suppose that $M$ is not maximum, and therefore that some larger matching $M'$ exists. Let $E'$ be the set of edges $M \oplus M'$ – then $(U, V, E')$ forms a bipartite graph with degree at most two. A degree-two graph consists of a collection of paths and cycles, and in this graph each path and cycle must have edges of $M$ alternating with edges of $M'$.

But since $M'$ has more edges than $M$, $E'$ must contain more $M'$ edges than $M$ edges. The cycles in $E'$ have an equal number of edges from each, so there must be at least one path in $E'$ with more edges from $M'$ than from $M$ (exactly one more, in fact, since its edges alternate). This is the desired augmenting path.

**Finding the Augmenting Path:**

The natural way to find the path is by breadth-first search. We have to make sure that we find a path with alternating edges – to do this we use BFS in a directed graph where the $M$ edges go from $V$ to $U$ and the non-$M$ edges go from $U$ to $V$.

We need to carry out a BFS from each vertex in $U$, to see whether we can find a nonempty path to a free vertex of either $U$ or $V$. This could take $O(|U||E|)$ time. But we can save some time by avoiding duplication in our search.

If we start at a vertex $u$ and *fail* to find a path to a free vertex, we know that not only $u$ but *every* $U$ vertex we encountered in the search lacks a path to a free vertex. If we mark all these vertices and avoid searching them again, we save time. Now we visit each vertex only once for each edge coming into it – this gives us a total time of $O(|E|)$.

Let's now generalize the bipartite matching situation. Suppose that we have two matroids $(E, I)$ and $(E, J)$ with the same set of elements. That is, $I$ and $J$ are two distinct definitions of "valid" sets. Our goal is to find a *maximum cardinality* set that is in both $I$ and $J$.

Note that we have no weighting function here, and there is no reason to believe that a greedy strategy will work.

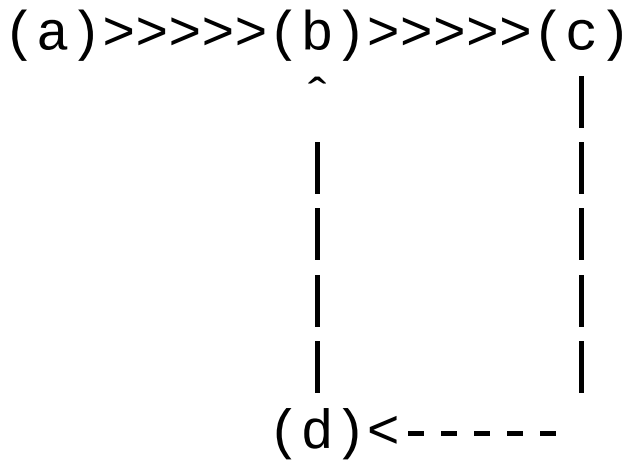
In the bipartite matching problem, we could let $I$ be the sets of edges that share no vertex on the left, and $J$ be the sets that share no vertex on the right.

A second example of matroid intersection is the problem of finding a **branching** in a directed graph, which is a sort of directed spanning tree.

**Definition:** A branching in a directed graph $(V, E)$ is a set of edges $E' \subseteq E$ such that the undirected version of $(V, E')$ is a tree and all the edges of $E'$ point away from a single vertex, called the root.

A set of edges is a branching with root $r$ iff its undirected version is a tree, the in-degree of $r$ is zero, and the in-degree of every other vertex is one.

The greedy algorithm, looking for a branching with root $a$, can fail on the following graph (Figure 3.8):




We have no criterion for choosing an order on the edges! If we choose the edge from $d$ to $b$ first, we cannot then choose the edge from $a$ to $b$ that we need.

But if we let $I$ be the sets of acyclic edges and $J$ be the sets of edges that meet the in-degree condition, then a maximum cardinality set in $I \cap J$ is what we need. If it has $n - 1$ edges, it is a branching. If not, no branching exists.

**The General Algorithm:** Instead of an augmenting path, we look for an augmenting sequence of elements. If we currently have a set $M \in (I \cap J)$, an augmenting sequence is a sequence of elements $\{e_1, \ldots, e_k\}$ where $k$ is an odd number and:

- $M + e_1$ is in $I$ but not in $J$
- $M + e_1 - e_2$ is in $I \cap J$
- $M + e_1 - e_2 + e_3$ is in $I$ but not $J$
- until finally...
- $M + e_1 - e_2 + e_3 - \ldots + e_k$ is in $I \cap J$.

If such an augmenting sequence exists, we have found a set in $I \cap J$ that is one element larger than $M$. If no such sequence exists, it turns out that $M$ is maximum. It follows that repeatedly finding an augmenting sequence will eventually give us a maximum set.

The proof that this method solves the matroid intersection problem is somewhat involved so we will only sketch it here. Some of the ideas should become a little clearer as we look at *how to find* an augmenting sequence if one exists.

**How to find an augmenting sequence?**

For a given set $M$, we build an **auxiliary graph** $D(M)$ as follows:

- The vertices are the elements of $E$
- If $x \in M$, $y \notin M$, and $M + y - x$ is in $I$, $D(M)$ has the edge $(y, x)$
- If $x \in M$, $y \notin M$, and $M + y - x$ is in $J$, $D(M)$ has the edge $(x, y)$

We let $U$ be the set of elements $u \in E \setminus M$ such that $M + u \in I$, and $V$ the set of elements $v \in E \setminus M$ such that $M + v \in J$. Note that if $U \cap V \neq \emptyset$, then we have an augmenting sequence of length 1 and $M$ is definitely not maximal.

- If $x \in M$, $y \notin M$, and $M + y - x$ is in $I$, $D(M)$ has the edge $(y, x)$
- If $x \in M$, $y \notin M$, and $M + y - x$ is in $J$, $D(M)$ has the edge $(x, y)$

Any augmenting sequence corresponds to a path of even length in $D(M)$ from some element $s \in U$ to some other element $t \in V$. (Actually it must be a *minimum length* such path. We'll omit the proof of this correspondence.)

But we can argue that if there is *no* path in $D(M)$ from $U$ to $V$, then $M$ is maximal in $I \cap J$. First, let $U'$ be the elements of $M$ that have paths in $D(M)$ from $U$, and let $V'$ be those that have paths to $V$. (Since there is no path from $U$ to $V$, $U'$ and $V'$ are disjoint.)

- If $x \in M$, $y \notin M$, and $M + y - x$ is in $I$, $D(M)$ has the edge $(y, x)$
- If $x \in M$, $y \notin M$, and $M + y - x$ is in $J$, $D(M)$ has the edge $(x, y)$

The key point is that since $U'$ has no edges to $V$, it is maximal in $U \cup U'$ for $J$. Pick any $y \in U$ and look at $U' + y$. If it were in $J$, we could add elements from $M$ to it and stay in $J$ by the Exchange Property, since $M \in J$. By the time we have a set of size $|M|$, we have $M + y - x$ for some $x \in V'$. But $M + y - x$ cannot be in $J$ as there is no edge from $x$ to $y$ in $D(M)$.

By a similar argument, $V'$ is maximal in $V \cup V'$ for $I$. Any set in $I \cap J$ of size greater than $|M|$ would have to either be bigger than $V'$ in $V \cup V'$ or bigger than $U'$ in $U \cup U'$. This is a contradiction, so $M$ is maximal.

We can easily use BFS to find a shortest possible path from $U$ to $V$ in $D(M)$, or demonstrate that no such path exists. The running time turns out to be $O(|E|^3)$ times the time to test a set for membership in $I$ or $J$ – a bad polynomial but still polynomial.

A variation of this algorithm (see Papadimitriou and Stieglitz) can be used to find the maximum weight set in $I \cap J$, given a non-negative weight function on $E$.

But finding the maximum-cardinality set in the intersection of *three* matroids is in general a much harder problem (as far as we know). Consider, for a directed graph $G = (V, E)$ with fixed vertices $s$ and $t$:

- $I$ is the matroid of sets of edges that are acyclic when viewed as undirected edges,
- $J$ is the matroid of sets of edges that have no edges into $s$ and at most one into any other vertex, and
- $K$ is the matroid of sets of edges that have no edges out of $t$ and at most one out of any other vertex.

A set of edges in $I \cap J \cap K$ is a **simple path** from $s$ to $t$. The graph has a **Hamilton path** from $s$ to $t$ if and only if there is a set of size $|V| - 1$ in $I \cap J \cap K$. The Hamilton path problem is **NP-**complete, and thus not in **P** unless $\mathbf{P} = \mathbf{NP}$.