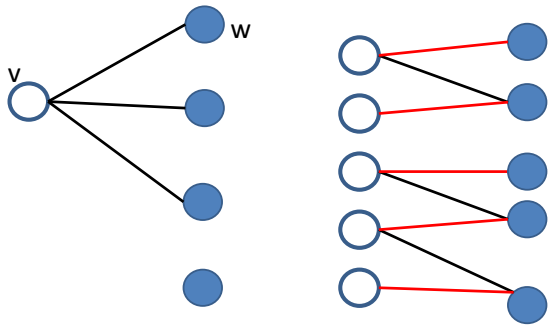
In some situation finding approximately optimal solution to a maximization or minimization problem is good enough in the application. This may be useful if we either lack all the information as we have to make decisions (online algorithms) or we don't have the time to find a fully optimal solution (say as the problem is too hard). In the remaining part of the course, we will explore algorithms for such situations.

1 Online Matching

We start by an online version of the bipartite matching problem. The bipartite matching problem is given by two sets of nodes X and Y, and a set of edges $E \subset X \times Y$, and the problem is to find a matching $M \subset E$ of maximum size. We have seen how to do this in polynomial time using a flow algorithm. In this section, we will consider a version of this problem, where Y is given upfront, nodes $x \in X$ appear one-at-a time, as a node $v \in X$ appears, we are also informed about all its adjacent edges in E, and we need to make a decision to add an edges to M adjacent to v (or not) without knowledge of further nodes that will show up later. So for example, in the graph on the left of the figure below, when the first node v shows up, there seems to be no good way to decide which node to match it on the opposite side. However, once we match v, say using the edge (v, w), there will be no way to change this edge in the matching, we will not be able to use augmenting path to update our matching. A typical application of this style of matching problem, is matching advertisement to web pages. One side of the matching are web users, the other side are advertisements we may want to show the users. When a user v shows up online, we may have an opportunity to show many possible ads, and we need to decide instantly, which add to show the person.



Under these circumstances, we cannot expect to be able to find the true maximum matching. We will be analyzing here the simple greedy algorithm: select a edge adjacent to v that can be added to the matching, whenever such an edge is possible. This greedy algorithm is not optimal. For example, if we use (v, w) in the example above, and the next node is v' with a single edge (v', w) adjacent, then we could have selected a matching of size 2 (had we matched v to a different adjacent node), but unfortunately, our resulting matching is just one edge. However, we can say something positive about the matching the algorithm creates.

We say that the algorithm is a 2-approximation algorithm is the matching it finds has size at least $1/2$ of the maximum possible. Note that for this argument, we need to show a bound on the size of the

maximum matching, and we'll do this without computing the maximum matching explicitly.

Claim. The above online matching algorithm is a 2-approximation, find a matching M is size at least $1/2$ of the maximum possible matching.

Proof. Consider the matching M and the the maximum matching M^*. Let Y be the side that is given, and nodes on the Y side arrive one-at-a-time. Consider the graph just consisting of edges M and M^*, an example of which is shown on the right of the figure, where red edges are edges in M^* and black edges are edges in M. Each component of this graph is a path alternating between M and M^* edges. Consider a node $v \in X$ matched by M^* by an edge (v, w). Either v is also matched in M (to possibly a different node), or w is matched in M, as otherwise the edge (v, w) would have been an option to add when v appeared. So any edge $(v, w) \in M^*$ is adjacent to at least one edge in M, and an edge in M can only be adjacent to at most 2 edges in M^* proving that $|M^*| \leq 2|M|$ as claimed.