Matchings: bipartite graphs.

Q. How can you determine if a graph is bipartite?

no odd cycle $\Leftrightarrow$ bipartite, but how do you test this?

Given: $G = (A \cup B, E)$ bipartite graph (undirected), find a maximum-size matching.

Can be solved via flows:

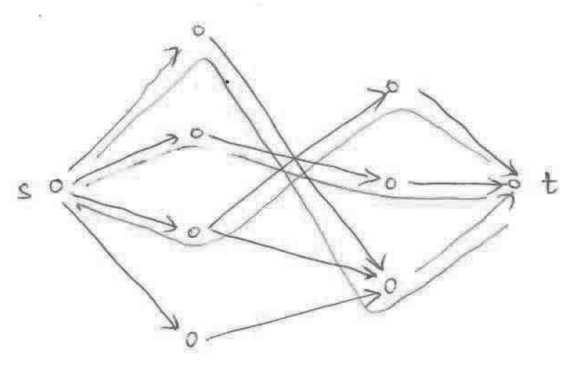
directed graph $D = (V', E')$: $V' = \{s,t\} \cup A \cup B$

$E' = \{(s,a)\ \forall a \in A\} \cup E$ (directed from A to B) $\cup \{(b,t)\ \forall b \in B\}$

$c(e) = 1 \quad \forall e \in E'$. Find maximum flow.

If value of max. flow $= k$, then size of max. matching $= k$, given by edges of E used in flow.

example:

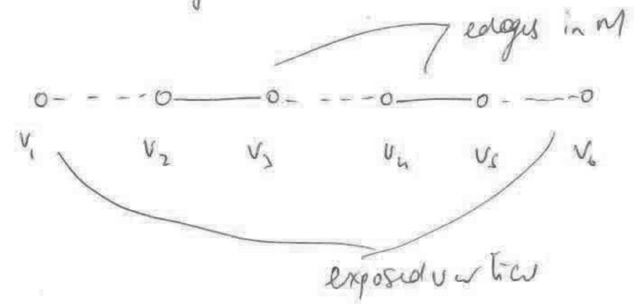


$c(e) = 1 \quad \forall e$

Maximum matchings in bipartite graphs – another algorithm.

Let M be a given matching. A path $p = (v_1, \dots, v_k)$ of odd length (i.e., # vertices is even) is an augmenting path if

1. v_1, v_k are exposed vertices (i.e., no edge in M is incident)
2. alternate edges are in M.



eg.: v_1 o- - - -o v_2 is an augmenting path if v_1, v_2 are exposed.

If $\exists$ an a.p., then we can construct a matching of size $|M|+1$ by simply switching edges along the a.p., i.e., if e is in M, take it out, and if it is not a matching edge, put it in (convince yourself this works). Hence, if $\exists$ a.p., M is NOT a maximum matching.

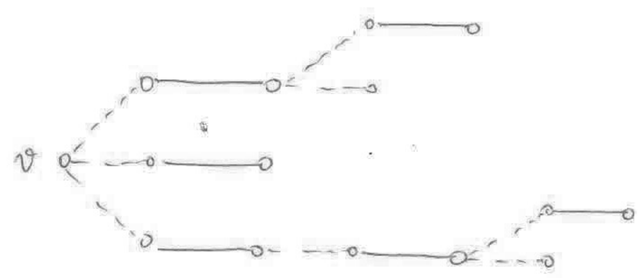
The other direction is true also:

Berge's Lemma: For any graph G, M is a maximum matching iff there is no a.p.

We will show proof of other direction in today's lecture.

Given a matching M (possibly empty), let us first write down an algorithm to find augmenting paths. For this, we choose an exposed vertex v, and find the alternating tree rooted at v:

Say $v \in A$, where $V = A \cup B$.



A	B	A	B	A	B	A
even	odd	even	odd	even	odd	even

A vertex is even or odd depending on its distance to the root.

Given an exposed vtx. v, our algo proceeds as follows:

~~main~~

```
∀ u ≠ v,  u.mark = NULL
          v.mark = even,  v.parent = NULL
          Enqueue (Q, v)
          while (notempty (Q)) {
                ~~[u = Dequeue (Q)]~~
                AltTree (Q)
          }

Alt-Tree (Q)  {
     u = Dequeue (Q)
     if u.mark = even
          for all e = (u,w) ∈ E   {
                if   w.mark = NULL   {
                       w.parent = u , w.mark = odd
                       Enqueue (Q, w)
                }
          }
```

else // u.mark = odd {

if $\exists\, e = (u, w) \in M$ AND w.mark = NULL

w.mark = even, w.parent = u,

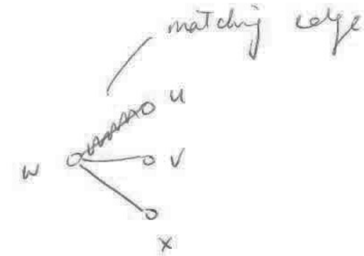
Enqueue (Q, w)

else print "alternating v-w path found"] —— (*)

}

we note the following about the algorithm:

(0) Not all vertices in $A \cup B$ are in tree, e.g.



For non-tree edges:

(i) no edges between vertices on the same level, since that would give an odd cycle.

(ii) no edges between vertices if diff. in levels ≥ 2, since [~~that~~] we are using queue / bfs-style search.

from (i) & (ii): only non-tree edges are between vertices with difference in levels = 1.

(iii) every "even" vertex except for v has a matching edge in the tree

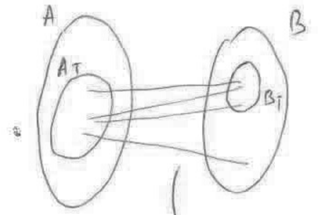
Thus: any non-tree edge, since it is between an even & odd vertex, is not in M. This gives the following claim.

<u>Claim</u>: If leaf w is odd, there is a v-w augmenting path

This justifies step —— (*) in the algorithm.

Thus, if there is no augmenting path, all leaves are even.

Now let A_T, B_T be the vertices of A, B in tree. Note that $|A_T| = |B_T| + 1$. Further, $N(A_T) = B_T$, since:



such an edge cannot be present

say $\exists$ edge (u,w) with $u \in A_T$, $w \notin B_T$. But then u is an even vertex, and the algorithm searches all nodes adjacent to even vertices, hence w would have been in tree.

Hence vertices in A_T must be matched to vertices in B_T, and any matching must leave at least one vertex in A_T exposed.

Now, if v is the only exposed vertex in matching M, this proves that M is a maximum matching.

Suppose that in matching M, vertices $A' = \{v_1, \dots, v_k\} \subseteq A$ are exposed. Further, there is no augmenting path found in alternating tree rooted at any of these vertices. We will show that M is ~~an~~ a maximum matching. Note that $|M| = |A| - |A'| = |A| - k$.

Let T_1 be alternating tree from v_1, A_{T_1}, B_{T_1} defined as previously.
Let T_2 be alternating tree from v_2 in $G \setminus T_1$, and A_{T_2}, B_{T_2} nodes of A, B in T_2 respectively. By same logic, $|A_{T_2}| = |B_{T_2}| - 1$.

$\vdots$

Let T_k be alternating tree from v_k in $G \setminus (T_1 \cup \dots \cup T_{k-1})$, and A_{T_k}, B_{T_k}

nodes of A, B in T_k. Then $|A_{T_k}| = |B_{T_k}| - 1$.

Thus: $\sum_{i=1}^{k} |A_{T_i}| = \sum_{i=1}^{k} |B_{T_i}| - k$.

By construction, $A_{T_i} \cap A_{T_j} = \phi$, $B_{T_i} \cap B_{T_j} = \phi$. Hence: $|A_{T_1} \cup \ldots \cup A_{T_k}| = \sum_{i=1}^{k} |A_{T_i}|$, and same for B_{T_i}.

Further, $N(A_{T_1} \cup \ldots \cup A_{T_k}) = B_{T_1} \cup \ldots B_{T_k}$.

Thus, any matching must leave at least k vertices in $A_{T_1} \cup A_{T_2} \cup \ldots \cup A_{T_k}$ unmatched, and hence M is a maximum matching.