

Maximum Bipartite Matching

Adapted from the CLRS book slides

EXAMPLE APPLICATIONS OF MATCHING

- Assigning job candidates to interview slots.
 - One vertex in L for each candidate.
 - One vertex in R for each interview slot.
 - Edge (l, r) , where $l \in L$ and $r \in R$, if candidate l is available for an interview in slot r .
 - A matching assigns candidates to interview slots.
 - Want a *maximum matching*: a matching of maximum size. That maximizes the number of candidates interviewed.

MAXIMUM MATCHING IN BIPARTITE GRAPHS

Graph $G = (V, E)$, matching $M \subseteq E$.

- ***Matched vertex***: has an incident edge in M . Otherwise, the vertex is ***unmatched***.
- ***Maximal matching***: a matching that you cannot add any other edges to: if M is a matching, it is a maximal matching if for all $e \in E - M$, $M \cup \{e\}$ is not a matching. A maximum matching is always maximal, but a maximal matching is not always maximum.
- ***M-alternating path***: a simple path whose edges alternate between being in M and $E - M$.
- ***M-augmenting path***: an M alternating path whose first and last edges belong to $E - M$. (Also called an augmenting path with respect to M .) Contains 1 more edge in $E - M$ than in $M \Rightarrow$ has an odd number of edges.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (continued)

M-augmenting paths

The key to Hopcroft-Karp. If you have a matching M and an M -augmenting path P , then make a new matching with 1 more edge than M by

- Removing from M the edges that are in P .
- Adding to M the edges in P that are in $E - M$.

Since P contains 1 more edge from $E - M$ than from M , the new matching has 1 edge more than the old matching.

Symmetric difference: For sets X, Y , $X \oplus Y = (X - Y) \cup (Y - X) = (X \cup Y) - (X \cap Y)$ = the elements in either X or Y , but not in both.

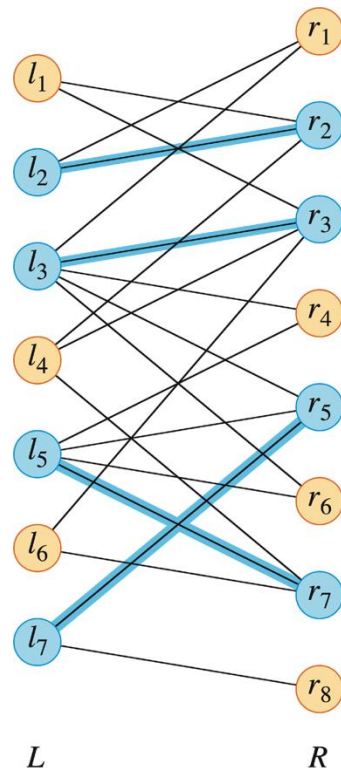
\oplus is commutative and associative. Identity is the empty set.

Formally, use symmetric difference to create a new matching from M and an M -augmenting path:

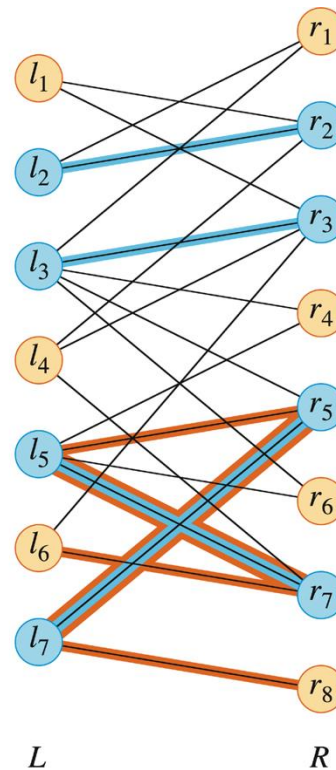
MAXIMUM MATCHING IN BIPARTITE GRAPHS (continued)

Lemma

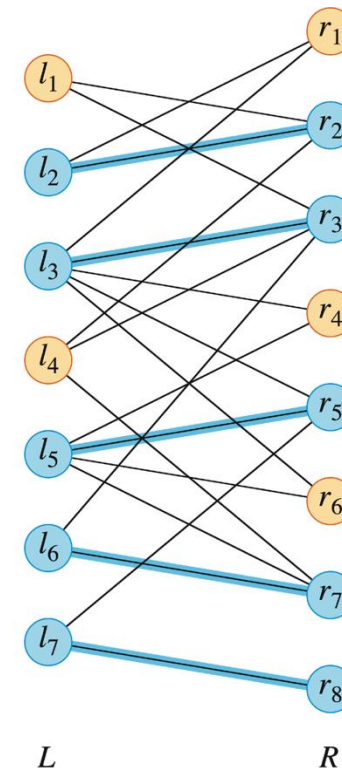
Let M be a matching and P be an M -augmenting path. Then $M' = M \oplus P$ is also a matching with $|M'| = |M| + 1$.



(a)



(b)



(c)

MAXIMUM MATCHING IN BIPARTITE GRAPHS (continued)

Corollary

Let M be a matching and P_1, P_2, \dots, P_k be vertex-disjoint augmenting paths. Then $M' = M \oplus (P_1 \cup P_2 \cup \dots \cup P_k)$ is a matching with $|M'| = |M| + k$.

Proof Induction on i using the lemma shows that $M \oplus (P_1 \cup P_2 \cup \dots \cup P_{i-1})$ is a matching with $|M| + i - 1$ edges and P_i is an augmenting path with respect to $M \oplus (P_1 \cup P_2 \cup \dots \cup P_{i-1})$. ■

MAXIMUM MATCHING IN BIPARTITE GRAPHS (continued)

Lemma

Let M and M^* be matchings in $G = (V, E)$. Consider $G' = (V, E')$, where $E' = M \oplus M^*$. Then, G' is a disjoint union of simple paths, simple cycles, and/or isolated vertices. Edges in each simple path or simple cycle alternate between M and M^* . If $|M^*| > |M|$, then G' contains $\geq |M^*| - |M|$ vertex-disjoint M -augmenting paths.

Can design an algorithm to find a maximum matching by repeatedly finding augmenting paths to increase the size of the matching. Stop when there are no more augmenting paths:

Corollary

M is a maximum matching if and only if there are no M -augmenting paths.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (continued)

Have enough for a maximum-matching algorithm already:

- Start with matching M empty.
- Repeatedly run either BFS or DFS from an unmatched vertex, taking alternating paths, until finding another unmatched vertex.
- Use the M -augmenting path to increment the size of M .
- Runs in $O(VE)$ time.

HOPCROFT-KARP ALGORITHM

Runs in $O(\sqrt{V}E)$ time.

Starts with an empty matching M . Repeatedly finds a maximal set of vertex-disjoint M -augmenting paths and updates M according to the corollary of the first lemma above. Stops when there are no M -augmenting paths, so that the matching is maximum according to the corollary of the second lemma above.

HOPCROFT-KARP ALGORITHM (continued)

HOPCROFT-KARP(G)

```
1   $M = \emptyset$ 
2  repeat
3      let  $\mathcal{P} = \{P_1, P_2, \dots, P_k\}$  be a maximal set of vertex-disjoint
        shortest  $M$ -augmenting paths
4       $M = M \oplus (P_1 \cup P_2 \cup \dots \cup P_k)$ 
5  until  $\mathcal{P} == \emptyset$ 
6  return  $M$ 
```