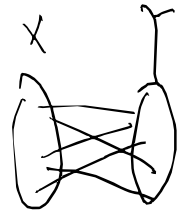
Bipartite matching

Given an undirected graph, a matching is a subgraph with max degree ≤ 1.
Equivalently, it's a set of edges with disjoint endpoints.

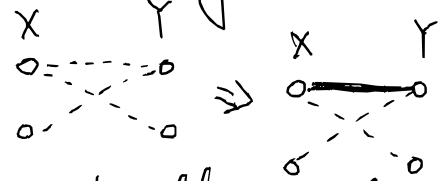
Problem: Given an undirected graph G, compute a matching containing maximum number of edges.

Today: bipartite graph G.

$G = (X \cup Y, E)$

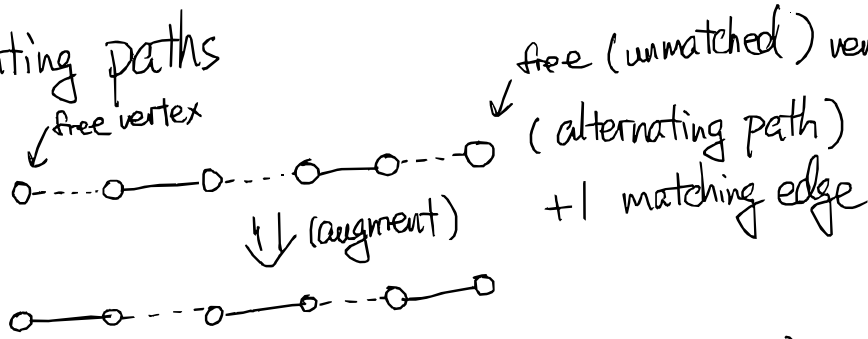


Main algo: start with the empty, iteratively improve the size of the matching until cannot be improved further.



(simply adding an edge to the matching might not always work)

Augmenting paths



Algorithm for bipartite graph $G=(X \cup Y, E)$

Repeat
  mark all vertices as unvisited
  for each free vertex $x \in X$
    DFS from x to search for an augmenting path, mark all visted vertices as visited
    if an augmenting path is found
      Augment!
Until no augmenting path found.

Thm: A matching has maximum size iff there's no augmenting path.

Proof: $\Rightarrow$: trivial.

$\Leftarrow$: Prove the contrapositive.

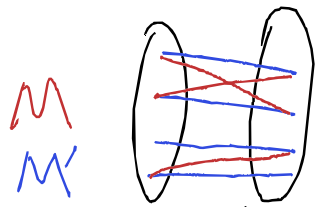
Let M be any matching of size less than the maximum. WTS: $\exists$ an augmenting path.

Let M' be a maximum matching.

Consider the symmetric difference of M and M'.

What can each connected component be?

- must be a path or a cycle with edges coming from M and M' alternatively.

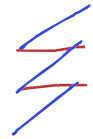


Each cycle has the same number of edges from M and M'.

Each path may contain one more edge from one matching than the other.

$|M| < |M'|$. $\exists$ path with one more edge from M'.

This is an augmenting path for M.



□

Corollary: If $|M'| - |M| = k$, then M has k disjoint augmenting paths. The shortest has length $\leq \frac{n}{k}$.

Proof:
$\exists$ k connected components with one more edge from M'. □

Running time: $O(n \cdot m)$.
- Each iteration takes $O(m)$ time (DFS).
- At most n iterations.

Hopcroft-Karp algo.
(Augment all shortest augmenting paths.)
One phase of Hopcroft-Karp:
- BFS from all free vertices from X together to compute for each v, $d[v]$, the min length of an alternating path from a free

vertex to v.

- Let k be the length of the shortest augmenting path ($k = \min_{v:\text{free} \in Y} d[v]$).
- Keep finding augmenting paths of length k that is disjoint from the previously found ones.

(A maximal set of length-k disjoint aug paths)

- Augment all at once.

Hopcroft-Karp:

Repeat phases until no augmenting path.

Lemma: The length of the shortest augmenting path must increase after a phase.

Assuming the lemma, we have:

Thm: Hopcroft-Karp terminates in $O(\sqrt{n})$ phases.

Proof: After $\sqrt{n}$ phases, the length of the shortest aug path is $\geq \sqrt{n}$.

By the corollary, the matching size is within $\sqrt{n}$ of the max. It takes at most $\sqrt{n}$ more phases to reach max matching. □

Each phase can be implemented in $O(m)$ time.

Total time: $O(m\sqrt{n})$.