Hopcroft-Karp algorithm for maximum matching in bipartite graphs

Follows the presentation in the text book “The design and analysis of algorithms” by Dexter Kozen.

A characterization of maximum matching

- Let G be a bipartite graph and M be a matching in G.
- A vertex in G is said to be a *free* vertex with respect to M if there are no edges of M incident on it.
- A vertex in G is said to be a *matched* vertex with respect to M if there is an edge of M incident on it.
- A path in G is said to be an *alternating path* w.r.t M if its edges alternate between an edge in M and an edge not in M.
- An alternating path is said to be an *augmenting path* w.r.t. M if it starts with a free vertex and ends with a free vertex.
- **Claim:** A matching M in G is maximum if and only if it has no augmenting paths w.r.t. M.
  - Suppose there is an augmenting path P in G w.r.t. M.
  - Consider the set of edges $M' = M \oplus P$.
  - It is easy to check that M' is a matching with $|M'| = |M| + 1$.
  - The other direction follows from Lemma 1:

    Lemma 1: Let M and N be two matchings with $|N| > |M|$. Then, there are $k = |N| - |M|$ vertex-disjoint M-augmenting paths in $N \oplus M$.

Hopcroft-Karp bipartite maximum matching algorithm

1. Initially, $M \leftarrow \emptyset$.
2. Repeat until no more augmenting paths
   (a) Construct a shortest paths DAG H from G and M.
   (b) Construct a maximal set $\mathcal{S}$ of vertex-disjoint minimum length augmenting paths.
   (c) Update M.

Constructing a shortest paths DAG H from G and M

Let $G = (L, R, E)$ be a bipartite graph. Let M be a matching in G.

- Construct a layered DAG H such that:
  - i is the shortest path distance from the source to all the vertices in layer i.
- Let D_i denote the vertices in layer i.
- Let $Indegree(u)$ be the number of incoming edges to u in the DAG being constructed.
- Shortest paths DAG construction algorithm (using breadth-first search):
  1. For all $u \in V$, $Indegree(u) = 0$.
  2. Add all free vertices from L to D_0.
  3. $i = 0$.
  4. Repeat Until all vertices have been classified or a layer with free vertices of R is found:
     (a) For all $u \in D_i$
        i. For all w adjacent to u such that the edge (u, w) is an unmatched edge and w is not in an earlier layer do
           A. If w is not in D_{i+1}) then include w in D_{i+1} and set $Indegree(w)$ to be 0.
           B. If w is in D_{i+1}) then increment $Indegree(w)$.
     (b) (All vertices in D_{i+1} are from R.)
        If any of the vertex in D_{i+1} is a free vertex in R then
        i. Delete all matched vertices from D_{i+1}.
        ii. Let $k = i + 1$.
        iii. Go to Augment Stage.
     (c) (None of the vertices in D_{i+1} are free vertices of R.)
        For all $u \in D_{i+1}$
        i. For w that is adjacent to u using a matched edge do
           A. If w is not already included (in an earlier layer or in D_{i+1}) then include w in D_{i+1}.
     (d) $i \leftarrow i + 2$.
  5. No augmenting paths in G with respect to M and hence M is maximum.
- Time: $O(m)$.

Finding augmenting paths in the shortest paths DAG H

- Layer 0 of the DAG H consists of free vertices from L.
- Layer k consists of free vertices from R.
- Construct a *maximal* set $\mathcal{S}$ of vertex-disjoint augmenting paths of length k.
- If there is a vertex $u \in D_k$ then there is a path (an augmenting path) from some vertex in D_0 to u.
- If there is a vertex w in D_0 it is not guaranteed that there is a path from w to a vertex in D_k.
- Hence, to find an augmenting path in H, we start from a vertex in D_k and trace back.
- Maximal set of minimum length augmenting paths construction algorithm:

  1. While there is a vertex u in D_k do:
     (a) Trace backwards from u to a free vertex in D_0 to obtain an augmenting path.
     (b) Place this path in the set $\mathcal{S}$.
     (c) Add all the vertices along this path to a deletion queue.
     (d) While the deletion queue is non-empty do:
        i. Let u be the next vertex in the deletion queue.
        ii. Delete u and its incident edges.
        iii. If this deletes an edge (u, w) with $u \in D_j$ and $w \in D_{j+1}$ for some j then $indegree(w)$ is decremented. (This is because from w we cannot trace back to u anymore since u is deleted.) If $indegree(w)$ becomes 0 then add w to the deletion queue.

- Time: $O(m)$.

Time taken by HK algorithm

Lemma 2: Let M be a matching and P be a shortest M-augmenting path. Let $M' = M \oplus P$ be the matching obtained by augmenting M by P. Let Q be an M'-augmenting path. Then,

$$|Q| \geq |P| + 2|P \cap Q|.$$

- If P and Q are vertex-disjoint then Q is an M-augmenting path In this case, the lemma holds since P is a shortest M-augmenting path and $P \cap Q = \emptyset$.
- Suppose P and Q are not vertex-disjoint.
- Let $N = M' \oplus Q$ be the matching obtained by augmenting M' by Q.
- Then, $|N| = |M| + 2$.
- By Lemma 1, there are two odd length vertex-disjoint M-augmenting paths P_1 and P_2 in $N \oplus M$.
- These augmenting paths canot be shorter than P since, by hypothesis, P is a shortest M-augmenting path.
- Now, $N \oplus M = M' \oplus Q \oplus M = M \oplus P \oplus Q \oplus M = P \oplus Q$.
- Therefore, there are two M-augmenting paths in $P \oplus Q$ that are not shorter than P.
- Therefore, $|P \oplus Q| \geq 2|P|$.
- Since $|P \oplus Q| = |P| + |Q| - 2|P \cap Q|$, we have $|P| + |Q| \geq 2|P| + 2|P \cap Q|$.
- The lemma follows.

Time taken by HK algorithm

Call each execution of the Repeat loop in the matching algorithm a *phase*.

Lemma 3: After each phase, the length of a shortest augmenting path increases by at least two.

- Let the matching M be augmented by a maximal set $\mathcal{S}$ of vertex-disjoint shortest length M-augmenting paths in a phase. Let the resulting matching be M'.
- Let Q be an M'-augmenting path.
- If Q does not share any vertex with any path P in the set $\mathcal{S}$, $|Q| > |P|$.
  - Q is also an M-augmenting path.
  - If $|Q| \leq |P|$ then S is not maximal.
- If Q shares a vertex with some $P \in \mathcal{S}$ then $|Q| > |P|$.
  - Every vertex in P is matched in M'.
  - Therefore, $|P \cap Q|$ has at least one edge in M'.
  - The claim follows using Lemma 2.

Time taken by HK algorithm

- The running time for the HK matching algorithm is $O(\sqrt{n}m)$.
  - Each phase can be executed in $O(m)$ time.
  - The number of phases is $O(\sqrt{n})$.
    - Let M be the matching after $\sqrt{n}$ phases.
    - Suppose M is not maximum.
    - Let M^* be a maximum matching.
    - Then, the size of the matching has to increase by $|M^*| - |M|$.
    - By Lemma 1, $M^* \oplus M$ has $|M^*| - |M|$ vertex-disjoint M-augmenting paths.
    - By Lemma 3, each of these M-augmenting paths has length $\geq 2\sqrt{n}+1$. Therefore, the number of M-augmenting paths in $M^* \oplus M$ is $\leq \frac{\sqrt{n}}{2}$. (This is because the number of vertices is n.)
    - Each phase increases the size of the matching by at least 1.
    - Therefore, there are at most $\frac{\sqrt{n}}{2}$ more phases before a maximum matching is computed.
    - Hence, there are at most $O(\sqrt{n})$ phases.