# Hopcroft-Karp algorithm for matching in bipartite graphs

Let $G = (V_1, V_2, E)$ be a bipartite graph. Let $M$ be a matching in $G$.

**Constructing a shortest paths DAG**

The algorithm below constructs a layered DAG $H$ such that $i$ is the shortest path distance from the source to all the vertices in layer $i$. It also computes for each vertex $u$, except those at layer 0, the number of incoming edges to $u$. This construction is done using a modified breadth-first traversal of $G$. Let $L_i$ denote the vertices in layer $i$.

1. Add all free vertices from $V_1$ to $L_0$.

2. $i = 0$.

3. Repeat Until all vertices have been classified or a layer with free vertices of $V_2$ is found:

   (a) For all $u \in L_i$

      i. For all $w$ adjacent to $u$ using an unmatched edge do

         A. If $w$ is not from an earlier layer add $w$ to $L_{i+1}$ if it is not already included.

         B. $indegree(w) \leftarrow indegree(w) + 1$.

   (b) (All vertices in $L_{i+1}$ are from $V_2$.)
   If any of the vertex in $L_{i+1}$ is a free vertex in $V_2$ then

      i. Delete all matched vertices from $Li + 1$.

      ii. Let $t = i + 1$.

      iii. Go to Augment Stage.

   (c) (None of the vertices in $L_{i+1}$ are free vertices of $V_2$.)
   For all $u \in L_{i+1}$

      i. For $w$ that is adjacent to $u$ using a matched edge do

         A. If $w$ is not from an earlier layer add $w$ to $L_{i+2}$ if it is not already included.

   (d) $i \leftarrow i + 2$.

4. No augmenting paths in $G$ with respect to $M$ and hence $M$ is maximum.

### Finding augmenting paths in the shortest paths DAG $H$

The DAG $H$ is a layered graph with layer 0 consisting of free vertices from $V_1$ and layer $t$ consisting of free vertices from $V_2$. The algorithm below constructs a set $\mathcal{P}$ of vertex-disjoint minimum length augmenting paths.

(If there is a vertex $u$ in $L_t$ then there is a path (an augmenting path) from some vertex in $L_0$ to $u$. If there is a vertex $w$ in $L_0$ it is not guaranteed that there is a path from $w$ to a vertex in $L_t$. Hence, to find an augmenting path in $H$, we start from a vertex in $L_t$ and trace back. )

1. While there is a vertex $u$ in $L_t$ do:

   (a) Trace backwards from $u$ to a free vertex in $L_0$ to obtain an augmenting path; place this path in the set $\mathcal{P}$.

   (b) Place all the vertices along this path on a deletion queue.

   (c) While the deletion queue is non-empty do:

      i. Remove a vertex and all its outgoing edges.

      ii. Whenever an edge $(u, w)$ in $H$ is deleted, $indegree(w)$ is decremented. (This is because from $w$ we cannot trace back to $u$ anymore since $u$ is deleted.) If $indegree(w)$ becomes 0 then place $w$ on the deletion queue.

**Time taken by this algorithm**

We will use the following two lemmas:

**Lemma 1:** The length of the shortest augmenting path increases in each phase.

(Proof omitted)

**Lemma 2:** Let $M$ be a matching that is not maximum. Let $M^*$ be a maximum matching. Let $|M^*| - |M| = k$. Then, there are $k$ vertex-disjoint augmenting paths in $M^* \oplus M$.

**Proof:**

- Observe that in $M^* \oplus M$, the number of $M^*$-edges is $k$ more than the the number of $M$-edges.

  In $M^* \oplus M$, the number of $M^*$-edges is $|M^* - M|$ and the number of $M$-edges is $|M - M^*|$. We have,

$$\begin{aligned} |M^* - M| &= |M^*| - |M^* \cap M| \\ &= |M| + k - |M^* \cap M| \\ &= |M - M^*| + k. \end{aligned}$$

- The edges in $M^* \oplus M$ can be classified as one or more of the following categories with edges alternating from $M^*$ and $M$: (a) even length cycles, (b) even-length paths and odd-length paths.

  No vertex can have more than two incident edges from $M^* \oplus M$.

- Note that no odd-length path in $M^* \oplus M$ can start (and hence end with) an edge from $M$.

  Such a path would be an augmenting path with respect to $M^*$. But, $M^*$ is a maximum matching.

- Note that the collection of all odd-length paths are vertex-disjoint augmenting paths with respect to $M$ each contributing one more $M^*$-edge than $M$-edge. Since there are $k$ more $M^*$-edges than $M$-edges, the number of such augmenting paths must be $k$.

**Claim:** The running time for this algorithm is $O(\sqrt{n}m)$.

**Proof:** First note that each phase can be executed in $O(m)$ time. It remains to show that 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. By Lemma 2, $M^* \oplus M$ has $|M^*| - |M|$ vertex-disjoint augmenting paths. By Lemma 1, each of these augmenting paths has length $\geq 2\sqrt{n}+1$. Therefore, the number of 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.