Matching

A matching in an undirected graph is a set of edges, no two having a common end vertex.

Bipartite graph: vertices can be partitioned into two sets, such that every edge has one end vertex in each set.

Maximum cardinality matching: find a matching containing as many edges as possible.

Maximum weight matching: in a graph with edge weights, find a matching with maximum total weight.

Bipartite vs. general graphs
<table>
<thead>
<tr>
<th>Bipartite</th>
<th>General</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
<td>Hopcroft &amp; Karp, 1971</td>
</tr>
<tr>
<td></td>
<td>$O(n^{1/2}m)$</td>
</tr>
<tr>
<td>Weighted</td>
<td>Fredman &amp; Tarjan, 1984</td>
</tr>
<tr>
<td></td>
<td>$O(n^2 \log n + nm)$</td>
</tr>
<tr>
<td></td>
<td>Gabow, 1985</td>
</tr>
<tr>
<td></td>
<td>$O(n^{2/3}m \log C)$</td>
</tr>
<tr>
<td></td>
<td>Gabow &amp; Tarjan, 1987</td>
</tr>
<tr>
<td></td>
<td>$O(n^{1/2}m \log (nc))$</td>
</tr>
</tbody>
</table>
Augmenting Paths

\[ \begin{align*}
\text{free} & \quad \Rightarrow \quad \text{one more matched edge} \\
\text{unmatched} & \quad \text{matched}
\end{align*} \]

\[ M = \text{any matching} \quad \bar{M} = \max \text{(card.) matching} \]

\[ M \oplus \bar{M} = \text{edges in exactly one of } M, \bar{M} : \]

\[ \bar{M} \quad \text{subgraph, all degrees} \leq 2 : \]

\[ \begin{align*}
M & \quad +1 \\
\bar{M} \quad -1
\end{align*} \]

\[ \bar{M} \text{ excess} \]

\[ \begin{cases} 
0 & \\
-1 & \text{M-augmenting}
\end{cases} \]

If \(|\bar{M}| - |M| = k\), \( M \oplus \bar{M} \) contains \( k \) M-aug. paths
Max card matching

Begin with empty matching.
Repeatedly find an augmenting path, augment.
Stop when no more augmenting paths.

Bipartite case:
O(n) time per augmentation.
O(n) augmentations
⇒ O(nm) time total.
Bipartite case faster

Build layered subgraph containing all shortest any paths by BFS

\[ A \quad B \quad A \quad B \quad A \quad B \]

\[ \text{free} \quad \text{free} \]

Find any paths in \( S \) 1 at a time by DFS

Total time per phase \( \leq O(n) \).

Length of shortest any path strictly increases after a phase

\( O(\sqrt{n}) \) phases \( \Rightarrow \) \( O(\sqrt{n}m) \) time
2 $\sqrt{n}$ phases:

Each phase increases matching size.

If $|\overline{M}| - |M| > \sqrt{n}$, $M \oplus \overline{M}$ contains $> \sqrt{n}$ any paths, at least one of length $< \sqrt{n}$ (only $n$ vertices).

$\Rightarrow$ After $2 \sqrt{n}$ phases, shortest any path has length $\geq \sqrt{n} \Rightarrow$ within $\sqrt{n}$ of max $\Rightarrow \leq \sqrt{n}$ more phases.
Each phase increases any path length. Let $d(v)$ be shortest dist from an $A$-free vertex to $v$ via an alternating path.

$d(v)$ strictly increase along any shortest any. path. New edges created by a shortest any. go from larger to smaller $d(v)$. Thus no shorter any. path created by a shortest any; after a phase, every any path contains at least one edge from larger to smaller $d(v)$ longer path.
Nothing better is known, even though the sum of lengths of shortest any paths is $O(n\log n)$.

Note: $k$ phases $\Rightarrow$ max to within $(1 - \frac{1}{k})$ factor: fast approximation

Generalizes to general graphs, weighted matchings, shortest paths, max flows $O(\sqrt{m}) \times \alpha$ and/or log factors
Max card matching on general graphs

Basic problem: how to find one any path
(a vertex can be an A-vertex or a B-vertex; a priori, one doesn't know which)

Edmonds: blossom-shrinking to find any paths

[Diagram with labels: stem, free, base, only unmatched edges, odd-length alt. cycle]
Thm: Let $G'$ be formed from $G$ by shrinking a blossom. Then $G'$ contains an any path iff $G$ does.

Pf. If $G'$ contains an any path, then $G$ does: expand blossom, link broken ends of path by going around blossom in correct direction (one broken end is blossom base).

Other direction is the hard part. If the blossom has a non-trivial stem, swap edges along it to make the base of the blossom free, obtaining $G_1$ from $G$ (and $G'_1$ from $G'$).
$G \rightarrow G'$

$G_1 \rightarrow G_1'$

$G(G')$ has an any path iff $G_1 (G_1')$ does. Thus we need only show that if $G_1$ has an any path, so does $G_1'$. Thus suppose $G_1$ has an any path. Either it is an any path in $G_1'$ or it hits the blossom, in which case the path from the end not the blossom base until it first hits the blossom is an any path in $G_1'$. 


Edmonds' alg to find an any path via blossom-shrinking (DFS version)

Start at any free vertex.
Grow an alt. search path.
If an edge extending the path hits the path, shrink a blossom if the path is of odd length; otherwise discard the edge.
When reaching a new free vertex, stop with success.
When at a vertex or blossom with no unexplored edges, delete the vertex or blossom.
After deleting a free vertex, start a new search at an undeleted free vertex.
Time per any path: \( O(m \cdot \alpha(n)) \)

\( \text{(need set union to maintain blossoms)} \)

Total time: \( O(n \cdot m \cdot \alpha(n)) \)

Can improve to take advantage of shortest any path idea:

very complicated
The cost scaling approach gives a time of $O(\sqrt{nm} \log(n \epsilon))$
for the assignment problem (weighted bipartite matching).

Compare with Hopcroft-Karp bound of $O(\sqrt{nm})$ for
unweighted bipartite matching, and

Fredman-Tarjan bound of $O(nm+n^2 \log n)$ for
a nonscaling algorithm.

For nonbipartite weighted matching, we obtain a time of

$O(\sqrt{n\log(n)} \log(n \epsilon) \log(n \log n))$

Compare with Micali-Vazirani bound of $O(\sqrt{nm})$ for
unweighted matching, Gabow-Galil-Spencer
bound of $O(nm \log \log \log m + n^2 \log n)$ for a
nonscaling algorithm.