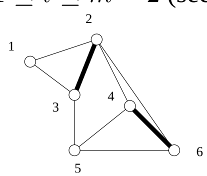
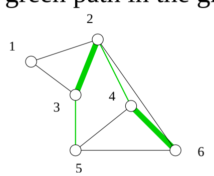
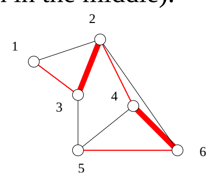
Maximum matching using alternating paths

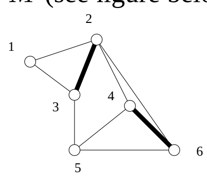
Let G be a graph and M any matching in G (see figure to the left). A path $P = v_1, \ldots, v_m$ is said to be an *alternating path with respect to* M or an M*-alternating path* if $\{v_i, v_{i+1}\} \in M$ if and only if $\{v_{i+1}, v_{i+2}\} \notin M$ for $1 \leq i \leq m-2$ (see the green path in the graph in the middle).

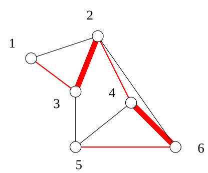


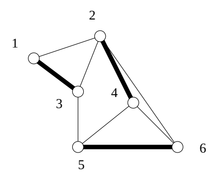




A vertex v is *exposed* (or *unmatched*, *unsaturated*, *not covered*) with respect to matching M if no edge of M is incident with v. Clearly if G contains an M-alternating path joining two exposed vertices then M cannot be a maximum matching, for one can easily obtain a larger matching by simply removing the lines in $P \cap M$ and adding those in $P - M$ (see figure below).







An alternating path joining two exposed vertices is called an M*-augmenting path* (see the red path in the graph in the middle).

Berge's result

A matching M in a graph G is a maximum matching if and only if there exists no augmenting path in G with respect to M.

We have dealt with the "only if" part (to re-iterate, by contradiction if there was an augmenting path we could enlarge the matching), we need to prove the "if" statement.

Let M be a matching in a graph G and assume there exists no augmenting path in G with respect to M.

Then, consider any maximum matching M' of G and take the graph[a] $M \oplus M'$.

[a] If G_1 and G_2 are graphs on the same set of vertices V, then $G_1 \oplus G_2$ contains all the edges in $E(G_1) \cup E(G_2)$ but those in $E(G_1) \cap E(G_2)$.

The components of such a graph are cycles and path (because if there was a vertex of degree three either M or M' wouldn't be a matching).

Furthermore, none of these components can have odd length for otherwise either M or M' would have an augmenting path and this is not possible for M (by assumption), and it is not possible for M' because M' is a maximum matching (we have already proved, in the contrapposite form, that maximum matchings do NOT have augmenting paths).

But then $|M| = |M'|$, because each component of $M \oplus M'$ contains the same number of edges from M and M'. So M is maximum cardinality as well!

... still we have NO efficient algorithm for finding a maximum matching of a bipartite graph!

But there is a way ...

Essentially all we need is a data structure to handle the augmenting paths in a graph.

A *forest* F is a collection of trees ...

Let $G = (V_1, V_2, E)$ be given along with some matching M (this may be found by GREEDY-MATCHING1 or it could indeed be just a single edge of G).

Let U_i be the set of exposed vertices in V_i, for $i = 1, 2$.

We build a (maximal) forest F in G with the following properties:

(P1) each vertex in $V(F) \cap V_2$ has degree two and belongs to $V(M)$;

(P2) each component of F contains a point in U_1.

Detailed forest construction

We build (greedily) a collection of trees T rooted at u for each $u \in U_1$.

Vertices at different depth in each tree will belong to different color classes.

Level zero is a vertex $u \in U_1$.

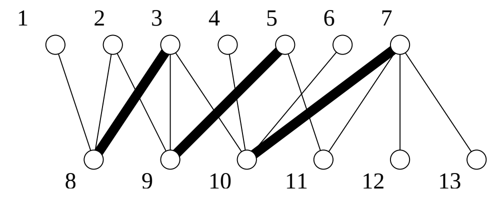
Vertices at level one are all vertices adjacent to u.

We further develop only those vertices adjacent to u which are in $V_2 \setminus U_2$ by adding for each leaf the corresponding edge in the matching (so we are back in V_1).

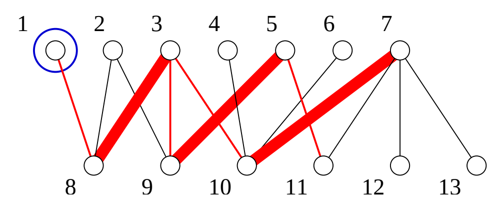
No contraint on the degree of vertices in V_1 then for every vertex in V_1 we put in the tree at the next level all vertices adjacent.

We stop building a tree if we find at least one leaf in U_2 (in which case we have found an augmenting path!) or we can't expand the tree any further (in which case we may start building another tree).

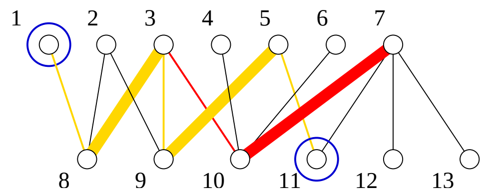
Example



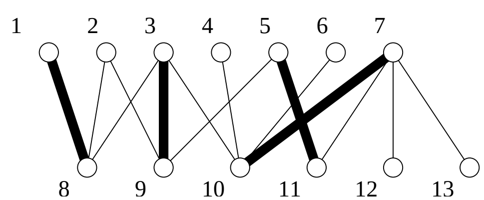
Example



Example



Example



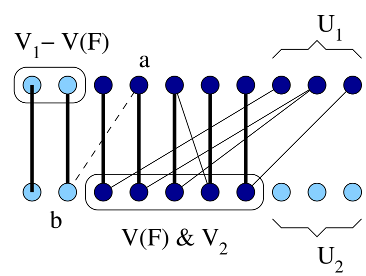
Hungarian algorithm

The approach given below seems to have first appeared in the work of König (1916, 1931, 1936) and Egerváry (1931) who reduced the problem with general non-negative weights on the edges to the unweighted case.

HUNGARIAN-MATCHING (G)
    let M be any matching in G
    repeat
        form a maximal forest F having
            properties (P1) and (P2)
        if there is an edge joining $V(F) \cap V_1$ to
                a vertex in U_2
            $M \leftarrow \text{Augment}(M, F)$
        else return M
    until TRUE

Correctness

Let $G = (V_1, V_2, E)$ be a bipartite graph, M be a matching in G, U_i be the set of vertices unmatched in V_i and F a maximal forest built by algorithm HUNGARIAN-MATCHING. Then M is a maximum matching if and only if no vertex in U_2 is adjacent to a vertex in F.



Proof

$(|M| = \nu(G) \Rightarrow \nexists x \in V(F), u_2 \in U_2 \; \{x, u_2\} \in E)$

Let's assume there exists $x \in V(F), u_2 \in U_2$ such that $\{x, u_2\} \in E$. Then, by the way in which F has been constructed there is a path P from x to some $u_1 \in U_1$. Therefore $P \cup \{x, u_2\}$ is an augmenting path in G. M is not maximum. Contradiction.

$(\nexists x \in V(F), u_2 \in U_2 \; \{x, u_2\} \in E \Rightarrow |M| = \nu(G))$

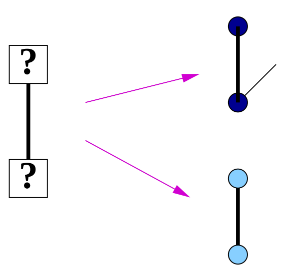
Define $\quad X = V_1 \setminus V(F) \qquad Y = V(F) \cap V_2.$

We will prove

1. $|X \cup Y| = |M|$
2. $X \cup Y$ is a vertex cover of G.

The result (and the correctness of the Hungarian algorithm) will follow from König's theorem, because we have

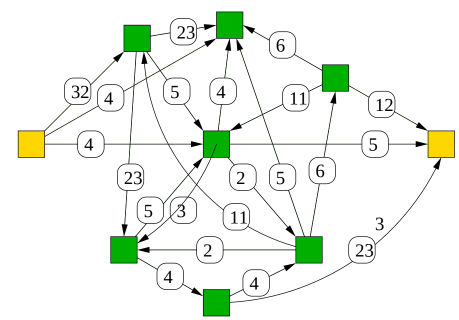
$$\tau(G) \leq |X \cup Y| = |M| \leq \nu(G)$$



Let $e \in M$. Then either e is incident to a $y \in Y$ (and in this case the other endpoint of e cannot be in X) or e is incident to some $x \in X$ (and hence both its endpoints are NOT part of $V(F)$). Therefore $|X \cup Y| = |M|$.

Suppose now that there exists an edge $\{a, b\}$ that is not covered by $X \cup Y$, with, say, $a \in V_1$. It must be the case that $a \in V(F)$ and $b \notin V(F)$ and $b \notin U_2$. By hypothesis M covers b by some edge $\{a', b\}$, and a' must be different from a. In such case we can extend F by adding $\{a, b\}$ and $\{a', b\}$. Contradiction!

Maximum matching using flows



Let $D = (V, E)$ be a directed graph with two distinguished vertices s (the *source*) and t (the *sink*). Let $c : E(D) \rightarrow \mathbb{R}^+$ be a function which associates with each edge $(u, v) \in E(D)$ a non-negative real number $c(u, v)$ called the *capacity* of the edge. The quadruple (D, c, s, t) is called a *network*.

Any function $f : E(D) \rightarrow \mathbb{R}^+$ is called a *flow* in (D, c, s, t) if it satisfies the following properties:

1. $\sum_u f(u, v) = \sum_w f(v, w)$ for all $v, w \in V(D) \setminus \{s, t\}$ (conservation of flow), and
2. $f(u, v) \leq c(u, v)$ for all $(u, v) \in E(D)$.

The *value* of a flow f is the quantity $\sum_u f(s, u) - \sum_u f(u, s)$.

An important computational problem (a.k.a. the *network flow problem*) is that of determining a flow of maximum value that can be "applied" to a given network D.

The best existing algorithms for solving the bipartite matching problem are based on network flows through the following reduction.

Given a bipartite graph $G = (V_1, V_2, E)$ one can define a network D_G as follows:

1. the set of vertices in the network is $V_1 \cup V_2 \cup \{s, t\}$ where s and t are two "new" vertices that will act as source and sink respectively.
2. the set of directed edges of D_G is formed by the edges of G directed from V_1 to V_2, $|V_1|$ "new" edges from the source to V_1, namely and edge (s, v) for each $v \in V_1$, and, similarly $|V_2|$ "new" edges from V_2 to the target t.
3. $c(u, v) = 1$ for every edge in the resulting graph.