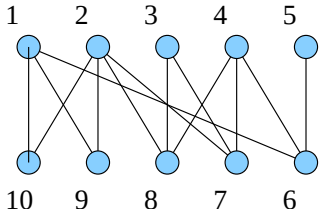
The Hungarian algorithm

We will start today's lecture by running through an example. Consider the graph below. We will compare the results of GREEDY-MATCHING1 with those of HUNGARIAN-MATCHING.

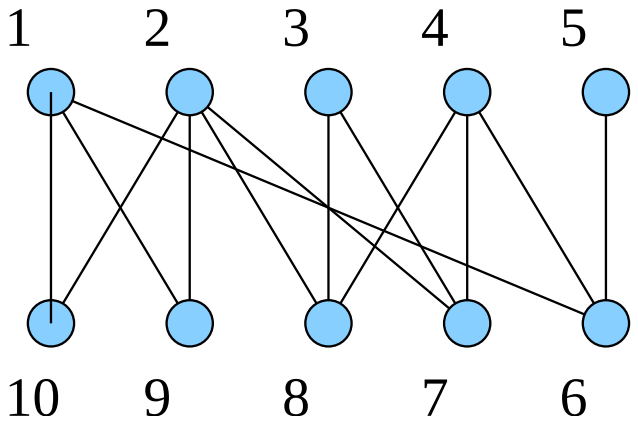


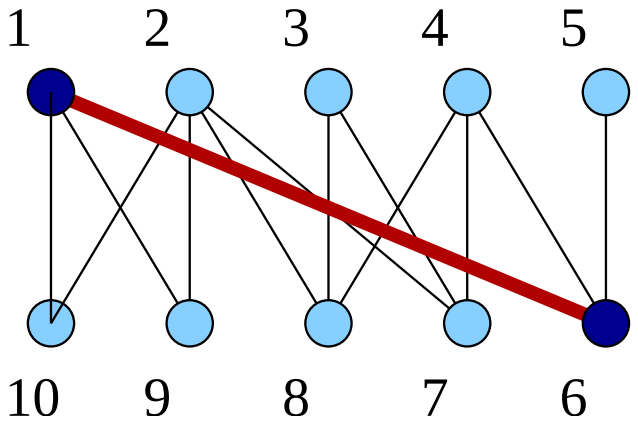
You are warmly invited to try this at home with paper and pencil first.

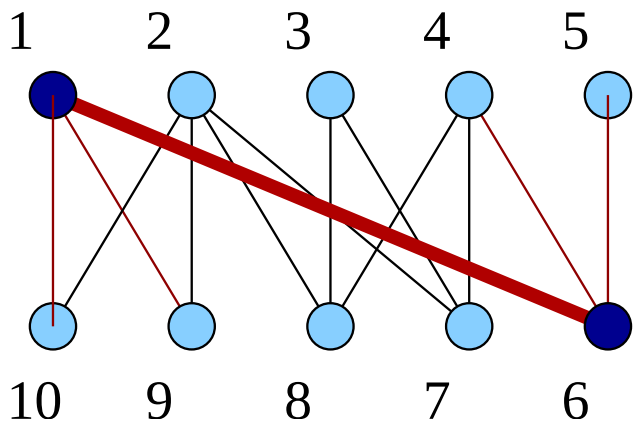
GREEDY-MATCHING1 (G)
$M \leftarrow \emptyset$
while $E(G) \neq \emptyset$
(*) pick the lexicographically first $e \in E(G)$
$M \leftarrow M \cup \{e\}$
remove e and all
edges adjacent to e
from $E(G)$
return M

Assume the edges in the given graph are *ordered* pairs (i.e. put a top-down direction on them).

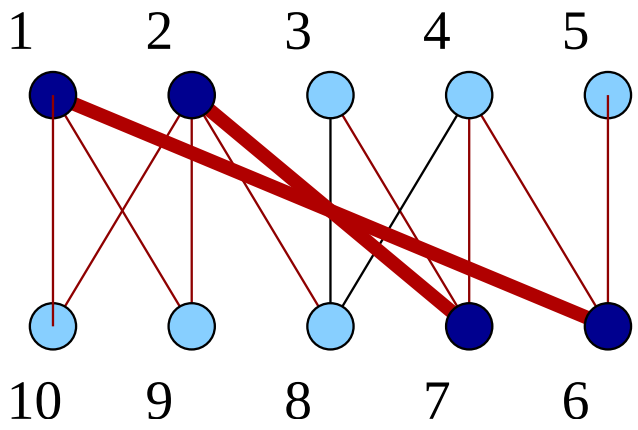
$E(G) = \{(1,6),(1,9),(1,10),(2,7),(2,8),(2,9),(2,10),(3,7),(3,8),(4,6),(4,7),(4,8),(5,6)\}$



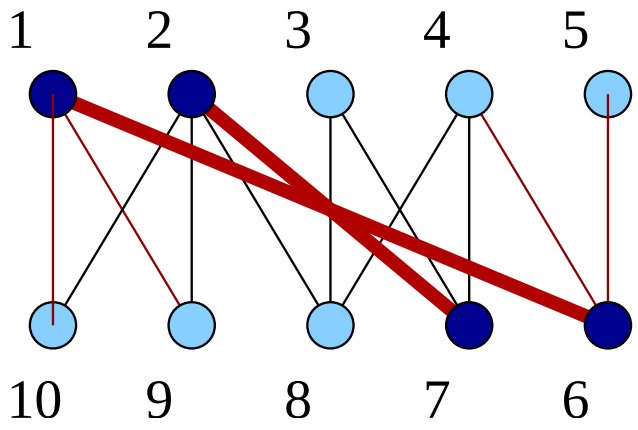




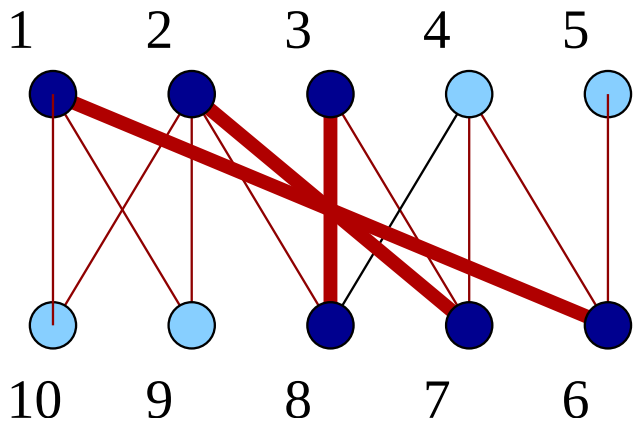
5



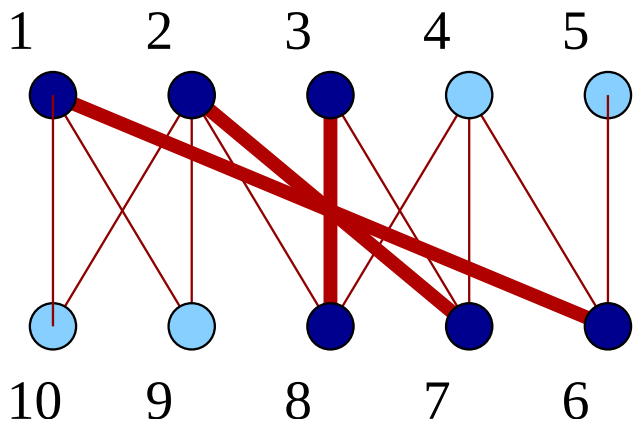
7



6



8



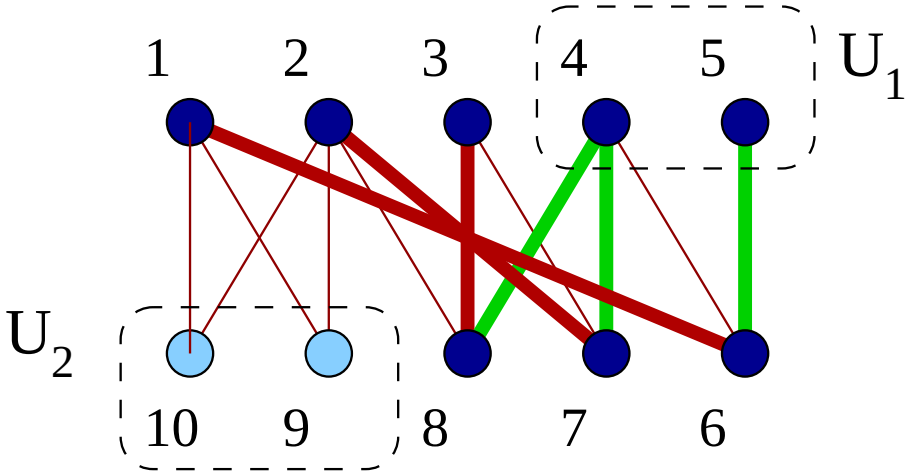
9

Hungarian algorithm

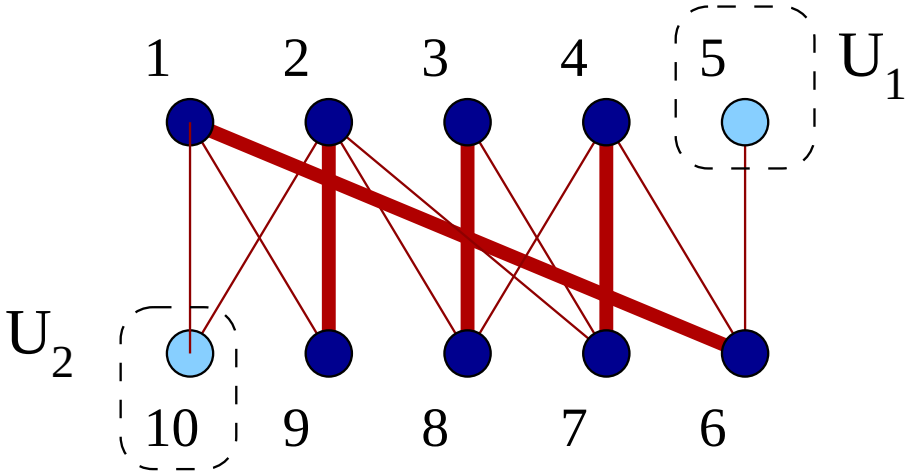
The approach given below seems to have first appeared in the work of König (1916, 1931, 1936) and Egerváry (1931) who reduced the problem with general non-negative weights on the edges to the unweighted case.

HUNGARIAN-MATCHING (G)
  let M be any matching in G
  repeat
    form a maximal forest F having
      properties 1. and 2.
    if there is an edge joining $V(F) \cap V_1$ to
        a vertex in U_2
      $M \leftarrow$ Augment(M, F)
    else return M
  until TRUE

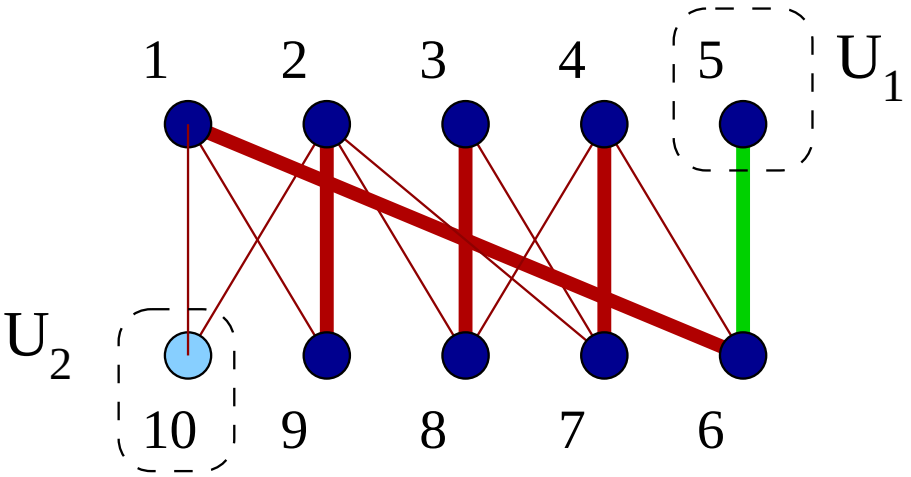
10

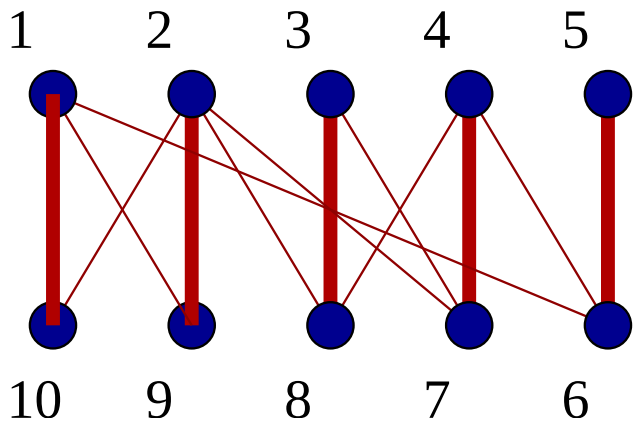


11



12





Maximum matching in general graphs

We presented an algorithm for finding a maximum matching in a bipartite graph.

From a mathematical point of view, this algorithm is essentially no more involved than the proof of König's equality.

For non-bipartite graphs the situation is quite different. Known poly-time algorithms for finding a maximum matching in a general graph are among the most involved combinatorial algorithms.

Most of them are based on augmentation along alternating paths. But important new ideas are needed to turn these tricks into polynomial time algorithms.

Edmonds' algorithm

The first polynomial time matching algorithm for general graphs was constructed by Edmonds.

In this algorithm the key idea of "shrinking" certain odd cycles was introduced.

Up to the present time most matching algorithms – certainly the most successful ones – are based (implicitly or explicitly) on this idea.

We begin with a lemma which will enable us to reduce the size of the graph under consideration in many cases.

The lemma help us understand the crucial step of "cycle shrinking" and lends us confidence that we are not losing necessary information when carrying out such shrinking.

Shrinking Lemma. Let G be a graph and M a matching in G. Let Z be a cycle of length $2k+1$ which contains k lines of M and is vertex-disjoint from the rest of M. Let G' be the graph obtained from G by shrinking Z to a single vertex. Then $M' = M \setminus E(Z)$ is a maximum matching in G' if and only if M is a maximum matching in G.

Proof

($|M'| = \nu(G') \Rightarrow |M| = \nu(G)$) Assume that $|M| < \nu(G)$. Then there exists an augmenting path P relative to M. Two cases arise:

P **vertex-disjoint from** Z In such case P is also an M'-augmenting path, and hence $|M'| < \nu(G')$. Contradiction!

P **does intersect** Z W.l.o.g. there must be an endpoint, say x, of P that is not in Z. Let z be the first vertex in the path P which also belongs to Z. The path Q from x to z is mapped onto an M'-augmenting path when Z is contracted. Hence $|M'| < \nu(G')$. Contradiction!

($|M| = \nu(G) \Rightarrow |M'| = \nu(G')$) This time assume M' is not maximum. Take a maximum matching N' in G'. Then expand Z and define a matching N in G. Then $|N| = |N'| + k > |M'| + k = |M|$, i.e. M is not a maximum matching. Contradiction!

Algorithm description

We now turn to an informal description of Edmonds Matching Algorithm.

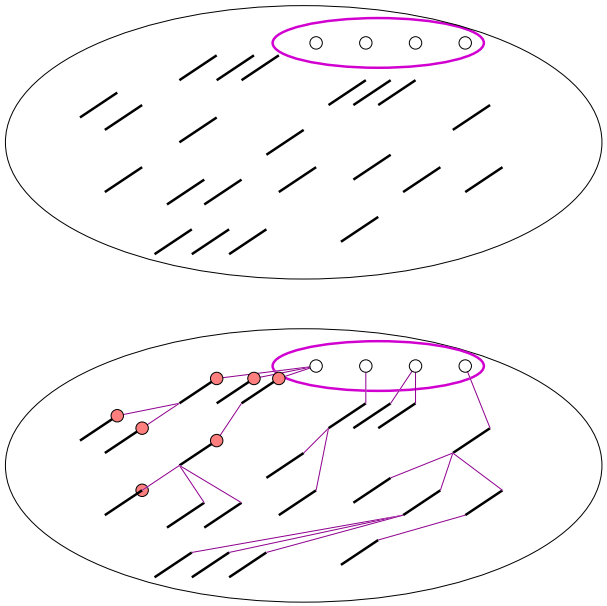
We are given a graph G. Let M be a matching in G.

If M is perfect we are done!

Otherwise let S be the set of vertices that are not covered by M.

Construct (as in the bipartite case) a forest F such that every connected component of F contains exactly one vertex[a] of S, every point of S belongs to exactly one component of F, and every edge of F which is at an odd distance from a point in S belongs to M.

[a] It may be defined as the root of the component under consideration.



Properties of F

Every vertex of F which is at an odd distance from S has degree two in F.

Such vertices will be called *inner* vertices, while the remaining vertices in F will be called *outer* vertices (in particular all vertices in S are outer).

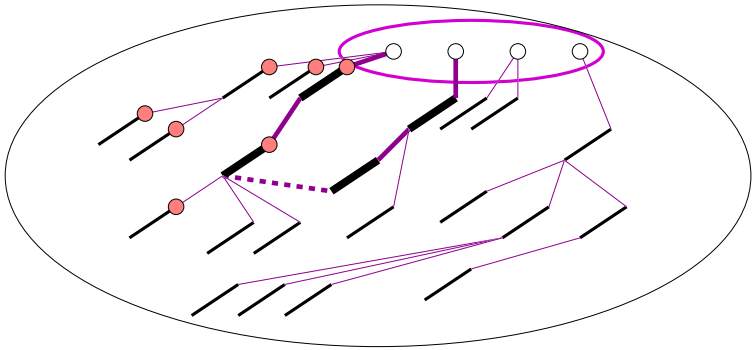
Such a forest is called M*-alternating forest*.

Clearly, the (trivial) forest with vertex set S and no line is an M-alternating forest (although not a very useful one!).

"External" outer vertices

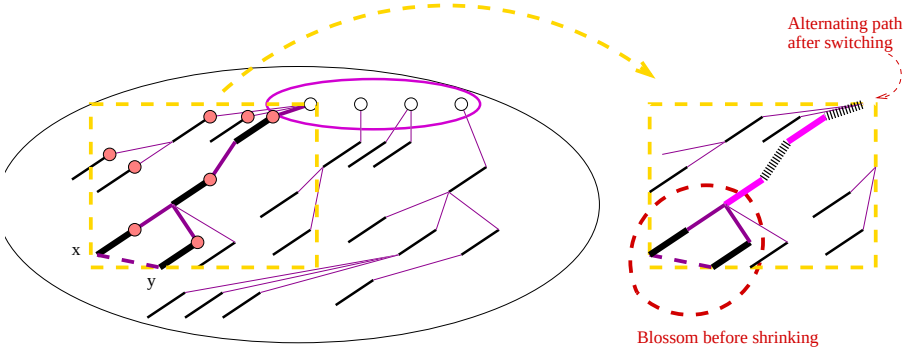
Next we consider the neighbours of outer vertices. If we find an outer vertex x adjacent to a vertex y not in F, then we can enlarge F by adding the edges $\{x, y\}$ and $\{y, z\} \in M$.

"Adjacent" outer vertices in different components



If F has two adjacent outer vertices x and y belonging to different components of F, then the roots of these two components of F are connected by an M-augmenting path. We can obtain a larger matching! And after this we restart the process by constructing a new (smaller) F.

"Adjacent" outer vertices in the same component



If F has two outer vertices x and y in the same connected component which are adjacent in G, then let C be the cycle formed by the line $\{x, y\}$ and the path from x to y in F. Let P denote the (unique) path[a] in F connecting C to a root of F. Clearly P is an M-alternating path, so if we "switch" on P, we obtain another matching M_1 of the same size as M. But M_1 and C satisfy the conditions of the shrinking Lemma, and so if we shrink C to a single point to obtain a new graph G', we have reduced the task of finding a matching larger than M in G to the task of finding a matching larger than $M_1 \setminus E(C)$ in the smaller graph G'.

[a] We allow C to pass through the root, in which case P consists of a single point.

Finally, if every outer vertex has only inner vertices as neighbours, then we claim that the matching M is already maximum. For suppose that F contains m inner vertices and n outer vertices. Clearly $|S| = n - m$. Furthermore if we delete all the inner vertices of F from G, the remaining graph will contain all the outer vertices of F as isolated points. Hence $\text{def}(G) \geq n - m = |S|$. But M misses exactly $|S|$ vertices, and so it must be a maximum matching.

In summary we can always do one of the following:

- enlarge F,
- enlarge M,
- decrease $|V(G)|$, or
- stop with a maximum matching!

Thus it is clear that the algorithm terminates in polynomial time with a maximum matching in G.