Maximum (cardinality) matchings in graphs

Design an algorithm that returns a maximum cardinality matching in a given graph.

Q1 Does it work? Are you sure it returns a maximum cardinality matching?

Q2 How long does it take?

Maximal matchings

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

GREEDY-MATCHING2 (G)
$M \leftarrow \emptyset$
while $E(G) \neq \emptyset$
(*) pick $e \in E(G)$
if e is not adjacent to any $f \in M$
$M \leftarrow M \cup \{e\}$
$E(G) \leftarrow E(G) \setminus \{e\}$
return M

Remarks

- in both cases M is a maximal matching (Exercise: prove it!)
- Question: does any of these algorithm return a maximum cardinality matching?

  The question is ill-posed since indeed each of the algorithms above really represents a class of algorithms. In fact I gave no details on how to perform the (*) step. Different algorithms are obtained implementing step (*) according to one of the following rules:
  - Choose e at random among the available edges.
  - Choose e as the "first" available edge.
  - Choose $e = \{u, v\}$ as the "first" edge among those for which $\deg(u) + \deg(v)$ is minimised.

Once we made either pseudo-codes into precise algorithms, what can be said about the size of resulting matching?

Claim. Any maximal matching M in a given graph G has at least half the edges of a MAXIMUM cardinality matching. In symbols, $|M| \geq \frac{\nu(G)}{2}$, for any maximal matching M in G.

WHY?

Supporting argument

Let M_1 and M_2 be two maximal matchings in G (in particular think of M_1 as a minimum cardinality maximal matching and M_2 as a MAXIMUM cardinality matching).

Some edges may be both in M_1 and M_2. We focus on the edges in M_2 which are NOT in M_1. Let $e \in M_2 \setminus M_1$.

By the maximality condition, the set $M_1 \cup \{e\}$ is not a matching anymore. Hence there exists $\phi(e) \in M_1$ which is adjacent to e and since M_2 is a matching (i.e. a collection of *independent* edges), $\phi(e) \in M_1 \setminus M_2$.

Indeed ϕ defines a function from $M_2 \setminus M_1$ to $M_1 \setminus M_2$.

Let f be one of the edges in the range of ϕ. A bound on the number of edges $e \in M_2 \setminus M_1$ that can be the pre-image of $f \in M_1 \setminus M_2$ is needed. In the worst case there can be at most 2 such e (incident to each end-point of f).

Hence $|M_2 \setminus M_1| \leq 2|M_1 \setminus M_2|$.

Therefore

$$|M_1| = |M_1 \cap M_2| + |M_1 \setminus M_2| \geq |M_1 \cap M_2| + \frac{|M_2 \setminus M_1|}{2} \geq \frac{|M_2|}{2}.$$

Independence Systems

An *independence system* is a pair $(X, \mathcal{F})$ where X is a finite set and $\mathcal{F}$ a collection of subsets of X with the property that whenever $F \subset G \in \mathcal{F}$ then $F \in \mathcal{F}$. The elements of $\mathcal{F}$ are called *independent sets*.
A *maximal independent set* is an element of $\mathcal{F}$ that is not a subset of any other element of $\mathcal{F}$.

The set of all matchings in a graph G, which we denote by $\mathcal{M}(G)$, forms an independence system (Exercise: prove it!).

Theorem. (Korte and Hausmann) $\nu(G) \leq 2\ \beta(G)$ for any graph G.

Maximum Matching in trees

Trees are very simple graphs, with a nice hierarchical structure.

The maximum matching problem can be solved very efficiently on trees.

We will present a simple greedy solution to the problem of finding a largest matching in a tree.

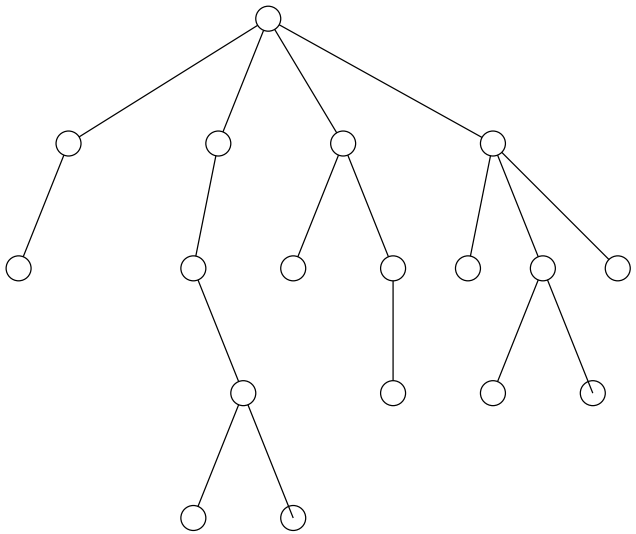
Algorithm

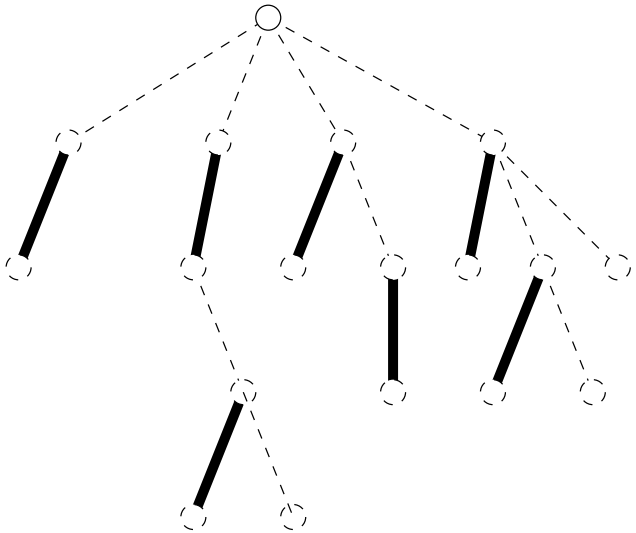
We assume the tree is given along with a vertex, chosen to be its root.

If all the children of a vertex v in the tree are leaves, then the subgraph induced by the vertex and its children is called a *fan*. The vertex v is called the *centre* of the fan.

The algorithm repeatedly finds a fan that is furthest from the root, chooses one of its edges to be added to the matching and then removes all other edges of the fan from the tree along with the edge from the centre of the fan to its father.

Remark. Notice that this is just a slight modification of GREEDY-MATCHING1: at each step we are choosing an edge to be added to the matching that is as far as possible from the root of the tree.





Analysis

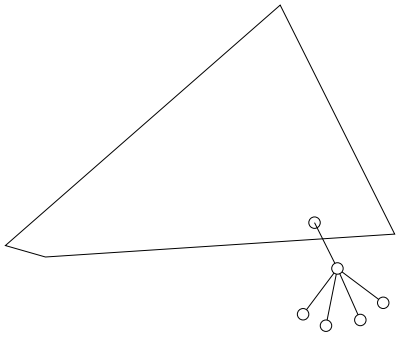
We need to prove the two usual properties:

Greedy-choice an optimal solution can be found by making greedy choices at each step.

Optimal substructure an optimal solution contains within it optimal solutions to subproblems.

Remember! These are properties of the solutions to the problem, not of the particular algorithm!!

The blob with the fan sticking out of it!



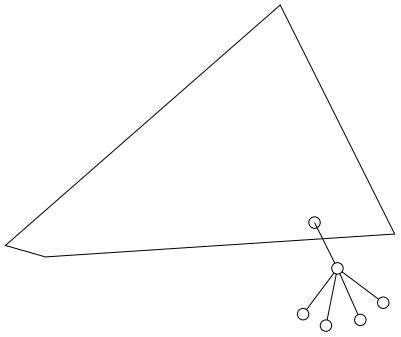
It's the deepest fan.

Our greedy algorithm will pick, say, the edge to the leftmost leaf.

Any matching which claims to be optimal MUST hit at least one of those edges!

Claim. Swapping such an edge with the one chosen by our algorithm can only improve things.
In the best case it will "free" the father of the centre of the fan!

The blob with the fan sticking out of it!



Claim. If our algorithm claims to be optimal, then it must find an optimal matching in the "blob".
If not, well, then we can prove the algorithm is not so good after all!
The algorithm returns a matching M of the whole tree that, by definition, has an edge e from the deepest fan and some matching M' covering the rest of the edges.
Assume M' is not best possible!
Well, pick a maximum matching of the blob M^* (notice that $|M^*| > |M'|$), then $|\{e\} \cup M^*|$ must be larger than $|M| = |\{e\} \cup M'|$, contradiction!