Trees and Prüfer codes

Let us start with the definition of a tree.

Definition. A graph is a *tree* if it is connected and contains no cycle. A *forest* is a graph which contains no cycle.

So, the components of a forest are trees.

We have already observed that deleting an edge, which is contained in a cycle, from a connected graph results in a connected graph. This shows that we can delete edges as long as the graph contains a cycle and finally get a spanning tree. This gives the following.

Proposition. *Every connected graph contains a spanning tree.*

The following is an alternative definition of a tree.

Proposition. *A graph is a tree if and only if for any two points there is a unique path joining them.*

If there are two paths between two points, then there is a cycle in the graph (the concatenation of one of the paths and the reverse of the other paths gives a closed walk which contains an edge which appears once).

Let us see some alternative definitions of trees.

Proposition. *A tree is a minimal connected graph. More precisely, a graph is a tree, if and only if it is connected but deleting any of its edges yields a non-connected graph.*

A tree is a maximal cycle-free graph. More precisely, a graph is a tree, if and only if it contains no cycle but adding any new edge yields a cycle.

We prove the first statement. If we delete an edge from a tree then there can be no path between the endpoints of that edge (otherwise that path with the deleted edge would give a cycle in the original tree). On the other hand, we have already seen that deleting an edge from a cycle would yield a connected graph.

Proposition. *Every tree on $n \geq 2$ vertices contains at least two vertices of degree one. They are sometimes called* leaves.

Consider a longest path in the graph. They are only connected to vertices of this path, but the graph contains no cycles.

Proposition. *A tree on n vertices contains $n-1$ edges.*

The proof goes by mathematical induction. Delete a vertex of degree one (this menas that we also delete the edge emanating from this vertex). The remaining graph clearly contains no cycle, and it is easy to see that it is connected (the path between any pair of points cannot contain the deleted vertex of degree one.) So the remaining graph has $n-1$ vertices and $n-2$ edges by the induction hypothesis, and the original graph has one more vertices and edges.

Using essentially the same idea one can prove that there is a simple recursive construction for trees.

Proposition. *Every tree on n vertices can be obtained from a tree on $n-1$ vertices by the following tree-growing procedure:*

- start with the tree F on $n-1$ vertices and add a new vertex w

- pick a vertex v of F and join it with w.

It is also true that this procedure yields a tree on n vertices.

We have noted before that any tree on n vertices has a vertex w of degree 1 (leaf). Deleting w yields a tree F on $n-1$ vertices.

We can combine parts of the original definition with the number of edges and find some alternative definitions of trees.

Proposition. *A graph on n vertices is a tree if and only if it is connected and has $n-1$ edges.*

A graph on n vertices is a tree if and only if it is contains no cycles and has $n-1$ edges.

So, we can now imagine connected graphs. They consist of a spanning tree plus extra edges. The spanning tree can be visualized in levels. Start from a vertex, the next level contains its neighbours, then the neighbours of neighbours and so on. This is how we listed the vertices of a graph (a component) by using breadth-first search.

Our next question is to count trees. This is dfficult if the vertices are not labeled. If they are labeled (by $1, 2, \ldots, n$, say), then the following theorem by Cayley gives their exact number.

Theorem (Cayley)**.** *The number of labeled trees on n vertices is n^{n-2}*

This theorem will be proved by constructing the so-called Prüfer code of the tree. In case of a tree on n vertices, the Prüfer code is a sequence of length $n-2$, and the elements of the sequence are the labels of vertices (without any restriction, so we have n possibilities for each element of the sequence). We will sjow that there is a bijection between labeled trees and Prüfer codes, so the number of trees is indeed n^{n-2}.

Assume we have a tree, whose vertices are labeled with $1, 2, \ldots, n$. Pick the leaf (vertex of degree one) with the smallest label. Write this in the first row of a $2 \times (n-1)$ matrix as the first element. The label of its uniques neighbour

will be written in the second row (as first element). Intuitively, we can say that we tear off the leaf. Continue to fill the two rows in the same way. This means that the two vertices in a column correspond to an edge of our tree. Since at each step the actual tree has at least two leaves, the above procedure associates a $2 \times (n-1)$ matrix to every tree. The *Prüfer code* itself will be the first $n-1$ elements in the second row. (The $2 \times (n-1)$ matrix itself is called the *extended, two-row P-code*.

As an example, the path $1-2-3-\ldots n$ will have P-code $2, 3, \ldots, n-1$.

The most important properties of the Prüfer code are listed on the explanatory figures (in another file). In general, we can use the Prüfer code when we have to solve an exercise about trees and degrees of vertices. As an example, let $d_1, \ldots d_n$ be a degree sequence with $d_i > 0$. Then there is a tree with these degrees: indeed, build a P-code in which „i" occurs ($d_i - 1$ times. This is a sequence of length $n-2$, so there is a tree belonging to this P-code. Similarly, one can show that a tree having a vertex of degree k has at least k leaves. We have to recall, that the vertex of degree k occurs $k-1$ times in the P-code and even if the reamining $(n-2)-(k-1) = n-k-1$ elements are different, there are at least $n-1-(n-k-1) = k$ labels that do not occur in the P-code. These are the leaves of the original tree.