LECTURE 6.

Abstract. Trees: elemenraty facts and the Prüfer code.

A graph (finite, undirected) is thought to be a finite set of points (called vertices), some of them being joined with lines (called edges). There may be lines joining a vertex with itself (loops); two vertices may be joined by more than one line (multiple, or parallel, edges).

A *simple path* in a graph is a sequence $e_1, \dots, e_k$ of pairwise distinct edges such that e_i and e_{i+1}, for all $i = 1, \dots, k$, have a common vertex, and e_i and e_j with $j \neq i \pm 1$ have no common vertices. A *cycle* is like a simple path with one exception: e_1 and e_k have a common vertex, too. A *tree* is a graph in which any two vertices can be joined by a simple path, and there are no cycles.

Theorem 1. *In a tree any two vertices are joined by a unique simple path.*

The idea of proof. Suppose there are two simple paths; let u_1 be the first vertex where they diverge, and u_2, the first vertex on the first path that belong to the second path as well. Then parts of the paths between the vertices u_1 and u_2 form a cycle. □

A vertex in a graph is called *hanging* if it is incident to one edge only (and this edge is not a loop).

Theorem 2. *Any tree has at least two hanging vertices. Deleting a hanging vertex together with the incident edge gives another tree.*

Proof. Take the longest simple path $e_1, \dots, e_k$ in a tree (why does it exist?), and let v_0, v_1 be the starting and the final vertex of this path; we prove that both are hanging. Indeed, v_1 is incident to the edge e_k of the path; suppose it is incident to another edge, e. A simple path cannot pass a vertex twice; so, the edge e does not enter the path. Therefore, $e_1, \dots, e_k, e$ is a simple path longer than $e_1, \dots, e_k$, contrary to the choice. The proof for v_0 is similar.

If one deletes a vertex and an edge from a graph having no cycles, cycles would not appear. Any two vertices of the graph T' obtained from a tree T by deletion can be joined by a simple path in T. This path cannot pass the deleted vertex (because it is hanging), so it is a simple path in T' as well. Hence, T' is a tree. □

Corollary 1. *If a tree has n vertices then it has $n - 1$ edges.*

Proof. Induction by the number of vertices: if $n = 1$ then a tree cannot contain edges (i.e. loops). Let T be a tree with n vertices and e edges; take a hanging vertex and delete it together with the incident edge. The graph obtained is a tree T' with $n - 1 < n$ vertices; by the induction hyothesis the number of its edges is equal to $e - 1 = n - 2$, so that $e = n - 1$. □

The Prüfer code is an algorithm relating to every tree T with n vertices a sequence $b_1 \dots b_{n-2}$ of integers, $1 \leq b_i \leq n$ for all $i = 1, \dots, n - 2$. It acts as follows: take a hanging vertex v_i of T with the maximal number (among the hanging vertices), and let b_1 be the number of the other end of the single edge incident to v_i. Delete v_i and the edge and repeat the procedure for the tree T' obtained, to get b_2, then b_3, etc.

A Prüfer code behaves nicely under the deletion of a hanging vertex together with the incident edge. Namely, if $b = b_1 \dots b_{n-2}$ is a Prüfer code for a tree T then the hanging vertices are exactly vertices whose numbers *do not* enter b. If one deletes a hanging vertex with the maximal number v together with the incident edge, and renumbers the vertices of the tree T' obtained skipping v (that is, all the vertices with the numbers $v_i < v$ preserve their numbers, and every vertex with the number $v_i > v$ gets $v_i - 1$ instead), then the Prüfer code for T' becomes $b'_2 \dots b'_{n-2}$ where $b'_i = b_i$ if $b_i < v$ and $b'_i = b_i - 1$ if $b_i > v$.

Theorem 3. *For any sequence $b = b_1 \dots b_{n-2}$ of integers such that $1 \leq b_i \leq n$ for all $i = 1, \dots, n - 2$ there exists exactly one tree with n vertices having b as its Prüfer code.*

Proof. Induction by n: for $n = 3$ the statement is trivial (check!). Suppose we know the statement for sequences of any length smaller than $n - 2$; now consider b. Let v be the maximal integer from 1 to n that does not enter b (it exists because the number of terms in b is less than n). Form a sequence $b' = b'_2 \dots b'_{n-2}$ by the rule described above (if $b_i < v$ then $b'_i = b_i$ and $b'_i = b_i - 1$ otherwise). The sequence b' contains $n - 3$ terms from 1 to $n - 1$ (obviously, it cannot contain n). So, there exists a unique tree T' such that b' is its Prüfer code. Change the numbering of vertices of T' appropriately (the vertices with the numbers $v_i < v$ retain their numbers, while every vertex with the number $v_i \geq v$ gets $v_i + 1$); a new tree T'' does not have a vertex numbered v. Form now a tree T joining the vertex b_1 of T'' with the new vertex numbered v; it is easy to see that the Prüfer code of T is b. So the existence of T is proved. The uniqueness: if T has b as its Prüfer code then the Prüfer code of T' is b'. By induction hypothesis, T' is unique, has only one vertex numbered b_1 which should be joined by an edge with the hanging vertex v — thus, T is uniquely restored. □

Corollary 2. *There exist n^{n-2} different trees with n vertices numbered* 1 *to* n.

Exercises

A pair of vertices i, j of a tree T is said to form an inversion if $2 \le i < j \le n$ and the (unique) simple path joining i with 1 passes j. A tree is called monotonic if it has no inversions.

Exercise 1. a) What is the biggest possible number of inversions in a tree with n vertices? Prove that for every n there exists exactly one tree with this number of inversions. b) Form a table: how many are there trees with n vertices and k inversions, for $n \le 4$ and all possible k? c) Which sequences $b_1 \dots b_{n-2}$ are Prüfer codes of the monotonic trees? How many are there monotonic trees with n vertices? d) How to find the number of inversions in a tree T using its Prüfer code? e) How many are there trees having exactly 1 inversion? f*) Exactly 2 inversions?