Reading
January 29: Sections 4.3 and 4.4
January 31: Section 4.5
February 5: Section 4.7
February 7: Section 5.1

Homework #4. due Thurs, 2/7/2013: Chapters 3 and 4.
Hand in exercises 3.18*, 3.22, 3.29, 4.12*, 4.13*.
Graduate: exercises 3.8, 4.11.

Notes
* 3.18 Prove for all $k \geq 0$.
* 4.2 Stop the recursion whenever a multigraph whose underlying graph is a tree is encountered; to be noted in class, $\tau$ is the product of the edge multiplicities in this case.
* 4.12 Show the relevant data for each step of the tree reconstruction.
* 4.13 Show the relevant data for each step of the Pr"ufer code construction.

Correction for Previous Handout. On Handout 3, the line in the homework section which reads
* 2.23 should read ”Show that if a simple graph $G$ on $n \geq 1$ vertices...”.
    should say instead
* 2.23 should read “Show that if a simple graph $G$ on $n > 1$ vertices...”.

Third week summary

Tue 1/22. Call for questions on HW 2. We went over connected component, said generally more edges imply fewer components, but gave an example to show two graphs where the first one had both more edges and more components. We checked this: Let $G$ be a graph, and
\( v \in V(G) \) one of its vertices; then the set

\[
\{ u \in V(G) : \text{there is a} \ (u, v) \ \text{path} \}
\]

is a connected component of \( G \). Also discussed the underlying logic of negating a statement of the form \( s \implies t \); this required one of de Morgan’s laws; mentioned contrapositive, converse. Worked problem 1.20. Asked if anyone had looked at 2.16; they had not. Revisited definition of minor: \( H \) is a minor of \( G \) if \( H \) is isomorphic to the contraction of a subgraph of \( G \). Is every connected subgraph of \( G \) a minor of \( G \)? YES

Is every connected subgraph of \( G \) a contraction of \( G \)? NO

Is every induced, connected subgraph of \( G \) a contraction of \( G \)? NO

Took a second look at problems 2.34, 2.36.

**Thu 1/24.** How does adding an edge to a graph affect the number of connected components? We gave a definition of TREE which turned out to be a little different from the book’s, but then Theorem 3.7 shows that six different definitions all end up saying the same thing. Also, Theorem 3.4 asserts every tree with at least two vertices has at least two leaves.

We attempted to present the book’s proof of the theorem that in a tree with \( n > 1 \) vertices, and no vertex with degree greater than 3, the number of vertices of degree 3 is no more than \( \lfloor \frac{n}{2} \rfloor - 1 \). The proof is by induction, using the idea that when edges (bridges) are removed from a tree the resulting components are also trees.

We ended the class trying to demonstrate that the previous upper bound is infinitely often exact. The idea is an infinite sequence of trees \( T_2, T_3, \ldots \) constructed from the following recursion: for each leaf of \( T_n \), and two neighbors. The first member of the sequence, \( T_2 \) is a single central vertex of degree 3 whose 3 neighbors are all leaves. We determined
the following

<table>
<thead>
<tr>
<th>n</th>
<th>#leaves</th>
<th>#vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4 + 6 = 10</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>4 + 6 + 12 = 22</td>
</tr>
</tbody>
</table>

The total number of vertices is

\[ 4 + 3(2^{n-1} - 2) = -2 + 3 \cdot 2^{n-1}. \]

The total number of vertices of degree 3 is

\[ (-2 + 3 \cdot 2^{n-1}) - 3 \cdot 2^{n-2} = -2 + 3 \cdot 2^{n-2}. \]

In order to check that the upper-bound is tight, we must confirm

\[ -2 + 3 \cdot 2^{n-2} \equiv \left\lfloor \frac{-2 + 3 \cdot 2^{n-1}}{2} \right\rfloor - 1. \]

The completion of the check is left to the reader.